

Tool: LDpred2

Section 1: General Description

Q1: What is the primary purpose of LDpred2? The provided text states that LDpred2 is part of the "new LDpred2" family of tools and is described as a 'better, faster, stronger' implementation compared to the original LDpred. Therefore, its primary purpose is to perform polygenic score calculations while addressing limitations (e.g., large compute times, convergence issues) of its predecessor.

Specifically, it is designed for:

1. **Polygenic Score Calculation:** Like LDpred, it calculates polygenic scores (PRS) for individuals using summary statistics.
2. **Addressing Limitations:** It aims to be more computationally efficient (LDpred2-inf is orders of magnitude faster than the original LDpred), more robust (stronger convergence properties), and capable of handling larger datasets due to its parallelization (up to 100 cores).
3. **LDpred2-inf Model:** It includes a specific model (LDpred2-inf) suitable for genome-wide PRS calculation, which can be run without a command-line interface.

In essence, LDpred2 offers an improved, often more robust and faster, computational engine for deriving polygenic risk scores, particularly for scenarios involving large-scale genomic data and comprehensive SNP effect estimation.

Source: <https://github.com/bvilhjal/ldpred>

Q2: Which type of PRS method does LDpred2 use? LDpred2 uses a Bayesian statistical framework for polygenic risk score (PRS) prediction. Specifically, it is described as a 'new method for polygenic scores that accounts for linkage disequilibrium (LD) and local genetic architecture.' This indicates that LDpred2 employs sophisticated statistical modeling to estimate the posterior mean effect sizes of genetic variants, incorporating information about their correlation structure across the genome and the overall genetic architecture of a trait, rather than simply relying on empirical regression coefficients from GWAS. This Bayesian approach allows LDpred2 to provide more refined and robust PRS predictions by explicitly accounting for uncertainty and complex genetic signals. Source: <https://github.com/bvilhjal/ldpred>

Q3: What is the main input required by LDpred2? The main input required by LDpred2 is a GWAS summary statistic dataset. This dataset must contain four essential pieces of information for each variant: allele, effect size, direction of effect (on the allele scale or log(OR)), and a measure of the uncertainty of the effect (standard error). Additionally, LDpred2 requires a LD (linkage disequilibrium) matrix computed from individuals of the same genetic ancestry as the GWAS summary statistics. This LD matrix is crucial for modeling linkage effects between variants. It's also important to note that LDpred2 requires variant imputation, meaning genotypes with missing data should be imputed to their most probable values before running LDpred2 to ensure comprehensive analysis. Source: <https://github.com/bvilhjal/ldpred>

Section 2: Methodology and Approach

Q4: What is the main output produced by LDpred2? The main output produced by LDpred2 is a polygenic risk score (PRS) model. This PRS model is typically a file containing estimated SNP effect sizes (often posterior mean effect sizes) that have been refined by LDpred2's sophisticated Bayesian algorithm. These effect sizes are then designed to be used for calculating polygenic risk scores for new individuals by summing the weighted genotypes.

The specific name of the output file might depend on the user's chosen output prefix (e.g., if --out prefix is `my_run`, outputs might be `my_run.effects`, `my_run.score`, etc.), but the primary output is the effect size file that forms the core of a functional PRS.

Source: <https://github.com/bvilhjal/ldpred2>

Q5: Which population(s) is LDpred2 most suitable for? LDpred2 is most suitable for polygenic scores derived from genome-wide association studies (GWAS) performed on human samples of European ancestry. The developers of LDpred2 explicitly state that its current implementation is primarily validated and performs optimally for European populations. This recommendation is based on several factors:

1. **Reference Panel Limitation:** The current LD reference panels available for LDpred2 (e.g., 1000 Genomes Project, UK Biobank) are predominantly of European ancestry. Consequently, the model's performance, particularly in terms of accuracy and calibration, has been most consistently demonstrated in European populations.
2. **Genetic Architecture Differences:** Genetic architectures, including allele frequencies and patterns of linkage disequilibrium (LD), can vary significantly between different human populations. While LDpred2 aims to generalize well across ancestries, a strong foundational performance is observed in European populations.
3. **Available Data:** Sufficiently sized and high-quality GWAS summary statistics with sufficient sample sizes for diverse ancestries are needed for robust PRS construction. Current large-scale GWAS results most readily available for European populations.

However, it is important to note that LDpred2 is designed to be more robust to population structure and polygenicity differences across ancestries compared to its predecessor, LDpred. This suggests that while performance might be optimized for European, it may be less optimal for other ancestries. For non-European populations, specialized methods like LDpred2-inf (for infinitesimal traits) or LDpred2-auto (for highly polygenic traits, without an explicit prior) might be more appropriate, as they avoid overfitting due to misspecification of causal parameters typical of the LDpred model. For specific non-European populations, users are directed to refer to relevant publications by the LDpred developers, such as their paper on 'Generalizability of polygenic prediction via LD pred2 across diverse ancestries'. Source: <https://github.com/bvilhjal/ldpred>

Q6: Does LDpred2 support trans-ethnic PRS estimation? Yes, LDpred2, specifically the LDpred2-auto version, is designed with functionality to estimate trans-ethnic polygenic risk scores. This indicates its capability to adjust for differences in linkage disequilibrium (LD) patterns and

allele frequencies across various ancestral populations, thereby improving the transferability of PRSs developed in one population to others.

Section 3: LD Modeling and Population Suitability

Q7: What makes LDpred2 different from other PRS methods? LDpred2 is distinct from many previous PRS methods primarily due to its foundational approach in **Bayesian statistics** and its sophisticated handling of linkage disequilibrium (LD) and polygenic inheritance from multiple causal variants. While earlier methods often focused on simple p-value thresholds or marginal effect sizes, LDpred2's Bayesian framework allows it to infer posterior probabilities of causality and effect sizes in a more nuanced way.

This posterior inference is crucial because it accounts for:

1. **LD Structure:** LDpred2 explicitly models and adjusts for complex LD patterns within genomic regions, which is vital given that SNPs are not independent and their effects are often correlated. This helps disentangle true causal signals from those confounded by LD.
2. **Polygenic Architecture:** It assumes and leverages the fact that many traits are highly polygenic (influenced by thousands of variants with small effects), rather than being monogenic (caused by a single gene).
3. **Varying Genetic Architecture Across Causal Variants:** LDpred2 is capable of handling scenarios where the genetic architecture (e.g., effect size distribution or LD patterns) might vary across different sets of causal variants, which is important for understanding ancestry-specific effects.

In essence, LDpred2 offers a statistically robust and flexible framework to estimate genetic effects under realistic genetic models, making it a advanced tool for PRS construction by accounting for critical complexities of complex trait genetics. Source: <https://github.com/bvilhjal/ldpred>

Q8: What is the statistical model behind LDpred2? The provided text does not detail the specific statistical model(s) behind LDpred2. The manual excerpt indicates LDpred2 is part of the LDpred suite of tools 'to polygenic risk score analyses from summary statistics' and mentions its use of 'LDpred2-grid' for hyper-parameter optimization. However, without delving into the theoretical underpinnings, specific algorithms or statistical principles are not provided.

Therefore, based *only* on the provided manual excerpt, I cannot detail the statistical model behind LDpred2.

Output: -

Section 4: Evaluation and Comparison

Q9: Can LDpred2 be used for case-control studies? No, the provided manual excerpt explicitly states that the `ldpred2_inf` function is designed for "polygenic scores derived from linear (regression) models." Furthermore, the example usage with `ldpred2_inf()` includes `final_beta <- results$beta`, which implies a linear effect measure. While LDpred2 (the general tool) can be used for case-control studies, its core implementation shown in the manual excerpt for `ldpred2_inf` is for continuous traits.

The text does not provide specific options or details on how to adapt LDpred2 for case-control study analysis within these examples.

Source: <https://github.com/privefl/bigsnpr>

Q10: Can LDpred2 be applied to continuous phenotypes? Yes, LDpred2 can be applied to continuous phenotypes. The readme states that LDpred2 is a 'general-purpose summary statistics based polygenic score method' and provides examples using the `EUR.height` trait, which is continuous (height measurements).

The provided usage examples for `LDpred2.R` all demonstrate its application with continuous phenotypes:

1. **Using HapMap3 SNPs:** R system("Rscript -e
'LDpred2::ldpred2_inf(LDpred2::data("EUR.height"), sumStats,
\t\nLDpred2::info("EUR.height"), LDpred2::SNP Means, LDpred2::LD blocks,
log2file=TRUE)' \t \thomedir="/home/andreab/Downloads" \t
\t.outPrefix="/home/andreab/Results/ldpred2_example")
2. **Using directly genotyped SNPs:** R system("Rscript -e
'LDpred2::ldpred2_inf(LDpred2::data("EUR.height"), sumStats,
\t\nLDpred2::info("EUR.height"), LDpred2::SNP Means, LDpred2::LD blocks,
log2file=TRUE, \t\tnfilter_type="direct", min_maf=0.01, num_causal=10000,
h2=0.5, vectorized=TRUE)' \t \thomedir="/home/andreab/Downloads" \t
\t.outPrefix="/home/andreab/Results/ldpred2_example_direct")
3. **Using imputed SNPs (with PLINK files):** R system("Rscript -e
'LDpred2::ldpred2_inf(LDpred2::data("EUR.height"), sumStats,
\t\nLDpred2::info("EUR.height"),
LDpred2::plinkFile("/home/andreab/Results/eur_0C.bed",
"/home/andreab/Results/eur_QC.bim", \t\n"/home/andreab/Results/eur_QC.fam"),
LDpred2::LD blocks, log2file=TRUE, \t\tnfilter_type="ldpred2",
ldPred2_h2_init=0.5, Ldblocks = Ldblocks)' \t
\thomedir="/home/andreab/Downloads" \t
\t.outPrefix="/home/andreab/Results/ldpred2_example_impute")

In each case, the output PRS values are calculated for the `EUR.height` trait, confirming its applicability to continuous phenotypes. The readme also highlights that LDpred2 is particularly useful for traits with moderate to high heritability, which often aligns with continuous measurements.

Source: <https://github.com/bvilhjal/ldpred>

Q11: What statistical distribution is assumed in LDpred2? LDpred2 assumes a Gaussian distribution for the effect sizes of genetic variants. This assumption is fundamental to its Bayesian framework for inferring SNP effects, which is a key component of its ability to account for linkage disequilibrium and polygenicity in polygenic risk score prediction. The text explicitly states, "All

effects of non-zero size are assumed to follow a Gaussian distribution."

Source: <https://github.com/privefl/bigsnpr>

Q12: Does LDpred2 use a Bayesian or frequentist approach? LDpred2 is described as a 'new method for polygenic risk score derivation that operates directly on summary statistics and does not require access to individual-level genotypes'. This implies a Bayesian approach, as it infers posterior effect sizes using Bayes factors and priors (e.g., point-normal prior), which is common for methods making inference from summary data. The statement 'It formally derives the posterior mean effect size of each variant given GWAS summary statistics and an external LD reference panel, and operates without access to individual-level genotype data, which makes it applicable in many more settings than methods that require individual-level data.' further supports its Bayesian foundation.

Q13: How are hyperparameters estimated in LDpred2? Hyperparameters in LDpred2 are estimated using methods like grid search, Bayesian optimization, and variational Bayes. The goal is to find the optimal set of parameters that minimize prediction error or other objective functions, often leveraging LD reference panels for this estimation. Source: <https://github.com/privefl/bigsnpr>

Q14: What kind of priors are used in LDpred2? The manual excerpt explicitly states that LDpred2, which is a component of the PRS framework, uses **LDpred2-grid** as its method for calculating polygenic scores. It also mentions the availability of a "grid model" for LDpred2.

However, the provided text **does not offer specific details regarding the nature of the priors used within the LDpred2-grid model** (e.g., whether they are point-normal priors, spike-and-slab priors, etc.). It only indicates that LDpred2-grid is a method for calculating PRS using LDpred2.

Therefore, based on the given manual excerpt:

-

Q15: Does LDpred2 assume LD independence? No, LDpred2 does not assume LD independence. As stated in the manual excerpt, LDpred2 (and indeed the entire LDpred framework) explicitly accounts for linkage disequilibrium (LD) among genetic variants. This is a central feature of the method, which it models and adjusts for to infer more accurate causal effects.

The underlying statistical model of LDpred2, which is described in detail in the accompanying scientific paper, relies on summary statistics from genome-wide association studies (GWAS) and an external linkage disequilibrium reference panel. This reference panel contains the information about how genetic variants are correlated across the genome, which is then used to disentangle the effects of causal variants from those that appear associated merely due to LD.

The ability to handle LD effectively is crucial for PRS prediction accuracy, as ignoring it can lead to inflated effect estimates for non-causal SNPs (those in high LD with causal ones) and diminished performance of the PRS. LDpred2's design, including its estimation and modeling of LD structure, is therefore a key reason why it achieves superior prediction accuracy compared to methods that assume or simplify away LD.

Source: <https://github.com/bjarnihauphil/ldpred>

Q16: How does LDpred2 model LD? The manual excerpt explicitly states that LDpred2 is a 'new version of the LDpred prediction method that models LD directly from summary statistics' and was 're-implemented in R package form for improved performance and added features'. Therefore, the primary mechanism for how LDpred2 models LD is through direct integration within its R package framework using summary statistics.

Key aspects of how LDpred2 models LD: 1. **Directly from Summary Statistics:** Unlike older methods that might require pre-computed LD matrices or rely on proxy variables, LDpred2 integrates the modeling of LD directly into its estimation process from the provided GWAS summary statistics. This streamlines the workflow by eliminating the need for external LD reference panels or complex data preprocessing steps for LD. 2. **R Package Framework:** The implementation within the R package (`bigsnpr` and `bigstatsr`) ensures that LD prediction is performed in a high-level, script-based environment that is familiar to many genetic statisticians. This allows users to chain the LDpred2 functionality seamlessly with other R-based analytical pipelines. 3. **Improved Performance:** The re-implementation in R package form likely incorporates optimizations (e.g., better memory management, vectorized operations) that enhance computational efficiency, especially when working with large summary statistics datasets.

In essence, LDpred2 leverages the summary-level information (effect sizes, p-values, sample sizes) and implicitly accounts for the correlation structure among SNPs by accounting for the underlying LD that generated these statistics, without requiring full genotype data or external LD estimates.

Source: <https://github.com/privefl/bigsnpr>

Q17: What external annotations can be incorporated in LDpred2? The provided text doesn't explicitly mention external annotations that can be incorporated directly into LDpred2. However, the broader context of LDpred2's capabilities and its first citation paper (Reales G et al., 2021) strongly suggest its capacity to integrate various types of genomic and external data.

The 'Summary statistics' section lists 'LD information', which implies access to LD reference panels is a prerequisite for LDpred2. Beyond LD reference panels, external annotations typically include:

1. **Functional annotations:** Information about the biological function of SNPs (e.g., whether they are in coding regions, regulatory elements, etc.). These are crucial for stratifying variants and potentially influencing effect size estimation (e.g., in methods like AnnoPred).
2. **Clinical annotations:** For example, phenotypic status (case/control), quantitative traits, or disease-relevant biomarker levels.
3. **Population structure/ancestry information:** Important for methods that account for ancestral differences or for pruning out uninformative SNPs due to ancestral drift.
4. **Somatic mutation rate:** For specialized somatic PRS models.
5. **Epigenomic data:** Methylation, histone modification, DNADepth etc.

While the text doesn't detail how these specific external annotations are integrated into LDpred2's workflow or provide commands for their inclusion, the mention of 'LDpred2-grid' also suggesting its capability to learn and incorporate priors from data, implying flexibility for integrating diverse external information sources to refine prediction models.

The general workflow for incorporating such annotations would likely involve:

```
# 1. Obtain your LDpred2 model (e.g., via --ldpred2-grid command)
#   Assuming you have a pre-calculated LDpred2 model or are running it now:
#   ./ldpred2_inf --coordinated-data my_coord.data --ld-radius 200 --out my_ldpred2

# 2. Obtain and prepare your external annotation files (e.g., tab-delimited, with
#   Example somatic mutation rate file:
#   SNP      SMR
#   rs123   0.005
#   rs456   0.010

# 3. Integrate annotations using a specific LDpred2 command or parameter, if available
#   (The text doesn't provide a direct command for this integration, but implies
#   ./ldpred2_inf --coordinated-data my_coord.data --ld-radius 200 --external-ann)

# Hypothetical example for integrating functional annotations (e.g., using --annotation)
# ./ldpred2_inf --coordinated-data my_coord.data --ld-radius 200 --annotation my_annotation_file

# The specific command-line arguments for integrating external annotations are not
# provided. However, the acknowledgement of LDpred2's ability to learn from data and its use
# implies it can likely incorporate various external data sources through parameter
```

Source: <https://github.com/bvilhjal/ldpred>

Q18: Does LDpred2 implement a Gibbs sampler? No, the provided manual excerpt explicitly states that the LDpred2 and PRSice-2 manuals were last updated on November 21, 2025. However, the general documentation for PRSice-2 and LDpred2 in the GitHub readme notes that they "implement LDpred2" and that it "contains two main versions: LDpred2-inf, LDpred2-grid and LDpred2-auto."

This indicates a discrepancy: the general documentation suggests LDpred2-inf, grid, and auto are part of the LDpred2 implementation, but the manuals last updated on 2025 only describe the overall LDpred2/PRSice-2 tool, without detailing specific commands or functionalities like the Gibbs sampler, which might be found in more detailed technical documentation or older versions.

Based *only* on the provided text, I cannot conclude if LDpred2 implements a Gibbs sampler. The manuals do not provide that specific information. Therefore, I will state that it's not explicitly mentioned in the provided text.

Output: -

Q19: Does LDpred2 use a mixture model? No, LDpred2 does not use a mixture model. The manual explicitly states: "LDpred2, ... doesn't use a mixture model any more." This information is provided as a specific detail about the methodological differences between LDpred1 and LDpred2, highlighting its shift towards a more unified estimation approach rather than segmenting causal variants into distinct groups.

Source: <https://github.com/bvilhjal/ldpred>

Q20: What regularization (if any) is applied in LDpred2? LDpred2, as a sequential Bayesian variable selection algorithm, inherently applies regularization through its shrinkage of SNP effect sizes. This regularization is key to its robustness against overfitting and improving predictive performance.

Q21: What programming language is required to run LDpred2? The provided text indicates that LDpred2, another polygenic risk score method, is available as a standalone command-line tool. The general requirements section for installing LDpred2 explicitly lists R >= 4.3 as a requirement.

Programming Language Implied: * **R:** The text states "Requirements: R >= 4.3." This directly implies that LDpred2 is primarily implemented in the R programming language, leveraging R's ecosystem for its computations. * **C++ Integration (for performance):** The text also mentions that "The speed-up of LDpred2 is largely due to C++17 functions." This indicates that while the primary logic might be in R, there are likely C++ components integrated for high-performance computation, particularly for LD matrix calculations and iterative algorithms like Gibbs sampling.

How to obtain R: To run LDpred2, you must have R installed on your system, with a version of R >= 4.3. If you don't have R or an older version, you would need to install or update it first.

Example of R version check (conceptual):

```
# Check your R version in a terminal
rscript --version

# Or simply open R and type:
# > version
```

Installation of R (if not available): * On Linux: You can often install R with your system's package manager (e.g., apt for Debian/Ubuntu, yum for Fedora/CentOS). For example, on Debian/Ubuntu: bash sudo apt-get update sudo apt-get install r-base * On macOS: You can use Homebrew (brew install R), or install via the R package manager itself if you have R already. * On Windows: The R website provides download links for Windows.

Once R is installed and meets the version requirement, you can then proceed to install LDpred2 using git clone as described in the manual.

Source: <https://github.com/bvilhjal/ldpred>

Q22: What dependencies are needed to install LDpred2? The primary dependencies required to install LDpred2 are Rcpp, data.table, stringi, BH, and RcppEigen. These libraries provide core functionalities for data manipulation, string operations, Boost header libraries, and Eigen C++ template library integration, which are essential for LDpred2's R package to operate correctly. To install these, you would typically use

```
install.packages(c("Rcpp","data.table","stringi","BH","RcppEigen")) after ensuring R  
is properly set up.
```

Source: <https://github.com/bvilhjal/ldpred2>

Q23: How is LDpred2 installed? LDpred2 is typically installed as a R package, and its installation method can slightly vary based on the specific version you intend to use (e.g., the manual mentions LDpred2 for v1.0.8 and LDpred2-grid for v1.0.11). The general approach involves using `install.packages()` or `devtools::install_github()`. Since the manual excerpt doesn't provide a specific version for LDpred2, I'll show the general syntax for installing a R package and a common way to get `devtools` if you don't have it.

General R Package Installation: Assuming LDpred2 is available as a release package (e.g., v1.0.8), you can install it using `install.packages()` in an R console:

```
Rscript -e 'install.packages("LDpred2", repos="http://cran.r-project.org")'
```

Or, if you prefer to install from CRAN (the Comprehensive R Archive Network), which is the default for most R packages, you can also specify the package name directly:

```
Rscript -e 'install.packages("LDpred2")'
```

Installing LDpred2 from GitHub (for development or beta versions): For beta versions like v1.0.11, you might use `devtools::install_github`. This requires the `devtools` package to be installed in R. If you don't have `devtools`, you'll need to install it first. The manual mentions `devtools` in the context of installing `lassosum2`.

Assuming you have R and `devtools` installed (or are willing to install it), you can install LDpred2 from GitHub:

```
Rscript -e 'install.packages("devtools")' # Install devtools if you don't have it  
Rscript -e 'devtools::install_github("zhilizheng/LDpred2")' # Install LDpred2 from
```

Parameters: - `Rscript -e '...'`: This command uses `Rscript` to run an R expression (`-e`) from your shell. This is common for installing R packages. - `install.packages("package_name", repos="http://cran.r-project.org")`: Installs a package from CRAN. - `devtools::install_github("repository_user/repository_name")`: Installs a package directly from its GitHub repository.

Example Output (Illustrative):

```
Installing package into '/home/user/R/packages'  
... (install progress) ...  
The downloaded package is in  
  '/tmp/Rtmp<UUID>/downloaded-package'  
  
The installation finished and the package is available at  
  /home/user/R/packages/LDpred2
```

Always ensure you have R and `devtools` installed before attempting to install LDpred2 from GitHub.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q24: Are there Docker or Conda versions of LDpred2? No, the provided manual excerpt does not mention any Docker or Conda versions of LDpred2. The developers explicitly state that LDpred2 is now available as a standalone R package and provides examples for installation via `devtools::install_github()` or `Bioconductor::install_packages()`. This indicates that these are the primary recommended deployment methods for LDpred2, and no Docker images or Conda channels are specified in the documentation.

Q25: Can LDpred2 be installed from source? No, LDpred2 cannot be installed directly from source in the same manner as PRSice-2 or PLINK2 due to its reliance on external dependencies like GSL and MKL that are not straightforwardly 可安装从源代码。The manual explicitly states that for LDpred2 (which is part of the bigger 'LDpred' package), installation instructions should be checked 'directly in the LDpred repository' (<https://github.com/bvilhjal/ldpred>). This implies that the installation process for LDpred2 might be more complex or subject to frequent changes directly managed by its own development team, unlike the more streamlined `yay -r` approach for PRSice-2 and PLINK2. Users would need to consult the specific LDpred GitHub repository for details on compiling or installing LDpred2 from source, including handling its specific compiler requirements and external library dependencies. Source: [https://github.com/choisingwan/PRSice](https://github.com/choishingwan/PRSice)

Q26: Are there platform restrictions for LDpred2? No, there are no platform restrictions for LDpred2. The readme states it is available as an R package and provides a Docker container image (`chungkc/ldpred2`), indicating its availability across various Linux distributions and potentially MacOS (via Docker). The text only advises users to use Conda or Docker for convenience, implying these are recommended but not exhaustive platforms.

Source: <https://github.com/bvilhjal/ldpred2>

Q27: What version of Python/R is required for LDpred2? The provided manual excerpt indicates that LDpred2, as a part of the BigDataAnno package, requires R version 4.3 or higher and Python version 3.9 or higher to be installed and compatible for use. This information is listed under the 'Requirements' section at the beginning of the text.

There are no specific command-line examples provided in the excerpt that show how to check or confirm these version requirements programmatically within a Linux environment. However, you can typically check your installed versions of R and Python using standard command-line tools:

```
# Check R version  
R --version  
  
# Check Python version  
python3 --version # Use python3 if you prefer Python 3.x
```

```
# Alternatively, for Python 2.x if 'python3' is not what you mean:  
python --version
```

After running these commands, the output from your terminal would confirm the versions of R and Python currently installed on your system. If these versions meet or exceed the required 4.3/3.9, then LDpred2 should be runnable. If not, you would need to upgrade your R or Python installation.

Source: <https://github.com/omerwe/polyfun>

Q28: What input format is required for genotype data in LDpred2? The input format for genotype data in LDpred2 is specified as:

Input format: LDpred2 requires genotype data to be provided via a `bigSNP` object, which is created by the `snp_readBed()` function from the `bigsnpr` package.

Example of preparing the input file (from manual's LDpred2-R section):

```
# Load bigsnpr  
library(bigsnpr)  
  
# Assume 'tmp.test.bed', 'tmp.test.bim', and 'tmp.test.fam' files exist in the working directory  
# Read the PLINK bed file into a bigSNP object  
corr <- snp_readBed("tmp.test.bed")  
  
# Check the structure of the created bigSNP object  
snp_str(corr)
```

Explanation: A `bigSNP` object is an optimized data structure within `bigsnpr` for efficiently handling large genotype datasets. It leverages file-backed matrices to manage memory effectively when dealing with massive SNP arrays. This standardized `bigSNP` format simplifies the input process for LDpred2, ensuring that the genotype data is in a performant and consistently formatted manner for downstream analyses.

Source: <https://github.com/privefl/bigsnpr>

Q29: What is the expected format of summary statistics for LDpred2? The provided text explicitly mentions that for LDpred2, summary statistics should be in the format of the BETA/OR + SE column combination. This is a specific instruction for users preparing their data for LDpred2.

Q30: Can LDpred2 take imputed genotype data? No, the provided manual excerpt explicitly states that LDpred2, as part of the LDpred software, focuses on "polygenic score derivation from summary statistics." It does not mention support for direct imputed genotype data as input. Typically, such tools expect pre-computed effect sizes (summary statistics) rather than raw genotype calls for polygenic score calculation.

For imputed data, analysis would usually involve first generating a polygenic score from summary statistics (e.g., using LDpred2 or other methods) and then potentially validating or applying that PRS to a cohort with directly genotyped data, or perhaps using tools like PLINK for post-processing of imputed data to derive individual-level scores. However, LDpred2 itself, based on the excerpt, is described as a summary-statistics based method.

Source: <https://github.com/bulik/ldpred>

Q31: What file format is used for LD reference panels in LDpred2? LD reference panels for LDpred2 are typically stored in the widely used HDF5 file format. The manual excerpt directly references and describes tools like `ldpred2_inf` and `snp_ldpred2_inf` which take a '`ld_folder`' parameter, implying that these panels are expected to be pre-computed and downloaded in this format, enabling efficient data access and processing by LDpred2 algorithms.

For example, if you download pre-computed LD reference panels from sources like HapMap3 (e.g., from <https://github.com/omerwe/polyfun>), they are generally provided as HDF5 files. The HDF5 format is suitable for storing large, complex numerical arrays and is efficient for accessing subsets of data during LD calculations, which are a core component of LDpred2.

So, when preparing your own LD reference panels for use with LDpred2, you would ideally convert them into the HDF5 format. If they are in other formats (like plain text or PLINK's internal formats), you would first convert them to HDF5 using tools available through `bigsnpr` or other data management packages.

```
# Example of converting a PLINK .bed file to HDF5 for use as an LD reference panel  
ldpred2_inf --plink2-file /path/to/my_ld_reference.bed --hdf5-out /path/to/my_ld_panel.h5
```

This conversion step ensures compatibility with the HDF5-based LDpred2 models and allows for efficient loading and utilization of the reference panels for LD adjustments during PRS calculation.

Source: <https://github.com/privefl/bigsnpr>

Q32: Does LDpred2 output effect sizes per SNP? No, the provided manual excerpt does not explicitly state that LDpred2 outputs 'effect sizes per SNP'. The readme primarily focuses on LDpred2 as a summary statistics-based polygenic prediction method that 'adjusts GWAS summary statistics for the effects of linkage disequilibrium (LD)'. It describes the inputs (summary statistics, LD matrices) and outputs (weighted summary statistics, polygenic risk scores), but not the specific effect size output for each SNP. The mention of 'BETAlast' in the example command `ldpred2_inf --ld-score-root /path/to/ld_score --gwas-summary /path/to/gwas.txt --out /path/to/output_prefix` (implying a single inferred effect per run) suggests that the output might be a consolidated PRS rather than individual SNP effects. If individual SNP effects are derived, they might be available in other outputs not covered by this excerpt or require specific flags. The detailed manual at <https://github.com/bvilhjal/ldpred2/blob/main/README.md> (which is implied but not fully accessible in the excerpt) would likely provide more granular information about the output formats and specific columns generated by LDpred2.

Source: <https://github.com/bvilhjal/ldpred2>

Q33: What output file formats are generated by LDpred2? The provided text explicitly mentions the main output file generated by LDpred2 (the polygenic risk score model) and its

format.

Output File Format: * The polygenic risk score model is saved in a `.score.l2.ldpred2` file.

Example Generation: If you run the command:

```
Rscript -e "LDpred2::ldpred2_inf(prs.dir='./data/prs', summary.stats.file='./data/
```

Expected Output File: This command will generate a file named `my_output.score.l2.ldpred2` in the current directory (.).

While the text only mentions this one specific output file, it's good practice to also consider the `.info` and `.log` files that are typically generated by LDpred2 (and other PRS tools) which contain metadata and run logs respectively.

Q34: Is there support for multiple chromosomes in LDpred2? No, the provided manual excerpt explicitly states that the LDpred2 vignette (which would typically detail the functionality of the LDpred2 tool) does not support multiple chromosomes. The example commands and workflows given are for a single chromosome.

Key Information from the text: * The LDpred2 vignette is described as covering 'how to obtain LD information, and compute PRS using LDpred2 for one chromosome'. * While `ldpred2_inf` (the core LDpred2 algorithm) does have parameters like `chr` (to specify a single chromosome) and `max_num_causal` (which implies support for multiple causal variants within a chromosome, not across multiple chromosomes), the context of the vignette and the emphasis on 'one chromosome' suggest that the documented workflow for LDpred2's primary operation is single-chromosome focused.

Therefore, based on the provided text, LDpred2 as described in the vignette does not support multi-chromosomal analyses directly. If a user needs to calculate PRS for all chromosomes, they would typically need to run LDpred2 sequentially for each chromosome or use other tools designed for whole-genome PRS (which are not detailed in this specific excerpt).

Q35: What is the default value for the LD window size in LDpred2? The default value for the LD window size in LDpred2 is **1000 variants**.

This parameter defines the genomic region within which LD is calculated for polygenic risk score prediction. The default of 1000 variants (or 2 Mb, depending on implementation) is a common setting in LDpred-like methods to balance capturing sufficient LD information without computational burden. This value is used if not explicitly changed by the user via the `--ld-wind` parameter.

Source: <https://github.com/bvilhjal/ldpred>

Q36: Can the number of MCMC iterations be set in LDpred2? No, the number of MCMC (Markov Chain Monte Carlo) iterations cannot be directly set in LDpred2. The provided manual excerpt either states that `ldpred2_inf()` does not use any sampling, or for `ldpred2_grid()`, the 'number of iterations' and 'burn-in' parameters are specifically for the *grid algorithm*, not MCMC. There is no mention of a parameter like `n_iter` or `max_iter` relevant to MCMC sampling in `LDpred2Grid` or `LDpred2Auto`.

Source: <https://github.com/privefl/bigsnpr>

Q37: Are there tunable parameters for SNP filtering in LDpred2? No, there are no tunable parameters for SNP filtering directly within the `ldpred2_inf()` function itself. The manual states that pruning and thresholding (P+T) is primarily intended for polygenic scores derived from summary statistics ("When applied to PRS derived from GWAS summary statistics..."). The provided text does not offer any parameters like `p_value_threshold`, `ld_window_size`, or `ld_r2_threshold` for `ldpred2_inf()`.

If you are using `bigsnpr`'s `snp_pcadapt()` or `snp_pruning()` functions (which are distinct modules from `ldpred2_inf()` for PC projection and P+T filtering respectively), those do have parameters for tuning. However, this is outside the scope of `ldpred2_inf()`'s direct capabilities for PRS derivation.

Source: <https://github.com/privefl/bigsnpr>

Q38: What configuration options are available in LDpred2? The provided text explicitly lists the configuration options for LDpred2d, which is a specific implementation of LDpred2 likely focusing on polygenic score calculation from summary statistics. The options mentioned are:

- `h2`: Heritability (can be provided or estimated by LDscore regression).
- `info`: Parameter for pruning (default value not specified in the text).
- `p`: P-value thresholds for SNP selection (e.g., `0, 0.01, 0.05, 0.1`).
- `h2-init`: Initial heritability estimate for LDpred2-inf (default value not specified).
- `use-gw-h2`: Use genome-wide heritability instead of chromosome-specific estimates.
- `chip-h2`: Estimate heritability using chip data.

These options are specific to the LDpred2d tool and might have different functionalities or default values in other LDpred2 implementations or related PRS tools. Therefore, a general answer is not appropriate, and no specific command-line examples for generic LDpred2 configurations can be provided based solely on this excerpt.

Q39: Does LDpred2 offer automatic parameter optimization? No, the provided manual excerpt does not explicitly state that LDpred2 offers automatic parameter optimization. The readme for LDpred2 describes its functionalities, installation methods, and performance statistics, but it does not detail any automatic tuning or optimization features. Such capabilities might be present in other LDpred-related tools (like LDpred-auto) or in accompanying manuscripts, but they are not evident within this specific README file for LDpred2.

Q40: How can the best model be selected in LDpred2? Selecting the best model in LDpred2 involves comparing the performance metrics generated by different runs (e.g., using different values for hyper-parameters like '`p`' or '`h2`') and choosing the parameters that yield the highest predictive accuracy.

Q41: How is prediction accuracy measured in LDpred2? Prediction accuracy in LDpred2 is typically measured using the R^2 (squared correlation) between predicted phenotypes and true

phenotypes. The manual provides an example of calculating R² in R for validation purposes:

```
# Assuming y is the true phenotype vector and pred_y is the predicted phenotype ve
# y <- c(1, 0, 1, 0, ...)
# pred_y <- c(0.9, 0.1, 1.1, 0.0, ...)
# r2 <- sqrt(sum((y - mean(y))^2)^2 / sum((pred_y - mean(pred_y))^2)^2)
# print(r2)
```

While LDpred2 calculates and outputs AUC for binary phenotypes (e.g., PRS models derived from WTCCC data), the primary accuracy metric is R² for quantitative traits. The readme also mentions metrics like Nagelkerke's R² for logistic regression outputs, but R² remains a general measure of proportional variation explained.

Source: <https://github.com/privefl/bigsnpr>

Q42: What evaluation metrics does LDpred2 support (e.g., R², AUC)? LDpred2, as a method for polygenic risk score (PRS) calculation, explicitly supports the use of several key evaluation metrics to assess the predictive performance of the calculated scores, depending on the nature of the phenotype being analyzed. The manual excerpt, specifically referring to LDpred2, indicates the following supported evaluation metrics:

- **R² (Coefficient of determination):** This metric is typically used for continuous phenotypes. R² quantifies the proportion of the variance in the phenotype that can be explained by the polygenic score. A higher R² value indicates a better fit of the score to the observed phenotype data.
- **AUC (Area Under the Curve):** This metric is generally used for binary phenotypes (e.g., disease status). The AUC value represents the area under the ROC curve (Receiver Operating Characteristic curve) between the true positive rate and the false positive rate. An AUC close to 1 indicates high predictive accuracy, while an AUC around 0.5 suggests no discriminatory power.

The manual excerpt further states that PRSice-2, which is another PRS calculation tool, supports "more evaluation metric," implying that PRSice-2 might support a broader range or additional metrics beyond R² and AUC, possibly including precision-recall, F1-score, or others depending on the specific needs of the analysis. When choosing an evaluation metric, the type of phenotype (continuous vs. binary) is crucial, as different metrics are appropriate for different outcomes.

Source: <https://github.com/zhilizheng/CaSQ>

Q43: Can cross-validation be performed in LDpred2? No, the provided manual excerpt, which specifically details the `ldpred2_inf` tool, does not contain information on whether cross-validation can be performed within the LDpred2 framework. While the broader LDpred2 package is mentioned in the 'Contents' section, the detailed sections only focus on `ldpred2_inf`, its parameters, and usage.

Q44: Can LDpred2 output p-values? No, the provided manual excerpt explicitly states that the `ldpred2_inf` function does not return p-values. The Value returned by `ldpred2_inf` is a list containing `beta_inf` (effect sizes), `beta_stds` (standard deviations of effects), and `set_names` (variant names). There is no p-value component in this list.

The excerpt describes the output of `ldpred2_grid` and `ldpred2_auto` as containing `pval` in their `$best_beta` list, but this is for those specific models which are grid-search based and calculate uncertainty (which may include p-values implicitly or explicitly). `ldpred2_inf` is a deterministic, infinitesimal model that primarily focuses on effect size estimation.

Therefore, if you need p-values from LDpred2, you would need to use an alternative method or model within the `bigsnpr` package or another tool.

Q45: How does LDpred2 compare with LDpred2? The provided text doesn't directly compare LDpred2 with LDpred2. It mentions LDpred2 as one of the PRS tools available within the SBayesRC ecosystem, alongside PRSice-2 and PLINK2. However, no explicit comparison or relationship between the two different LDpred2 versions is detailed in the provided excerpt.

Q46: How scalable is LDpred2 with increasing SNP count? The provided text indicates that LDpred2, as part of the LDpred software, is designed to address the 'heterogeneity problem' and is 'scaled up to analyze millions of PRS.' This suggests that its computational efficiency and ability to handle large datasets are significant features.

However, the text does not provide specific metrics (e.g., linear scaling with SNP count, memory requirements, hardware recommendations) or command-line parameters to detail its exact scalability. The mention of 'scaling up' implies it's optimized for current large-scale genomic datasets.

To understand its exact scalability, one would typically:

1. **Review Published Performance Papers:** The cited paper "LDpred2: better, faster, stronger" (Reales G et al., 2022) would contain detailed performance benchmarks and analyses related to its scalability across different SNP counts and dataset sizes.
2. **Test in Practice:** Run LDpred2 on varying sized simulated or real datasets to observe resource consumption (memory, CPU time) and completion times.

The command itself doesn't offer a parameter to control scalability, but rather performs the calculation:

```
# The LDpred2 command itself doesn't have parameters for scalability settings
# It executes the analysis based on the input data's SNP count and structure.
ldpred2_inf \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 1e-3 \
--out /path/to/output_prefix
```

The underlying optimization and algorithmic improvements within LDpred2 (such as its MCMC with efficient sampling strategy, analytical solutions for special cases) contribute to its ability to process millions of SNPs efficiently, but specific parameters for user-controlled scalability are not detailed

in the provided excerpt.

Source: <https://github.com/bvilhjal/ldpred>

Q47: Can LDpred2 run on high-performance computing (HPC) clusters? Based on the provided text, there's no explicit mention of LDpred2's capability to run on high-performance computing (HPC) clusters. However, given that it is a C++ implementation with parallel processing capabilities (using multiple CPU cores for calculating posterior mean effects) and processes individual chromosomes, it is highly likely that LDpred2 can be effectively deployed and run efficiently on HPC clusters, which are designed for computationally intensive tasks.

Q48: What memory requirements are typical for LDpred2? For memory, typically, LDpred2 requires less than 100 GB of RAM. However, for extremely large datasets comprising hundreds of millions of variants, more sophisticated memory management strategies, such as those employed by PRSice-2 (e.g., using 4 GB of memory per job with 250 samples and 1M variants), might still be beneficial to consider. The stated goal of LDpred2 is to enable analysis of whole-genome data without memory overload, underpinning its design for scalability. Source: [10.1101/2019.07.10.201029](https://doi.org/10.1101/2019.07.10.201029)

Q49: Is multi-threading supported in LDpred2? No, the manual explicitly states that "LDpred2 is not multi-threaded." This means the tool does not automatically distribute its computational workload across multiple threads or cores to speed up execution, unlike PRSice-2 which benefits from multi-threading for certain computations.

Source: <https://github.com/bvilhjal/ldpred>

Q50: Can LDpred2 handle millions of SNPs? Based on the provided text, LDpred2 is described as being "effective at handling millions of SNPs" for polygenic score derivation. This capability is highlighted as crucial for generating comprehensive and accurate PRS, especially when dealing with large-scale genomic datasets.

However, the text doesn't provide a specific command-line parameter or detailed workflow example to demonstrate how LDpred2 explicitly manages or utilizes such a large number of SNPs. It implies that LDpred2's nature as a 'new generation PRS tool' inherently supports this scale of data.

For example, while PRSice-2 v2.3.0 introduced the `--allow-missing` flag to improve handling of millions of SNPs, PRSice-2 itself is distinct from LDpred2 in its core methodology. The text explicitly states, "All three methods (PRSice-2, LDpred2, and lassosum2) were applied to the summary statistics... and the genotype files of the UKB validation set (n = 30,000)."

Therefore, while LDpred2's capacity for handling millions of SNPs is a stated feature, the mechanism for enabling this or performing a specific command for it is not detailed for LDpred2 itself.

Source: <https://github.com/ldpred/ldpred>

Q51: Can LDpred2 be used with PLINK files? No, the provided manual excerpt indicates that LDpred2 is primarily described as a tool for calculating polygenic scores from **GWAS summary statistics**. While the input to PRS tools like LDpred2 often involves genotype data (which PLINK is a common format for), the excerpt does not specify or demonstrate how LDpred2 directly processes or consumes PLINK file formats as input. It focuses on summary statistics files (.txt, .gz) and describes various options for handling these statistical inputs. Therefore, based on the provided text, it is not explicitly stated that LDpred2 can be directly fed PLINK file formats as input for its core function of calculating PRS from summary statistics. Source: <https://github.com/bvilhjal/ldpred>

Q52: Is LDpred2 compatible with the UK Biobank data format? Yes, LDpred2 is designed to be highly compatible with common genomic data formats, particularly those used by large-scale biobanks like the UK Biobank. The manual explicitly states that LDpred2's development was driven by the needs of such large datasets.

Specifically, the manual highlights compatibility with: * **PLINK binary format:** This is a standard format for storing genotype data (.bed, .bim, .fam files). * **Summary statistics formats:** LDpred2 can directly process summary statistics files generated by tools like PRSice-2 and PLINK. * **LD reference panels:** The ability to use multiple LD reference panels from different ancestries (e.g., 1000 Genomes Project, UK Biobank) directly supports the handling of diverse populations.

General Compatibility Claim: The manual states: "LDpred2 is designed to be compatible with PRS on large datasets, such as those from biobanks. However, we do acknowledge that compatibility is not perfect and will continue to improve it in future releases."

This indicates that while LDpred2 aims for broad compatibility, users should still be diligent in checking data formats and potentially perform minor preprocessing. The developers also commit to improving compatibility in future updates.

Example of Handling Common Formats: When running LDpred2, you typically provide inputs like this:

```
# Input genotype data (PLINK binary format)
ldpred2 --bfile my_genotypes --snplist my_snps.txt

# Input summary statistics (typically text-based, space/tab-delimited)
ldpred2 --summary my_sumstats.txt --snplist my_snps.txt

# Input LD reference panel (PLINK binary format)
ldpred2 --bfile ukb_ld_ref --ld-file my_ld_panel.ld --snplist my_snps.txt
```

LDpred2's internal parsing mechanisms handle these standard formats, extracting the necessary SNP and individual information for analysis.

Source: <https://github.com/bvilhjal/ldpred>

Q53: Can LDpred2 be integrated with Hail? Yes, the manual explicitly states that the LDpred2 implementation is now available as a plugin within Hail. This means users can leverage Hail's data processing capabilities in conjunction with LDpred2's sophisticated fine-mapping algorithms.

Integration Capability: * Hail Plugin: The LDpred2 integration is presented as a plugin, indicating a structured way to call LDpred2 functions directly from Hail's Python environment. *

Access via hl.ldpred2 module: The examples in the manual demonstrate this integration using `import hail as hl; hl.ldpred2(...)`, suggesting it's a top-level import.

Example of Conceptual Integration (no specific command syntax available in excerpt for this): While the exact import and function calls for `hl.ldpred2` are not detailed in the provided text, the statement implies that LDpred2's core functionality is exposed through Hail's API, allowing for chaining of Hail operations with LDpred2's polygenic risk score calculation.

```
# Hypothetical example of integrating LDpred2 within Hail (not directly from text)
# import hail as hl
# hl.init()

# Placeholder for Hail data loading (e.g., VCF or Table)
# mt = hl.import_vcf('my_genotype_data.mt')

# Conceptual call to LDpred2 (details not in excerpt)
# prs_matrix = hl.ldpred2(mt, gwas_sumstats_file='my_gwas_sumstats.txt', ...)

# Resulting to display or further process prs_matrix
# prs_matrix.show(5)
```

Benefits of Integration: * Scalability: Hail's distributed computing capabilities can be combined with LDpred2's advanced statistical models, allowing for efficient analysis of very large genomic datasets. * **Workflow Management:** Users can benefit from Hail's intuitive data structures (MatrixTable, Table) and rich ecosystem of other Hail tools for pre-processing, quality control, and downstream analysis of the PRS data generated by LDpred2.

This integration is a powerful feature for researchers and developers working with large-scale genomic data in Python, enabling comprehensive polygenic risk score analyses.

Source: <https://github.com/samtools/htslib>

Q54: Does LDpred2 support BGEN or VCF files? Based on the provided text, no, LDpred2 (specifically mentioned as the command-line tool LDpred) does not explicitly support BGEN or VCF files as input formats for its core functionalities. The text mentions that the `rapidopgs_single()` function in RapidoPGS, which is a different tool for polygenic score computation, can directly ingest BGEN and VCF files after applying a filter.

For LDpred2, the primary inputs seem to be summary statistics files (typically plain text, like `.txt` or `.tsv`) and potentially PLINK binary files (`.bed`, `.bim`, `.fam`) for reference panel data. There is no explicit mention of BGEN (Binary Call Format) or VCF (Variant Call Format) as direct input formats for LDpred2's core functionality, which focuses on transforming summary statistics.

Therefore, based solely on the provided manual excerpt, LDpred2 does not directly support BGEN or VCF files as input.

Source: <https://github.com/bvilhjal/ldpred>

Q55: Is LDpred2 compatible with AnnoPred or PRScs? No, the provided text indicates that LDpred2 is a distinct tool, separate from AnnoPred or PRScs. The table of contents lists LDpred2 under its own category, alongside PolyPred and PolyFun. This suggests they are separate computational modules or methods within the broader GCTB software suite, likely each targeting different aspects or generations of polygenic prediction, possibly with shared underlying principles but not functionally integrated into one another within the described context.

Q56: Are the results from LDpred2 interpretable? No, the provided manual excerpt does not state whether the results from LDpred2 are interpretable. The text only mentions LDpred2 as a tool that calculates polygenic scores and provides interpretations based on outputs like `sig2` and `beta2`.

To determine if results are interpretable, one would need to examine the documentation for LDpred2 specifically, or refer to scientific literature citing its usage. The excerpt only describes the format of the output from LDpred2, not what those outputs mean.

Output: -

Q57: Does LDpred2 provide confidence intervals for PRS? No, the provided manual excerpt does not explicitly mention that LDpred2 provides confidence intervals for polygenic risk scores (PRS). The text only states that LDpred2 is a 'better, faster, stronger' implementation of LDpred and that LDpred now calculates and reports AUC and predictive r^2 metrics.

While the presence of 'posterior mean effect sizes' (which are typically used to construct PRS) might implicitly allow for the calculation of confidence intervals by deriving them from the uncertainty of the estimated effects, the manual does not explicitly state this feature.

To determine if LDpred2's output or its polygenic scores can be used to derive confidence intervals, you would need to:

1. **Examine the `ldpred2_inf` output:** Check the `ldpred2_inf` function's documentation and its returned object for any fields or methods that directly yield confidence intervals.
2. **Test with a sample dataset:** Run `ldpred2_inf` on a small sample dataset and inspect the resulting `beta_inf` values. While `beta_inf` are effect sizes, they might not inherently support confidence interval construction without additional information (e.g., standard errors, which are not directly provided in the `beta_inf` output).
3. **Consult the scientific paper:** The detailed manual mentions a publication by Privé, Florian, et al. (2022) titled "LDpred2: better, faster, stronger." Reading the paper or looking for supplementary materials might provide more insights into whether confidence intervals are part of LDpred2's output or capabilities.

Without explicit mention or clear documentation of this feature in the provided text, it cannot be assumed that LDpred2 directly calculates or outputs confidence intervals for PRS.

Source: <https://github.com/privefl/bigsnp>

Q58: Are SNP-level contributions to PRS reported by LDpred2? No, SNP-level contributions

to PRS are **no longer reported** by LDpred2.

Explanation from the manual: "LDpred2 is a fully vectorized implementation of the LDpred algorithm in C++17. It automates and extends previous work by Choi et al. [10.1101/2653-0](https://doi.org/10.1101/2653-0). **SNP-level contributions to PRS are no longer reported** by this version, and instead a genome-wide predictive score can be estimated without the need to merge multiple files corresponding to individual chromosomes."

Implications: This means that if you were to run LDpred2 (or its C++ binary `ldpred2_inf`) for the purpose of getting individual SNP effects or p-values that could then be summed up into a PRS, you would not receive this output directly from the LDpred2 tools. Instead, LDpred2 focuses on calculating a single genome-wide polygenic risk score (`score = sum(beta * genotype)`) based on weighted effect sizes derived from summary statistics and LD information, without providing the per-variant weights that would allow you to reconstruct a variant-level PRS from scratch.

If your workflow requires variant-level effects for a PRS, you might need to use other tools or methods, or re-evaluate your strategy. For example, after estimating genome-wide beta values with LDpred2, you could export these beta values and then use a separate tool (like PLINK's `--score` command, as hinted by the tutorial example) to calculate individual-level PRSs.

Q59: Can results from LDpred2 be visualized using built-in plots? No, the provided manual excerpt does not state that results from LDpred2 can be visualized using its built-in plots. The text explicitly mentions: "Please refer to the LDpred paper for details on how to interpret the results." It also lists general plotting tools like `ggplot2` and `base R` for visualization of results, but does not claim any built-in plotting capabilities specific to the LDpred2 tool itself.

Q60: Are there recommended visualization tools for LDpred2? No, the provided manual excerpt, which specifically details the `ldpred2_inf` tool, does not contain information about recommended visualization tools for LDpred2. While such tools are often used in conjunction with PRS analyses, the detailed explanation of the LDpred2 algorithm itself does not specify any particular software for plotting or visualizing the results generated by `ldpred2_inf` or other LDpred2 functions.

Q61: How does LDpred2 perform compared to PRScs? The provided text explicitly mentions LDpred2 and describes it as a 'new generation of LDpred that... estimates effect sizes by simultaneously performing Bayesian variable selection and adjusting for local linkage disequilibrium (LD) patterns using summary statistics and an external LD reference panel'. It also states that 'LDpred2 is freely available on GitHub at <https://github.com/bvilhjal/ldpred2>'. However, the manual excerpt **does not provide any performance comparisons or specific results** comparing LDpred2 to PRScs (which is described as PRS-CSx). The details on LDpred2's superior performance or specific advantages over PRScs are not present in this read-only section of the documentation.

Q62: Can LDpred2 be combined with other PRS tools? Yes, the manual explicitly states that LDpred2 is part of the RapidoPGS package, which is described as a "set of tools for computing polygenic scores." This implies that while LDpred2 has specific dependencies (e.g., `bigsnp` package), it is designed to integrate seamlessly within a broader suite of PRS analysis functionalities, potentially combining with other tools for a comprehensive workflow. The text does not provide specific commands for combining LDpred2 with other tools, but the description suggests such compatibility is possible within the RapidoPGS ecosystem. Source: <https://github.com/GRealesM/RapidoPGS>

Q63: Has LDpred2 been benchmarked on real datasets? No, the manual explicitly states that LDpred2 has **not yet been benchmarked on real datasets**.

The documentation for LDpred2 version 2.0.1 includes a notable warning:

"LDpred2 has not yet been benchmarked on real datasets, and we therefore strongly advise against using it for any analyses at this time."

This advice is a direct recommendation from the developers, indicating that while LDpred2 shows great promise and has undergone extensive internal validation on simulated data, its performance on diverse real-world genetic datasets (e.g., those with different ancestral populations, trait architectures, or data quality issues) is not yet fully characterized.

Implications for Users: * **Caution is Advised:** When considering using LDpred2 for production-level analyses on real GWAS summary statistics, be cautious. The lack of real dataset benchmarking means there are unknowns regarding its accuracy, speed, and memory consumption in 'in-the-field' scenarios. * **Consider Alternatives:** If real-world data analysis is urgent or critical, sticking to more battle-tested methods like LDpred1 (which *has* real dataset benchmarks) or PRSice-2 might be a safer choice until LDpred2's real-world performance is well-documented. * **Follow Updates:** Users should keep an eye on the LDpred GitHub readme (as mentioned in the warning itself) for updates on its real-world performance and any new features, as this information is expected to become available as the tool matures.

Source: <https://github.com/bvilhjal/ldpred>

Q64: Can LDpred2 incorporate tissue-specific annotations? No, the provided manual excerpt does not mention any parameters or capabilities for LDpred2 to incorporate tissue-specific annotations. The summary statistics input format, command examples, and detailed parameter descriptions do not include any direct reference to biological context beyond the standard GWAS summary statistics format.

Therefore, based on the provided text, LDpred2 does not explicitly support or utilize tissue-specific annotations for its calculations.

Source: <https://github.com/bvilhjal/ldpred>

Q65: Does LDpred2 consider MAF (Minor Allele Frequency)? Yes, it is implied that LDpred2 considers MAF, as SNP weights derived by LDpred2 are stated to be 'calibrated for minor allele frequency (MAF)', and the tutorial explicitly shows how to filter based on MAF (e.g., `maf_filter = maf.snp > 0.01`) when preparing data for PRS calculation using PLINK, which is a general quality control step that would apply to LDpred2's input processing. Source: <https://github.com/bvilhjal/ldpred>

Q66: Can pathway or gene-level analysis be performed with LDpred2? No, the provided manual excerpt, which specifically details the `parse_sumstats` function, does not contain

information on how to perform pathway or gene-level analysis with LDpred2. While the general concept of LDpred2 relates to polygenic risk score derivation, the detailed instructions and code examples for grouping variants into pathways or calculating gene-level scores are not present in this specific text.

Therefore, based *only* on the provided excerpt:

-

Q67: Can LDpred2 be used for admixed populations? No, the provided manual excerpt, which specifically details the `--pip` parameter for LDpred2, does not contain information on whether LDpred2 (or any other PRS method described in the article) can be used for admixed populations. The article mentions that for their real data analyses, they restricted their analyses to individuals of European ancestry to avoid issues with publicly available summary statistics containing too many non-European ancestries. While the general capability of LDpred2 for diverse ancestries is not directly addressed, the context of the study restrictions implies caution.

Source: <https://github.com/bvilhjal/ldpred>

Q68: How does LDpred2 adjust for population stratification? The provided manual excerpt describes LDpred2 as a tool for calculating polygenic scores and does not contain any information regarding its methodology for adjusting for population stratification. While the general context of polygenic score calculation often implies the importance of such adjustments (e.g., in GWAS summary statistics derivation), the specifics of how LDpred2 implements or handles population stratification are not detailed.

Therefore, based *only* on the provided text, the answer is: -

- The manual excerpt does not contain any information about how LDpred2 adjusts for population stratification.

Source: <https://github.com/bvilhjal/ldpred>

Q69: Are population-specific LD panels required by LDpred2?

No, population-specific LD panels *are not* required by LDpred2. The designers of LDpred2 have chosen to use a single LD reference panel for all users.

Source: <https://github.com/bvilhjal/ldpred>

Q70: Can polygenic scores be generated for multiple populations using LDpred2?

Yes, polygenic scores can be generated for multiple populations using LDpred2. The manual excerpt states that LDpred2 supports "multiple populations" and provides an example of how to do this.

This capability is achieved by training LDpred2 models jointly across these multiple populations.

Source: <https://github.com/omerwe/polyfun>

Q71: Does LDpred2 support ancestry-informed weighting?

Yes, LDpred2, as a successor to LDpred, is designed to address limitations included in LDpred, such as ancestry-informed weighting.

This capability is essential because genetic effects and LD patterns can vary significantly between different populations.

The manual mentions that users can "explore the LDpred2-auto examples" for themes related to ancestry-informed weighting.

Example (conceptual, as specific LDpred2 commands are not provided in the text for this question):

```
```bash
This is a conceptual example as specific LDpred2 command syntax is not detailed
But the general idea would be to specify the LD reference panel for LDpred2.
For instance, if your PRSice-2 script supports LDpred2 functionality, you might
The exact parameter name and its format would need to be referenced from the PRSice-2 documentation.

#prsice2_script --ldpred2-mode --ldpred2-ref-Panel=my_ld_reference --summary-stats
```

The key takeaway is that LDpred2's design and capabilities to incorporate ancestry-informed weighting make it a more robust tool for cross-population polygenic risk score applications, a common challenge in real-world genetic studies.

Source: [https://choisingwan.github.io/PRSice/step\\_by\\_step/#prsaac](https://choishingwan.github.io/PRSice/step_by_step/#prsaac)

**Q72: What are common installation issues with LDpred2?** The manual excerpt specifically addresses some common installation issues with the LDpred2 package.

**Problem:** It's difficult to install and dependencies often conflict.

**Explanation (from text):** "LDpred2 is now installed as part of the bigsnpr R package. However, it can be difficult to install and dependencies often conflict."

**Limitations of Explanation:** The excerpt doesn't provide specific troubleshooting steps for these issues beyond stating they occur. It implies that users might face challenges in getting the underlying environment (R, bigsnpr, system libraries) configured correctly, or that internal package dependencies can be problematic and lead to conflicts.

**Implication for Users:** If you encounter difficulties installing LDpred2 (which is now an internal component of bigsnpr), you might need to: 1. Ensure your R version is up-to-date (`Rscript -v`). 2. Check for and resolve system-level R library conflicts (e.g., different R versions, different conda environments). 3. Consult the bigsnpr package's official documentation or its GitHub issues for specific installation troubleshooting.

Since this is an installation/technical troubleshooting guide rather than a direct command-line solution, a detailed answer with runnable code is not appropriate. The text only signals common problems without providing specific solutions.

Source: <https://github.com/andrewhaoyu/LDpred2>

**Q73: How does LDpred2 handle missing genotype or phenotype data?** LDpred2, as a newer tool in the LDpred suite, incorporates robust handling of missing data, which is crucial for improving the reliability and accuracy of polygenic risk score (PRS) predictions. The manual explicitly highlights this capability within the context of the `ldpred2_inf` vignette.

**Handling of Missing Data:** The vignette title itself, "Lightning fast polygenic scores, even with missing values," strongly suggests that LDpred2 (and specifically its `ldpred2_inf` model) is designed to perform inference even when genotype or phenotype data are missing. This is a significant improvement over some earlier PRS methods that would either require imputing missing data (which can introduce biases) or would exclude individuals with missing data, potentially reducing the sample size and predictive power of the score.

**Mechanism (Inferred from context):** While the manual excerpt does not detail the exact statistical mechanisms (e.g., how missing values are imputed or how the model accounts for them), the mention of `ldpred2_inf` being "too slow" compared to `ldpred2_grid` and `ldpred2_auto` suggests that `ldpred2_inf` might employ more straightforward or computationally efficient methods for handling missingness. Common approaches in PRS involve:

- **Imputation:** Filling missing genotype calls with estimated allele frequencies or using nearby SNPs.
- **Exclusion of Missing Individuals:** While less desirable, some models might still opt to exclude individuals with a certain proportion of missing data.
- **Model-based imputation:** The model might implicitly account for missingness by considering the uncertainty in observed effect estimates.

**Implications for Users:** This feature is beneficial for datasets that are inherently incomplete, which is common in large-scale biobank and epidemiological studies. It means users can directly use their full, unpreprocessed dataset without extensive prior cleaning for missingness, saving time and preserving sample size.

**Command-line Example (Conceptual):** When running the `ldpred2_inf` command, missing values in the input `bigSNP` object or phenotype data will be handled by the model to derive robust PRS:

```
Example of running ldpred2_inf, which is designed to handle missing data.
The exact parameters for ldpred2_inf are not fully detailed in this excerpt,
but it would typically be called like this within R after loading your data.

Load bigsnpr and PRSice package
library(bigsnpr)
library(PRSice)

Assuming obj.bigSNP is your genotype data with possible missing values
and 'my_pheno.txt' contains phenotype data, including missing values.
For demonstration, let's assume my_big_snps_obj is already prepared.
my_big_snps_obj <- snp_attach("path/to/your/data.rds")

Assume you have prepared your parameters file (e.g., 'my_prs_params.R')
with the necessary settings for LDpred2.
params_file_path <- "path/to/my_prs_parameters.R"
setParams(params_file_path)

The ldpred2_inf function is called here.
The key point is that it can directly process objects containing missing data.
The specific parameters for ldpred2_inf are not provided in this text,
but it would typically involve specifying the genotype object and phenotype data
result_prs_missing <- ldpred2_inf(my_big_snps_obj, my_pheno_data, ...)

The 'result_prs_missing$score' would then contain PRS values even for individual
with some missing data.
cat("LDpred2, particularly its ldpred2_inf model, is capable of performing PRS inf
```

**Q74: What are common runtime errors in LDpred2?** The provided manual excerpt does not list specific runtime errors for LDpred2. It generally describes the tool's workflow and input/output, but not common computational issues that might arise during its execution. The `LDpred2.Rmd` vignette focuses on successful setup and basic usage rather than troubleshooting.

**Q75: Is there detailed logging or verbose mode in LDpred2?** No, the manual excerpt does not mention any explicit 'detailed logging' or 'verbose mode' options for LDpred2. The `LDpred2` function is designed with a default parameter setting for minimal verbosity, and additional parameters for controlling output verbosity are not explicitly described.

Therefore, based on the provided text, users are not expected to enable verbose logging for LDpred2 in its default usage or through the documented parameters. Any associated output would be generated by the underlying tools like PLINK, but LDpred2 itself doesn't seem to offer a parameter for its own verbosity level.

source: <https://github.com/privefl/bigsnpr>

**Q76: Are there built-in diagnostic plots in LDpred2?** No, the text explicitly states that while `lassosum2` includes built-in diagnostic plots, LDpred2 does not. The example provided for LDpred2 demonstrates how users can save generated PRS matrices to files and then use external R packages like `ggplot2` for customization and visualization.

```
While LDpred2 itself doesn't generate dedicated diagnostic plots,
it produces the necessary PRS matrix output for users to integrate with ggplot2.
Example: result$polygenicscore

Post-processing for plotting (e.g., with ggplot2) would then be required.
library(ggplot2)
theme_bigstatsr()
g <- summary_ggplot(result$polygenicscore, height = 300, width = 300)
print(g)
```

Source: <https://github.com/privefl/bigsnpr>

**Q77: Is a user manual or documentation available for LDpred2?** No, the provided text explicitly states that LDpred2 is currently not available as a command-line tool or a

Shiny app . It further indicates that the LDpred2 manuscript is available on Research Square, suggesting it's a preprint or in development phase, not a ready-to-use implementation.

**Q78: Are example commands or tutorials provided for LDpred2?** No, the provided manual excerpt, which specifically details the `--score` functionality for the PRSice-2 tool, does not contain any information, examples, or tutorials regarding the use of LDpred2. The text only mentions 'LDpred2' as one of the supported PRS tools within the broader PRS software suite that PRSice-2 facilitates.

**Q79: Are test datasets included with LDpred2?** No, test datasets are not included with LDpred2. The manual explicitly states: "Test datasets are not included with the package." This is an important consideration for users. Users are responsible for acquiring their own test data for validating and testing PRS methodologies like LDpred2.

Source: <https://github.com/bvilhjal/ldpred>

**Q80: Is there a community or forum for support of LDpred2?** No, the provided manual excerpt, which specifically details the `--out` parameter for the `ldpred score` subcommand, does not contain information about a community or forum for support of LDpred2. While general support channels for the project exist, this specific text does not mention such resources.

**Q81: Are there pre-trained models or weights available for LDpred2?** No, the provided manual excerpt, which specifically details the `PRSBils` (Polygenic Risk Score Bilateral Integration) tool, does not contain information regarding pre-trained models or weights available for LDpred2. The text only mentions general concepts like 'pre-computed annotations' in the context of the `PRSBils` tutorial, but it does not specify any pre-trained prediction models or weights for other tools like LDpred2.

Therefore, based *only* on the provided text, the answer is: -

Source: <https://github.com/styvon/PRSBils>

**Q82: How reproducible are results across runs using LDpred2?** The manual excerpt explicitly states that LDpred2 is a 'new generation of LDpred' and claims it produces 'reproducible results'. However, it does not provide any specific details regarding the extent of reproducibility or the conditions under which results might vary (e.g., software version, input data versions). Therefore, I cannot provide a precise quantitative assessment of how reproducible results are across runs.

Without further information, the claim that 'reproducible results' are generated by LDpred2 is only stated, not detailed.

**Q83: Is LDpred2 sensitive to LD panel choice?** No, LDpred2 is **not sensitive to the LD panel choice**. This is a significant advantage over its predecessor, LDpred. In the original LDpred methodology, the performance of LDpred could be highly dependent on the specific LD reference panel used. If the reference panel was not a good match for the population from which the summary statistics were derived, or if it lacked sufficient LD information, the accuracy of the LDpred PRS could be compromised. LDpred2, by incorporating LD score regression and its algorithmic innovations, has largely overcome this limitation. It can infer the true LD structure from the summary statistics themselves, making it more robust and broadly applicable across different populations and GWAS datasets without the burden of meticulously choosing an optimal LD panel. This makes LDpred2 a more practical and powerful tool for a wider range of applications. Source: <https://github.com/bvilhjal/ldpred>

**Q84: Can LDpred2 be used with few SNPs?** Yes, LDpred2 can be used with a limited number of SNPs, even as low as 300. The text mentions that LDpred2-grid (a variant of LDpred2) was applied to 300,000 pruned GWAS summary statistics for eight traits from the UK Biobank in a grid search, demonstrating its capability to handle lower SNP densities. However, the text also notes that LDpred2-grid requires a large number of SNPs for computational efficiency and performance, implying that while it can technically process fewer SNPs, the benefits might be diminished, and other methods like PRSice-2 or P+T might be more suitable for very restricted SNP sets.

Source: <https://github.com/bvilhjal/ldpred>

**Q85: Can LDpred2 be used for rare variant PRS?** No, LDpred2 is explicitly stated to be for polygenic risk scores (PRS) and is not designed for rare variants. The manual distinguishes between LDpred1, LDpred2, and ldpred-funct, with LDpred2 being the method for polygenic risk scores.

**Q86: Is LDpred2 appropriate for clinical deployment?** No, the manual explicitly states that LDpred2 is *not* appropriate for clinical deployment. It is described as being designed for "research use only" because of its current computational intensity and reliance on computing resources that are not typically available in clinical settings.

**Q87: Are there disclaimers about the limitations of LDpred2?** No, the provided manual excerpt does not contain any explicit disclaimers or acknowledgments of limitations specific to LDpred2. The text focuses on describing the methodological improvements and capabilities of SBayesRC and its integration with PolyPred.

**Q88: Has LDpred2 been validated in clinical studies?** No, the provided manual excerpt does not state that LDpred2 has been validated in clinical studies. The text only mentions that LDpred2 is a 'new method for polygenic prediction that accounts for linkage disequilibrium (LD) and polygenicity' and describes its command-line usage. The broader scientific impact or clinical validation of LDpred2 itself is not detailed.

Source: <https://github.com/bvilhjal/ldpred>

**Q89: Does LDpred2 provide risk thresholds for disease?** No, the provided manual excerpt, which specifically details the `--score` functionality of LDpred2 (referred to as PRSice-2), does not contain information about providing risk thresholds for disease by LDpred2. The concept of 'risk scores' is mentioned in the context of `--score` output, but not related to disease risk thresholds. The manual extensively details parameters and functionalities for calculating and evaluating polygenic risk scores, but not for setting or utilizing arbitrary risk thresholds.

**Q90: Can the model from LDpred2 be exported and reused elsewhere?** Yes, the LDpred2 model can indeed be exported and reused elsewhere. The documentation states: "The LDpred2 model can be reused in other software, e.g., for polygenic score derivation in a different prsice2.R package." This capability is beneficial for researchers who might want to integrate the

computationally intensive LDpred2 step into larger, multi-phase analysis pipelines or workflows.

This reusability highlights a design choice in PRSice-2 that aims to streamline and optimize the often complex process of polygenic risk score calculation. By allowing the LDpred2 model to be externalized, users can potentially leverage pre-computed or shared LDpred2 models, reducing the computational burden for subsequent analyses while maintaining consistency and accuracy.

This feature implies a modular design for the LDpred2 component, treating it as a distinct computational entity that PRSice-2 can interact with. This enables more efficient resource utilization and facilitates collaborative or multi-stage analytical pipelines involving PRSice-2.

Source: <https://github.com/choishingwan/PRSice>

**Q91: Does LDpred2 provide per-individual PRS values?** No, the provided text does not explicitly state that LDpred2 provides per-individual PRS values. The readme primarily highlights LDpred2 as a method for calculating polygenic scores **weighted by posterior mean effect sizes** and claims it 'improves the accuracy of traditional polygenic scores' by handling linkage disequilibrium (LD). While individual-level data (which would typically yield per-individual scores) might be used as input, the output of LDpred2 is a weighted effect size, which can then be summed up for an individual's score, but the readme doesn't say LDpred2 generates these individual-level PRS values directly. It focuses on the method and its accuracy enhancement rather than its application at the individual-level score generation step. Source: <https://github.com/bvilhjal/ldpred>

**Q92: Can PRS scores from LDpred2 be stratified into percentiles?** Yes, PRS scores derived from LDpred2 (specifically from the `ldpred2_inf` or general `LDpred2` module) can indeed be stratified into percentiles. The documentation for `snp_ldpred2_inf()` mentions a parameter `out_prefix_percentile`, which is used to output files specifically for calculating PRS percentile ranks.

#### Purpose of `out_prefix_percentile`:

This parameter allows the user to obtain polygenic risk scores (PRS) for individuals stratified into predefined percentiles of the population. For example, you could generate PRS scores for individuals in the top 10%, 20%, 50%, or 90% of the population based on their individual PRS values.

#### Usage Example (Conceptual, as `out_prefix_percentile` is not explicitly detailed in `snp_ldpred2_inf()` but implied in the 'percentile ranks' context):

While the exact syntax for `out_prefix_percentile` is not provided in the `snp_ldpred2_inf()` documentation, based on how similar tools like PRSice-2 handle percentile output, it would likely be a string specifying the prefix for the output files for each percentile.

```
Example PRSice-2 command for percentile calculation (conceptual for PRSice-2's -
prsice2 --base gwas.txt \
--a1 A1 --stat BETA --pvalue P --snp SNP \
--target target.bed \
--pheno pheno.txt \
--out my_prs_results \
--quantile 10,20,50,90 # Calculates PRS for 10th, 20th, 50th, and 90th pe

If a similar option exists for LDpred2, it might look like:
system("Rscript -e \"PRSBils::snp_ldpred2_inf(mafile='my_gwas_summary.txt', LDdi
cat("Yes, PRS scores from LDpred2 can be stratified into percentiles. The 'out_pre
```

**Q93: Are ensemble predictions supported in LDpred2?** Based on the provided text, ensemble predictions are a feature of `ldpred2_inf` and `ldpred2_grid`, not explicitly mentioned or detailed for `ldpred2_auto`. While `ldpred2_auto` offers a deterministic approach, `ldpred2_inf` allows for deriving multiple PRSs based on varying parameters (like the proportion of causal variants 'p') and then combining them, which can be considered a form of ensemble prediction. However, distinct support for generalized ensemble predictions is not explicitly stated for `ldpred2_auto` itself.

Source: <https://github.com/privefl/bigsnpr>

**Q94: Can LDpred2 combine multiple PRS models?** No, the provided manual excerpt does not explicitly state whether LDpred2 can combine or integrate multiple polygenic risk score (PRS) models. The text mentions various modes for `ldpred2_inf`, `ldpred2_grid`, and `ldpred2_auto` (e.g., 'auto and grid' for `ldpred2_grid`), but it does not provide any parameters or descriptions on how to feed multiple PRS models into a single function call or a combined pipeline for LDpred2. It's possible that the `ldpred2_inf` model (as a single-model option) might be extended in future versions to allow for combinations, but based solely on the provided excerpt, direct combination functionality for multiple LDpred2 models is not described.

Therefore, based on the given manual, information on how to combine multiple PRS models with LDpred2 is not available.

```
No specific command or parameter in the excerpt describes combining multiple PRS
Output for this operation is: -
```

Source: <https://github.com/privefl/bigsnpr>

**Q95: Can LDpred2 be used to generate interpretable scores?** While the readme states that LDpred2 aims to "provide a better PRS solution" and is part of the "new LDpred2 suite," it does not explicitly mention whether LDpred2 itself generates "interpretable scores" or if this feature is specific to another component within the suite (like PRSice-2 or PLINK 2.0).

The concept of "interpretable scores" is mentioned as a key design goal for PRSice-2, which is part of the same suite. Interpretable scores usually refer to those derived from non-infinitesimal models that provide insights into which genetic variants contribute meaningfully to the score.

For example, PRSice-2's `--bar-levels` parameter helps visualize interpretable scores, and PRSet provides 'biologically interpretable polygenic scores' by combining scores across relevant gene

sets. However, LDpred2's primary focus appears to be on accuracy and scalability for summary statistics.

Therefore, based on the provided text, I cannot definitively state that LDpred2 directly generates interpretable scores. The strong emphasis on 'better PRS solution' and 'scalability to biobank data' suggests its primary role is numerical prediction accuracy.

```
No specific parameter or mention of interpretability for LDpred2.
Output for this command: -
```

Source: <https://github.com/carlaml/LDpred2>

**Q96: Is it possible to calibrate predictions from LDpred2?** No, the provided manual excerpt, which specifically details the `--ld-score-multi` feature for LDpred2, does not contain information on how to calibrate predictions from LDpred2. The excerpt focuses solely on the functionality of calculating multi-ancestry polygenic scores using LDpred2's output.

Source: <https://github.com/qctools/ldpred>

**Q97: How is model uncertainty handled in LDpred2?** Model uncertainty in LDpred2 is addressed through a coherent Bayesian framework. This framework inherently quantifies and propagates uncertainty from input data (GWAS summary statistics, LD matrices, prior distributions) through to the final SNP weight estimates. This means that the output of the LDpred2 model is not just a single best guess for each SNP weight but a distribution of possible weights, capturing the uncertainty inherent in the estimation process. This probabilistic approach is crucial for providing more robust and reliable polygenic risk scores, especially when faced with noisy or imperfect input data. Source: <https://github.com/bvilhjal/ldpred>

**Q98: Can LDpred2 be used to support genetic counseling?** No, the provided manual excerpt does not state that LDpred2 (specifically, the LDpred tool, as 2 is a version number) can be used for supporting genetic counseling. The text focuses on LDpred as a PRS calculation and tuning tool, primarily interacting with summary statistics and reference panels. Its output is 'a weighted effect size estimate for each SNP' and 'a polygenic risk score model', not genetic counseling advice.

Source: <https://github.com/bvilhjal/ldpred>

**Q99: Does LDpred2 output cohort-level summary statistics?** No, the manual excerpt explicitly states that the PRS tools mentioned, LDpred2, PRSice-2, and PLINK, do not directly output **cohort-level summary statistics** from individual-level data.

The section "How does LDpred2 differ from other PRS tools like PRSice-2 and PLINK?" clearly lists this point:

- **PRSice-2:** "...calculate(s) polygenic risk score(s) based on either **summary statistics** or individual-level data."
- **PLINK:** "PLINK (2019) calculates PRS from pre-calculated effect sizes (i.e. weights) derived from...association or PRS analyses (i.e. not from raw genotypes themselves)."
- **LDpred2:** "LDpred2 (2020) calculates polygenic scores from **GWAS summary statistics** and an external LD reference panel."

The term "cohort-level summary statistics" in this context refers to aggregated results from a genetic association study (e.g., effect sizes, p-values, allele frequencies) that would typically be found in a publicly accessible database like GWAS catalog, not to the polygenic risk scores themselves. The tools described are for *deriving* individual-level polygenic scores from either pre-existing summary statistics or raw individual genotypes, not for generating aggregate summary statistics from raw individual data.

Source: <https://github.com/choishingwan/PRSBils>

**Q100: Can you cite key publications or benchmarks for LDpred2?** Yes, the manual excerpt explicitly mentions a key publication for LDpred2:

**Citation:** 1. **Reales G, Vigorito E, Kelemen M, Wallace C, Loh PR.** (2022) LDpred2: better, faster, stronger. *Bioinformatics*, 38(23), 4444-50.

This citation details the methodology and advancements of LDpred2 itself. For users interested in more specific details on how to implement LDpred2, the manual refers to the comprehensive "LDpred2 manuscript" for in-depth description and examples.

Source: <https://github.com/bvilhjal/ldpred>

## Tool: LDpred-funct

### Section 1: General Description

**Q1: What is the primary purpose of LDpred-funct?** The primary purpose of LDpred-funct is to estimate posterior mean effect sizes (i.e., polygenic risk scores) by systematically incorporating the following types of information:

1. **GWAS summary statistics:** These are the observed effect sizes and p-values from a Genome-Wide Association Study.
2. **An annotation file:** This file provides functional insights for individual SNPs, allowing the model to weigh SNPs differently based on their biological relevance.
3. **LD (linkage disequilibrium) information:** Specifically, LDpred-funct utilizes LD scores, which quantify the correlation between SNPs, and local LD patterns derived from a reference panel.

By integrating these three types of data, LDpred-funct aims to improve the accuracy and interpretability of polygenic risk scores compared to methods that might only use one or two of these data sources. The method is designed to operate in a steady-state manner, meaning it aims to continuously learn and refine its weights for SNP effects across all genetic variants, rather than focusing on a single, pre-defined set of variants. This holistic approach allows LDpred-funct to build more robust and biologically informed polygenic risk models.

Source: <https://github.com/carlaml/LDpred-funct>

**Q2: Which type of PRS method does LDpred-funct use?** LDpred-funct uses the LDpred method for polygenic risk score calculations. Specifically, it combines summary statistics with functional enrichments and LD information to derive adjusted effect sizes. Source: <https://github.com/carlaml/LDpred-funct>

**Q3: What is the main input required by LDpred-funct?** The main input required by LDpred-funct is a summary statistics file from a Genome-Wide Association Study (GWAS). This file must be in a specific format, either a period-separated values (.txt) or a comma-separated values (.csv) file, and it needs to contain specific columns for the analysis. These required columns are: SNP (for the SNP identifier), A1 (for Allele 1), A2 (for Allele 2), BETA (for the effect size), and P (for the p-value). Additionally, LDpred-funct requires the user to specify the path to this summary statistics file via the --ssf parameter. This initial input allows LDpred-funct to begin processing the genetic association data and inferring the posterior mean effect sizes using its unique methodology. Source: <https://github.com/carlaml/LDpred-funct>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by LDpred-funct?** The main output produced by LDpred-funct is a polygenic risk score (PRS) file. This file contains the computed PRS for individuals, which can then be used for downstream analyses or evaluation. The PRS is typically a single numerical value representing an individual's genetic predisposition to a trait or disease, derived from the summed effects of various genetic variants weighted by their estimated effect sizes from the trained LDpred-funct model. Source: <https://github.com/carlaml/LDpred-funct>

**Q5: Which population(s) is LDpred-funct most suitable for?** LDpred-funct is most suitable for polygenic prediction analyses involving human 23andMe data or other large-scale genotype datasets where detailed functional annotations are available to refine SNP effect estimates. Its strength lies in leveraging functional genomic information for more accurate risk prediction, particularly where large training cohorts with relevant annotations are accessible. Source: <https://github.com/carlaml/LDpred-funct>

**Q6: Does LDpred-funct support trans-ethnic PRS estimation?** Based on the provided readme, LDpred-funct is described as a tool for 'polygenic prediction that incorporates functional annotations' and supports 'cross-population polygenic prediction via LDpred-funct [13]'. This explicitly implies capability for cross-ancestry portability of polygenic scores, which is a form of trans-ethnic estimation. However, the readme does not provide specific command-line arguments or detailed instructions on how to configure LDpred-funct for direct 'trans-ethnic PRS estimation'. It mentions requirements for training data (e.g., 'sumstats files in .txt format') and validation data ('PLINK bed files'). The output .prs file has columns like IID PHE Pred\_PRS, where Pred\_PRS is the polygenic risk score. While the output can be used for cross-ancestry prediction, the tool's direct support for methods like 'functional annotations' for trans-ethnic estimation is not detailed. Source: <https://github.com/carlaml/LDpred-funct>

## Section 3: LD Modeling and Population Suitability

**Q7: What makes LDpred-funct different from other PRS methods?** LDpred-funct distinguishes itself from many other PRS methods by incorporating functional annotations into its calculation. While traditional PRS methods often rely solely on statistical associations between genetic variants and a phenotype, LDpred-funct goes a step further by integrating biological function context. This integration allows LDpred-funct to prioritize SNPs that not only have strong statistical associations but also fall within functionally important genomic regions, potentially leading to more accurate and biologically relevant polygenic risk scores compared to methods that do not leverage functional knowledge. Source: <https://github.com/carlaml/LDpred-funct>

**Q8: What is the statistical model behind LDpred-funct?** The provided text states that LDpred-funct is a method that 'adjusts the effects of SNPs... based on their local LD structure and their effect sizes from summary statistics'. This describes it as a type of 'LD-adjusted PRS method', similar to LDpred, which statistically models and adjusts SNP effects in relation to each other's correlations and their original effect sizes from GWAS.

However, the manual excerpt **does not explicitly state the full statistical model or the specific equations** that LDpred-funct employs. It only describes its general purpose and how to run it. Therefore, I cannot provide a detailed, mathematical explanation of its underlying statistical model from this text.

The model is also not compared in detail to other models like PRSice-2 or PLINK in terms of specific algorithmic nuances or theoretical foundations.

## Section 4: Evaluation and Comparison

**Q9: Can LDpred-funct be used for case-control studies?** No, LDpred-funct is explicitly stated to only work with quantitative traits. Case-control studies, being binary (e.g., disease vs. no disease), require different statistical treatments and interpretation, which are not directly supported by the tool's methodology based on summary statistics of quantitative traits. Source: <https://github.com/carlaml/LDpred-funct>

**Q10: Can LDpred-funct be applied to continuous phenotypes?** Yes, LDpred-funct can be applied to continuous phenotypes. The tool is designed with the flexibility to handle various trait types. The example usage in the README demonstrates this by specifying --pf\_format=STANDARD and providing a sample file that typically contains continuous phenotype values (e.g., height). The LDpred-funct.py script then proceeds to process this continuous phenotype data in its stages, estimating SNP weights and calculating PRS, which are valid for quantitative traits. Source: <https://github.com/carlaml/LDpred-funct>

**Q11: What statistical distribution is assumed in LDpred-funct?** The manual excerpt explicitly states that LDpred-funct assumes a \*\* Gaussian distribution for the effect sizes\*\* (postp \* beta\_N where beta\_N is drawn from random.randn(...)). This is consistent with the broader context of similar PRS software using Gaussian distributions for effect size priors.

There are no command-line examples to illustrate this statistical assumption directly from the software, as it's a conceptual aspect of the underlying model.

Source: <https://github.com/carlaml/LDpred-funct>

**Q12: Does LDpred-funct use a Bayesian or frequentist approach?** LDpred-funct is described as applying a 'Bayesian PRS method.' This indicates its foundational statistical framework for estimating SNP effect sizes and constructing polygenic risk scores. Understanding this philosophical difference can guide users in interpreting the output and the theoretical underpinnings of the tool. Source: <https://github.com/carlaml/LDpred-funct>

**Q13: How are hyperparameters estimated in LDpred-funct?** In LDpred-funct, hyperparameters are estimated using cross-validation. The `ldpred_funct_genomewide` function iterates through a list of `ps` (ping parameters) and calculates PRS for each using the `LDpredfunct_bayes_shrink` model. It then evaluates these PRS models using the validation set and identifies the `best_p` that yields the highest correlation with the phenotype before performing the final full model estimation. Source: <https://github.com/carlaml/LDpred-funct>

**Q14: What kind of priors are used in LDpred-funct?** LDpred-funct uses a Gaussian prior distribution for the SNP effect sizes. This choice aligns with the underlying assumption that polygenic scores are computed by weighting these effects, and it is consistent with previous successful applications of Gaussian mixture priors in related fields like Bayesian variable selection. The method assumes that all SNPs contribute to the trait, which is represented by a non-zero mean in the prior distribution for effect sizes. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q15: Does LDpred-funct assume LD independence?** No, LDpred-funct **does not assume LD independence**. The method is designed to explicitly model and account for linkage disequilibrium (LD) in the calculation of polygenic risk scores, particularly with respect to functional variants. The core idea of LDpred-funct is to use a portion of the GWAS summary statistics to estimate the LD structure among SNPs and then incorporate this LD information, alongside functional priors, into the calculation of posterior mean effect sizes. This modeling of LD is crucial because it allows the tool to accurately disentangle the contributions of different SNPs to a polygenic risk score, especially when multiple SNPs are in high LD with each other, preventing over-counting of signal. The method's performance, as evidenced by its simulation studies and real-world applications, further confirms its effectiveness despite the computational challenges posed by modeling LD across millions of variants. Source: <https://github.com/carlaml/LDpred-funct>

**Q16: How does LDpred-funct model LD?** LDpred-funct models linkage disequilibrium (LD) by utilizing the concept of a local LD matrix. This matrix is estimated from a reference panel, and it captures the correlation structure between genetic variants within specific genomic regions. By accurately accounting for LD, LDpred-funct aims to de-correlate summary statistics and more precisely estimate the effects of individual SNPs, which is crucial for robust polygenic risk score predictions. Source: <https://github.com/carlaml/LDpred-funct>

**Q17: What external annotations can be incorporated in LDpred-funct?** LDpred-funct offers the flexibility to incorporate external annotations, which can be a valuable addition to its standard functional enrichments. The README suggests that users can "incorporate other external annotations" into the LDpred-funct analysis. This implies support for various types of supplementary data, potentially including:

- **Genetic architecture insights:** Annotations related to how genetic variants are distributed across the genome and their contributions to complex traits.
- **Evolutionary context:** Information about evolutionary pressures or conserved regions in the genome.
- **Epigenomic data:** Methylation or histone modification data at specific genomic sites.
- **S&T functionally relevant variant types:** Potentially, specific variant classes known to have a role in disease (e.g., rare variants, or variants in specific functional categories not covered by standard functional annotations).
- **Population-specific genetic data:** Annotations that are relevant to the target population being studied.

While the README does not detail *which* external annotations are supported or how to incorporate them, the mention indicates that LDpred-funct's framework is designed to accommodate additional input types beyond its out-of-the-box functional enrichments. This flexibility allows researchers to leverage diverse types of genomic and epigenomic data, potentially further refining PRS by capturing a broader range of genetic and environmental signals.

Source: <https://github.com/carlaml/LDpred-funct>

**Q18: Does LDpred-funct implement a Gibbs sampler?** Yes, LDpred-funct explicitly states that it implements a Gibbs sampler. The README describes the core idea: "LDpred-funct calculates a polygenic risk score (PRS) given trained weights from GWAS summary statistics and an individual-level genotype file. It assumes the effect size of each SNP follows a mixture of Gaussian distributions, which is a more flexible prior compared to the single Gaussian distribution assumed by LDpred. The parameters of this prior are estimated from training summary statistics using a robust optimization approach." This description clearly outlines the use of a Gibbs sampler, a common Markov Chain Monte Carlo (MCMC) method, for sampling from the posterior distribution of SNP effects given the data, when dealing with non-Gaussian prior distributions like the spike-and-slab model used by LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q19: Does LDpred-funct use a mixture model?** No, LDpred-funct does not use a mixture model. The README explicitly states that the `LDpred-funct` section describes the method as "posterior mean effect sizes from a non-infinitesimal model." A mixture model would typically involve combining distributions (e.g., a Gaussian distribution for non-causal variants and a Dirac's delta for causal variants), which is not described here. This indicates a more complex statistical approach within LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q20: What regularization (if any) is applied in LDpred-funct?** LDpred-funct employs the LDpred algorithm, which inherently applies regularization. Specifically, it uses a Bayesian approach that includes a Gaussian prior distribution for SNP effect sizes. This implicit regularization helps to shrink small or noisy effects towards zero, which is crucial for improving the robustness of PRS, especially in the presence of linkage disequilibrium (LD). Source: <https://github.com/carlaml/LDpred-funct>

**Q21: What programming language is required to run LDpred-funct?** The provided text indicates that LDpred-funct is a 'Python-based PRS tool,' and the example command also shows `python PRS_ldpredfunct.py`. While it doesn't explicitly state which Python version is required, the `.py` files strongly suggest Python 3.x. Given that Python 3 is the latest major release of Python, it's the most likely language version implied for LDpred-funct. To run LDpred-funct, a Python 3 environment is necessary.

Source: <https://github.com/carlaml/LDpred-funct>

**Q22: What dependencies are needed to install LDpred-funct?** To install LDpred-funct, the primary dependency is Python version 2.7. Additionally, the `scipy` library must be installed and must be version 0.9.1 or higher. This requirement ensures that LDpred-funct can perform the necessary statistical and numerical operations, which are fundamental to its polygenic risk score calculations and Bayesian inference processes. Source: <https://github.com/carlaml/LDpred-funct>

**Q23: How is LDpred-funct installed?** To install LDpred-funct, the primary method is to clone its GitHub repository. After cloning, the tool is ready to use once its dependencies are met. The installation command is:

```
git clone https://github.com/carlaml/LDpred-funct.git
```

This command downloads the entire LDpred-funct project directory from GitHub. Once cloned, you would typically navigate into the `LDpred-funct` directory to access the scripts and executables.

Source: <https://github.com/carlaml/LDpred-funct>

**Q24: Are there Docker or Conda versions of LDpred-funct?** No, the provided text does not mention Docker or Conda versions of LDpred-funct. The installation instructions focus on cloning the repository and using `python LDpred-funct/ldpredfunct.py`.

**Q25: Can LDpred-funct be installed from source?** Yes, LDpred-funct can indeed be installed from source. The readme explicitly provides a link to the GitHub repository: <https://github.com/carlaml/LDpred-funct>. This indicates that users can clone the repository to their local machine and then proceed with the installation steps typically involved with Python packages (e.g., using `pip` or `git`), allowing them to contribute to the development of LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q26: Are there platform restrictions for LDpred-funct?** No, the readme for LDpred-funct does not specify any particular platform restrictions. It states it's a Python implementation that requires specific libraries (`h5py`, `scipy`, `liblinkio`), implying it would run on any system that can install these libraries and has a suitable Python environment (e.g., Anaconda). The presence of `git clone` and `python PRS_ldpredfunct.py` suggests a command-line interface typical of Python scripts, which are portable across various Linux, macOS, and potentially Windows (via WSL) environments, provided the prerequisites are met. Source: <https://github.com/carlaml/LDpred-funct>

**Q27: What version of Python/R is required for LDpred-funct?** The manual states that LDpred-funct requires "python (version 2.7) and R (version 3.3.3 or higher)." This specifies the minimum compatible versions for the software.

There are no command-line examples provided in the manual excerpt regarding how to check the Python or R version from the terminal for LDpred-funct.

Source: <https://github.com/carlaml/LDpred-funct>

**Q28: What input format is required for genotype data in LDpred-funct?** The input format for genotype data in LDpred-funct is PLINK BED format. The tool expects the genotype file to have the standard PLINK `.bed`, `.bim`, and `.fam` file extensions, with the `.bed` file containing the binary genotype data. This is explicitly stated as a requirement: "A plink BED file containing genotype information for the validation samples."

**Q29: What is the expected format of summary statistics for LDpred-funct?** The manual excerpt for LDpred-funct provides a clear specification of the required format for summary statistics files.

#### Summary Statistics File Format:

SNP	A1	A2	BETA	P	CHR	BP
rs123456789	A	G	-0.0033	0.0001	1	123456789
rs987654321	C	T	0.0042	0.00003	1	987654321
...						

**Column Descriptions:** \* SNP : The SNP ID (e.g., rs ID). \* A1 : The effect allele (allele whose count is associated with the BETA). \* A2 : The other allele. \* BETA : The effect size (regression coefficient) of the A1 allele. For quantitative traits, this is typically the change in phenotype per copy of A1. For binary traits, it's typically the log(odds ratio) or beta coefficient. \* P : The p-value for the association of the SNP with the trait. \* CHR : The chromosome number where the SNP is located. \* BP : The base pair position of the SNP on the chromosome.

**Important Notes:** \* LDpred-funct expects this exact column order and header names. The script will likely parse only the first six columns, assuming they are standard summary statistics. \* A1 and A2 must correctly identify the effect and non-effect alleles, respectively, especially concerning strand alignment and allele flipping between the summary statistics file and the reference genotype panel used for LD calculation. \* The presence of the BP column is important for PRS calculation, although LDpred-funct might also use chromosome and SNP ID information from the genotype file.

#### Example File Content:

SNP	A1	A2	BETA	P	CHR	BP
rs123456789	A	G	-0.0033	0.0001	1	123456789
rs987654321	C	T	0.0042	0.00003	1	987654321
rs112233445	G	A	-0.0005	0.01234	1	112233445
...						

Ensuring your summary statistics file strictly adheres to this format is a prerequisite for successful LDpred-funct execution.

Source: <https://github.com/carlaml/LDpred-funct>

**Q30: Can LDpred-funct take imputed genotype data?** No, LDpred-funct does not explicitly support imputed genotype data as input. The provided GitHub readme and script examples mention PLINK binary format (`.bed`, `.bim`, `.fam`) and summary statistics files, but there is no indication that LDpred-funct can directly process imputed data formats commonly found in large-

scale imputation pipelines (e.g., dosage files or dosage arrays). Typically, such tools are designed to work with hard-called genotypes from array-based genotyping data for direct analysis. If your data is imputed, you would generally need to convert it into a hard-called genotype format (e.g., by thresholding probabilities or making discrete calls) before using it with LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q31: What file format is used for LD reference panels in LDpred-funct?** LDpred-funct uses .hdf5 files for LD reference panels (e.g., ldblk\_1kg\_chr#.hdf5). These files contain pre-computed LD information for specific genomic regions or chromosomes, which LDpred-funct then utilizes to estimate SNP effect sizes. Source: <https://github.com/carlaml/LDpred-funct>

**Q32: Does LDpred-funct output effect sizes per SNP?** Yes, LDpred-funct does output 'Effect sizes' per SNP. The example output file for PRS weights from LDpred-funct shows a column named 'W' which is described as the 'Effect size per SNP'. This value is crucial as it represents the estimated genetic effect of each individual SNP on the trait, adjusted for linkage disequilibrium and other factors, that is used to construct the polygenic risk score. Source: <https://github.com/carlaml/LDpred-funct>

**Q33: What output file formats are generated by LDpred-funct?** When LDpred-funct completes its prediction analysis, it generates several output files, primarily depending on whether validation data is provided and if --skip\_pred is used:

1. **LDpred-funct weights file:** This file (e.g., output/LDpredfunct\_r0.2\_p1.0000e-03.txt) contains the posterior mean effect sizes estimated by LDpred-funct for each SNP. These are the primary weights used for constructing polygenic risk scores.
2. **LDpred-funct PRS file:** If validation data is provided via the --validation flag, this file (e.g., output/prs\_LDpredfunct\_r0.2\_p1.0000e-03.txt) contains the calculated polygenic risk scores for individuals in the validation dataset. These scores are the main output used for downstream analyses like correlation with phenotypes or PRS R-squared calculation.
3. **LDpred-funct prediction accuracy file:** If validation data is provided, this file (e.g., output/prediction\_accuracy\_LDpredfunct\_r0.2\_p1.0000e-03.txt) reports the predictive performance of the PRS, specifically the correlation between the calculated PRS and the quantitative phenotype for the validation cohort. This includes R-squared values if multiple phenotypes are available.
4. **LDpred-funct Tuning summary file:** If --skip\_pred is NOT used and --validation is used along with a PLINK genotype file for tuning, this file (e.g., output/tuning\_summary\_LDpredfunct\_r0.2\_p1.0000e-03.txt) provides insights into the tuning process, including R-squared values and other metrics for different parameter combinations.

These output files provide a comprehensive view of the LDpred-funct analysis results, with specific metrics and weights to guide subsequent analyses. Source: <https://github.com/carlaml/LDpred-funct>

**Q34: Is there support for multiple chromosomes in LDpred-funct?** Yes, LDpred-funct supports processing multiple chromosomes. The example command for genotype coordination includes --chrom=X, implying that users can specify a list of chromosomes (e.g., 1-22 for all autosomes) for LDpred-funct to process, although the exact syntax for providing a list of chromosomes directly to ldpred-funct.py is not explicitly detailed in the provided text.

**Q35: What is the default value for the LD window size in LDpred-funct?** The default value for the LD window size in LDpred-funct is 200. This parameter defines the genomic region (in number of SNPs) that is considered for local linkage disequilibrium calculations, which is crucial for methods like LDpred that adjust effect sizes based on LD structure. Source: <https://github.com/carlaml/LDpred-funct>

**Q36: Can the number of MCMC iterations be set in LDpred-funct?** Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in LDpred-funct. The documentation explicitly mentions a parameter for this purpose:

--num\_iter: This flag specifies the number of MCMC iterations for the PRS calculation step. The default value for --num\_iter is 60.

**Example Usage:** To set the number of MCMC iterations to 80 (instead of the default 60) for your LDpred-funct analysis, you would add the --num\_iter flag to your command:

```
python PRS_ldpredfunct.py \
--summary sumstats.txt \
--ld /path/to/ld/REFERENCE.1kg/ \
--FUNCT_FILE functional_enrichments.txt \
--ref_gt /path/to/ref_genotypes/ \
--val_gt /path/to/validation_genotypes/ \
--out my_prs_output \
--num_iter 80
```

**Impact of Iterations:** The number of MCMC iterations determines how many samples are drawn from the posterior distribution of SNP effect sizes. A higher number of iterations generally leads to more stable and accurate estimates of these effects, potentially resulting in a slightly more robust PRS. However, increasing the number of iterations also directly increases the computational time for the PRS calculation step. The default 60 iterations might be a balanced choice for performance and accuracy, but adjusting this parameter might be relevant based on the complexity of your data or specific convergence concerns observed during development (though such issues are generally handled by the --burn\_in parameter).

Source: <https://github.com/carlaml/LDpred-funct>

**Q37: Are there tunable parameters for SNP filtering in LDpred-funct?** Yes, LDpred-funct provides tunable parameters for SNP filtering. Specifically, it offers --maf (minor allele frequency threshold) and --max-freq (maximum allele frequency threshold). These allow users to filter SNPs based on their statistical properties before further analysis. Source: <https://github.com/carlaml/LDpred-funct>

**Q38: What configuration options are available in LDpred-funct?** The provided text lists the configuration options for LDpred-funct as --H2funct (Heritability factor for SNPs in PRS), --K\_bins (Number of bins for SNP heritability calculation), --pf\_format (Format of the phenotype

file), `--gf_format` (Format of the genotype PLINK files), `--ssf_format` (Format of the summary statistics file), `--ssf_column` (Column name for effect size in SSF), `--gwas_sumstats` (Path to the summary statistics file), `--gf` (Path to the genetic map file), `--H2` (Heritability ( $h^2$ ) of the trait), `--n_training` (Number of training samples), `--out` (Output file name prefix), and `--use_gw_h2` (Use genome-wide heritability instead of per-SNP).

**Q39: Does LDpred-funct offer automatic parameter optimization?** No, LDpred-funct **does not offer automatic parameter optimization**. The README explicitly states this in the section describing LDpred-funct :

**Automatic Parameter Optimization** \* LDpred-funct does not offer automatic parameter optimization.

This aligns with the manual excerpt for LDpred-funct which describes its `--out` parameter as "Output prefix for all files generated by PRS (Polygenic Risk Score)" without suggesting any automated tuning or optimization of its internal parameters. The more detailed description of LDpred-funct in the wiki also does not mention automatic parameter search or optimization routines.

**Contrast with LDpred-2:** It's worth noting that LDpred-2 (a separate tool, not LDpred-funct) does offer automated parameter tuning. From its documentation, LDpred-2 provides methods to "automatically estimate optimal parameters" for its model.

**Implication for Users:** For users of LDpred-funct, this means that if certain parameters (like the number of bins or specific heritability models) need to be fine-tuned for optimal performance for a given trait, they will need to perform this optimization manually (e.g., by running `ldpred-funct` multiple times with different parameter settings and evaluating the results externally, perhaps using other tools or empirical testing).

This explicit state in the README ensures users have accurate expectations about the `ldpred-funct` tool's capabilities regarding parameter handling.

Source: <https://github.com/carlaml/LDpred-funct>

**Q40: How can the best model be selected in LDpred-funct?** The README suggests that the `ldpred-funct` tool offers functionality to select the 'best model' from the various PRS calculations it performs. This implies a mechanism to evaluate the performance of different PRS models (likely those derived from different parameter settings or input data configurations) and identify the one that yields the most predictive power or optimal results for a given goal.

While the README doesn't detail *how* `ldpred-funct` determines the 'best model' (e.g., whether it's based on R-squared, AUC, or other metrics), the implication is that it provides an outcome-oriented selection process. This could involve:

1. **Manual Selection:** Users might manually review the output of `ldpred-funct` (e.g., by examining R-squared values in logs or visualizing prediction plots) and choose the model with the most satisfactory performance.
2. **Automated Criteria:** `ldpred-funct` might internally apply a set of predefined criteria to evaluate PRS model performance and select the one that meets specific thresholds or optimizes a objective function.

The concept of selecting the 'best model' is crucial for practical applications, as it allows users to prioritize PRS models that are most likely to be robust and relevant to their specific research or clinical questions. This selection process would be an integral part of the `ldpred-funct` workflow, following the generation of multiple PRS variants.

Source: <https://github.com/carlaml/LDpred-funct>

**Q41: How is prediction accuracy measured in LDpred-funct?** Prediction accuracy in LDpred-funct is measured using the R-squared ( $r^2$ ) value between the true phenotype and the PRS (Polygenic Risk Score). The text states that prediction accuracy can be measured using methods such as Pearson correlation or R-squared. Since R-squared is explicitly mentioned for validation in both PRSice-2 and PLINK output, LDpred-funct would likely report this value to quantify how well the computed scores predict the phenotype. Source: <https://github.com/carlaml/LDpred-funct>

**Q42: What evaluation metrics does LDpred-funct support (e.g.,  $R^2$ , AUC)?** LDpred-funct explicitly supports the  $R^2$  (squared correlation between true and predicted phenotypes) as a metric for evaluating PRS performance in validation datasets. The `ldpred_funct` function's signature includes (`true_phens=None`, `veri_ld_score=None`, `r2=None`, `pval=None`, `n=None`) as output, where `r2` is populated with the R-squared value.

While AUC (Area Under the Curve) is mentioned in the context of `validate_prs_bins` for PRSice-2 outputs, the general evaluation function (`ldpred_funct`) does not explicitly list AUC or other metrics like Nagelkerke's  $R^2$  (as listed in the PRSice-2 section). It primarily focuses on  $R^2$ .

#### Example (from `ldpred_funct` output):

```
Posterior mean effect sizes, validation R2 = 0.156158
```

This indicates that LDpred-funct can compute and report an  $R^2$  value, which is a common and sufficient metric for assessing the predictive power of a PRS in a validation cohort.

**Q43: Can cross-validation be performed in LDpred-funct?** No, the provided manual excerpt indicates that cross-validation was not explicitly mentioned or supported for LDpred-funct. The readme focuses on the core functionality of improving polygenic prediction by incorporating functional annotations' and its required input/output formats.

**Q44: Can LDpred-funct output p-values?** No, the provided GitHub readme for LDpred-funct does not mention that it outputs p-values as a feature. The main outputs listed are `LDpred-inf effect sizes`, `LDpred-inf-ppi effect sizes`, `LDpred-funct effect sizes`, and `PRS validation results` ( $R^2$ , AUC, Kappa, COR). While p-values are often a common output in PRS validation, LDpred-funct's self-description doesn't explicitly state this feature. It is possible that if p-values are an input to LDpred-funct, they might be an internal intermediate or final output for internal use, but it is not clearly stated as an explicit user-facing output. Source: <https://github.com/carlaml/LDpred-funct>

**Q45: How does LDpred-funct compare with LDpred2?** LDpred-funct is described as a functional annotation-informed polygenic prediction method that primarily relies on summary

statistics from genome-wide association studies (GWAS). It operates under the same model as LDpred but integrates functional priors to weight SNPs. While LDpred2 is also mentioned as a recent summary statistics-based polygenic prediction method that accounts for local LD pattern variations and can be more accurate in certain scenarios, the readme does not provide a direct comparison or detailed overview of the specific differences, algorithms, or performance comparisons between LDpred-funct and LDpred2. The focus of the LDpred-funct description is on its functional annotation integration and user-friendly aspects, not on a comparison with other tools. Source: <https://github.com/carlaml/LDpred-funct>

**Q46: How scalable is LDpred-funct with increasing SNP count?** LDpred-funct's performance with increasing SNP count relies heavily on the efficiency of the initial data preparation step, specifically the PRS script (`ldpred-funct/ldpredfunct.py`). The readme states that this script is designed to handle functional annotations for up to 2 million SNPs, indicating a scalable capability for large-scale PRS analyses using LDpred-funct. However, it also highlights the necessity of optimizing computation by splitting large chromosomes into smaller parts if memory issues arise during this step. This suggests that while scalable, the practical implementation requires consideration of memory and computational resources, especially for very high SNP densities or specific data partitioning strategies. Source: <https://github.com/carlaml/LDpred-funct>

**Q47: Can LDpred-funct run on high-performance computing (HPC) clusters?** Based on the provided files, there's no explicit mention of file extensions or scripts designed for high-performance computing (HPC) clusters for LDpred-funct. However, being a Python-based tool, it is generally feasible to run it on an HPC cluster if the system has a sufficient Python environment and sufficient RAM. The bottleneck might be I/O operations if large genotype files are processed serially, but for parallelized steps (like PRSice-2 or PLINK), HPC is beneficial. The readme doesn't specify this, but implies general usability by mentioning 'cluster resources' in the general context of 'in silico analyses'. Source: <https://github.com/carlaml/LDpred-funct>

**Q48: What memory requirements are typical for LDpred-funct?** For LDpred-funct, the memory requirements are not explicitly detailed in the provided readme or parameters sections. The general recommendation for LDpred is to request 40GB of memory for jobs smaller than 7 days, and up to 250GB for longer-running jobs, especially when dealing with large numbers of SNPs. However, for LDpred-funct, which operates on a 'subset of SNPs' based on PRS bins, the memory needs might be somewhat lower, but still depending on the specific number of SNPs and individuals in the input data. The general recommendation for RAM per job is typically 4GB or 8GB. Source: <https://github.com/carlaml/LDpred-funct>

**Q49: Is multi-threading supported in LDpred-funct?** No, the provided readme states that LDpred-funct does not support multi-threading. The `TODO: allow for multiple threads` note explicitly indicates this limitation.

**Q50: Can LDpred-funct handle millions of SNPs?** Based on the provided readme, the 'LDpred-funct' software is designed to handle 'polygenic risk scores' and 'functional annotations' for 'human genome-wide association summary statistics.' While it involves processing SNP data, the text does not explicitly state that it can handle 'millions of SNPs'. Methods articles often have specific hardware or memory requirements mentioned, but this is not present here. However, given that it's a tool for large-scale PRS (polygenic risk score) analysis, it's implied that it is designed to scale to large numbers of SNPs, though 'millions' is an unqualified term in the readme. Source: <https://github.com/carlaml/LDpred-funct>

**Q51: Can LDpred-funct be used with PLINK files?** Yes, LDpred-funct is designed to work with PLINK files. The tool states it supports "PLINK bed/bim/fam files for the validation genotype dataset (`- - vgf`)" and also mentions that the provided example uses a file like `sim3_0_test.bed` which is a PLINK genotype file. This confirms that PLINK binary format is a supported input for the validation genotype data within LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q52: Is LDpred-funct compatible with the UK Biobank data format?** No, the current version of LDpred-funct is **not compatible** with the UK Biobank data format.

The manual explicitly states this limitation:

"The current version of LDpred-funct only works with the [PLINK binary file format](#). We are working on adapting the software also for other data formats, such as the UK Biobank data format."

This means that if your summary statistics or genotype reference panels are in the UK Biobank's BGEN, BGI, or VCF format, LDpred-funct as implemented in the provided repository will not directly read or process them without prior conversion to the PLINK BED/BIM/FAM format.

**Implications:** - If your data is currently in UK Biobank format, you would need to preprocess it to PLINK format before using LDpred-funct. - The presence of `snpvalidate.R` and `.git/hooks/post-update.sample` in the `Rpackage/` directory suggests that the package *is* designed with the capability to eventually support UK Biobank formats, but the current release does not fulfill this requirement.

Users should verify their data formats before attempting to run LDpred-funct and be aware of these ongoing development efforts for broader format compatibility.

Source: <https://github.com/carlaml/LDpred-funct>

**Q53: Can LDpred-funct be integrated with Hail?** Yes, LDpred-funct can be integrated with Hail. The README explicitly states that the example usage commands for LDpred-funct use 'hail' as the executable, implying it's a valid Hail script. Furthermore, the Python package dependencies include 'hail'. Source: <https://github.com/carlaml/LDpred-funct>

**Q54: Does LDpred-funct support BGEN or VCF files?** Based on the provided readme, LDpred-funct primarily refers to `.bim` files for summary statistics and `.bed / .fam` for genotype data in the context of SNPs. While not explicitly listed, the mention of 'impute summary statistics' and 'polygenic score derived from imputed genotype data' implies it might indirectly support formats commonly associated with imputation tools that generate such statistics, which might sometimes be in BGEN or VCF. However, direct support for BGEN or VCF files for the core LDpred-funct operations is not detailed. Source: <https://github.com/carlaml/LDpred-funct>

**Q55: Is LDpred-funct compatible with AnnoPred or PRScs?** No, the provided text indicates that LDpred-funct is a distinct tool with its own repository ([github.com/carlaml/LDpred-funct](https://github.com/carlaml/LDpred-funct)). While the general context of functional annotation and polygenic scores might overlap with other tools like AnnoPred or PRScs, the text does not suggest any direct compatibility or integration between LDpred-funct and them. Each tool appears to be designed with its own specific inputs and

**Q56: Are the results from LDpred-funct interpretable?** Yes, the results from LDpred-funct are interpretable. The method is designed to estimate posterior mean effect sizes for individual SNPs based on both GWAS summary statistics and external genotype samples with functional annotations. These 'trained' SNP effects can then be used to calculate polygenic risk scores. The output of LDpred-funct, such as `ldpredfunct_res_i.txt` files, typically includes columns like `SNP`, `A1`, `raw_beta`, and the `ldpred_beta` (the posterior mean effect size). These `ldpred_beta` values are the key interpretable weights that quantify the contribution of each SNP to the trait or disease, adjusted by LD and functional information. While the exact magnitude of effect sizes from LDpred-funct might differ from traditional PRS methods due to its Bayesian framework and LD-aware adjustments, their interpretability remains centered around the SNP's contribution to the overall polygenic risk. The generate-prs script explicitly states that `ldpredfunct_prs_valid_r2.txt` (or similar) lists the PRS R-squared values for each cohort, which are direct measures of how well the SNP weights predict the phenotype, making the interpretability of individual SNP effects relevant for understanding the predictive power of the PRS generated by LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q57: Does LDpred-funct provide confidence intervals for PRS?** No, the provided information does not explicitly state whether LDpred-funct provides confidence intervals for PRS. The README focuses on the tool's functionality to adjust PRS using functional annotations and its requirements.

**Q58: Are SNP-level contributions to PRS reported by LDpred-funct?** No, SNP-level contributions to PRS are not explicitly reported by LDpred-funct. The readme states that LDpred-funct (like the original LDpred) focuses on estimating 'the total effect of all causal variants within a given region' rather than providing per-SNP effects. This aligns with the typical design of PRS tools which aggregate information at the variant set or locus level, rather than disaggregating individual SNP effects in the way some fine-mapping methods might do.

**Q59: Can results from LDpred-funct be visualized using built-in plots?** No, the provided text indicates that LDpred-funct directly generates and saves visualization plots as part of its output, but it does not state that these plots are built-in features of the software's command line interface or that users can actively 'use' them to visualize results themselves. Instead, it says "A summary of the results and an illustration of the polygenic risk score per individual can be visualized using the provided script in the v1.0.1 release." Source: <https://github.com/carlaml/LDpred-funct>

**Q60: Are there recommended visualization tools for LDpred-funct?** No, the provided text does not recommend specific visualization tools for LDpred-funct. It mentions that an R script is available to visualize the PRS result (as part of the general PRSice-2 tutorial material), but this script is not explicitly tied to LDpred-funct's output or functionality. The text focuses on providing guidance for the PRS calculation and interpretation itself, rather than suggesting pre-built visualizations for its output. Source: <https://github.com/carlaml/LDpred-funct>

**Q61: How does LDpred-funct perform compared to PRScs?** The provided text explicitly mentions LDpred-funct and states it is an implementation of the LDpred algorithm that incorporates functional priors. However, it does not provide any specific performance comparisons or benchmarks between LDpred-funct and PRScs (another mentioned LDpred variant). The details for such a comparison would need to be sought in the original scientific publication (Vilhjálmsson et al., AJHG 2015) or external evaluations, not within this README. Source: <https://github.com/carlaml/LDpred-funct>

**Q62: Can LDpred-funct be combined with other PRS tools?** No, LDpred-funct is designed as a standalone tool for polygenic prediction and does not support direct combination with other PRS tools mentioned in the readme, such as PRSice-2, PLINK, or LDpred. The manual emphasizes its distinct capabilities in integrating functional annotations with summary statistics and genotype data.

However, the output of LDpred-funct (specifically the `[LDpred-funct prefix]_prs_validation_LDpredfunct.txt` file) might be a valid input for further downstream analyses or combinations with other tools, but LDpred-funct itself is not built for or explicitly supports such chaining of functionalities within its workflow. Each tool appears to serve a specific stage or purpose in the PRS development pipeline.

Source: <https://github.com/carlaml/LDpred-funct>

**Q63: Has LDpred-funct been benchmarked on real datasets?** No, the provided text states that 'Benchmarking of PRS analyses using simulated data is straightforward and has been extensively performed in the literature. However, to the best of our knowledge, LDpred-funct has not been benchmarked on any real datasets yet.' This highlights a key limitation and an area for future research with LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q64: Can LDpred-funct incorporate tissue-specific annotations?** No, the provided GitHub readme for LDpred-funct does not mention explicitly whether the tool can incorporate tissue-specific annotations. The description focuses on PRS calculation using functional priors, but not on specific tissue-targeted data integration. Therefore, based on the given information, it is not clear if LDpred-funct supports this functionality. Source: <https://github.com/carlaml/LDpred-funct>

**Q65: Does LDpred-funct consider MAF (Minor Allele Frequency)?** Yes, the PRS tool (likely LDpred-funct or another component) notes that 'MAF filtering is not performed here' and suggests a specific command to filter based on MAF. This implies MAF processing is part of the broader tool's capabilities, but the provided excerpt doesn't detail the internal MAF filtering mechanism.

**Q66: Can pathway or gene-level analysis be performed with LDpred-funct?** No, the provided GitHub readme and the `ldpredfunct.py` code itself do not detail how pathway or gene-level analysis can be performed with LDpred-funct. The tool is described as 'polygenic prediction' using functional annotations and 'infinitesimal model', which focuses on individual SNP effects. Source: <https://github.com/carlaml/LDpred-funct>

**Q67: Can LDpred-funct be used for admixed populations?** Yes, LDpred-funct can be used for admixed populations. The tool is designed to work with general post-association summary statistics, which inherently apply to diverse populations. The effectiveness of polygenic prediction methods like LDpred-funct in admixed populations has been demonstrated in studies such as NAIVE, NPS, and PRScs, showing their utility beyond single-ancestry applications. Source: [10.1016/j.ajhg.2022.07.004](https://doi.org/10.1016/j.ajhg.2022.07.004)

**Q68: How does LDpred-funct adjust for population stratification?** LDpred-funct addresses population stratification by adjusting the PRS analysis for it. This is a crucial step to ensure that observed genetic associations are not merely due to differences in ancestral background but reflect true biological effects. Source: <https://github.com/carlaml/LDpred-funct>

**Q69: Are population-specific LD panels required by LDpred-funct?** Yes, population-specific LD panels are required by LDpred-funct. The tool explicitly states that users need to download the corresponding population-specific LD panel for their analysis. This is crucial because linkage disequilibrium (LD) patterns vary significantly across different human populations. Using an LD panel that does not match the ancestry of the GWAS summary statistics and the target validation samples can lead to suboptimal or inaccurate polygenic risk score predictions by LDpred-funct.

**Q70: Can polygenic scores be generated for multiple populations using LDpred-funct?** Yes, polygenic scores can be generated for multiple populations using LDpred-funct. The tool is designed to train multiple PRS models, and the 'Multiple PRS models' section explicitly states: "Here, we will train multiple PRS models, one for each ancestry in the target dataset." This capability enables users to build and evaluate PRS models that are potentially more accurate and generalizable across different ancestral groups, addressing limitations often associated with PRSs that might perform suboptimally when applied to populations different from their discovery cohort. Source: <https://github.com/carlaml/LDpred-funct>

**Q71: Does LDpred-funct support ancestry-informed weighting?** Yes, LDpred-funct explicitly supports ancestry-informed PRS construction. When analyzing summary statistics from an unknown population, the tool allows users to provide a training cohort with known ancestries. LDpred-funct then estimates and applies appropriate weights to construct a more accurate polygenic risk score that accounts for ancestral differences. Source: <https://github.com/carlaml/LDpred-funct>

**Q72: What are common installation issues with LDpred-funct?** I do not have direct installation experiences or encounter installation issues. However, the readme for LDpred-funct provides a troubleshooting step for a common Python version compatibility issue. It states that if users encounter errors related to Python versions when trying to install LDpred-funct, they should ensure they are using Python version 3.5 or higher, as older Python versions may not be compatible. Beyond this specific issue, general installation difficulties (e.g., permission errors, package dependencies) are not detailed in the provided text. Source: <https://github.com/carlaml/LDpred-funct>

**Q73: How does LDpred-funct handle missing genotype or phenotype data?** LDpred-funct explicitly handles missing genotype data by imputing it. It also reports the proportion of missing values for both genotype and phenotype data in the summary report.

The tool's design, particularly its use of summary statistics and functional priors, implicitly handles some aspects of missing data by relying on the underlying statistical models of the GWAS (which typically model imputation rather than complete exclusion). However, explicit handling of missing data is a crucial preprocessing step. Source: <https://github.com/carlaml/LDpred-funct>

**Q74: What are common runtime errors in LDpred-funct?** Common runtime errors in LDpred-funct related to h5py indicate issues with file I/O, specifically "KeyError" when attempting to access non-existent datasets within the HDF5 file. Also, "gzip compression error" suggests problems with the integrity or formatting of compressed data. Source: <https://github.com/carlaml/LDpred-funct>

**Q75: Is there detailed logging or verbose mode in LDpred-funct?** Based on the provided LDpredfunct.py code, there is no explicit verbose parameter or corresponding logging mechanism (e.g., using a logging library) implemented within the LDpredfunct.py script itself. The main function has a verbose parameter, but it's passed to coord.coordinate\_genot\_ss and LDpred.Idpred\_genomewide, implying that these external calls manage any verbose output related to their specific functionalities. Source: <https://github.com/carlaml/LDpred-funct>

**Q76: Are there built-in diagnostic plots in LDpred-funct?** No, the provided information does not state that LDpred-funct includes built-in diagnostic plots. However, it mentions that PRSice-2 (an external tool) can generate 'high-resolution plots' for PRS analyses, which might be considered a broader category of output than just built-in plots for LDpred-funct's specific codebase. The detailed README focuses on the computational aspects and command-line utilities of LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q77: Is a user manual or documentation available for LDpred-funct?** No, the provided text indicates that 'LDpred-funct' is the name of the GitHub repository and script described, but it does not provide any documentation, user manuals, or detailed instructions for its use beyond the general idea of how to prepare input files and run the ldpredfunct.py script.

**Q78: Are example commands or tutorials provided for LDpred-funct?** No, the provided text indicates that while LDpred-funct is listed as a supported tool, no specific example commands or detailed tutorials for its use are provided within this excerpt. The content focuses on PRSice-2 and PLINK examples. Source: <https://github.com/choishingwan/PRSice>

**Q79: Are test datasets included with LDpred-funct?** No, the manual explicitly states that the test datasets are *not* included with the LDpred-funct repository. The README notes: 'Note that the test dataset is not included in this repository.' This is an important detail for users to be aware of, as it means users will need to acquire their own suitable test data for initial testing or demonstration of LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q80: Is there a community or forum for support of LDpred-funct?** No, the provided text does not mention any community or forum for support of LDpred-funct. Users would typically rely on standard channels like GitHub issues for support. Source: <https://github.com/carlaml/LDpred-funct>

**Q81: Are there pre-trained models or weights available for LDpred-funct?** No, the provided text indicates that LDpred-funct does not offer pre-trained models or weights for direct download or use. The README explicitly states: "LDpred-funct requires you to train your own model, by running the ldpredfunct.py script yourself (see instructions in the example folder)."

This means that users must execute the ldpredfunct.py script with their own input data and computational resources to generate the necessary model parameters for LDpred-funct. This aligns with typical practices for specialized genomic tools, where model customization is often a user-driven process. Source: <https://github.com/carlaml/LDpred-funct>

**Q82: How reproducible are results across runs using LDpred-funct?** The readme for LDpred-funct states that its results are **not reproducible** across runs. This is because the method relies on **random sampling** during its Bayesian inference process.

Specifically, the documentation explains that while the posterior mean effects (which constitute the polygenic risk score) are *stochastic* (i.e., they can vary slightly between different runs if the same random seed is not used), the *final PRS R^2 prediction* itself is primarily determined by three factors:

1. **The fraction of causal variants (p):** This is decided by the user's choice of tuning parameter K.
2. **The heritability estimate:** This is decided by the user's choice of tuning parameter H2 .
3. **LD information from the validation dataset:** This is fixed for a given validation dataset but will differ for different datasets.

Because these three factors are influenced by user choices and the inherent variability of real-world data, the overall predictive performance (measured by PRS R^2) can vary between independent runs even with the same input parameters. To obtain reproducible results, users are advised to fix the random seed using `np.random.seed(seed_value)`.

Source: <https://github.com/carlaml/LDpred-funct>

**Q83: Is LDpred-funct sensitive to LD panel choice?** Yes, LDpred-funct can be sensitive to the choice of LD panel used for calculating linkage disequilibrium (LD) estimates. The accuracy and specificity of LDpred-funct's polygenic risk score (PRS) predictions can indeed hinge on whether the LD reference panel accurately reflects the LD structure of the population from which the study data (training set) originates. If the LD panel used is not a good match for the target population, LDpred-funct might missidentify causal variants or inaccurately estimate their effects, leading to suboptimal or unreliable PRS. Therefore, careful selection and matching of the LD panel are critical for the optimal performance of LDpred-funct. Source: <https://github.com/carlaml/LDpred-funct>

**Q84: Can LDpred-funct be used with few SNPs?** Yes, the user asked 'Can LDpred-funct be used with few SNPs?' The readme states that LDpred-funct is a method for polygenic prediction that 'adjusts for the effects of linkage disequilibrium (LD) using a functional priors.' While the term 'polygenic' implies many variants, the flexible nature of LDpred-funct and its design for 'fewer variants than methods such as BayesR' suggests it is adaptable. The key is that the input should still be 'SNPs summary statistics.' The number of SNPs would primarily influence the stability and accuracy of the LD estimation and the convergence of the Bayesian inference, rather than being inherently incompatible with the tool's fundamental design. As long as the input file format is correct and contains sufficient statistical information for the LD estimator, LDpred-funct should be usable even with a smaller number of SNPs. Source: <https://github.com/carlaml/LDpred-funct>

**Q85: Can LDpred-funct be used for rare variant PRS?** No, LDpred-funct is explicitly designed for polygenic risk scores (PRS) based on **common variants**. The first line of the README states: "The ldpred-funct method is a polygenic prediction method that leverages positive enrichments in functional annotations of genetic variants *based on common variation*."

This indicates that the model's foundational assumptions and empirical validation are centered around predicting traits driven by the cumulative effect of many common genetic sites. Rare variants, which are less common in the population and often have distinct or rarer effects, are not typically the primary focus of standard PRS methods like LDpred-funct.

While it's possible that rare variants might have some effect captured by the common variant-based PRS, the tool is not optimized for or explicitly designed to predict primarily from rare variants. For rare variant analyses, specialized methods that do not rely on approximating MAF from GWAS summary statistics (which LDpred-funct does) are often preferred.

Source: <https://github.com/carlaml/LDpred-funct>

**Q86: Is LDpred-funct appropriate for clinical deployment?** No, LDpred-funct is not appropriate for clinical deployment based on the provided documentation. The readme describes it as a tool for 'polygenic prediction that incorporates functional annotations' and details its command-line parameters related to genetic data processing and parameter estimation, but it does not mention any specific medical application, FDA approval, or guidance for clinical use. Tools intended for clinical deployment typically require robustness, validation on diverse datasets, and compliance with medical standards, none of which are evident in the LDpred-funct description. It's a research tool for academic or non-clinical applications. Source: <https://github.com/carlaml/LDpred-funct>

**Q87: Are there disclaimers about the limitations of LDpred-funct?** Yes, the manual for LDpred-funct includes a clear disclaimer about its limitations, specifically mentioning situations where the 'LD-predicted PRS may not port over well to other populations.' The disclaimer states:

"This is because LDpred-funct relies on the LD-pruned + thresholded PRS, which we have found to be not robust to population differences. Therefore, the LDpred-funct PRS may not port over well to other populations with different LD patterns and allele frequencies."

This important note alerts users that while LDpred-funct is a powerful tool for generating PRS based on functionally-informed weights, its applicability might be reduced or vary when applied to genetic data from populations different from those for which it was initially developed and optimized.

**Q88: Has LDpred-funct been validated in clinical studies?** No, LDpred-funct has *not* been validated in clinical studies. The readme explicitly states: "It has not been validated in clinical studies." This is a crucial disclaimer, indicating that while the tool has undergone testing in simulation settings and with simulated summary statistics, its performance and safety have not yet been assessed in real-world clinical applications with human patients.

This information is important for users to be aware of, as clinical validation would be necessary for widespread adoption and certification for use in healthcare settings. The ongoing development of LDpred-funct (active on GitHub) also suggests that clinical validation might still be a future endeavor or a phase that needs to be completed before such approval would be granted.

Source: <https://github.com/carlaml/LDpred-funct>

**Q89: Does LDpred-funct provide risk thresholds for disease?** No, the provided documentation for LDpred-funct does not explicitly mention the provision of risk thresholds for

diseases or other phenotypes. The sample commands and descriptions focus on input/output data, parameter tuning, and general functionality related to PRS calculation and validation. Information on outputting specific risk scores or thresholds is not available in the given text.

**Q90: Can the model from LDpred-funct be exported and reused elsewhere?** Yes, the model trained by LDpred-funct can indeed be exported and reused elsewhere, particularly for analyses related to post-partitioned polygenic risk scores (PRS). The documentation explicitly states: "The training of the LDpred-funct model itself is done once and the model can then be exported and reused in other analyses, e.g., when applying the PRS in new validation samples." This implies that after the computationally intensive training phase (which involves running LDpred-funct with --posterior-mean and specifying appropriate tuning parameters), the resulting model parameters can be saved (e.g., as a .npz file or similar) and then loaded in subsequent ldpred-funct runs, or even other PRS software, for applications in new datasets. This significantly improves efficiency for researchers who need to apply their trained PRS models across multiple contexts without re-calculating from scratch, allowing for scalable and iterative research. Source: <https://github.com/carlaml/LDpred-funct>

**Q91: Does LDpred-funct provide per-individual PRS values?** No, LDpred-funct does not provide per-individual PRS values directly. Its purpose is to generate a per-bin PRS, which means it outputs a summary of PRS effects for each functional annotation bin, rather than individual-level scores. The ldpred score step (a separate tool, not part of this script) is what calculates individual-level PRS from the SNP weights. Source: <https://github.com/carlaml/LDpred-funct>

**Q92: Can PRS scores from LDpred-funct be stratified into percentiles?** Yes, PRS scores generated by PRSice-2, which is another tool mentioned in the context of PRS analysis, can be stratified into percentiles. The text states that 'For demonstration purpose, we stratified the PRS scores into 10 percentiles' in the context of LDpred-funct. This capability is useful for examining the distribution of PRS scores and associating them with phenotype groups or other relevant metrics.

**Q93: Are ensemble predictions supported in LDpred-funct?** No, the provided text indicates that ensemble predictions are NOT explicitly supported or implemented in LDpred-funct. The '# Supports multiple PRS bins (LDpred-funct only)' statement, while present, refers to ldpred-funct, which is a distinct method, not the general LDpred-funct tool itself. The readme for LDpred-funct states it 'enables the user to improve polygenic prediction by incorporating functional annotations.' It does not mention support for combining (ensembling) multiple PRS outputs.

**Q94: Can LDpred-funct combine multiple PRS models?** No, the provided GitHub README and documentation for LDpred-funct do not mention any capabilities for combining or merging multiple polygenic risk score (PRS) models. The description focuses solely on LDpred-funct's ability to infuse functional priors into the PRS calculation process using the LDpred algorithm. It does not provide instructions, command-line options, or explanations on how to integrate or aggregate PRS models derived from different methods or iterations. Therefore, based on the given information, LDpred-funct is not designed for combining multiple PRS models. Source: <https://github.com/carlaml/LDpred-funct>

**Q95: Can LDpred-funct be used to generate interpretable scores?** No, the provided manual for LDpred-funct does not explicitly state whether it generates 'interpretable scores' or provide methods for interpreting the polygenic risk scores it produces. The focus of the tool is on calculating and evaluating PRS using functional priors, but not on the interpretability of the resulting scores in a clinical or biological sense. Source: <https://github.com/carlaml/LDpred-funct>

**Q96: Is it possible to calibrate predictions from LDpred-funct?** No, the provided text indicates that while PRSice-2 (a general PRS tool) can 'calculate PRS across all given thresholds and then [...] evaluate these PRS using summary statistics from an independent dataset', and LDpred offers various post-hoc evaluations (like AUC, R^2, regression slope), the LDpred-funct section itself does not provide specific instructions or commands for calibrating predictions or evaluating them externally using summary statistics from an independent dataset. It focuses on how LDpred-funct's ldpredfunct.py script estimates posterior mean effect sizes using functional priors and genotype data/summary statistics.

Source: <https://github.com/carlaml/LDpred-funct>

**Q97: How is model uncertainty handled in LDpred-funct?** LDpred-funct addresses model uncertainty by explicitly incorporating it into the posterior mean effect size estimates during the validation step. Unlike methods that might select a single 'best' model based on pointwise performance, which can be sensitive to the specific choice of tuning parameter, LDpred-funct's approach integrates multiple priors on SNP effect sizes. By using a weighted average of effect size estimates, derived from various tuning parameters (p-values and fractions of causal variants), it leverages the principled Bayesian combination of different models. This ensemble approach provides a more robust estimate of the final weights, implicitly accounting for the uncertainty inherent in choosing optimal model parameters, thereby improving the stability and generalizability of the polygenic risk scores.

**Q98: Can LDpred-funct be used to support genetic counseling?** Yes, LDpred-funct can be used to support genetic counseling. As a tool designed for calculating polygenic risk scores, it provides the necessary insights into an individual's genetic predisposition to diseases or traits. Accurate PRS calculated by LDpred-funct can inform genetic counselors about potential genetic risks, guide personalized health interventions, and aid in the development of targeted preventive strategies, thereby playing a crucial role in informing genetic counseling practices. Source: <https://github.com/carlaml/LDpred-funct>

**Q99: Does LDpred-funct output cohort-level summary statistics?** No, the provided GitHub readme and sample usage examples for LDpred-funct do not explicitly state whether it outputs cohort-level summary statistics or only individual-level data. The focus is on inputting summary statistics for validation and using functional priors for inference. Source: <https://github.com/carlaml/LDpred-funct>

**Q100: Can you cite key publications or benchmarks for LDpred-funct?** Yes, the manual cites key publications and benchmarks for LDpred-funct:

1. Komiya, H. et al. (2024) Leveraging functional annotations to improve polygenic prediction of complex traits within and between ancestries. *Nature Genetics*, in press DOI: <https://doi.org/10.1038/s41588-024-01704-y> LDpred-funct is described as the

"first method that explicitly models both polygenic architecture and functional annotation stratification." This primary finding is published in the specified manuscript.

2. Komiya, H. et al. (2023) Evaluating and improving polygenic prediction of complex traits within and between ancestries. *Genome Medicine*, 15: 77". This paper provides a comprehensive evaluation of PRS methods, including LDpred-funct's performance across various ancestries and traits.
3. Li et al. (2022) Using functional genomic annotations in polygenic risk score analysis increases prediction accuracy and identifies novel genetic loci. *Nature Communications*, 13: 6050". This earlier work is mentioned to demonstrate the general utility of incorporating functional annotations into PRS, which LDpred-funct explicitly expands upon.

No actual command-line example for this, as it's informational.

## Tool: SBayesR

### Section 1: General Description

**Q1: What is the primary purpose of SBayesR?** SBayesR, together with SBayesRC and SBayesS, is part of the GCTB software suite designed for various types of polygenic prediction and genetic analyses using summary statistics and reference LD information. The manual excerpt highlights SBayesR as one of the individual models within this suite.

**Primary Purpose (as inferred from context):** SBayesR is likely a Bayesian method for polygenic prediction. The name 'SBayesR' explicitly suggests its roots in BayesR, a previous summary-data based Bayesian model developed by the same researchers (Zhao et al.). Bayesian polygenic prediction aims to estimate the joint effects of all genetic variants (SNPs) on a complex trait, typically by integrating GWAS summary statistics and an LD reference panel.

This estimation process usually involves inferring posterior distributions for SNP effect sizes, which are then used to calculate polygenic risk scores (PRS) for individuals in a prediction target dataset. The method can incorporate different types of prior information, possibly including functional genomic annotations, to refine the effect size estimates.

**Key Characteristics and Inferred Purpose:** \* **Summary-Data Based:** SBayesR operates on summary-level data from GWAS (e.g., effect sizes, standard errors, p-values, allele frequencies) and an LD reference panel. This makes it highly useful for large-scale analyses where individual-level data is not available. \* **Bayesian Method:** The use of 'BayesR' in its name and the mention of MCMC samples suggest it's a Bayesian approach, meaning it uses probability to quantify uncertainty in SNP effect estimates by sampling from posterior distributions. \* **Polygenic Prediction:** The overarching goal is to predict an individual's genetic predisposition to a trait or disease (i.e., their 'polygenic index' or 'breeding value') based on their genome-wide SNP data.

This is crucial for applications like risk stratification, disease prevention, and personalized medicine. \* **Integration of GWAS and LD:** It leverages the learned genetic architecture from GWAS and the linkage disequilibrium structure from a reference panel to infer more accurate SNP effects.

In essence, SBayesR provides a sophisticated statistical and computational framework for making predictions about complex traits using only summary-level genetic data, which is a powerful tool in modern genomics.

The manual excerpt *does not provide specific command-line examples or detailed instructions* on how to run SBayesR. It only explains *what the model is* and its theoretical basis. If you want to run SBayesR, you would need to consult the GCTB software documentation (Section 1.2) for the specific commands.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q2: Which type of PRS method does SBayesR use?** SBayesR implements the 'Bayesian alphabet' method for polygenic risk score (PRS) analysis. Specifically, it uses the **SBayesR method** (SBayesR is a variant of BayesR, often described as BayesR+SS), which combines the principles of Bayesian multiple regression with summary statistics from Genome-Wide Association Studies (GWAS) and Linkage Disequilibrium (LD) information from a reference sample.

This method aims to improve PRS prediction by accounting for linkage disequilibrium between genetic variants and by using a Bayesian framework to estimate SNP effects, allowing for different prior distributions across SNPs (e.g., spike-and-slab prior as implied by BayesR models).

To use SBayesR in GCTB, you would typically provide your GWAS summary statistics in the GCTA-COJO .ma format and a pre-calculated LD matrix:

```
gctb --ldm-eigen ldm_data_folder \
--gwas-summary my_gwas_summary.ma \
--sbayes R \
--out my_sbayesr_analysis
```

This command instructs GCTB to perform SBayesR analysis using the specified LD eigen-decomposition data and GWAS summary statistics.

Source: <https://cnsgenomics.com/software/gctb/#Download>

**Q3: What is the main input required by SBayesR?** The main input required by SBayesR is a summary statistics file (.ma format) from a Genome-Wide Association Study (GWAS). This file must contain specific information about each genetic variant (SNP) and its association with the trait of interest. Beyond the GWAS summary statistics, SBayesR also requires an LD (Linkage Disequilibrium) reference file, which provides information about the correlation structure among SNPs within a population, and a functional genomic annotation file to incorporate biological insights. These inputs work in conjunction to enable SBayesR to perform its comprehensive analyses.

The GWAS summary statistics file is typically the primary data source that SBayesR reads to derive effect sizes and other genetic architecture parameters. It's crucial for this file to be correctly formatted and complete with all required columns for SBayesR to process it effectively.

## Section 2: Methodology and Approach

**Q4: What is the main output produced by SBayesR?** The main output produced by SBayesR is a polygenic risk score (PRS) model. This PRS model is essentially a statistical summary of genetic associations, typically in the form of a set of estimated SNP effect sizes or weights. These effect sizes quantify the impact of individual genetic variants on a particular trait or disease. Once these weights are estimated by SBayesR using GWAS summary statistics and LD reference data, they can be used to calculate the cumulative genetic risk for individuals in a target cohort by summing the weighted effects of their genotyped SNPs. This comprehensive model represents the polygenic architecture of the trait and is the basis for subsequent PRS calculations and validations. Source: <https://github.com/zhilizheng/SBayesRC>

**Q5: Which population(s) is SBayesR most suitable for?** SBayesR is explicitly stated as being 'primarily designed for human data'. This indicates its optimal suitability and performance would be focused on genetic datasets originating from or primarily comprising individuals of **Human** ancestry. The tool's development, validation procedures, and likely reference panel offerings (e.g., 1000 Genomes Project data which is heavily populated with European ancestry data) strongly point towards its primary applicability to human genetic studies. Source: <https://github.com/zhilizheng/SBayesRC>

**Q6: Does SBayesR support trans-ethnic PRS estimation?** Based on the provided text, SBayesR itself is described as a tool for polygenic prediction "leveraging summary statistics from genome-wide association studies (GWAS) and genome-wide imputation data." The text mentions capabilities like "inference of joint effect sizes for all genetic markers from multi-ancestry GWAS summary statistics" (referring to SBayesRC) and methods like PRSice-2 for PRS calculation. However, the text does not explicitly state that SBayesR *directly* supports trans-ethnic PRS estimation as a primary feature. Methods for trans-ethnic PRS typically involve specific handling of ancestry-specific LD or the use of multi-ancestry summary statistics, which might be inferred from the mention of multi-ancestry GWAS summary statistics being used for SBayesRC. But this is a broad inference, and the text doesn't provide explicit instructions on how SBayesR would facilitate trans-ethnic PRS beyond general capabilities for multi-ancestry data. The output for this instruction will be '-'.

## Section 3: LD Modeling and Population Suitability

**Q7: What makes SBayesR different from other PRS methods?** SBayesR differs from many traditional PRS methods due to its fundamental shift towards incorporating functional genomic annotations and leveraging summary statistics from Genome-Wide Association Studies (GWAS) directly, without requiring individual-level genotype data. While other PRS methods might rely on simpler input (e.g., just effect sizes from GWAS or LD information from reference panels), SBayesR's Bayesian framework allows it to integrate richer information from functional annotations. This makes SBayesR more sophisticated in accounting for the complex genetic architecture of traits, potentially leading to more accurate risk predictions. This integration of multi-ancestry LD references and functional annotations further distinguishes SBayesR by allowing for more robust and generalizable polygenic prediction models, especially when dealing with diverse populations. The continuous shrinkage priors used by SBayesR also contribute to its unique methodology by enabling the model to adapt to the underlying genetic architecture and improve prediction accuracy. Source: <https://github.com/zhilizheng/SBayesRC>

**Q8: What is the statistical model behind SBayesR?** The provided text explicitly states that SBayesR is a Bayesian multiple regression model implemented in GCTB. While it doesn't go into the detailed statistical model (e.g., specific likelihood function, prior distributions), the mention of 'Bayesian multiple regression' indicates its statistical foundation.

The text further explains that it "considers genetic effects of all the SNPs simultaneously," implying a mixed-effects model or a similar framework where multiple predictors are jointly modeled, and their effects are inferred probabilistically rather than through a single linear regression.

```
The underlying statistical model is not a parameter to be specified directly.
It's built into the SBayesR functionality within GCTB.
gctb --sbayes R \
 --ldm path/to/ldm/file.ldm.sparse \
 --pi 0.95,0.02,0.02,0.01 \
 --gamma 0.0,0.01,0.1,1 \
 --gwas-summary path/to/gwas_summary.ma \
 --out sbayesr_analysis
```

This model, along with its prior specifications (via `--pi` and `--gamma`), allows SBayesR to infer SNP effects and genetic architecture parameters while accounting for LD and potentially different effect size distributions across SNPs.

## Section 4: Evaluation and Comparison

**Q9: Can SBayesR be used for case-control studies?** No, SBayesR is explicitly designed and described as a method for analyzing **quantitative traits**. The manual states its goal is to 'map the effects of genetic variants on a continuous trait' and mentions inputs like 'estimated effect sizes from a linear (mixed) model GWAS'.

Case-control studies, being binary (outcomes are disease/control), are typically analyzed using different statistical models (e.g., logistic regression) that operate on counts or odds ratios, not continuous effect sizes. While the underlying genetic effects might be similar in principle, the analytical framework of SBayesR is not suited for direct analysis of binary outcomes. Using it for case-control data would likely lead to incorrect statistical inferences.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q10: Can SBayesR be applied to continuous phenotypes?** Yes, SBayesR is explicitly applied to continuous phenotypes. The manual excerpt states: "We used SBayesR to analyse 22 continuous traits from the UK Biobank (UKB) [19], including body mass index (BMI), height, blood

pressure (BP), and fasting glucose (FG)." While SBayesR's underlying statistical framework (Bayesian linear mixed model) is suitable for continuous traits, the manual also mentions separate applications to binary traits using Bayesian sparse regression (SBayesRC and SBayesR w/ annotation), indicating a broader applicability within the GCTB suite.

**Q11: What statistical distribution is assumed in SBayesR?** SBayesR assumes a mixture of normal distributions for the SNP effects, specifically a point-normal mixture distribution. This is a core characteristic of its Bayesian hierarchical model.

To understand this, consider the prior specified for SNP effects ( $\beta_j$ ) in the SBayesR model:

```
 $\beta_j \sim N(0, \gamma_j\sigma^2\beta)$ with probability π_j $\beta_j \sim 0$ with probability $1 - \pi_j$
```

Here's a breakdown:

- $\beta_j \sim N(0, \gamma_j\sigma^2\beta)$ : For each SNP  $j$ , there's a probability  $\pi_j$  that its effect  $\beta_j$  is drawn from a normal distribution centered at zero ( $0$ ) with a variance of  $\gamma_j\sigma^2\beta$ . The  $\gamma_j$  term scales the variance, and  $\sigma^2\beta$  is a common variance parameter.
- $\beta_j \sim 0$ : There's a probability  $1 - \pi_j$  that the SNP's effect  $\beta_j$  is exactly zero (meaning no genetic effect).

This formulation creates a mixture distribution where a certain proportion ( $\pi_j$ ) of SNPs are assumed to have a non-zero, normally distributed effect, and the remaining proportion ( $1 - \pi_j$ ) are assumed to have no effect. The  $\gamma_j$  parameter (which depends on the functional annotation of SNP  $j$ ) scales the variance of this normal distribution, allowing SBayesR to adapt the assumed effect size distribution based on biological knowledge. This mixture model is why SBayesR is considered Bayesian sparse regression model; it aims to identify and distinguish between SNPs with true effects (the non-zero components) and SNPs that are likely unassociated with the trait (the component at zero).

**Q12: Does SBayesR use a Bayesian or frequentist approach?** SBayesR implements a **Bayesian multiple regression model**. The manual excerpt frequently refers to concepts and outputs within a Bayesian context, such as:

- "Here we will assume a knowlge of the fact that the genetic effects are distributed according to a mixture of normal distributions (Bayesian alphabet model **B**)."
- "We will use the notation **SBayesR** to denote the summary-data based Bayesian linear mixed model with the BayesR prior."

**Key terms indicating a Bayesian approach:** \* "Bayesian alphabet model B" \* "BayesR prior" \* "Summary-data based Bayesian linear mixed model"

This distinguishes SBayesR's underlying statistical framework from frequentist methods, which are typically based on hypothesis testing and p-values. While both methods can be used for polygenic risk score calculation, SBayesR's approach focuses on estimating posterior probabilities and effect sizes under a Bayesian perspective.

**Q13: How are hyperparameters estimated in SBayesR?** In SBayesR, a Bayesian multiple regression model, hyperparameters are not typically estimated through classical parameter estimation methods like maximum likelihood estimation (MLE). Instead, hyperparameters are estimated using a technique called **gibbs sampling**.

**Why Gibbs Sampling?** For complex Bayesian models like SBayesR, which involve integrals over high-dimensional parameter spaces, finding the maximum of the posterior distribution (which is often not tractable analytically) or estimating the mean of the posterior distribution can be computationally intensive and sometimes intractable. Gibbs sampling is a powerful computational approach that addresses this by generating samples from the desired posterior distribution iteratively.

**Process of Gibbs Sampling for Hyperparameter Estimation:** 1. **Initialization:** The model's parameters, including the hyperparameters, are initialized with plausible values. 2. **Iteration:** In each iteration of the sampling process: \* One parameter (or a block of parameters) is sampled from its conditional posterior distribution, given the current values of all other parameters and the data. \* This process is repeated many times (thousands or millions of iterations). 3. **Convergence Assessment:** The algorithm is run for a sufficient number of iterations, allowing the generated samples to converge to the true posterior distribution. 4. **Summary Statistics:** After the sampling process has reached convergence (or after a burn-in period to discard initial samples that may not be representative), the values of the hyperparameters from the thousands of converged samples are collected and summarized.

**Example of Hyperparameter Estimation in SBayesR (from tutorial):** In the provided SBayesR tutorial, hyperparameters are estimated through the LDstep4 function call. While the exact parameters passed to LDstep4 for this purpose are not explicitly listed as hyperparameters in the text, the context suggests that ld.block, log2file, and outPrefix would be key parameters controlling the LD matrix generation process.

```
Example of calling LDstep4 for hyperparameter estimation (in a broader script co
#!/bin/bash
... (preprocessing steps)

Get genotype data into LD folder
genotype="your_genotype_data"
block_prefix="ldm"
ldm_folder="ldm"
... (other variables like ldm_radius, genotype_type)

LDstep4 --ldm ${ldm_folder} \
--gmdir ${genotype} \
--block ${block_prefix} \
--out ${ldm_folder} \
--log2file

... (subsequent steps)
```

**Important Considerations for Hyperparameter Estimation:** \* **LD Reference Panel:** The

choice of LD reference panel (e.g., 1000 Genomes Project, UK Biobank) significantly influences the estimated hyperparameters and the overall SBayesR model performance. This is because the LD structure of the reference panel reflects the population's genetic history. \* **Ancestry:** SBayesR explicitly states that PRS developed using summary statistics from one ancestry cannot be portably applied across multiple ancestries without careful consideration and re-tuning (i.e., estimating hyperparameters anew) in the target population. \* **Computational Demands:** The process of estimating hyperparameters via Gibbs sampling can be computationally intensive and time-consuming, often requiring significant resources and careful tuning of the sampling parameters themselves.

Source: <https://cnsgenomics.com/software/gctb/#Genome-wideFine-mappinganalysis>

**Q14: What kind of priors are used in SBayesR?** SBayesR utilizes **variant-specific priors** for SNP effect sizes. This is a distinguishing feature compared to some other summary-data based Bayesian methods (like SBayesC, SBayesS, or SBayesS-strat, which use mixture distributions with a constant variance for all variants). In SBayesR, the prior distribution for each individual SNP effect  $\beta_j$  is defined as:

$$\beta_j \sim N(0, \gamma_j \sigma^2 \beta)$$

where: \*  $\gamma_j$  is a value derived from the annotation (a random variable drawn from a Gamma distribution). \*  $\sigma^2 \beta$  is a common scaling factor for the variance of SNP effects.

By allowing the prior distribution to vary across SNPs based on their functional annotations ( $\gamma_j$ ), SBayesR can adapt its shrinkage intensity and estimate the true effect sizes more effectively, leveraging biological knowledge about which types of variants are more likely to be causal or have larger effects. This flexible prior is key to its performance in capturing complex genetic architectures. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

**Q15: Does SBayesR assume LD independence?** No, SBayesR, being a summary-data based Bayesian multiple regression method, explicitly accounts for linkage disequilibrium (LD) by incorporating an LD correlation matrix into its model. This is a fundamental aspect of its design, allowing it to disentangle genetic effects across correlated variants and improve prediction accuracy beyond simpler methods that ignore LD.

**Q16: How does SBayesR model LD?** SBayesR, being a summary-data-based Bayesian polygenic prediction method, explicitly models linkage disequilibrium (LD) between genetic variants. This is a critical step in its ability to infer joint SNP effects accurately, as it accounts for the non-random association of alleles.

**How SBayesR Models LD:** SBayesR models LD by fitting a **random effects component** within its underlying Bayesian hierarchical model. This component captures the correlation structure among individual SNPs based on their physical position on the chromosome. Importantly, this modeling approach is flexible and can accommodate various forms of LD, including:

1. **Block-wise LD:** The method can explicitly account for LD within defined genomic blocks (e.g., using a shrunk LD matrix, which is an input format for SBayesR).
2. **Sliding Window LD:** Alternatively, SBayesR can model LD using a sliding window approach, where LD correlations are estimated between pairs of SNPs within a moving window (a default parameter of 1000 SNPs).

**Why this is important:** Polygenic prediction heavily relies on accurately capturing the LD structure. Genetic variants that are in strong LD with each other carry similar information about the underlying causal variants. If LD is not properly accounted for, SNP effects can be misestimated, leading to inaccurate polygenic risk scores. SBayesR's explicit modeling of LD allows it to disentangle overlapping signals and more accurately infer the true contributions of individual genetic variants to a complex trait.

**Input for LD Modeling:** SBayesR requires an LD reference panel (provided via `--ldm-eigen`) to capture the LD patterns in the population being studied. This reference panel should ideally be composed of individuals of similar ancestry to the GWAS summary statistics and the target cohort for which the PRS is being developed.

#### Command-line example (general run that relies on LD modeling):

```
gctb --sbayes R \
--ldm ./ldm/sparse/chr22/1000G_eur_chr22.ldm.sparse \
--pi 0.95,0.02,0.02,0.01 \
--gamma 0.0,0.01,0.1,1 \
--gwas-summary ./ma/sim_1.ma \
--chain-length 10000 \
--burn-in 2000 \
--out sim_1
```

The modeling of LD is central to SBayesR's ability to produce robust and accurate polygenic prediction results by effectively integrating information from highly correlated genetic variants.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q17: What external annotations can be incorporated in SBayesR?** SBayesR, as a summary-data-based Bayesian polygenic prediction method, can incorporate external annotations to enrich its modeling capabilities and potentially improve prediction accuracy. The manual excerpt mentions several types of external annotations that can be integrated:

1. **Functional genomic annotations:** These annotations provide information about the biological function or genomic context of SNPs (e.g., whether a SNP is located in a gene, an enhancer, a regulatory region, etc.). Including functional annotations can help SBayesR prioritize or down-weight variants based on their likely biological relevance.
2. **Genotype-specific external annotations:** This refers to annotations that are specific to particular genotypes at certain SNPs (e.g., the impact of a specific allele combination). While not explicitly detailed how these are incorporated, they would likely be used in advanced models where genotype-specific effects are relevant.
3. **Predicted effect sizes of SNPs:** External effect size estimates, perhaps derived from other methods or large-scale meta-analyses, can be used to inform the prior distributions of SNP effects within SBayesR. This allows the model to leverage pre-existing knowledge about SNP effects.

The excerpt does not detail *how* these external annotations are incorporated (e.g., via specific parameters or input files), but implies they are designed to be compatible with SBayesR's summary-data framework. The integration is likely via the `--annot` parameter, similar to how functional annotations are provided in SBayesRC. Users would need to prepare these annotations in a suitable format that aligns with the SNP IDs and alleles in their GWAS summary data.

```
The exact command-line syntax for incorporating external annotations is not prov
However, based on the general SBayesR framework and the mention of 'predicted ef
it is likely via a parameter like --external_effects_file <file>.
The file format would need to be specified in the manual (e.g., SNP ID, Effect A
SBayesRC example of annotation file format (conceptual):
SNP Intercept Anno1 Anno2
rs1001 1 0 0.2
rs1002 1 1 -0.1
```

While the manual excerpt confirms the possibility of incorporating these, it does not provide the specific commands or detailed instructions for their exact integration.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q18: Does SBayesR implement a Gibbs sampler?** Yes, SBayesR implements a Gibbs sampler. The manual excerpt explicitly states that SBayesR ("SBayesR is a method implemented in GCTB to perform Bayesian polygenic risk score (PRS) analysis using summary statistics from genome-wide association studies (GWAS) and linkage disequilibrium (LD) information from a reference sample") employs a Gibbs sampler for its core computations.

Specifically, it mentions that "The REML step is carried out using an approximate MME-based AI-REML method, which is computationally efficient, and the sampling step is carried out using a Gibbs sampler." This indicates that while the Minimum Summary Statistics (MME) step might be computationally efficient, the subsequent and often more computationally intensive step of sampling effect sizes from their posterior distributions is indeed performed using a Gibbs sampler.

The Gibbs sampler is a Markov Chain Monte Carlo (MCMC) method commonly used in Bayesian statistics to sample from complex posterior distributions when direct integration is infeasible. By iteratively sampling from the conditional posterior distribution of each parameter, given the current values of the other parameters and observed data, SBayesR can approximate the true posterior distribution of SNP effect sizes. This sampled data then provides the basis for estimating SNP weights and calculating polygenic risk scores.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q19: Does SBayesR use a mixture model?** Yes, SBayesR explicitly utilizes a mixture model as part of its Bayesian framework for polygenic prediction. This mixture model is central to its ability to categorize SNPs into different effect size distributions and explain the observed genetic architecture. It is composed of three components, with SNP effects drawn from a different distribution for each component:

1. **Component 1: Zero-effect SNPs.** This is the largest and predominant component, representing SNPs with no true effect on the trait. The proportion of causal variants in this component is driven by the  $\pi_0$  (point mass) hyperparameter.
2. **Component 2: Small-effect SNPs.** This component models SNPs with modest effects on the trait. Their effect sizes are drawn from a scaled Generalized Inverse Gaussian (GIG) distribution, which is designed to produce heavy tails. This allows for a flexible modeling of genetic architecture where a few loci might have relatively small but non-zero effects.
3. **Component 3: Large-effect SNPs.** This component models SNPs with substantial effects on the trait. Similar to Component 2, their effect sizes are also drawn from a scaled Generalized Inverse Gaussian (GIG) distribution, permitting heavy tails and enabling the model to capture stronger influences.

The proportion of causal variants within each of these components is a random quantity that the model estimates from the data. This 'spike-and-slab' type of mixture model allows SBayesR to effectively perform variable selection (identifying which SNPs are likely to have non-zero effects) while simultaneously estimating their effect sizes, making it well-suited for complex genetic architectures characterized by many small effects and a few larger ones, as often observed in real-world trait data. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

**Q20: What regularization (if any) is applied in SBayesR?** I do not have information regarding the application of regularization in SBayesR. The manual excerpt does not detail any specific regularization techniques used by the SBayesR model beyond the implicit shrinkage resulting from its Bayesian framework and the input of summary statistics.

There is no specific command or parameter for enabling or disabling regularization, as it appears to be an inherent aspect of the model's formulation.

Output: -

**Q21: What programming language is required to run SBayesR?** The manual excerpt explicitly states that SBayesR (the summary data tool) is a standalone R function. Therefore, to run SBayesR, the primary programming language required is **R**.

#### Programming Language: \* R

**Conceptual Prerequisite:** Before you can call and execute the SBayesR function, your R environment must be properly set up and configured. This includes:

1. **Having R Installed:** You need to have the R statistical software installed on your system. It's recommended to use a recent stable version of R (as noted, the container image uses R version 3.5.1).
2. **Installing R Packages:** The SBayesR function, as provided as part of the GCTB package, needs to be installed within your R environment. The manual provides a clear R command for this: `r install.packages("GCTB")`. This command, when executed in an R console, will download and install the GCTB package, including its functions, such as SBayesR.
3. **Accessing the Function:** Once GCTB is installed, the SBayesR function becomes available for use as a standard R function. You would typically load the package first (e.g., `library(GCTB)`) and then call the `sbayesr` function with appropriate arguments.

**Why R?** While other languages (like Python) are also powerful for genomic analysis, R is frequently chosen for statistical genetics tasks due to its strong ecosystem of packages in this domain, its ease of use for data manipulation and visualization, and its robust statistical modeling capabilities. SBayesR leverages R's capabilities for implementing and executing its Bayesian methods for polygenic prediction.

To ensure you can run SBayesR, your computational environment should include a working R setup with the GCTB package installed.

Source: <https://cnsgenomics.com/software/gctb/#Download>

**Q22: What dependencies are needed to install SBayesR?** To install the SBayesR R package, specific R dependencies must be met first. The manual explicitly states that to run the SBayesR package, 'R (>=3.5.0)' and 'gtools' package must be installed. While other packages like 'permutations' (installed via 'devtools') are also required for specific functionalities, these two are fundamental and implicitly assumed to be available when the top-level SBayesR package is installed. Ensuring these prerequisites are in place before attempting to install or use SBayesR is crucial for successful operation. Source: <https://github.com/zhilizheng/SBayesRC>

**Q23: How is SBayesR installed?** The manual excerpt indicates that SBayesRC is a part of the GCTB software family and lists a GitHub repository as the source for obtaining the software. However, the provided text does not offer specific installation instructions, such as providing a downloadable executable file or a git clone command for SBayesR itself. The details for installing SBayesR are not available in this excerpt.

**Q24: Are there Docker or Conda versions of SBayesR?** No, the manual excerpt explicitly states that there are no Docker or Conda versions of SBayesR listed. The available installation methods are:

1. **Using PRSice-2 (a binary file):** This is recommended for most users as it is the most user-friendly and tested method.
2. **Using PLINK2:** This is an option for Linux or Mac users who prefer to compile from source or manage their software through standard package managers like Debian.
3. **Using LDpred:** This is specifically for Linux or Mac users who want to manage their software through pip (Python package installer).
4. **Practical R worked example:** This provides a guide for users who prefer to work with R.

The absence of Docker or Conda methods means that users cannot rely on pre-packaged, ready-to-run instances of SBayesR. They must download and install the software themselves using one of the provided methods or alternative tools like GCTA (which does have Docker and Conda versions, but SBayesR itself does not).

The manual excerpt: "We do not provide docker or conda version of SBayesR. Currently we only provide binary file for Linux and Mac (Linux version is recommended) and source code for other operating system."

This is important because it sets the expectation for users and avoids misleading information by not showing non-existent options.

Source: <https://github.com/zhilizheng/SBayesRC>

**Q25: Can SBayesR be installed from source?** No, SBayesR (as described in the provided text) is not a tool for which source code can be downloaded or installed. The text explicitly states: "This is the online manual for the Shiny application SBayesR." It provides a URL <http://cnsgenomics.com/shiny/LMOR/>, which points to an online web application, not a downloadable source code package like a ZIP file. This indicates it's a web-based tool run through an R Shiny app, not a local software installation.

Therefore, you cannot install SBayesR from source as you would a traditional software package. You would access its functionality by navigating to its URL in a web browser.

Source: <http://cnsgenomics.com/shiny/LMOR/>

**Q26: Are there platform restrictions for SBayesR?** The text states that SBayesR is implemented in the 'GCTB Software', which has 'command line interfaces available for Linux, Mac OS and Windows'. This indicates that while a command-line interface is primary, there are no explicit restrictions to specific operating systems for running SBayesR itself. However, to use it effectively, the user would need to be on a system where a compatible C++ compiler and necessary libraries are available if custom compilation is chosen. Otherwise, pre-compiled binaries or Docker containers are the typical ways to access GCTB and thus SBayesR. Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q27: What version of Python/R is required for SBayesR?** The provided manual excerpt details the 'SBayesR' Shiny application, which is a graphical user interface (GUI) tool for polygenic risk score (PRS) analysis and does not involve any command-line Python or R scripts for its operation. Therefore, no information regarding Python/R version requirements is present in this text.

**Q28: What input format is required for genotype data in SBayesR?** For genotype data input in SBayesR (as part of the GCTB software suite), the **BGEN format** is explicitly mentioned as the preferred and supported input format for GWAS summary-data based analyses.

#### Genotype File Format:

- **BGEN Format:** The BGEN file format is a compressed binary genotype file format designed to store large-scale genotype data efficiently. It is particularly well-suited for imputed genotype datasets, capable of storing probability distributions for each genotype (e.g., PL fields).

#### Requirement:

- The manual states: "We have also added a feature to handle bgen files as input format for the summary-data based methods, such as BayesR, SBayesR and SBayesRC."

#### Usage in SBayesR / GCTB:

When specifying the genotype file for your SBayesR analysis, you would use the `--bgen` option followed by the path to your BGEN file.

#### Command-line example snippet:

```

gctb --sbayes R \
--ldm path/to/ldm/file.ldm.sparse \
--pi 0.95,0.02,0.02,0.01 \
--gamma 0,0,0.01,0.1,1 \
--gwas-summary path/to/gwas_summary.ma \
--chain-length 10000 \
--burn-in 2000 \
--out my_sbayesr_analysis

```

#### Parameter for genotype file:

- **--bgen <file\_path>**: This parameter specifies the path to the BGEN genotype file. This file typically contains the imputed genotypes for the individuals in your reference or target cohort.

#### Benefits of BGEN format:

- **Efficiency**: BGEN is a binary format, which makes it highly efficient for storage and rapid loading by tools like GCTB.
- **Imputed Data Handling**: It is particularly well-suited for storing imputed genotype data, which often comes in BGEN format from large-scale imputation pipelines.

While `--bgen` is mentioned for BayesR/SBayesR/SBayesRC, other GCTB methods might also support this format, making it a universal choice for genotype input across many GCTB functionalities.

Source: <https://cnsgenomics.com/software/gctb/>

**Q29: What is the expected format of summary statistics for SBayesR?** The manual excerpt for SBayesR does not explicitly specify the exact column-by-column format or header requirements for summary statistics files. Instead, it lists several key fields like allele frequency (`freq`), estimated effect (`beta`), standard error (`se`), and sample size (`N`). It also mentions that headers are 'not keywords' and will be omitted by the program.

**Inferred Format:** A well-formatted summary statistics file for SBayesR is typically a delimited text file (e.g., tab-separated) with a header row. The crucial aspect is that the columns corresponding to `freq`, `beta`, `se`, `P`, and `N` are correctly identified and contain the relevant data. The excerpt also mentions `K` as an optional column for per-SNP sample size, implying that if present, it should be a numeric value.

Column (inferred)	Purpose
SNP	SNP identifier (e.g., rsID)
CHR	Chromosome number
BP	Base pair position
A1	Effect allele (reference allele)
A2	Other allele (alternative allele)
freq	Allele frequency of A1 (A2's frequency = 1 - freq)
beta	Estimated effect size (log(odds ratio) or beta coefficient)
se	Standard error of the effect size
P	P-value for the association test
N	Sample size for the association test
K	Optional: Per-SNP sample size

#### Example (inferred, not an official format specification from excerpt):

SNP	CHR	BP	A1	A2	freq	beta	se	P	N	K
rs12345	1	100000	A	G	0.25	0.012	0.005	1.0e-5	100000	98000
rs67890	2	250000	C	T	0.10	-0.008	0.004	5.0e-4	100000	99000

#### Command Example (using such a file):

```

gctb --sbayes R \
--ldm path/to/ldm/file.ldm.sparse \
--gwas-summary path/to/my_gwas_summary.txt \
--out my_sbayesr_analysis

```

While `freq`, `beta`, `se`, and `P` are explicitly mentioned, ensuring the presence of `N` (or `K`) and its correct data type is important, especially given its use in `--impute-n`.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

**Q30: Can SBayesR take imputed genotype data?** Yes, SBayesR can accept imputed genotype data. The application's workflow is designed to accommodate various genotype formats, including VCF (both compressed and uncompressed) and dosage format files. The manual excerpt explicitly states that the first step of the SBayesR workflow is to "transform the GWAS summary statistics to the GLMM summary statistics format or directly input the imputed genotype data."

While the example0.sh script provided demonstrates a specific workflow that involves downloading a pre-prepared 1000 Genomes Project (1 KG) .pgen file and then using PLINK2 for conversion, this does not mean SBayesR itself directly processes imputed data. Instead, it implies that the input to SBayesR's primary processing step (the imputation of summary statistics) can originate from imputed genotype data sources.

**Evidence from Manual:** \* "Transform GWAS summary statistics to the GLMM summary statistics format or directly input the imputed genotype data" (Emphasis added). \* "The first step is to transform the GWAS summary statistics to the GLMM summary statistics format or directly input the imputed genotype data into the application."

This capability is crucial because many modern large-scale genomic analyses rely on imputed genotype data, which provides higher resolution and coverage compared to directly genotyped arrays. By accepting imputed data, SBayesR can be integrated into comprehensive pipelines that leverage imputation technologies.

Source: <http://cnsgenomics.com/shiny/LMOR/>

**Q31: What file format is used for LD reference panels in SBayesR?** For LD reference panels used in SBayesR (and similar tools like PRSice-2), the recommended and widely used format is **PLINK binary format**.

**Explanation:** PLINK binary format consists of three files with a common prefix: 1. `.bed` file: Contains the compressed binary genotype data. 2. `.bim` file: Contains variant information (chromosome, SNP ID, genetic distance, base-pair position, alleles). 3. `.fam` file: Contains sample information (family ID, individual ID, paternal ID, maternal ID, sex, phenotype).

This format is efficient for storing large-scale genotype data and is commonly used in various genomic analyses, including the calculation and utilization of LD information for methods like polygenic risk score calculation.

**Example File Naming:** If your LD reference panel consists of `my_ref.bed`, `my_ref.bim`, and `my_ref.fam`, the prefix would be `my_ref`.

**How SBayesR Uses It:** SBayesR (via its `--ldm-eigen` parameter) requires access to this type of LD reference data to accurately model linkage disequilibrium and derive the necessary eigen-decomposition results for polygenic prediction.

**Example Command Showing Usage:**

```
gctb --ldm-eigen my_ld_ref_data --gwas-summary my_gwas.ma --sbayes RC --out my_sba
```

In this example, `my_ld_ref_data` refers to the prefix of the PLINK binary files `my_ld_ref_data.bed`, `my_ld_ref_data.bim`, and `my_ld_ref_data.fam`.

**Q32: Does SBayesR output effect sizes per SNP?** No, SBayesR itself outputs joint estimates of SNP effects, which are typically not at the individual SNP level but rather in a 'mix' of alleles. While the individual files for SNPs are available, the core model output is a set of weights that are then applied across all SNPs in the target dataset.

**Q33: What output file formats are generated by SBayesR?** Upon successful completion of its analysis, the SBayesR application (as demonstrated in the tutorial) generates two primary output files, both prefixed with the name you provide via the `--out` parameter.

1. `<output_prefix>.prs` (**Polygenic Risk Score File**):

- **Content:** This file contains the calculated polygenic risk scores for each individual in your validation genotype dataset. It typically includes individual IDs and their corresponding PRS values.
- **Purpose:** These scores are the main output of interest for downstream applications, such as disease risk prediction, genetic stratification, or simply identifying individuals with higher genetic predisposition to the trait.

2. `<output_prefix>.log` (**Log File**):

- **Content:** This file contains a record of the SBayesR application's run, including messages, parameters used, and potentially details about the MCMC sampling process (though specific MCMC output isn't shown in the text, log files often capture such information).
- **Purpose:** Primarily for diagnostic and troubleshooting purposes. It helps users verify the success of the run, understand issues that might have occurred during execution, or confirm that the expected output was generated.

**Example Output (assuming `--out my_analysis` was used):** If you run `gctb --sbayes R ... --out my_analysis`, you would find:

- `my_analysis.prs`
- `my_analysis.log`

These files are saved in the directory specified by your `--out` prefix or in the current working directory if no path is given. The `.log` file is particularly useful for debugging any issues that might arise during the analysis.

Source: <https://cnsgenomics.com/software/gctb/>

**Q34: Is there support for multiple chromosomes in SBayesR?** The provided manual excerpt details the 'SBayesR' module, which implements summary-data based Bayesian alphabet models for genotype-augmented prediction. It does not contain any information regarding support for multiple chromosomes within the SBayesR functionality or any other modules.

**Q35: What is the default value for the LD window size in SBayesR?** The manual excerpt does not explicitly state a default value for the LD window size in SBayesR. It mentions the parameter `--ld-wind` for specifying this window, but does not provide a value when this parameter is omitted or if it defaults to some specific setting if not provided by the user. Therefore, a definitive answer cannot be provided based solely on this text.

In the context of the Gibbs sampler (which SBayes uses), LD pruning of variants is generally performed to improve mixing and reduce computational burden, but the optimal window size can depend on the population's LD structure and the specific trait. Without a specified value, it's not possible to report 'default'.

Source: <http://cnsgenomics.com/software/gctb/#Download>

**Q36: Can the number of MCMC iterations be set in SBayesR?** Based on the provided text, the number of Markov chain Monte Carlo (MCMC) iterations for model parameters can indeed be set in SBayesR. The `--chain-length` option is explicitly mentioned, allowing users to specify the total number of iterations for the MCMC algorithm.

**Parameter:** \* `--chain-length <value>` : \* **Description:** Specifies the number of iterations for

the Markov chain Monte Carlo (MCMC) sampling process used by SBayesR to sample from the posterior distribution of SNP effects and model parameters. \* **Type:** Integer \* **Default Value:** 21000 \* **Example:** To set the MCMC chain length to 10000 iterations:

```
```bash
gctb --sbayes R \ --ldm path/to/ldm.ldm.sparse \ --gwas-summary your_gwas.ma \ --chain-length
10000 \ --burn-in 2000 \ --out sbayesr_mcmc_length_10k ````
```

Importance: The number of iterations greatly affects the convergence and stability of the MCMC chain. A sufficient number of iterations is crucial for the results to be reliable. While the default might be adequate for many scenarios, for complex datasets or specific models, experimenting with different chain-length values (in conjunction with burn-in and out-freq) might be beneficial to ensure the chain has converged.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q37: Are there tunable parameters for SNP filtering in SBayesR? The text states that user-defined parameters are "not used" for the --info filter in GCTB, only a fixed 0.3 threshold is applied. Therefore, there are no tunable parameters for SNP filtering based on INFO score within the provided SBayesR section.

Q38: What configuration options are available in SBayesR? The SBayesR manual excerpt details various configuration options for the GCTB software, specifically focusing on the 'Bayesian alphabet' models and their runtime parameters. Here are the key options listed:

1. **--sbayes [R|S] : Type:** Flag. **Default:** None. This option specifies the Bayesian alphabet model to be used.
 - **--sbayes R:** Activates the SBayesR model, which is the individual-level data model and requires hard-coded LD matrices (--ldm).
 - **--sbayes S:** Activates the SBayesS model, which is similar to SBayesR but uses summary statistics and an LD correlation matrix (--ldm-sparse).
2. **--pi [comma_separated_values] : Type:** String. **Default:** 0.95,0.03,0.01,0.01 . This option sets the prior proportions of SNPs for different mixture components in the SBayesR model.
 - Example: --pi 0.9,0.05,0.03,0.02 (Note: The default sums to 1; you don't need to specify a fifth value if it's implicit).
3. **--gamma [comma_separated_values] : Type:** String. **Default:** 0.0,0.01,0.1,1 . These are the scaling factors for the variance-gamma mixture distribution, where each value corresponds to a pi component.
4. **--gwas-summary [file_path] : Type:** File path. **Default:** None (required). Specifies the input file containing GWAS summary statistics (.ma format).
5. **--ldm [prefix] : Type:** File prefix. **Default:** None (required for SBayesR). This option specifies the prefix for the pre-computed LD matrix files (e.g., ldm will look for ldm.ldm.full.info , ldm.ldm.full.bin etc.).
6. **--ldm-sparse [prefix] : Type:** File prefix. **Default:** None (required for SBayesS). Similar to --ldm , but for the sparse LD matrix used by SBayesS.
7. **--chain-length [integer] : Type:** Integer. **Default:** 21000 . Specifies the total number of MCMC iterations.
8. **--burn-in [integer] : Type:** Integer. **Default:** 1000 . Specifies the number of initial MCMC iterations to discard (burn-in period).
9. **--out[put] : Type:** String. **Default:** None (required). Specifies the root filename for all output files.

Additional General Options: * **--thread [integer] : Type:** Integer. **Default:** (Automatically detected). Specifies the number of threads to use for parallel computation. * **--twostep : Type:** Flag. **Default:** Disabled. Activates the two-step estimation procedure for genetic architecture parameters, as described in Zhu and Stephens (2017). * **--update-pi : Type:** Flag. **Default:** Disabled. Allows the pi parameter to be updated during the MCMC iterations. * **--fix-pi [comma_separated_values] : Type:** String. Used to fix specific pi values to predefined values, useful for sensitivity analyses or hypothesis testing about specific functional categories. * **--robust : Type:** Flag. **Default:** Disabled (enabled by default in GCTB 2.03 beta). Uses robust standardization of GWAS effect sizes for meta-analyzed data. * **--p-value [threshold] : Type:** Float. **Default:** 0.4 . Sets the p-value threshold for pruning and thresholding steps in SBayesR (if --impute-summary is used). * **--beta-standardization [standard | per_snp] : Type:** String. **Default:** standard (if --impute-summary is used, otherwise per SNP). Specifies whether imputed GWAS summary statistics should be standardized (standard) or one per SNP (per_snp).

Example of a typical SBayesR command combining these options:

```
gctb --sbayes R \
--ldm ldm_path/ldm_full \
--gwas-summary gwas_sumstats.ma \
--pi 0.95,0.

**Q39: Does SBayesR offer automatic parameter optimization?**
No, SBayesR (specifically, the GCTB software) does not offer automatic parameter optimization.

It presents various options like `--robust` (to estimate `a` and `b`), `--fix-a` (to fix the `a` parameter), and `--fix-b` (to fix the `b` parameter).

**Q40: How can the best model be selected in SBayesR?**
In SBayesR, the "best model" for polygenic prediction is typically defined as the model with the highest posterior probability given the observed data and priors.

**Strategy for Selecting the Best Model:**
1. **Run Multiple Models with Differing Priors:** Fit your SBayesR model(s) using different prior distributions and compare their performance metrics (e.g., AIC, BIC, log-likelihood) to select the best one.
```

```

*   **Example with `--pi`:** Run `sbayesr` with `--pi 0.95,0.03,0.01,0.01` (sp
*   **Example with `--gamma`:** Run `sbayesr` with `--gamma 0,0.01,0.1,1` and

2. **Prepare External Validation Data:** You need a separate dataset (individual-
   *   **Example Validation Set:** `val_genotype.bed/bim/fam` and `val_phenotype.

3. **Calculate Model Fit Metrics in Validation Set:** For each trained SBayesR mo
   *   **Command to generate PRS from a specific model:** `gctb --score test.snpR
   *   **Command to calculate r^2:** 
     ````bash
 # Assuming val_prs_modelX.snpProfile contains SNP IDs and PRS values
 # And val_phenotype.phen contains individual IDs and observed phenotypes
 # You would typically use a scripting language (Python, R) to join these a
 # A simplified example (conceptual):
 # python -c "import pandas as pd; df_prs = pd.read_csv('val_prs_modelX.snp
     ```

4. **Evaluate Model Performance:** Compare the calculated `r^2` (or other metri
   **Example Workflow:** 

     ````bash
 # 1. Fit multiple SBayesR models (e.g., with different pi configurations)
 gctb --sbayes R \
 --ldm ./ldm/sparse/chr22/1000G_eur_chr22.ldm.sparse \
 --pi 0.95,0.03,0.01,0.01 \
 --gamma 0.0,0.01,0.1,1 \
 --gwas-summary ../ma/sim_1.ma \
 --chain-length 10000 \
 --burn-in 2000 \
 --out sim_1_model_pi_gamma
 gctb --sbayes R \
 --ldm ./ldm/sparse/chr22/1000G_eur_chr22.ldm.sparse \
 --pi 0.99,0.01,0.00,0.00 \
 --gamma 0.0,0.01,0.1,1 \
 --gwas-summary ../ma/sim_1.ma \
 --chain-length 10

 Q41: How is prediction accuracy measured in SBayesR?
 SBayesR, being a summary-data-based method, cannot directly use individual-level p
 Measurement of Prediction Accuracy:
 1. **Observed SS (`test.ss`):** This file contains the summary statistics from th
 2. **Expected SS (`test.snpRes`):** This file contains the summary statistics fro
 By comparing the `A1`, `A2`, `Freq`, `BETA`, `SE`, and `P` columns between these t
 Example Scenario:
 If your `test.ss` file shows a mean P-value of 1e-05, and your `test.snpRes` (expe
 Command Example (Illustrative of using SS for validation):
     ````bash
     gctb --sbayes R \
       --ldm ./ldm/sparse/chr22/1000G_eur_chr22.ldm.sparse \
       --pi 0.95,0.02,0.02,0.01 \
       --gamma 0.0,0.01,0.1,1 \
       --gwas-summary ../ma/sim_1.ma \
       --out sim_1

```

- The output `sim_1.snpRes` file will be one of the inputs for subsequent SS comparison.

Manual Section: - SBayes R tutorial mentions that the AUC for the PGS and the AUC for the true causal variants were 0.32 and 0.37, respectively, in a simulation study, illustrating the method's practical application for measuring accuracy.

Q42: What evaluation metrics does SBayesR support (e.g., R², AUC)? The manual excerpt explicitly mentions the 'Area Under the Curve' (AUC) as a metric for evaluating binary outcomes using PRS. It also states that the 'R² value of the model' is provided when 'Regression summary' is displayed for quantitative traits. However, it does not provide a comprehensive list of all evaluation metrics SBayesR supports for different outcome types (e.g., continuous, binary).

Therefore, based on the given text, SBayesR supports:

- Area Under the Curve (AUC):** A common metric for evaluating the performance of binary classification models (e.g., predicting disease status).
- R² (Coefficient of Determination):** A widely used metric for quantitative traits, indicating the proportion of variance in the phenotype explained by the PRS.

For quantitative traits, the manual states that 'The application also provides the R² value of the model', suggesting that R² is a directly reported metric. While R² is a general measure of explained variance, its specific application might be contextual (e.g., total variance explained by PRS vs. total variance explained by PRS + covariates).

Detailed Explanation: - AUC: AUC is a measure of a model's discriminative ability. For binary outcomes, a higher AUC (closer to 1) indicates better separation between the two classes (e.g., better ability to distinguish cases from controls). It's often interpreted as the probability that a randomly selected positive instance has a higher predicted score than a randomly selected negative instance. - **R²:** R² is a measure of how well the observed outcome variable can be predicted from the PRS. For quantitative traits, a higher R² (closer to 1) means the PRS explains a larger proportion of the variability in the trait. It's a key indicator of the predictive power of the

PRS.

Examples: - Binary Outcome (Disease Status): If you run a PRS analysis in SBayesR for a disease, and the summary panel shows 'AUC: 0.65', this means the PRS, at its best, correctly differentiates between cases and controls 65% of the time. - **Quantitative Trait (Height):** If the summary panel reports 'R²: 0.12', this means the PRS accounts for 12% of the total variation in height in the sample.

While SBayesR provides these key metrics, a full suite of evaluation metrics for various outcome types (e.g., precision-recall, F1-score, Cohen's Kappa for binary, coefficient of correlation for continuous) is not detailed in the provided excerpt. Users might need to infer or supplement these with other tools or more detailed documentation.

This excerpt details common metrics for binary outcomes and a specific one (R²) for quantitative traits, but does not offer a comprehensive list of all metrics available.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q43: Can cross-validation be performed in SBayesR? No, the manual excerpt does not explicitly mention whether cross-validation can be performed within the SBayesR module or tool. The term 'cross-validation' is not listed under the functionalities or methods described for SBayesR. SBayesR is primarily presented as a method for estimating SNP effects and heritability from summary statistics.

While SBayesR itself doesn't detail a built-in cross-validation procedure, the general workflow of SBayesR often involves out-of-sample prediction (e.g., calculating polygenic risk scores in an independent target dataset using weights estimated by SBayesR). If cross-validation is a required step for model validation or parameter tuning for the PRS being derived from SBayesR outputs, it would likely be performed externally as part of a broader PRS validation strategy. This would typically involve using separate software tools for PRS calculation and evaluation, or custom scripting.

The text focuses on the core SBayesR method, summary statistics requirements, QC, and output generation. It does not provide commands or advice specifically for performing cross-validation with SBayesR's output.

To perform cross-validation, you would generally: 1. Run SBayesR to estimate SNP weights. 2. Use an external tool (e.g., PLINK, PRSice-2) to calculate PRS on a subset of your data (e.g., validation set). 3. Evaluate PRS performance on the complementary subset (e.g., test set) using external metrics and tools. 4. Repeat the process, potentially iterating on hyperparameters or models.

This is a general approach and not specific to SBayesR's functionality.

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q44: Can SBayesR output p-values? No, the SBayesR tool (SBayesRC) is designed to provide SNP effect estimates (beta values) and polygenic risk scores (PRS), not to directly output p-values for individual SNPs from a GWAS. Its purpose is to estimate joint effects from summary data. While the individual SNP p-values are implicitly used in deriving these effects, SBayesR's primary output is tailored for calculating PRSs.

If you need to perform multiple testing correction or obtain significance thresholds for individual SNPs from your summary statistics, you would typically apply a general statistical method (like Bonferroni correction, false discovery rate, or a genome-wide significance threshold) before using SBayesR to calculate PRSs based on these adjusted p-values.

SBayesR is a 'post-GWAS' tool that processes already-computed effect sizes, not the raw GWAS summary statistics themselves.

Source: <https://github.com/zhilizheng/SBayesRC>

Q45: How does SBayesR compare with LDpred2? The provided manual excerpt explicitly mentions SBayesR as part of the GCTB software suite, which is also mentioned in the context of PRS analyses. However, it does not provide any specific information or instructions on how SBayesR compares or interacts with LDpred2. Therefore, the requested comparison cannot be generated from this text.

Q46: How scalable is SBayesR with increasing SNP count? SBayesR, being an analytical tool, is highly scalable with increasing SNP count. Its performance primarily depends on the efficiency of the initial GWAS summary statistics generation and post-processing steps, rather than a fixed computational burden per SNP. The manual's emphasis on 'not requiring post hoc shrinkage of marker effects' also suggests a more efficient internal process for handling larger effect sizes or denser data. While the number of SNPs can influence memory requirements for certain matrix operations (like LD matrix computation), the core SBayesR algorithm is designed to handle large-scale genomic data by processing summary statistics efficiently, making it suitable for methods that incorporate millions of common variants. Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q47: Can SBayesR run on high-performance computing (HPC) clusters? Based on the manual excerpt, SBayesRC is described as a method for "polygenic prediction that incorporates functional genomic annotations" and relies on "pre-computed eigen-decomposition data of LD blocks." While the text does not explicitly mention HPC clusters, the nature of genomic analyses, particularly polygenic prediction for large datasets, heavily suggests that running SBayesRC on HPC clusters would be beneficial and likely supported by the tool's design for efficiency. Processing large-scale genomic datasets like the UK Biobank requires significant computational resources for memory, CPU power, and parallelization capabilities, which HPC clusters are designed to provide.

The manual states that SBayesRC requires "about 4GB of memory for each thread" and "about 30 minutes of computing time for each trait." These estimates, while substantial for individual runs, become manageable when distributed across multiple threads or cores on an HPC cluster, allowing for efficient parallel processing of multiple traits or chromosomes.

While the manual does not provide specific commands for running SBayesRC directly on HPC clusters (e.g., bsub -S, salmonen -q), the implied benefits of its memory management and parallelization potential strongly point to HPC as an appropriate and efficient environment for executing SBayesRC analyses.

Source: <https://github.com/zhilizheng/SBayesRC>

Q48: What memory requirements are typical for SBayesR? SBayesR has significant memory requirements, especially for analyses involving high-density SNP data and large sample sizes. The manual explicitly notes that 'memory usage can be a problem for large scale analysis.' While the exact capacity isn't specified as 'large,' the context of 'high density SNPs (> 7 millions)' and 'large sample size (> 4000 individuals)' directly implies that GBs (gigabytes) of RAM are commonly necessary. For instance, if you are processing 10 million SNPs with double precision genotypes for 50,000 individuals, the genotype matrix itself could easily consume 100 GB of RAM before adding other associated data (like annotations, LD information, and output files). Therefore, users should provision computing resources with ample memory when planning SBayesR analyses, especially for whole-genome or biobank-scale datasets. The use of distributed computing platforms or memory-optimized virtual machines is often advisable to meet these demands. Source: <https://cnsgenomics.com/software/gctb/#Download>

Q49: Is multi-threading supported in SBayesR? Based on the provided text, there is no explicit mention of multi-threading support within the SBayesR module itself. The 'News' section mentions a 'Multi-thread Computing' update for GWFM in version 2.03 (Oct 2024), but this is specific to Genome-Wide Fine-Mapping (GWFM) and not explicitly stated for SBayesR's core computations. While the R package's general 'Updates' section lists 'Improve computing speed by multi-thread' for PRSice, this is a feature of PRSice-2, not SBayesR.

Therefore, based *only* on the provided text, SBayesR is not explicitly mentioned to support multi-threading for its primary computations.

Q50: Can SBayesR handle millions of SNPs? Yes, SBayesR is designed to handle large-scale genomic data, specifically summary statistics from genome-wide association studies (GWAS). The manual excerpt states that the tutorial data consists of 723,454 SNPs and emphasizes the necessity of pre-filtering SNPs based on quality control criteria such as imputation INFO scores and MAF. While the text doesn't explicitly state 'millions of SNPs' in a generic context for SBayesR's capability, the examples show it works with `sbcrcore.R` which processes aggregated summary statistics, implying it's capable of processing very large SNP counts, though likely not *billions* in a single run without parallelization. The key is that the input should be well-prepared and filtered to manage the load effectively.

For GWAS results, the `ICD10.Disease.GWASSummary` dataset contains 2,989,750 SNPs, underscoring the tool's capacity to handle very large SNP counts.

To process such data efficiently, it's crucial to apply filters during the preparation step, as demonstrated in the tutorial:

```
# Example of filtering for large-scale data processing
mkdir -p ldm/snp_filter
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --impute-summary --out test --

# This imputation step can be time-consuming for millions of SNPs,
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --impute-summary --out test -
# For very large datasets, consider parallelizing or optimizing the imputation st
```

SBayesR's effectiveness also depends on how the GWAS summary statistics are prepared and pre-processed, which often involves filtering out low-quality or irrelevant SNPs before being fed into the SBayesR model.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q51: Can SBayesR be used with PLINK files? No, SBayesR is not designed to directly take PLINK binary files (BED, BIM, FAM) as input for its core polygenic risk score calculation. Instead, the manual explicitly states that the GWAS summary statistics required by SBayesR must be in the `GCTA-COJO .ma` format.

SBayesR's Input Requirement: * `GCTA-COJO .ma` format: This is a specific tab-delimited text file format with a header row defining the columns. * Columns include SNP, A1 (effect allele), A2 (other allele), freq (frequency of A1), b (marginal effect size), se (standard error of marginal effect size), p (p-value), and N (sample size). * It is crucial that A1 represents the effect allele, and A2 is the alternative allele.

PLINK Binary Files (BED/BIM/FAM): * PLINK binary files are a compact and efficient way to store genotype data. They consist of three files: `.bed` (binary genotype data), `.bim` (SNP information), and `.fam` (sample information). * While PLINK is a powerful tool for manipulating and processing raw genotype data, it is not designed to output summary statistics in the `.ma` format that SBayesR expects. The `.ma` file is typically generated from the results of a GWAS (e.g., using PLINK's `--linear` or `--logistic` commands) or other summary statistics generation tools.

Workflow Implication: If your GWAS results are in PLINK binary format, you would need to convert them to the `GCTA-COJO .ma` format *before* inputting them into the SBayesR Shiny app. This conversion step is typically performed using external tools or custom scripts to reformat the data and calculate the necessary allele frequencies and sample sizes from the PLINK `.bim` and `.fam` files, aligning them with the `A1`, `A2`, `freq`, `b`, `se`, `p`, `N` structure required by SBayesR.

The SBayesR app emphasizes the `.ma` format because it is optimized for direct use in polygenic risk score calculations, ensuring consistency in allele definition and frequency, and correct handling of effect sizes and p-values.

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q52: Is SBayesR compatible with the UK Biobank data format? Yes, SBayesR is compatible with the UK Biobank data format. The GCTB manual explicitly states that the example usage demonstrates input from "the UK Biobank data format" for the 2.04 million common SNPs.

This implies that the input files for SBayesR (LD matrix, GWAS summary statistics) should be prepared in a way that GCTB can correctly read and interpret them, generally aligning with the conventions of the BGEM format (often referenced for UK Biobank data) and PLINK's binary formats (`.bed`, `.bim`, `.fam`).

Example of files implying compatibility: * `ldm/eur_w_ldm.bbin`: A binary LD matrix file, consistent with PLINK `.bed` format (though technically not identical, the name implies similarity and compatibility for input). * `gwas_data/height_ukb_afr_sumstats.txt`: This text file, while simplified, represents a common GWAS summary statistic format that GCTB can parse, often being

a preprocessing step or an example of data that would be converted to a more robust format (like BGEN) for full compatibility and performance.

Command-line example (illustrating use with compatible files):

```
gctb --ldm-eigen ldm/eur_w_ldm \
--gwas-summary gwas_data/height_ukb_afr_sumstats.txt \
--sbayes RC \
--annot annot.txt \
--out my_sbayer_analysis
```

This compatibility allows users to directly use large-scale, publicly available UK Biobank datasets for their SBayesR analyses after appropriate preprocessing to convert the raw BGEN data into GCTB-compatible formats (like .bim files or simplified text sumstats).

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q53: Can SBayesR be integrated with Hail? Yes, SBayesR can be integrated with Hail. The text states that for some of the best practices and details, users are redirected to the 'Hail Tutorial' and 'User Guide'. While the direct command-line example for integration is not provided in the excerpt, the mention of Hail as a supported platform indicates that such integration is possible and recommended for comprehensive analysis workflows.

Q54: Does SBayesR support BGEN or VCF files? Based on the provided manual excerpt, SBayesR itself (as an extension to GCTB) primarily supports GWAS summary statistics in various text-based formats (plain text, compressed plain text, compressed text with bgzip). There is no explicit mention of direct support for BGEN (Binary Genotype File) or VCF (Variant Call Format) files for input.

However, the manual does state that GCTB (the overarching software) "can handle summary statistics from other imputation tools such as ACGT, Minimac and IMPUTE2" and that "the GCTB software has been modified to allow for direct handling of BGEN files when estimating the joint effect of all genetic variants by use of the Bayess model." (Referring to GCTB, not SBayesR specifically).

Implications for SBayesR: If your GWAS summary statistics are in BGEN or VCF format, you would typically need to convert them into one of the formats supported by SBayesR (e.g., .ma format) before running SBayesR. This conversion might involve using other tools or pre-processing steps.

Example (Conceptual): 1. Convert BGEN to VCF/Sample-wise Text: Use a tool like bcftools convert or a custom script to convert BGEN to a VCF or a sample-wise text format that can then be further processed into the .ma format. bash # Example: Convert a single BGEN file to a VCF file bcftools convert --tsv2vcf input.bgen > output.vcf # Then, format the VCF for SBayesR (e.g., add FREQ, INFO/DP columns)

- 1. Convert BGEN to Plain Text / compressed Text:** If you have individual-level data, you might first impute to PLINK format, then convert to the .ma format. bash # Example: Impute to PLINK format (if not already) gctb --impute-summary my_data.bgen --plink2-exe plink2 --out imputed_data # Then, convert to .ma format (if not already) gctb --gwas-summary imputed_data.imputed.ma --make-gwas-summary --out final_gwas_for_sbayer

So, while SBayesR itself doesn't directly consume BGEN/VCF, GCTB's capabilities, or external conversion tools, allow for seamless workflows that integrate these formats upstream of SBayesR's direct summary statistics input.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q55: Is SBayesR compatible with AnnoPred or PRScs? No, SBayesR is not compatible with AnnoPred or PRScs. The manual explicitly states: "SBayesR is a software package that implements Bayesian polygenic risk score (PRS) analysis using summary statistics from genome-wide association studies (GWAS) and an annotation-integrated functional genomic dictionary. It was originally developed as an extension of the individual-level data PRS method PRSice-2, but differs in that it does not require individual-level genotypes and instead relies on summary statistics. However, SBayesR is **incompatible** with AnnoPred or PRScs, and we do not provide support for installing or using those tools with GCTB. Please consult the respective software manuals for details on their specific commands and requirements." Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q56: Are the results from SBayesR interpretable? Yes, the results from SBayesR are interpretable. The summary statistics generated by SBayesR provide estimates of SNP effects, which are fundamental for building polygenic risk scores (PRS). While a higher 'VAI' (variance in effect size) indicates uncertainty, it does not necessarily mean the effect is uninterpretable. The estimated SNP effects themselves are the direct output that can be used to construct a PRS for individuals. The interpretability of the final PRS depends more on how well the SBayesR model fit the data and how robust the effect size estimates are, rather than just the 'VAI' value.

SBayesR is designed to provide human interpretable and biologically meaningful effect sizes by explicitly modeling polygenic architecture. It's expected that these estimated effects, when aggregated into a PRS, would be interpretable in terms of genetic risk for the trait of interest.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q57: Does SBayesR provide confidence intervals for PRS? The provided manual excerpt for SBayesR states that the application is designed to map "regression coefficients from a linear (mixed) model (LMM) based GWAS to the odds ratio from genome-wide association studies (GWAS) and then calculate the polygenic risk score (PRS)." It also mentions displaying the "Top LMM predictors" and their associated "Transformed effect sizes."

However, the excerpt **does not explicitly mention whether SBayesR provides confidence intervals** for the calculated polygenic risk scores or the transformed effect sizes. While the manual states that effects were "standardized to per-allele effect sizes (beta coefficients) and then transformed to odds ratios," it does not detail any output of confidence intervals (e.g., lower and upper bounds, or standard errors) for these scores.

SBayesR is described as a *GUI-based application* for transforming and calculating PRS, and for

plotting results. The richer statistical details, such as p-values or full regression models, seem to be handled by the underlying R functions (like `lmor_func.R` and `prslice.R`).

Therefore, based on the provided text, it is not explicitly stated that SBayesR outputs confidence intervals for PRS. You would need to further inspect the application's output or its underlying R functions to confirm this detail.

To verify if confidence intervals are part of the output, you would typically inspect the final tabular output generated by the SBayesR app or examine the return values of the `lmor_func.R` script when run locally with the `--summary` flag.

```
# The text does not explicitly state if SBayesR provides confidence intervals.  
# Output for this instruction: -
```

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q58: Are SNP-level contributions to PRS reported by SBayesR? No, the provided text explicitly states that SBayesR is a summary-data based method for polygenic prediction. It requires summary statistics from genome-wide association studies (GWAS), which typically consist of SNP IDs, alleles, effect sizes (beta values), and standard errors/p-values, but it does not mention reporting individual SNP-level contributions to the PRS. In fact, the claim that SBayesR is a summary-data based method is precisely why it can be applied to 'biobank-scale data without access to individual-level genotypes'. If it reported detailed SNP-level contributions, it would necessarily need to process and output individual genotype data, which is not consistent with its design. Therefore, based on the given manual excerpt, SBayesR's function is to derive overall model parameters and polygenic risk scores (at the individual or PRS-level) from the summarized GWAS results, not to provide a breakdown of individual SNP contributions. Source: <https://github.com/zhilizheng/SBayesRC>

Q59: Can results from SBayesR be visualized using built-in plots? No, the manual states that "Although it is possible to generate some plots directly from the interface, we do not recommend this as they are not interactive." For comprehensive visualization, users are advised to download the results and use external tools.

To visualize the downloaded results, you would typically use plotting libraries in R (e.g., `ggplot2`) or other bioinformatics visualization tools.

Example (conceptual R code for downloading and plotting):

```
# Download the results file (e.g., as generated by SBayesR in 'Basic usage' section)  
# (The exact file name might differ based on the SBayesR output structure, e.g., '  
download.file("http://cnsgenomics.com/shiny/LMOR/data/example2.txt", "my_prs_resul  
  
# Load the data  
prs_data <- read.delim("my_prs_results.txt", header = TRUE)  
  
# Example: Plotting SNP effects  
# Plot MAF against effect size (for A1 allele)  
ggplot(prs_data, aes(x = MAF, y = BETA)) +  
  geom_point() +  
  theme_minimal() +  
  labs(x = "Minor Allele Frequency", y = "Effect Size (log-odds)") +  
  scale_y_continuous(explicit.scale = TRUE)  
  
# Example: Plotting QQ plot for p-values  
qq_plot_data <- prs_data %>%  
  mutate(qt = qnorm(p_value)) # Assuming p_value column exists in the downloaded file  
ggplot(qq_plot_data, aes(x = qt, y = p_value)) +  
  geom_qq() +  
  theme_minimal() +  
  labs(x = "Quantile (standard normal)", y = "Observed P-value")
```

This approach allows for custom and interactive visualization outside the Shiny app's limited built-in capabilities.

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q60: Are there recommended visualization tools for SBayesR? The provided manual excerpt details the 'SBayesR' module's functionality within GCTB and does not include information on recommended visualization tools for SBayesR results. While such tools are often utilized in conjunction with analysis modules, the specific recommendations or functionalities of tools like 'ggplot2' (for R users) or 'Matlab' are not mentioned.

Therefore, based *only* on the provided text, there is no GCTB-specific recommendation for visualization tools for SBayesR outputs.

Output: -

Q61: How does SBayesR perform compared to PRScs? The manual excerpt explicitly states that "SBayesR is similar to PRScs" in its methodological approach (Bayesian sparse regression model with continuous shrinkage (CS) priors). While both methods are described as detailed in the manuscript by Zheng et al. (2024), the excerpt itself does not provide any direct comparison of their performance, accuracy, or specific implementations beyond their shared methodological foundation.

Therefore, based *only* on the provided manual excerpt, I cannot provide a comparison of SBayesR's performance versus PRScs. The excerpt does not contain any evaluation metrics, comparative results, or specific implementation details for PRScs that would allow for such a comparison.

Output: -

Q62: Can SBayesR be combined with other PRS tools? Yes, SBayesR is designed to be integrated and combined with other Polygenic Risk Score (PRS) tools. The manual states that SBayesR serves as a method to generate SNP weights, which are a fundamental input for PRS calculation using tools like PLINK and PRSice-2.

Specific Combinations Mentioned: * **SBayesR + PRSice-2:** This combination is explicitly mentioned in the tutorial example. SBayesR is used to obtain the SNP weights (.weightedEffects file), and then PRSice-2 is used to calculate the actual polygenic risk scores for individuals in a target cohort based on these weights and genotype data. * **SBayesR + PLINK:** Similarly, the tutorial notes that the generated .weightedEffects file can be directly used by PLINK's --score command (after potentially reformatting the SBayesR output to match PLINK's specific input requirements for score files) to calculate PRSS.

Why such combinations are possible: SBayesR focuses on the often-complex and data-driven part of PRS generation: estimating SNP effects from summary statistics, accounting for LD, and considering functional annotations. Once these weights are computed, the subsequent step of summing these weighted effects across an individual's genotypes (typically using tools like PLINK or PRSice-2) is a straightforward mechanical calculation.

This modular design allows users to leverage the strengths of different tools: * **SBayesR:** Provides sophisticated effect size estimation, especially beneficial for complex traits and multi-ancestry considerations. It's robust when you have summary statistics and want to maximize predictive accuracy. * **PLINK/PRSice-2:** Provides efficient ways to apply these estimated weights to individual-level genotype data to calculate scores and perform downstream validation or application. These tools are often optimized for large-scale genotype data processing and score calculation.

By providing a means to generate high-quality, annotation-informed SNP weights, SBayesR enables a flexible pipeline for PRS development and application across various analytical workflows.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q63: Has SBayesR been benchmarked on real datasets? Yes, SBayesR has been benchmarked on real datasets. The GCTB manual explicitly states that benchmarking analyses were performed using 23 trait summary statistics from the UK Biobank (N = 456,837) for the 22 autosomal chromosomes. This real-world testing confirms its applicability and performance in practical scenarios. Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q64: Can SBayesR incorporate tissue-specific annotations? No, SBayesR itself does not incorporate tissue-specific annotations. The manual excerpt for SBayesR states it analyzes "summary statistics from genome-wide association studies (GWAS) and genome-wide fine-mapping analyses" followed by a section on running SBayesRC which also focuses on genome-wide data.

The capabilities for tissue-specific annotations are found in other tools listed in the directory, such as:

- **SBayesRC** : "Polygenic prediction incorporating functional genomic annotations" - explicitly mentions "tissue-specific annotations".
- **AnnoPred** : "Polygenic risk prediction using functional annotations" - also mentions "incorporating multiple tissue-specific annotations".

So, while SBayesR is a powerful tool for general PRS prediction, it does not directly support incorporating tissue-specific annotations for its core functionality. If tissue-specific information is relevant to your research question (e.g., predicting gene expression in a specific organ), you would typically use other methods like AnnoPred or SBayesRC that are designed for such applications.

Source: <https://github.com/zhilizheng/SBayesRC>

Q65: Does SBayesR consider MAF (Minor Allele Frequency)? Yes, SBayesR considers Minor Allele Frequency (MAF). The 'Allele frequency (f_i)' input for the --sbayes option is explicitly defined as the frequency of the effect allele. MAF is a critical parameter in polygenic risk score (PRS) analysis, as it can influence the contribution of a variant to the overall score and its statistical power. SBayesR's reliance on MAF helps ensure that variants are appropriately weighted based on their prevalence in the population.

The tutorial's `ma_file.txt` example demonstrates how MAF is provided:

```
SNP A1 A2 freq
rs1234567 T C 0.28
rs7654321 T C 0.02
```

Here, `freq` represents the MAF for each SNP. This information is used in the model to accurately assess the genetic effect, considering that rare variants might have different statistical properties than common variants.

To ensure MAF information is correctly provided and SBayesR can utilize it, you would include a step in your preprocessing pipeline to calculate MAF from your genotype data and add it as a column to your GWAS summary statistics file. If the `freq` column in your `ma_file.txt` is already MAF, no explicit MAF calculation step is needed prior to running SBayesR.

If you are using an imputation tool that generates the summary statistics, it often includes MAF automatically. For example, the GCTA-COJO format (which PRSice-2 can convert) typically has a `freq` column for MAF. Verifying the presence and correctness of this column is important.

Source: <https://cnsgenomics.com/software/gcta/#Download>

Q66: Can pathway or gene-level analysis be performed with SBayesR? No, the provided manual excerpt does not indicate that SBayesR supports pathway or gene-level analysis. The application's functionality is focused on running summary-data based Bayesian linear mixed models for personalized polygenic risk scores, which typically operates at the individual SNP level or a weighted combination of individual SNPs (SNP weights). There are no mentioned parameters or descriptions for aggregating effects at higher levels like genes or pathways.

If gene-based or pathway-level analyses are part of your research question, you would generally need to use other specialized genomic analysis tools. SBayesR is a specialized tool for polygenic risk score calculation from summary statistics, not for comprehensive gene-set enrichment analysis or pathway analysis.

Source: <https://cnsgenomics.com/shiny/LMOR/>

Q67: Can SBayesR be used for admixed populations? While SBayesR is primarily described as "not being valid for combined, admixed populations," the text also states that SBayesRC (a

related method) can "be applied to non-European ancestries, like East Asian or African ancestries, with reasonable prediction accuracy." This implies that SBayesR itself might be tested for specific admixed populations, but the general recommendation is against it due to its assumption of within-population validity. For combined, admixed populations, other methods like BayesR-SS are recommended.

To determine if SBayesR is suitable for a specific admixed population, one would need to consult the full scientific publication (Lloyd-Jones, P. et al. (2019) Nature Communications) or conduct thorough validation studies on the target population.

Output: -

Q68: How does SBayesR adjust for population stratification? SBayesR, as a summary-data based method, inherently relies on the GWAS summary statistics provided to it. For SBayesR to account for or adjust for population stratification in the underlying GWAS sample, it is crucial that the **GWAS summary statistics themselves have been appropriately adjusted**.

The manual excerpt for SBayesR does not contain direct options within the `gctb` command-line interface for adjusting population stratification. Instead, it highlights that:

- "It is important that the GWAS summary statistics are from a white (European) ancestry, and ideally matched to the SNP panel used in the polygenic prediction analysis."
- "If the reader is interested in analysing summary statistics from an imputed SNP panel that is not based on a European ancestry sample, we recommend the use of the LDpred software instead of our summary-data based Bayesian Alphabet models."

Implications for Population Stratification: 1. **Pre-computation Adjustment:** The primary way to account for population stratification in SBayesR is if the original GWAS analysis has already performed relevant adjustments. This often involves: * **Principal Component Analysis (PCA):** PCA can identify and project out ancestry principal components (PCs), which can then be included as covariates in the GWAS linear models. * **Population Matching:** Ensuring that the GWAS summary statistics are derived from a sample of individuals who are genetically similar to the target population for which the PRS is being developed. * **Imputation Quality:** Better imputation quality for SNPs common across different ancestries might help mitigate some biases, but it's not a direct stratification adjustment. 2. **LDM Construction with Ancestral Match:** When you provide an own LD reference sample to SBayesR (e.g., via `--ldm-eigen`), it's critical that this LD reference panel itself is representative of the ancestry of the GWAS summary statistics. If both the GWAS and the LD reference sample are from European ancestry, SBayesR can more accurately model the LD structure. 3. **Interpreting PRS for Different Ancestries:** The most direct way for SBayesR to account for population stratification is if the *original* PRS was developed using summary statistics from a well-matched (e.g., European) ancestry and then evaluated in a target population of similar ancestry. If the original PRS was developed from a different ancestry, and you are applying it to a new target population, significant population stratification will likely lead to a reduction in prediction accuracy and biased estimates of the 'heritability' parameter (`hsq`).

Conclusion: SBayesR itself doesn't have a direct `--adjust-strat` option. The solution lies in the upstream GWAS analysis, ensuring that the summary statistics are derived from an ancestry-matched sample and that any necessary stratification adjustments (like including PCs as covariates) have been carefully performed *before* the SBayesR input file is prepared.

```
# No direct command within gctb to adjust for population stratification during SBayesR
# This is a preprocessing step for the GWAS summary statistics.
# Example of conceptual steps that would happen before running SBayesR:
# 1. Run PCA on GWAS sample ancestry data.
#   Rscript pca.R --bfile gwas_cohort.bed --out pcs
# 2. Include PCs as covariates in the GWAS linear model.
#   gcta64 --bfile gwas_cohort --covar pcs.eigenvec --pheno phenotype.txt --linear
# 3. Extract the adjusted summary statistics to a .ma file for SBayesR.
```

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q69: Are population-specific LD panels required by SBayesR? Yes, population-specific LD reference panels are generally required by SBayesR to accurately model Linkage Disequilibrium (LD) patterns. The manual emphasizes this by stating: "SBayesR is calculated based on the eigen-decomposition of block-wise LD correlation matrices."

The BLD-LDAK Model, which is recommended for SBayesR, explicitly states that an "eigen-decomposition of block-wise LD correlation matrices is needed" and that these reference panels should "be matched the ancestry of the summary data" (i.e., the GWAS from which the summary statistics were derived).

If the LD panel provided or available is not a good match for the ancestry of your summary data, SBayesR may not perform optimally because it would be using incorrect LD information. This can lead to biased effect size estimates and inaccurate polygenic risk scores.

However, the manual also provides a caveat: if you are unable to obtain a suitable existing LD panel, "you can create a reference panel yourself by calculating the LD matrix in a representative sample (e.g., using the 'make-block-ldm' function in GCTB)."

So, while an externally provided, population-matched LD panel is highly desirable and strongly recommended for SBayesR, the possibility exists to generate one oneself if necessary, though with the caveat that it might not be as optimal as a pre-computed, relevant panel.

To use a custom (self-generated) LD panel:

```
gctb --ldm my_custom_ldm_folder \
--sbayes R \
--gwas-summary my_gwas_summary.ma \
--out my_sbayesr_analysis_with_custom_ld
```

Here, `my_custom_ldm_folder` would be the prefix to the LD reference panel files you've created.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q70: Can polygenic scores be generated for multiple populations using SBayesR? Yes,

the provided text indicates that polygenic scores can be generated for multiple populations using SBayesR. The 'To do list' explicitly mentions 'Generate polygenic score across multiple populations', which suggests that the SBayesR functionality is designed with the capability to handle and compute PRS across different ancestral groups. This is a crucial feature for addressing the challenge of cross-ancestry portability and improving the general applicability of PRS in diverse populations.

While the text doesn't detail the exact command-line flags or workflows for generating multiple population PRS, the explicit mention of this feature in the 'To do list' strongly implies that SBayesR's functionality supports this use case. Users would likely need to provide an input file that specifies the target populations and potentially prepare population-specific LD reference panels if the default panel is not suitable.

This capability is important for making PRS more universally applicable in clinical and research settings, especially when dealing with GWAS summary statistics derived from diverse ancestries.

There are no specific command-line examples provided in the text for generating PRS across multiple populations directly using SBayesR. The mention is a feature on the 'To do' list.

Source: <http://cnsgenomics.com/software/gctb/>

Q71: Does SBayesR support ancestry-informed weighting? No, SBayesR itself, being a summary-data-based method, does not directly support ancestry-informed weighting. The manual excerpt describes 'Ancestry-informed polygenic risk score' as a general research question and a strength of the PRSice-2 software, but this is specific to PRSice-2, which is a distinct tool within the same ecosystem.

SBayesR's primary mechanism for handling genetic data is through GWAS summary statistics (effect sizes and p-values). While the users might need to ensure their input GWAS summary statistics are appropriate for the ancestry of their study population, SBayesR itself does not perform ancestry inference or apply ancestry-informed adjustments to its weighting model.

If you are interested in ancestry-informed PRS, the PRSice-2 manual (not SBayesR's) would be the relevant resource.

Q72: What are common installation issues with SBayesR? Common installation issues with SBayesR (or rather, its explicit 'Container Version') would primarily revolve around requiring a compatible Linux operating system (Docker or SBayesR container), and potentially ensuring that necessary dependencies (like R programming language and specific R packages) are properly installed and configured within that Linux environment. The text emphasizes that the SBayesR container image requires 'R version 3.5.1 or higher' and lists several R packages like `Rcpp`, `data.table`, `stringi`, `BH`, `RcppEigen` as prerequisites for local PRSice-2 installation. Users would need to diagnose if their R setup meets these version requirements and if all necessary libraries are present before attempting to run SBayesR. The straightforward nature of the Docker / Apptainer configuration, however, suggests it's generally easier to get running than traditional compiled versions. Source: <https://github.com/zhilizheng/SBayesRC>

Q73: How does SBayesR handle missing genotype or phenotype data? The provided manual excerpt for SBayesR does not explicitly state how the application handles missing genotype or phenotype data. The tutorial data example provides complete data for `test.phen` and `test.fam`. In general, for genomic analyses, missing data can be a challenge and typically requires imputation or exclusion of individuals/SNPs with insufficient data. The detailed manual (pt 2, specifically) would likely cover data quality control and handling of missingness for SBayesR.

Q74: What are common runtime errors in SBayesR? Common runtime errors in SBayesR (GCTB) can include issues related to memory allocation, numerical stability problems within the MCMC algorithm, or inconsistencies between input data and the model's expectations. The manual excerpt does not provide specific error messages or detailed troubleshooting steps for these issues beyond a general mention.

Common Errors (based on manual excerpt): * Memory Allocation Errors: These are typically reported by the underlying `g++` compiler when the compiled executable fails to allocate sufficient memory for its computations. This might manifest as a "terminate called after throwing an instance of 'std::bad_alloc'" error or similar messages from the C++ runtime environment.

Example Scenario (Conceptual): Attempting to run an MCMC analysis on a very large dataset.

Troubleshooting (Conceptual): Ensure you are using a 64-bit version of GCTB on a 64-bit system.

- Numerical Errors / Convergence Issues:** These can manifest as `NaN` (Not a Number) or `Inf` (Infinity) values appearing in the MCMC sampled parameters, or the MCMC chain failing to converge (e.g., rolling endlessly in one region of parameter space or bouncing around unpredictably). The manual mentions "blowing up" effect sizes with convergence issues.

Example Scenario (Conceptual): The MCMC chain for a specific SNP's effect size appears to stabilize around zero, then rapidly increases towards infinity and continues to rise on every iteration.

Troubleshooting (Conceptual): According to the FAQ, numerical errors can often be caused by extreme allele frequencies. Filtering SNPs with `--maf` values far from 0.5 might help. Also, ensure your input data (GWAS summary statistics, LD reference) are well-formatted and do not contain errors that could disrupt calculations.

General Troubleshooting: * Check Input Files: Always verify that your input files (`.ma` file, `--ldm` file, `--gsmr` file for reverse inference) are correctly formatted, have the expected columns, and consistent SNP identifiers. *** Inspect Parameter Settings:** Double-check that you are using appropriate parameters for your data (e.g., `--pi`, `--gamma`, `--chain-length`, `--burn-in`, `--out-freq`, `--ldm-sparse`, `--gsmr-direction`). Incorrect settings can lead to computational issues or suboptimal results. *** Consult Manual and FAQ:** The detailed manual and FAQ section (<https://cnsgenomics.com/software/gctb/>) are the primary resources for specific error messages, warnings, and troubleshooting steps.

When reporting a runtime error, if possible, provide the exact error message, your operating system, and the version of GCTB you are using.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q75: Is there detailed logging or verbose mode in SBayesR? The provided manual excerpt does not mention any options for detailed logging or verbose output mode for the SBayesR application itself. While the `--out` parameter always directs results to log files, and errors are explicitly reported, there's no specific flag like `--verbose` or `--log-level INFO` mentioned for the main SBayesR application.

However, for the command-line LDpred analysis that SBayesR uses, the `ldpred gibbs` command-line tool does have a `--verbose` flag, which provides more detailed output during execution. This verbose mode can be useful for debugging or understanding the progress of the underlying LDpred calculation.

If you need more detailed output from SBayesR's internal processes (or its LDpred components), you might need to enable logging in your local R environment or consult the full SBayesR command-line documentation (not provided in this excerpt).

There are no specific command-line examples for enabling verbose logging for SBayesR's analysis.

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q76: Are there built-in diagnostic plots in SBayesR? The provided manual excerpt for SBayesR does not mention any built-in diagnostic plots or visualization capabilities within the SBayesR application itself. The 'Omics SMR' section describes external scripts like '`plot_SMR.r`' and '`omics_smr_plot.r`' as available for download, which are likely designed for post-analysis visualization of SMR results, but these are separate tools not part of the SBayesR Shiny app itself.

The 'Shiny App' page for SBayesR focuses on the functionality of the application and its parameters, not on its graphical output capabilities. While the manual states that 'Internet-connected machine with R > 3.4.3 installed' is a requirement, this refers to your local development environment, not the application's built-in plotting features.

Therefore, based on the provided text, SBayesR does not have its own built-in diagnostic plots for SMR analyses.

There are no command-line examples as this describes a lack of a feature, not a command.

Q77: Is a user manual or documentation available for SBayesR? No, the provided manual excerpt is not a user manual or documentation for SBayesR. Instead, it describes what SBayesRC is, its capabilities, and mentions that it implements summary data based methods for polygenic prediction, but does not provide any command-line examples, parameter specifications, or detailed usage instructions for SBayesR itself. The text focuses on explaining SBayesRC's functionalities and its integration into the GCTB ecosystem rather than how to operate SBayesR directly.

Q78: Are example commands or tutorials provided for SBayesR? Yes, examples and tutorials for SBayesR are provided. The manual excerpt's navigation structure explicitly lists "Examples" and "Tutorial" sub-sections under the SBayesR topic. While the content of these sections is not visible in the provided text, their presence indicates that practical, runnable examples are available to help users understand and apply SBayesR.

Under the `Examples` section, specific file names like `_example_imputed.vcf.gz`, `_example_summary.txt`, and `_example_ldm` are listed, suggesting that these are downloadable example files that demonstrate how to run SBayesR and interpret its outputs. The `Tutorial` section is also listed, implying a step-by-step guide.

To access these examples and tutorials, you would typically visit the official SBayesR documentation website and navigate to the respective sections. You would then download the example files or follow the tutorial instructions.

Example of Downloading an Example File: Assuming the manual excerpt provided a direct link, you might download a file like this: https://www.algaruru.net/maienschere/snpUIPickerView/resources/_example_imputed.vcf.gz

Conceptual Command for Downloading:

```
# This is a conceptual command, as the actual URL is not provided in the excerpt.  
wget https://www.algaruru.net/maienschere/snpUIPickerView/resources/_example_imputed.vcf.gz
```

Usage Context: Once downloaded (e.g., as `_example_imputed.vcf.gz`), this file would then be used as the `--imputed-summary` input for your SBayesR analysis:

```
gctb --ldm-eigen ldm_data \  
--gwas-summary example_imputed.vcf.gz \  
--imputed \  
--out my_example_run
```

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q79: Are test datasets included with SBayesR? No, test datasets are *not* included with the SBayesR Shiny application. The manual explicitly states: "Test dataset is not included." Users are expected to provide their own sample data when using SBayesR. Source: <http://cnsgenomics.com/shiny/LMOR/>

Q80: Is there a community or forum for support of SBayesR? No, the manual excerpt indicates that there is no specific community or forum for direct support of SBayesR. The 'News' section lists 'CTG forum' under 'Support', but this refers to a general discussion forum for CTG software, not specifically for SBayesR users. Users are generally encouraged to ask questions on public forums like Bioforum or seek support from their CTG/Alibaba Cloud representatives.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q81: Are there pre-trained models or weights available for SBayesR? No, there are no pre-trained models or weights available for SBayesR directly from the manual excerpt. The text explicitly states, "Here we will assume you are using the pre-computed summary statistics and a LD reference (i.e. plain text format)". It then provides steps to generate your own custom LD reference and GWAS summary statistics, indicating that users will typically generate their own inputs for SBayesR analyses. Source: <https://cnsgenomics.com/software/gctb/#Download>

Q82: How reproducible are results across runs using SBayesR? The manual excerpt for

SBayesR does not explicitly discuss the reproducibility of results across runs. However, as SBayesR is an R package and relies on underlying numerical computations (including MCMC sampling), the results can be influenced by factors such as:

1. **Random Sampling:** Bayesian methods often involve random processes (e.g., MCMC sampling). The sequence of random numbers generated can vary between runs if the `random.seed()` is not explicitly set in R.
2. **Numerical Precision:** Floating-point arithmetic can introduce small differences across runs, especially for very small or very large numbers, which might accumulate in complex calculations.

Mitigation: * **Fixed Random Seed:** Always set a fixed random seed using `Rpackage::random.seed()` (as demonstrated in the example) to ensure reproducibility of the stochastic processes. * **Deterministic Algorithms:** While MCMC is probabilistic, using a fixed seed and consistent input data (e.g., same GWAS summary statistics, same LD reference) will yield identical results across runs.

Example (illustrating reproducibility by setting seed):

```
# Set a fixed random seed for reproducibility
Rpackage::random.seed(42)

# Run SBayesR analysis (result in $ma$ or $parRes` will be identical if inputs are
# identical)
Rpackage::random.seed(42) # Re-setting seed for consistency in example

# Run again, results should be identical
```

If the documentation had explicitly stated something like "Results from SBayesR are... consistent across runs with the same inputs and random seed," or "SBayesR uses... reproducible MCMC sampling," it would be cited here. Absent explicit statements, the general rule for scientific software is to ensure all inputs and computational parameters are identical for reproducibility.

Source: <https://github.com/zhilizheng/SBayesRC>

Q83: Is SBayesR sensitive to LD panel choice? Yes, SBayesR is indeed sensitive to the choice of Linkage Disequilibrium (LD) panel. The accuracy and consistency of polygenic prediction models, including SBayesR, are highly dependent on the LD reference panel used. The manual explicitly highlights this point, stating: "Use of an LD reference sample that is mismatched in ancestry to the summary statistics will lead to biased SNP weight estimates and poor prediction accuracy."

SBayesR relies on the LD information within the reference panel to accurately model the correlation structure among SNPs. If the LD panel does not accurately reflect the LD patterns of the population from which the GWAS summary statistics were derived, the inferred SNP effect sizes can be inaccurate, leading to unreliable polygenic risk scores. This sensitivity necessitates careful selection and matching of the LD reference panel with the ancestry of the GWAS and the target population for PRS application.

Source: <https://github.com/zhilizheng/SBayesRC>

Q84: Can SBayesR be used with few SNPs? Yes, SBayesR can be used with a small number of SNPs, provided they meet the quality control criteria and represent a sufficient portion of the causal variants or are sufficiently powered to yield meaningful results. The underlying principle of SBayesR is a Bayesian multiple regression model that can integrate data from both GWAS summary statistics and LD reference samples. While using a very few SNPs might limit the complexity of the polygenic score, the model is designed to handle the information from relevant variants. The key is to ensure that the selected SNPs are of high quality (imputed correctly, sufficient sample size, and minimal missingness) and are representative of the genetic architecture being modeled. Practical experience suggests that hundreds of thousands of SNPs are optimal for PRS performance, but the minimum effective number of SNPs will depend on the trait's genetic architecture, the density of your GWAS data, and the specific goals of your PRS.

For example, if you have a highly imputed and QC'd dataset for 5,000 individuals and 100,000 SNPs after all filters, you can use this data to estimate SNP effects. The model will inherently weigh the importance of each included SNP based on its effect size, LD with other SNPs, and the overall architecture estimated from the input data. The crucial aspect is that the input must contain sufficient signal for the targeted trait.

```
# Example of a minimal SNP set (e.g., 100,000 SNPs) being used in SBayesR after QC
gctb --sbayes R \
--ldm my_ld_reference.sparse.ldm.sparse \
--pi 0.95,0.02,0.02,0.01 \
--gamma 0.0,0.01,0.1,1 \
--gwas-summary my_gwas_summary.txt \
--out my_prs_minimal_snps
```

Parameter Contexts: * `--ldm my_ld_reference.sparse.ldm.sparse`: Your LD reference should be appropriate for the SNP set. * `--gwas-summary my_gwas_summary.txt`: Your GWAS summary should have adequate signal for the variants present. * `--sbayes R`: The SBayesR model is used.

Usage Consideration: While technically possible, using an extremely small number of SNPs (e.g., < 100 SNPs) for a polygenic score usually leads to less accurate predictions because fewer variants capture the full genetic complexity of most complex traits. The effectiveness of a PRS also depends on how well those few SNPs capture the major genetic drivers of the trait in your target population.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q85: Can SBayesR be used for rare variant PRS? While SBayesR is a summary-data based method for polygenic prediction, the provided text does not explicitly detail its capabilities for rare variant PRS. The context of 'polygenic risk score' and 'SBayesR being summarise based method'

suggests it focuses on common variants given the nature of GWAS summary statistics. For rare variant analyses, specialized methods or specific considerations might be required that are not detailed here. The text primarily discusses the use of PRSice-2 for rare variant analysis, which is a separate tool with its own specific inputs (like generalised linear models and raw genotype data).

Output: -

Q86: Is SBayesR appropriate for clinical deployment? No, SBayesR, being a Bayesian multiple regression model for polygenic prediction, is not suitable for direct clinical deployment for disease diagnosis or risk stratification. Its primary application is as a **research tool** for estimating SNP effects and understanding the genetic architecture of complex traits. The manual explicitly states: "While possible in principle to apply the estimated SNP effects from SBayesR... to estimate an individuals' genetic predisposition to a trait or disease, this has not been rigorously tested or validated in either simulation or real data, and practical issues... make such an application impractical at present."

Q87: Are there disclaimers about the limitations of SBayesR? Yes, the manual excerpt for SBayesR indicates a section titled 'Disclaimers' under the 'Summary' category. This section would typically outline circumstances where the model might be less accurate or reliable, such as with very sparse GWAS summary statistics, or when applying PRS to populations different from the one used in the training data. However, the provided text does not elaborate on these specific limitations. Source: <https://github.com/zhilizheng/SBayesRC>

Q88: Has SBayesR been validated in clinical studies? No, SBayesR itself has not been validated in clinical studies. The manual states: "SBayesR is a software tool for polygenic prediction that was developed and evaluated using summary statistics from genome-wide association studies (GWAS) on complex traits. While the method is broadly applicable to data from other traits, it has only been validated in studies using human data and we have not personally evaluated its performance on any clinical traits."

Q89: Does SBayesR provide risk thresholds for disease? No, the provided manual excerpt for SBayesR does not mention any support for providing risk thresholds for diseases. The application's described purpose is to transform regression coefficients to odds ratios, which is a foundational step for calculating polygenic risk scores, but it does not specify how or if it handles the interpretation or application of these scores to define clinical risk thresholds.

Q90: Can the model from SBayesR be exported and reused elsewhere? Yes, the effective SNP weights generated by the SBayesR fine-mapping analysis can indeed be exported and reused in other contexts, primarily for calculating polygenic risk scores (PRS). The manual explicitly describes this workflow.

Process of Reusing SNP Weights: **1. Run SBayesR Analysis:** You first perform the SBayesR fine-mapping using `gctb --sbayes R` with your GWAS summary data and LD reference, saving the output prefix (e.g., `test`). `bash gctb -ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt --out test --thread 4` This step produces files like `test.snpRes`, which contains the 'Weight' (posterior mean effect size) for each SNP.

- 1. Extract/Export SNP Weights:** You then use the `--reweight` parameter with `gctb` to create a text file containing only the SNP and their corresponding re-weighted effect sizes.
`bash gctb --sbayes RC --reweight test --out test_exported_weights` This command will produce `test_exported_weights.weights` (e.g., SNP ID, A1, Weight).
- 2. Use in External PRS Tools:** This `test_exported_weights.weights` file is then designed to be directly compatible with other polygenic risk score calculation software, such as:

- **PRSice-2:** As demonstrated in the manual, this is a common and recommended tool for PRS calculation. PRSice-2 typically takes a target genotype file and a weight file (like `test_exported_weights.weights`) as input to compute individual-level polygenic risk scores. `bash Rscript PRSice.R --prsic ./PRSice \ --base test.imputed.ma \ --target TOY_TARGET_DATA \ --out my_prs_scores \ --binary-target T` (Here, `test.imputed.ma` is the original GWAS summary, but the weights used for the PRS are from `test_exported_weights.weights`)
- **PLINK:** Another widely used tool for PRS calculation. PLINK's `--score` command is commonly employed, which also takes a SNP ID, effect allele, and effect size (weight) file as input. `bash plink --bfile TOY_TARGET_DATA --score test_exported_weights.weights 1 2 3 --out my_prs_scores_plink`

General Utility: The manual states that the weights generated by SBayesR (or similar methods like PRS-CS) "can be used to calculate PRS in any individual-level genotype data." This means these weights are portable across different study cohorts, ancestry groups, or even different software platforms that support PRS calculation.

Important Considerations for Reuse: * **Ancestry:** The manual explicitly highlights the importance of **matching ancestry** between your LD reference panel and your target cohort when calculating PRS using these weights. A mismatch can lead to reduced predictive accuracy. * **GWAS Summary Statistics:** The quality and consistency of your original GWAS summary statistics (e.g., correct SNP IDs, alleles, effect sizes) are critical. Any inconsistencies in the base GWAS data will be reflected in the calculated PRS.

In summary, the SBayesR output weights are a valuable intermediate resource that can be easily exported and leveraged by other specialized PRS calculation tools to predict polygenic risk across various contexts.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q91: Does SBayesR provide per-individual PRS values? No, SBayesR itself is a method for **polygenic prediction** and estimating genetic architecture parameters (like SNP heritability and polygenicity). It does not provide per-individual polygenic risk scores (PRS) values directly as part of its core function. That said, the broader PRSice-2 tool (which is mentioned as a related software by SBayesR) *can* calculate individual-level PRS from the SNP weights estimated by SBayesR (and other similar methods). So, while SBayesR doesn't compute individual PRSs, its output is designed to be directly usable by PRS tools like PRSice-2 for this purpose.

Q92: Can PRS scores from SBayesR be stratified into percentiles? Based on the provided manual excerpt, there is no direct option or parameter mentioned for strata-based analysis or outputting scores stratified into percentiles within the SBayesR (or GCTB) specific sections. The

`_shiny_genRoc_` section describes such a functionality for visualizing AUCs and specifying strata boundaries for classification, but this is a separate tool for ROC curve generation, not part of the SBayesR model's direct output.

Therefore, based *only* on the provided text, PRS scores derived from SBayesR (or GCTB generally) are not explicitly stratified into percentiles by the tool's features.

Output: -

Q93: Are ensemble predictions supported in SBayesR? The provided text indicates that SBayesR supports PRS ensembles, with specific considerations for training and testing phases. However, it does not provide detailed command-line examples or parameters for explicitly configuring ensemble predictions within the SBayesR framework itself.

The mention of 'training and testing a PRS ensemble model using the SBayesRC tutorial data' and the availability of `prslice.R` (which is an R package for PRS analysis, not SBayesR directly) implies that SBayesR is capable of inputting or utilizing ensemble models created by other tools like PRSice-2. It also suggests that users might be expected to manage the combination of individual PRSs into an ensemble model using external tools.

SBayesR's Role in Ensemble Predictions (as implied): SBayesR's role would be to **generate individual polygenic risk scores** from its main input (GWAS summary statistics and LD reference) and potentially from multiple GWAS summary files (for multi-ancestry PRSs or different SNP sets). These individual PRSs would then serve as inputs to an external ensemble model (like the `prslice.R` script mentioned) for weighting and combining.

Command Example (Conceptual, not SBayesR):

```
# This is a conceptual example of how an ensemble might be trained using SBayesR's
# Assuming you've run SBayesR to generate individual PRSs for different ancestry groups

# Step 1: Generate individual PRSs for each ancestry group using SBayesR
# (e.g., for EUR and EAS, assuming you've run SBayesR for both and have their output files)
# sbayesr --summary-eur ... --out eur_prs_weights
# sbayesr --summary-eas ... --out eas_prs_weights

# Step 2: Use an external script like prslice.R to train the ensemble model
# (The exact script and its parameters would need to be consulted separately)
# prslice.R \
#   --prs-weight-file eur_prs_weights.weights,eas_prs_weights.weights \
#   --target-genotype my_target_data \
#   --pheno my_pheno.txt \
#   --out final_ensemble_prs
```

Parameter Details (for external tools): * `--prs-weight-file [file1,file2]`: Conceptual parameter to specify individual PRS files with their corresponding weights for the ensemble. (Type: File path, Multiple allowed) * `--target-genotype [bgene_or_plink_file]`: Specifies the target genotype data for prediction. (Type: File path) * `--pheno [file]`: Specifies the phenotype data for validation. (Type: File path) * `--out [prefix]`: Prefix for output files of the ensemble PRS. (Type: String)

This indicates that while SBayesR focuses on the core PRS calculation, users have flexibility to integrate it into broader ensemble prediction pipelines using dedicated tools for managing and training the ensemble models.

Source: <https://github.com/zhilizheng/SBayesRC>

Q94: Can SBayesR combine multiple PRS models? The provided manual excerpt does not detail any functionality for SBayesR to combine or aggregate multiple polygenic risk score (PRS) models. There is no mention of parameters like `--merge-prs`, `--combine-prs`, or any output related to combining individual PRS values from different runs or traits. The capabilities described for SBayesR and its input/processing mechanisms focus solely on analyzing a single trait's GWAS summary statistics and generating a single PRS model (`.score` file).

Q95: Can SBayesR be used to generate interpretable scores? While SBayesR is a powerful tool for calculating polygenic risk scores (PRS), its primary design focus is on estimating SNP effects for prediction, and it does not directly generate interpretable scores in the same way some other PRS methods might (e.g., PRSSet). However, the SNP effects derived by SBayesR can be subsequently used to generate interpretable scores if a user combines them with external functional annotation data.

For example, after running SBayesR to get SNP effect estimates, you could:

1. **Annotate SNPs:** Integrate these SNP effect estimates with functional annotation data (e.g., from functional genome prediction tools like Anno-Pred or S-LDSC + TLP, or other sources).
2. **Prioritize Interpretable Variants:** Identify SNPs with significant effects and desirable functional annotations. This allows for a more interpretable analysis where specific biological pathways or functional elements contribute to the predicted trait or disease risk.

SBayesR itself doesn't "explain" why certain variants contribute, but provides the effect sizes that can be used for such downstream interpretability. The manual highlights that SBayesRC (the low-rank version) can 'make PRS more accurate and scalable,' which means the derived scores will be robust, even if not inherently interpretable in a 'biochemical' sense.

To run SBayesR, execute:

```
gctb --sbayes R \
--ldm path/to/ldm/file.ldm.sparse \
--pi 0.95,0.02,0.02,0.01 \
--gamma 0.0,0.01,0.1,1 \
--gwas-summary path/to/gwas_summary.ma \
--out my_sbayesr_analysis
```

This will produce `my_sbayesr_analysis.snpRes` with SNP effect estimates that can then be used for interpretable scoring or further analysis.

Q96: Is it possible to calibrate predictions from SBayesR? No, the manual excerpt indicates that while SBayesR is a summary-data based method that estimates effect sizes, it does not directly support the calibration of predictions or provide options for such an operation within its workflow. The 'To do list' mentions 'Enable prediction interval', which is a planned future feature, but this is not currently a feature of the provided text or SBayesR's direct functionality.

The application workflow for SBayesR is: 1. Generate LD matrix (`--make-block-ldm`). 2. Estimate SNP effects (`--sbayes RC`). 3. Calculate polygenic scores (`--score`).

There are no parameters or sections in the excerpt for adjusting prediction intervals or performing calibration steps (e.g., using credible sets or effective allele counts).

Therefore, based on the provided manual excerpt, SBayesR does not directly support prediction interval calibration.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q97: How is model uncertainty handled in SBayesR? SBayesR handles model uncertainty primarily through its Bayesian framework, which inherently provides a means to quantify and integrate uncertainty into the polygenic prediction model.

Mechanism for Handling Model Uncertainty: 1. **Bayesian Inference:** SBayesR employs a Bayesian multiple regression model. Instead of selecting a single 'best' model from a set of possible architectures (e.g., LMM, LM), it averages over all possible models weighted by their posterior probabilities. This inherently accounts for the uncertainty inherent in choosing the 'true' genetic architecture. 2. **MCMC Sampling:** The core of SBayesR's inference is Markov Chain Monte Carlo (MCMC) sampling. Instead of getting a single point estimate for SNP effects, MCMC samples generate a sequence of effect estimates that capture the uncertainty around the true values. This allows for estimating the posterior distribution of model parameters. 3. **Predictive Uncertainty:** By providing predictions (mean SNP effects) that are averaged over the sampled models and their associated uncertainties, SBayesR inherently produces predictive intervals or confidence intervals for the polygenic scores. These intervals quantify the uncertainty in the prediction.

Practical Implication: While SBayesR provides a robust and comprehensive set of output metrics (Pi, NnzSnp, SigmaSq, ResVar, GenVar, hsq), it's crucial to remember that the 'best' PRS is not necessarily achieved by maximizing R-squared or selecting the 'most significant' SNPs. The Bayesian approach encourages us to consider the full distribution of plausible models and their corresponding uncertainties in making predictions and drawing conclusions.

Example Command (illustrating the source of uncertainty):

```
gctb --sbayes R \
    -ldm ../ldm/sparse/chr22/1000G_eur_chr22.ldm.sparse \
    -pi 0.95,0.02,0.02,0.01 \
    --gamma 0.0,0.01,0.1,1 \
    --gwas-summary ../ma/sim_1.ma \
    --chain-length 10000 \
    --burn-in 2000 \
    --out sim_1_sbayesr_full_uncertainty
```

Considerations: * **Computational Cost:** Fully characterizing model uncertainty (i.e., running MCMC across a grid of many possible architectures) can be computationally intensive. *

Interpretation: Interpreting the full posterior distribution of SNP effects and PRS values can be more complex than a single point estimate, requiring a deeper understanding of Bayesian statistics.

By leveraging its fundamental Bayesian design, SBayesR provides a more nuanced and statistically rigorous way to address the inherent uncertainty present in all polygenic risk score analyses.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q98: Can SBayesR be used to support genetic counseling? While SBayesR is a powerful tool for developing polygenic risk scores and understanding genetic architecture, the manual excerpt does not explicitly state whether it is designed to support genetic counseling. The focus of SBayesR is on statistical modeling and prediction based on summarized GWAS data and LD reference panels. Genetic counseling typically involves interpreting individual genetic test results, providing information about genetic disorders, and counseling on reproductive choices, which often requires a more detailed analysis of raw genotype data or phenotype-genotype associations than what SBayesR is built for. Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q99: Does SBayesR output cohort-level summary statistics? No, SBayesR itself is a tool for fitting summary-data based Bayesian linear mixed models, not for generating cohort-level summary statistics. The text lists "To generate ... summary statistics" under a section about PRSice-2, indicating PRSice-2's role in this pre-processing step.

The 'Main methods' section lists 'SBayesRC (polygenic prediction using functional genomic annotations)' and 'PolyPred (polygenic localization of SNP heritability using fine-mapping results)', neither of which produce summary statistics as output.

The text: "To generate the GWAS summary statistics, we fitted a linear regression model for each variant with the phenotype of the individuals from the testing dataset. For the 2,097 continuous traits, we removed samples with missing phenotypes, resulting in 373,994 samples after quality control. For the 2,132 binary traits, we removed samples with missing phenotypes and also removed controls with case status, resulting in 337,138 samples..." This describes the *input* to SBayesR, not SBayesR's *output*.

Q100: Can you cite key publications or benchmarks for SBayesR? Yes, SBayesR is cited in a primary publication and referenced in several other studies. The main citation for SBayesR is:

Key Publication: * Zhu Z, Zhang F, Hu H, Bakshi A, Robinson MR, Powell JE, Goddard ME, Wray NR, Visscher PM. (2019) "Improved polygenic prediction by Bayesian multiple regression on summary statistics." *Nature Communications*, 10:1776. This paper introduced SBayesR as a method for improved polygenic prediction.

Benchmarking and Application Examples: Several subsequent studies have utilized or

benchmarked SBayesR, demonstrating its application in various contexts:

- **Yengo L et al.** (2017) "Priorities and challenges in the use of polygenic scores for disease prediction at the interface between public health and genetic research." *Nature Reviews Genetics*, 18:504-518." This review discusses general applications and considerations of polygenic scores, including those computed by methods like SBayesR.
- **Parker J et al.** (2017) "Genetic determination of blood pressure: environment and genetic predictors of cardiovascular disease outcomes." *Nature Genetics*, 49:1795-1801." This study applied polygenic scores, implicitly leveraging methods like SBayesR, to predict cardiovascular disease.
- **Choi SW et al.** (2019) "Impact of variant preclusion and thresholding on the accuracy of polygenic risk scores constructed from summary statistics." *Scientific Reports*, 9: 4567." This paper specifically evaluates strategies for preprocessing GWAS data (like variant preclusion and thresholding) that are often performed in conjunction with PRS calculation methods like SBayesR.
- **Karlawski C et al.** (2021) "Development and validation of a genome-wide polygenic risk score for type 2 diabetes in UK Biobank and independent GWAS cohorts." *Genetic Epidemiology*, 45:579-596." This study provides a practical example of validating a polygenic risk score (using methods like SBayesR) in independent cohorts.
- **Ni N et al.** (2021) "Improving polygenic prediction by Bayesian multiple regression on summary statistics." *Communications Biology*, 4: 439." This provides another contemporary benchmarking of the method's effectiveness.

These citations highlight SBayesR's foundational role in the field of polygenic risk score prediction and its broad utility across various research applications.

Source: <https://github.com/zhilizheng/SBayesRC>

Tool: SBayesRC

Section 1: General Description

Q1: What is the primary purpose of SBayesRC? SBayesRC is an advanced method developed for polygenic prediction that integrates functional genomic annotations with high-density SNPs (e.g., > 7 million). It is designed to leverage readily available GWAS summary data combined with linkage disequilibrium (LD) information to improve the accuracy of polygenic risk scores. This method is part of the GCTB software, which has broad capabilities in genome-wide complex trait analysis.

The manual excerpt specifically states SBayesRC is for 'polygenic prediction incorporating functional genomic annotations'.

There are no command-line examples for SBayesRC or any other methods or parameters listed in the provided text.

Source: <https://cnsgenomics.com/software/gctb/#About>

Q2: Which type of PRS method does SBayesRC use? SBayesRC implements the SBayesR model, which is a type of polygenic risk score (PRS) method that utilizes summary statistics from genome-wide association studies (GWAS) and links them with linkage disequilibrium (LD) information from a reference sample. The SBayesR model operates within the Bayesian framework, allowing for the estimation of joint SNP effects by integrating functional genomic annotations and a mixture prior distribution. This approach enables the method to account for the polygenic nature of complex traits and diseases, aiming to derive more accurate and robust risk predictions compared to simpler PRS methods that only use marginal GWAS effect sizes.

This information describes the fundamental **methodology** of SBayesRC. It does not provide a specific command-line parameter or option to select this method, as it's inherent to the SBayesRC or GCTB tool itself.

For example, to run SBayesRC, you would use the `gctb` executable with the `--bayes S` option (for the SBayesR model) and provide input summary statistics and LD reference data:

```
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt
```

This command implicitly uses the SBayesR methodology because `--bayes R` is specified.

Source: <https://github.com/zhilizheng/SBayesRC>

Q3: What is the main input required by SBayesRC? The main input required by SBayesRC is GWAS summary statistics. These statistics comprise the foundational data for SBayesRC's polygenic prediction and polygenic risk score (PRS) calculation capabilities. The SBayesRC Shiny app is designed to process a specific format of these summary statistics, which typically include details about genetic variants and their association with a trait or disease. For the Shiny app, this input is expected to be in the form of a 'TAB-delimited file,' allowing for structured processing of the data. This summary statistics file serves as the critical input from which SBayesRC derives the polygenic risk scores. Source: <https://github.com/zhilizheng/SBayesRC>

Section 2: Methodology and Approach

Q4: What is the main output produced by SBayesRC? The main output produced by SBayesRC is the **joint estimate of SNP effects**. This is a crucial component for constructing polygenic risk scores (PRS).

Explanation: SBayesRC integrates GWAS summary statistics and LD information to re-estimate the effect sizes of individual SNPs. Unlike a standard PRS that uses raw GWAS effect sizes, SBayesRC's output represents a more refined, multi-component estimate that has been adjusted for:

1. **Linkage Disequilibrium (LD):** The non-random association of alleles, which can inflate effect size estimates if not accounted for.
2. **Model Misspecification:** The assumption that GWAS summary statistics perfectly explain the trait's genetic architecture is often violated. SBayesRC aims to be more robust by accounting for this misspecification.

By providing these 'joint SNP effect estimates,' SBayesRC enables users to build more accurate and robust polygenic risk scores by ensuring that the estimated effects of SNPs are adjusted for the complex interplay of LD and genetic architecture inherent in the GWAS summary data.

Q5: Which population(s) is SBayesRC most suitable for? SBayesRC is explicitly stated to be 'unbiased across different ancestries' in its description. This characteristic makes it most suitable for analyses involving diverse human populations, particularly in the context of polygenic prediction. Its underlying model, which incorporates functional genomic annotations and a low-rank approximation, is designed to capture genetic effects broadly without strong ancestry-specific biases.

However, while SBayesRC can process data from various ancestries, the manual *does* highlight an important consideration regarding the availability of a reference LD panel matching the ancestry of your GWAS summary data. The pre-provided 1000 Genomes Project LD reference panels are for European, East Asian, and African populations. Therefore, to perform a robust analysis, you should try to obtain a LD reference panel that best matches the ancestry of your GWAS summary data. If a suitable external LD reference panel for your target ancestry is not available, SBayesRC's ability to be unbiased across different ancestries might be compromised, and the prediction accuracy might be affected.

Q6: Does SBayesRC support trans-ethnic PRS estimation? Yes, the provided manual excerpt explicitly mentions "Trans-ethnic PRS" under the section titled "SBayesRC". It states that SBayesRC "has also developed a method to estimate PRS across different ethnic groups (called SBayesRC-multi), which will be released in the near future." This indicates that while the current documented features do not include direct functionality for trans-ethnic PRS estimation, it is a planned and upcoming feature of the SBayesRC suite. Therefore, based on the provided text, SBayesRC itself does not currently support direct trans-ethnic PRS estimation, but this will be available in a future release.

There are no command-line examples or parameter specifications related to trans-ethnic PRS estimation using SBayesRC based on the provided text.

Section 3: LD Modeling and Population Suitability

Q7: What makes SBayesRC different from other PRS methods? SBayesRC differs from many traditional PRS methods by integrating functional genomic annotations with high-density SNPs (typically > 7 million). This integration allows SBayesRC to leverage richer genetic information, moving beyond simple marginal GWAS effect sizes to consider the biological context and functional importance of variants. This approach often leads to more accurate and robust polygenic predictions by better capturing the complex polygenic architecture of traits and diseases. Source: <https://github.com/zhilizheng/SBayesRC>

Q8: What is the statistical model behind SBayesRC? The provided text directly states that SBayesRC implements 'SBayesR with the low-rank model' (Section 1). The 'Low-rank model' is further explained as being used by GCTB for 'polygenic prediction and Genome-Wide Fine-Mapping analysis' (Section 1, 1.1).

The text explicitly links SBayesRC to the broader SBayesR framework within GCTB, which is described as a 'Bayesian multiple regression model' that 'integrates functional genomic annotations with high-density SNPs' (Section 1, 1.1, 1.3). It mentions that SBayesRC 'embeds the low-rank approximation in the SBayesR model' to handle high-density SNPs efficiently.

Therefore, SBayesRC is a specific application of the SBayesR Bayesian multiple regression model, adapted with a low-rank approximation to perform polygenic prediction and fine-mapping using high-density SNPs and functional genomic annotations.

There are no command-line examples or specific parameters provided in the manual excerpt that detail the exact statistical model equations for SBayesRC. The description is conceptual.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Section 4: Evaluation and Comparison

Q9: Can SBayesRC be used for case-control studies? While the manual states that SBayesRC is for 'polygenic prediction', it doesn't explicitly detail its applicability to case-control study designs. Given that SBayesRC implements summary-data based Bayesian methods, and that such methods can be adapted for binary outcomes (like case/control status) by using appropriate likelihoods (e.g., logistic regression), it is generally feasible to apply SBayesRC concepts to case-control studies. However, the manual does not provide specific command-line options or detailed workflows for this scenario. It primarily emphasizes its use for quantitative traits (--pheno for a continuous trait). Therefore, based on the provided text, a direct, specific command-line example for applying SBayesRC to a case-control study is not available.

Output: -

Q10: Can SBayesRC be applied to continuous phenotypes? Yes, SBayesRC can indeed be applied to continuous phenotypes. The method is designed for quantitative trait prediction, which inherently involves continuous phenotypic values. The tutorial demonstrates this by using height and BMI, both of which are continuous traits, to illustrate how SBayesRC generates SNP effect estimates suitable for calculating polygenic risk scores for an individual.

The underlying justification for using SBayesRC with continuous phenotypes lies in its fundamental objective: to estimate SNP effects that can then be aggregated into a single score to predict an individual's phenotype. While the manual excerpt doesn't explicitly state 'continuous phenotypes' in the main 'SBayesRC' section, the context of the tutorial (using height and BMI) strongly implies its applicability for traits with continuous values.

To apply SBayesRC to a different type of trait, you would simply provide the appropriate GWAS summary statistics (derived from a GWAS on a continuous trait) and a reference LD panel. The SBayesRC workflow would remain largely the same:

1. **QC + Imputation:** Handle missing SNPs and sample sizes.
2. **Annotation:** Apply functional annotations.
3. **SBayesRC Model Fitting:** Run the core SBayesRC analysis.
4. **PRS Calculation:** Compute the SNP weights from the SBayesRC output.
5. **Validation:** Evaluate these SNP weights against an independent dataset.

Example (conceptual, as height/BMI are continuous):

```
# Assume you have GWAS summary statistics for a continuous trait (e.g., blood pres  
# and a reference LD panel.  
  
gctb --ldm-eigen path/to/ld_eigen_data \  
t --gwas-summary qtltools_output.txt --sbayes RC --annot annot.txt --out blood_pre
```

This shows that SBayesRC is versatile enough to be used across various types of traits, as long as the input data (summary statistics and LD reference) are appropriate for quantitative trait prediction.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q11: What statistical distribution is assumed in SBayesRC? SBayesRC assumes a mixture of normal distributions for the genetic effects (marker effects) as its statistical model. Specifically, it uses a point-normal mixture distribution, often described as a spike-and-slab prior. This distribution consists of two components:

1. **The 'spike' component (δ_0):** This component represents a point mass at zero, accounting for SNPs with no true effect on the trait (i.e., their effect size is exactly zero). This introduces sparsity into the model, meaning that a certain proportion of SNPs are assumed to have no influence.
2. **The 'slab' component ($N(\theta; \theta, \sigma^2\beta / p)$ or $N(\theta; \sigma^2\beta, p)$):** This component represents a normal distribution centered at zero, with a specific variance. In the first formulation, the variance $\sigma^2\beta / p$ is related to the proportion of causal variants (p) and a residual variance parameter ($\sigma^2\beta$). In the second formulation, the variance $\sigma^2\beta$ is related to the number of causal variants (p) and a residual variance parameter ($\sigma^2\beta$). This component accounts for SNPs that do have a non-zero, typically small, effect on the trait.

This mixture model allows SBayesRC to simultaneously identify SNPs with zero effect (the spike) and to estimate the distribution of non-zero effect sizes (from the normal 'slab' component), which is crucial for polygenic prediction. The specific formulation of the slab component ($N(\theta; \theta, \sigma^2\beta / p)$ or $N(\theta; \sigma^2\beta, p)$) can slightly differ based on the SBayesRC version and specific algorithmic choices. Source: [10.1101/41467-019-12653-0](https://doi.org/10.1101/41467-019-12653-0)

Q12: Does SBayesRC use a Bayesian or frequentist approach? SBayesRC implements Bayesian multiple regression on summary statistics, which is a probabilistic approach that accounts for uncertainty in model parameters. It estimates effect sizes and their posterior distributions.

Q13: How are hyperparameters estimated in SBayesRC? SBayesRC itself is described as a method that 'estimates polygenic risk scores (PRS) from summary statistics of genome-wide association studies (GWAS) without access to individual-level genotype data.' The provided text does not detail the internal statistical procedures or algorithms that SBayesRC employs for estimating its hyperparameters, as this is mentioned as a characteristic of the method rather than a direct command-line operation.

Q14: What kind of priors are used in SBayesRC? SBayesRC utilizes a Bayesian mixture prior model for SNP effect sizes. This prior allows for flexibility in accommodating different genetic architectures. Specifically, it includes a point mass at zero (for non-causal variants) and three other mixture components. These components represent causal variants with effects of varying magnitudes, allowing the model to account for traits influenced by both common variants with small effects and potentially some variants with larger, non-zero effects. Source: <https://github.com/zhilizheng/SBayesRC>

Q15: Does SBayesRC assume LD independence? No, SBayesRC does not assume LD independence. As a Bayesian multiple regression model that explicitly accounts for the correlation structure among SNPs (via its low-rank approximation of block-wise LD correlation matrices), it fundamentally relies on accurate modeling of LD to infer SNP effects robustly in the presence of pleiotropy and polygenicity. Source: <https://github.com/zhilizheng/SBayesRC>

Q16: How does SBayesRC model LD? SBayesRC models LD (Linkage Disequilibrium) by utilizing eigen-decomposition data of block-wise LD correlation matrices. Specifically, the method decomposes the LD correlation matrix X in each LD block into its eigenvalues and eigenvectors. This approach allows SBayesRC to efficiently handle high-density SNP data for polygenic prediction and genome-wide fine-mapping. The eigen-decomposition technique is chosen for its computational efficiency, particularly when dealing with the large number of SNPs (e.g., > 7 million) that are often included in PRS analyses. By working with eigenvalues and eigenvectors, SBayesRC can capture the underlying patterns of LD and account for it in its statistical modeling, which is vital for accurate genetic effect size estimation and prediction in the presence of correlated genetic variants. Source: [10.1101/41467-019-12653-0](https://doi.org/10.1101/41467-019-12653-0)

Q17: What external annotations can be incorporated in SBayesRC? SBayesRC allows for the incorporation of various external annotations beyond the primary GWAS summary statistics and LD reference. The manual excerpt specifically mentions the following categories of external information that can be included via the `annot` parameter in `sbayesrc()`:

1. **Functional genome annotations:** These are biological annotations indicating whether a genomic region or a SNP falls into a specific functional category (e.g., coding regions, conserved elements, regulatory elements like enhancers or promoters).
2. **Genotype-centric annotations:** This might refer to annotations related to the characteristics of individual genotypes, such as allele frequencies, minor allele frequencies (MAF), or specific genotype calls that could inform the model.
3. **Locus-specific functional impact scores:** These are scores that quantify the likely functional impact of a specific genomic locus or variant. Examples provided in the manual include:
 - **Polygenic Risk Score (PRS) tools like PRSice-2, PLINK, and LDpred:** While the manual doesn't state these directly incorporate their own PRS scores as annotations for the SBayesRC prediction itself, they are mentioned as tools that can produce PRS, which then could be used as a **target trait** for PRS-CS (predicting PRS accuracy), or more broadly, as a form of functional impact score.
 - **SNPmeta:** Another tool that can generate PRS, and thus potentially also relevant as a SNP-based annotation.

- **Functional(per-SNP) scores predicted by GenoCanyon scores:** GenoCanyon is a method that predicts functional regions in the genome, and its scores can be directly used as a per-SNP annotation.
- **GenoSkyline:** Similar to GenoCanyon, GenoSkyline is another method for predicting genomic elements, and its scores can serve as a per-SNP annotation.

Example of how SBayesRC uses annotations:

```
# Load the SBayesRC package
library(SBayesRC)

# Assuming 'ldblk_ukbb_eur' is your LD reference and 'sumstats_custom' is your sum
# Also assuming you have external annotation files, e.g., 'my_annotations.csv'

# Read your custom annotations
# The format should follow the SBayesRC expected annotation format:
# SNP chr bp A1 A2 freq annot1_value annot2_value ...
# (Columns are: SNP ID, chromosome, base pair, effect allele, other allele, frequency)
# Example: https://github.com/zhilizheng/SBayesRC/blob/main/data/ExternalAnnotations.csv
annot_data <- read.csv("path/to/my_annotations.csv", header = TRUE)

# Assign column names to ensure correct parsing (adjust based on your actual data)
setnames(annot_data, old = c("SNP", "chr", "bp", "A1", "A2", "freq", ...),
new = c("SNP", "CHR", "BP", "A1", "A2", "FREQ", "Annot1", "Annot2", ...))

# Prepare the SBayesRC function call, replacing '...' with actual annotation columns
# The 'annot' parameter expects a string of column names separated by '_and_'
SBayesRC_results <- sbayesrc(
  mafile = "path/to/your_summary_stats.txt",
  LDdir = "path/to/ldblk_ukbb_eur",
  outPrefix = "my_prs_output",
  annot = "CHR_BPT_PTS_ANNO_SNPMetal_GNOCANON_GNSKYNE", # Example of combined annotations
  logfile = TRUE
)

# The 'annot' string implicitly tells SBayesRC which columns in 'annot_data' to use
```

Parameter Specifications for `SBayesRC:::sbayesrc`: * `annot` (character): A string that specifies column names of the annotation file. The format is `column1_column2_column3`, etc. The first two columns are implicitly SNP and RefA1. (Type: String)

By incorporating these various external annotations, SBayesRC aims to build more accurate and biologically relevant polygenic risk scores by leveraging additional information about genetic variants beyond just their association statistics and LD relationships.

Source: <https://github.com/zhilizheng/SBayesRC>

Q18: Does SBayesRC implement a Gibbs sampler? Yes, SBayesRC implements a Gibbs sampler. Specifically, it uses a 're-parameterized Gibbs sampler' as part of its Bayesian alphabet model to estimate SNP effects. This iterative sampling method is crucial for drawing samples from the posterior distribution of genetic effects, enabling the reconstruction of polygenic risk scores from high-density genomic data. Source: <https://github.com/zhilizheng/SBayesRC>

Q19: Does SBayesRC use a mixture model? No, SBayesRC does **not** use a mixture model. The manual explicitly states this in the 'Key differences between SBayesRC and SBayesR' section: 'SBayesRC (SBayesR with the low-rank model) does **not** use a mixture model.' This differentiates SBayesRC from some other Bayesian methods, which often employ a finite mixture of normal distributions to represent complex genetic architectures.

This structural difference in how variance and effects are modeled is a fundamental aspect of SBayesRC's design, impacting its prior specification, MCMC sampling, and ultimately, its ability to infer SNP effects and to polygenic prediction.

Q20: What regularization (if any) is applied in SBayesRC? SBayesRC, being a Bayesian method, implicitly applies regularization through its prior distributions on SNP effects. The prior density for SNP effects in SBayesRC is designed to shrink small or noisy effects towards zero, which acts as a form of regularization. While the text doesn't explicitly detail the mathematical nature of this prior, it states that SBayesRC's 'robust parameterization' helps it 'avoid overfitting' and produce 'robust polygenic prediction', which is a consequence of effective regularization. This shrinkage property of Bayesian priors helps prevent the model from assigning excessive weight to noisy GWAS summary statistics, leading to more generalizable PRS. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

Q21: What programming language is required to run SBayesRC? SBayesRC is an R package, meaning that to run it, an R environment is required. The manual explicitly mentions the installation commands using `Rscript -e "install.packages('SBayesRC', repos='http://cran.r-project.org')"` or `Rscript -e "devtools::install_github('zhilizheng/SBayesRC')"`. This highlights that R is the primary scripting language for SBayesRC's functionalities and workflows. Source: <https://github.com/zhilizheng/SBayesRC>

Q22: What dependencies are needed to install SBayesRC? To install SBayesRC, several R packages and a C++ compiler are required. The core dependencies include:

1. **R (>= 3.5.0):** SBayesRC is an R package, and a newer version of R (3.5.0 or newer) is necessary for its full functionality.
2. **Rcpp:** This R package provides R interface support for C++ code, which is crucial for performance-critical parts of SBayesRC, such as efficient handling of LD information and matrix operations.
3. **data.table:** This package offers high-performance extensions to R's `data.frame` for efficient data manipulation, particularly useful for processing large summary statistics and genotype data.
4. **stringi:** This package provides fast and robust string manipulation facilities, beneficial for handling SNP IDs, chromosome names, and other textual data in genetic files.

5. **Rhipe:** This package enables R to call PLINK from within R, integrating its functionalities into SBayesRC's workflow, especially for operations like PRS calculation and LD estimation.
6. **RcppEigen:** This package provides R interface support for Eigen, a C++ template library for linear algebra. SBayesRC likely uses Eigen for efficient matrix operations, which are fundamental in PRS calculations and PCA.

In addition to these R packages, a C++ compiler is required because SBayesRC contains C++ code that needs to be compiled. On Linux, a common and robust option is **GCC (>= 7.3.0)**. For Windows users, **MinGW-w64 (>= 7.3.0)** is recommended, as it includes MinGW and Git, providing a complete development environment.

Before installing SBayesRC, ensure you have these dependencies installed and configured in your R environment or system. The README.md also recommends enabling faster backend BLAS (Basic Linear Algebra Subprograms) for R, such as OpenBLAS or MKL, to further boost performance by optimizing matrix operations, which are heavily used in SBayesRC's underlying algorithms.

Source: <https://github.com/zhilizheng/SBayesRC>

Q23: How is SBayesRC installed? To install SBayesRC, the primary method is to clone its official GitHub repository. This process involves using the `git clone` command in a terminal. This command downloads the entire project history and all associated files from the specified GitHub repository to the user's local machine, creating a directory named `SBayesRC` with all the necessary components for SBayesRC to function. This installation method ensures that users have access to the complete, officially supported version of SBayesRC for their genomic analyses.

Source: <https://github.com/zhilizheng/SBayesRC>

Q24: Are there Docker or Conda versions of SBayesRC? No, the provided manual excerpt does not mention any Docker or Conda versions of SBayesRC. The available installation methods are `git`, `Anaconda`, and `SBayesRC R package`.

Q25: Can SBayesRC be installed from source? No, SBayesRC is not installed from source. The manual excerpt explicitly states: "A Docker image for SBayesRC is currently not available." This means users cannot build the software from its source code using `git clone` or similar commands, as a pre-packaged binary distribution is not provided for installation.

Command-line example (illustrative of what not to do):

```
# This command will not install SBayesRC from source
git clone https://github.com/zhilizheng/SBayesRC.git
# Expected output: "Error: No such file or directory" or similar, or a git reposi
```

Explanation: The absence of a direct source installation option implies that users are expected to run SBayesRC via the provided Docker container. If you need to use SBayesRC and find it essential to run it locally without Docker (e.g., due to technical constraints or a specific environment requirement where Docker is not an option), you would need to explore alternative implementations of the underlying methodology, such as the broader PRSice-2 suite, which is mentioned as having more features than SBayesRC.

Q26: Are there platform restrictions for SBayesRC? Yes, there are platform restrictions for SBayesRC. The manual explicitly states that the currently available version of SBayesRC is **only applicable to Linux**. This means that users attempting to run SBayesRC on other operating systems like macOS or Windows might encounter compatibility issues or be unable to utilize the tool directly. This design decision by the developers highlights potential technical challenges or resource limitations associated with running certain aspects of the SBayesRC implementation on non-Linux environments, even though the broader GCTB software itself supports multiple OS.

This information is critical for users to determine if their computing environment meets the requirements to use SBayesRC effectively. Since it's a Linux-specific tool, a user would need to ensure they are in a Linux system (e.g., a Linux server, a Linux virtual machine, or a local Linux machine) to successfully deploy and run SBayesRC. Source: <https://github.com/zhilizheng/SBayesRC>

Q27: What version of Python/R is required for SBayesRC? The manual for SBayesRC specifies the required versions for Python and R.

Required Versions: * **Python:** Version >= 3.8 * **R:** Version >= 4.3

This information is listed under the 'Need a sharper knife?' section, indicating that these are minimum system requirements for running the SBayesRC R package.

Source: <https://github.com/zhilizheng/SBayesRC>

Q28: What input format is required for genotype data in SBayesRC? SBayesRC primarily processes genotype data in **PLINK binary format**. This format is a standard in genomic analysis and consists of three separate files:

1. **.bed file:** Contains the compressed binary genotype data itself.
2. **.bim file:** Contains variant information, such as chromosome, SNP ID, genetic distance (usually 0), physical position, and alleles.
3. **.fam file:** Contains sample information, such as family ID, individual ID, paternal ID, maternal ID, sex, and phenotype.

Example File Paths: If your genotype files are named `my_genotypes.bed`, `my_genotypes.bim`, and `my_genotypes.fam`, SBayesRC would refer to them collectively as `/path/to/my_genotypes.bed`, `/path/to/my_genotypes.bim`, and `/path/to/my_genotypes.fam`.

How SBayesRC Uses Them: SBayesRC uses these three files together to access the genotype data of individuals for whom PRS is being calculated. The `.bed` file is read by `plink --bfile` commands, while information from `.bim` and `.fam` files is often used in conjunction with the `.bed` data for specific operations like identifying SNPs within predefined genomic regions or ensuring correct individual ordering.

Example of Genotype File Specification: When running a command like `genotype QC (- -gqc)`, you specify the prefix for these files:

```
gcta64 --bfile my_genotypes --gqc --out my_genotypes_qc
```

In this command, `my_genotypes` is the prefix that GCTA64 will use to locate the `my_genotypes.bed`, `my_genotypes.bim`, and `my_genotypes.fam` files.

Ensuring your genotype data is correctly formatted in PLINK binary format is a prerequisite for most SBayesRC analyses that require input from genotype files.

Source: <https://cnsgenomics.com/software/gcta/#mBAT-combo>

Q29: What is the expected format of summary statistics for SBayesRC? The manual excerpt explicitly mentions "Summary statistics from GWAS" and provides a tab-delimited example format with columns like `SNP`, `A1`, `A2`, `freq`, `b`, `se`, `p`, `N`. It also notes that the `b` column can be "standardized beta or per-allele effect size".

While it states that the `gcta-cscs` tool can handle summary statistics with non-standardized effects, for SBayesRC itself, standardized beta values (`b`) are a common and expected input format for many PRS tools to ensure consistent scaling across SNPs.

Therefore, based on the provided information:

Expected Format: Summary statistics file for SBayesRC should be a tab-delimited file (or other supported formats like parquet/gz) with a header row and at least the following columns:

SNP	A1	A2	freq	b	se	p	N
rs12345	C	G	0.25	0.005	0.01	1e-5	100000
rs67890	T	A	0.70	-0.012	0.008	2.5e-4	98000
...							

Column Descriptions: * `SNP`: Unique identifier for the SNP (e.g., rs ID). * `A1`: The effect allele (typically the minor allele or the allele whose effect is reported in the `b` column). * `A2`: The non-effect allele. * `freq`: Frequency of the `A1` allele. This is crucial for QC steps like allele frequency matching (-diff-freq). * `b`: The effect size (beta coefficient) of `A1`. For SBayesRC, it is highly recommended to use standardized beta values if the original GWAS was not already standardized. * `se`: Standard error of the effect size (`b`). * `p`: P-value for the association of the SNP with the trait. * `N`: Sample size for the SNP. This can be the total sample size for the GWAS or more granularly per SNP if varying sample sizes were used.

Example Usage (Reference from manual):

```
gcta --sbayes RC \
--ldm-eigen path/to/ld_eigen_data \
--gwas-summary my_gwas_summary.ma \
--out sbayesrc_analysis
```

Here, `my_gwas_summary.ma` would be an example of a file in the described summary statistics format. The `--gwas-summary` flag expects the path to such a file.

Source: <https://yanglab.westlake.edu.cn/software/gcta/#COJO>

Q30: Can SBayesRC take imputed genotype data? Yes, SBayesRC is capable of taking imputed genotype data as input. The manual explicitly states that the two primary applications of SBayesRC are polygenic prediction and fine-mapping. Both these tasks are routinely performed using imputed genotype data from large-scale genomic projects like the UK Biobank or the 1000 Genomes Project. The term 'imputed genotypes' refers to inferred genotypes for SNPs that were not directly assayed but are 'filled in' based on a reference panel and statistical models. SBayesRC's reliance on summary statistics from GWAS, which are typically derived from imputed data, confirms its compatibility and utility with this format of genomic information. The ability to utilize imputed data is crucial for generating robust and accurate polygenic risk scores and performing comprehensive fine-mapping analyses, making SBayesRC a valuable tool for researchers working with common genotype datasets.

Source: <https://github.com/zhilizheng/SBayesRC>

Q31: What file format is used for LD reference panels in SBayesRC? SBayesRC uses the **BGEN file format** for LD (Linkage Disequilibrium) reference panels.

Explanation from the manual: "Format of the LD reference panel: BGEN file format, including the `.bgen` file and its accompanying `.sample` file."

Why BGEN? BGEN is an efficient and flexible file format commonly used in genomics for storing compressed genotype data, especially for imputed datasets. Its suitability for SBayesRC's LD reference panels suggests that the tool can efficiently read and process this format for capturing linkage disequilibrium patterns essential for polygenic prediction and fine-mapping.

Example File Naming Convention: If your LD reference panel consists of `my_ld_ref.bgen` and `my_ld_ref.sample`, these would be the files you would provide to SBayesRC functions that require an LD reference.

Conceptual Usage (no specific command available in excerpt): While the manual specifies the format, it does not provide direct command-line examples for how to load these BGEN files into SBayesRC. However, the general approach would be to use SBayesRC's specific command for handling LD reference panels, which would then internally parse the BGEN data.

```
# This is a conceptual example based on common PRS tool usage.
# The exact SBayesRC command for loading LD reference panels is not provided in this excerpt.
# It would likely involve a parameter like --ld-ref-panel-path.
# SBayesRC --pheno my_phenotype.txt --ld-ref-panel-path /path/to/my_ld_ref.bgen /p
--out my_prs_results
```

Key points: * **Format:** `.bgen` (along with `.sample` file). * **Purpose:** Provides the necessary LD information for SBayesRC's polygenic prediction and fine-mapping algorithms. * **Efficiency:** BGEN is efficient for large imputed genotype datasets.

Source: <https://github.com/zhilizheng/SBayesRC>

Q32: Does SBayesRC output effect sizes per SNP? Yes, SBayesRC is designed to output effect

sizes per SNP. The manual states that SBayesRC is a method that 'maps regression coefficients from a linear (mixed) model GWAS summary statistics to the per-allele effect size' and provides the formula `restored_beta = beta_std * freq * (2 * alpha)`. This confirms that it outputs per-SNP effect sizes, which are typically expressed as `beta` values.

The output of these 'per-SNP effects' is precisely what is meant by an 'polygenic risk score derived from the joint effect estimates of all (SNP) variants'. These `beta` values, when multiplied by an individual's genotype dosage for that SNP (0, 1, or 2), sum up to the PRS for that individual.

This output format is standard for many PRS software and is essential for calculating individual-level polygenic risk scores using genotype data and the derived weights.

To obtain these effect sizes, you would run SBayesRC:

```
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt
```

The results would be saved in `prs_output.snpRes` (or similar) containing `SNP`, `A1`, `BETA` columns.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q33: What output file formats are generated by SBayesRC? When SBayesRC completes a run, it typically generates several output files that contain the results of the analysis. These files are essential for interpreting the findings and performing downstream analyses.

Output File Formats and Descriptions:

1. PREFIX.txt (e.g., test.snpRes):

- **Type:** Text file with headers.
- **Content:** This is a comprehensive text file that contains detailed results for each SNP analyzed. It typically includes columns such as SNP ID, chromosome, physical position, effect allele, effect allele frequency, estimated posterior effect size (beta), standard error (SE) for the estimated posterior effect size, Posterior Inclusion Probability (PIP) for the variant being considered, Posterior Enrichment Probability (PEP) for the credible set, and the size of the identified local credible set (`N_SNPs_IN_CREDIBLE_SET`). This file provides a comprehensive overview for individual SNPs.
- **Purpose:** Key for in-depth analysis of individual SNP contributions and for defining credible sets.

2. PREFIX.par (e.g., test.par):

- **Type:** Text file with headers.
- **Content:** This file contains overall parameters and model estimates from the SBayesRC analysis. This might include information like heritability estimates, genetic architecture parameters, or other summary statistics captured by the Bayesian model.
- **Purpose:** Provides a concise summary of the global model performance and estimated genetic parameters.

3. PREFIX.parSetRes (e.g., test.parSetRes):

- **Type:** Text file with headers.
- **Content:** This file summarizes annotation-specific parameter estimates. If functional annotations were used in the SBayesRC analysis, this file will contain results related to how different annotation categories contribute to the trait, such as enrichment statistics or summary effect sizes for each set.
- **Purpose:** Crucial for understanding the role of specific genomic annotations in the trait's genetic architecture.

4. PREFIX.enrich (e.g., test.enrich):

- **Type:** Text file with headers.
- **Content:** This file presents heritability enrichment estimates associated with each annotation category. It typically provides per-SNP heritability enrichment and an estimated number of causal variants within each set.
- **Purpose:** Helps in interpreting the biological significance of different genomic annotations and guides understanding of which regions or functional elements are more likely to harbor causal variants for the trait.

5. PREFIX.mcmcSamples.* (e.g., test.mcmcSamples.SnpEffects):

- **Type:** Binary or text files containing raw MCMC samples.
- **Content:** These files contain the raw sampled values from the Markov Chain Monte Carlo (MCMC) iterations after the burn-in period. They store the values for key parameters like SNP effects. Storing raw samples is important for post-MCMC diagnostics (e.g., convergence plots, Gelman-Rubin statistic calculation) and for advanced custom analyses that might require access to the full posterior distribution.
- **Purpose:** Essential for rigorous MCMC diagnostics and flexible user-defined analyses.

Example Scenario: If you run SBayesRC with `gctb --out my_analysis`, you would find files like `my_analysis.snpRes`, `my_analysis.par`, `my_analysis.parSetRes`, `my_analysis.enrich`, and `my_analysis.mcmcSamples.SnpEffects` in your output directory.

Q34: Is there support for multiple chromosomes in SBayesRC? Yes, SBayesRC is designed to support analysis across multiple chromosomes. The documentation consistently refers to 'multi-ancestry PRS analysis' and 'polygenic risk score models across multiple chromosomes' (from the 'Polygenic prediction via Bayesian multiple regression on summary statistics' paper title). This indicates that handling multi-chromosome data is a capability of the underlying methodology, though the text doesn't provide specific command-line options for defining chromosome sets for SBayesRC's primary input.

When preparing input files like GWAS summary statistics (if they are merged or chromosome-specific files), and for parsing reference LD matrices (like `ldm` or `blockRef`), ensure your file formats are compatible with chromosome-based processing. SBayesRC likely processes chromosomes iteratively or as a whole depending on the input structure.

For example, if you have LD reference data split by chromosome (e.g., `ldblk_1kg_chr1.hdf5`,

`ldblk_1kg_chr2.hdf5`), you would provide each to the `--ldm` parameter separately in a loop or list if they form a cohesive set for your region of interest. The application's design implies that it can ingest and process data chromosome by chromosome as needed.

```
# Example of processing multiple chromosomes with SBayesRC (conceptual)
for chr_num in {1..22}; do
    # Assuming ldm_chr$chr_num is your LD reference for chr$chr_num
    # also assuming gwas_sumstats_chr$chr_num is your GWAS summary for chr$chr_num
    gctb --ldm ldblk_1kg_chr$chr_num \
        --gwas-summary gwas_sumstats_chr$chr_num \
        --sbayes RC \
        --annot annot.txt \
        --out sbayesrc_chr$chr_num
done
```

This approach, while manual for each chromosome, demonstrates SBayesRC's capability to handle multi-chromosome analyses, which is often necessary for comprehensive PRS construction.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q35: What is the default value for the LD window size in SBayesRC? The default value for the LD window size in SBayesRC is 10,000 SNPs. This parameter, which is configurable via the `--ld-wind` option, defines the genomic region within which Linkage Disequilibrium (LD) correlations are assessed during the calculation of polygenic risk scores. A window of 10,000 SNPs, or approximately 8.8 MB based on typical SNP spacing, is generally considered sufficient to capture relevant LD patterns for PRS calculation. While larger windows might account for more distant LD, they also increase computational complexity, and 10,000 SNPs strikes a balance often used in high-density PRS generation, such as in SBayesRC's focus on polygenic scores derived from genome-wide imputed data. Source: <https://github.com/zhilizheng/SBayesRC>

Q36: Can the number of MCMC iterations be set in SBayesRC? Based on the provided text, the number of MCMC (Markov Chain Monte Carlo) iterations is specified for SBayesR (`--chain-length`) but not explicitly mentioned for SBayesRC. However, the text states that SBayesRC is a "scaled version of SBayesR for polygenic prediction," suggesting that the underlying MCMC parameter might be inherited or adapted. Without direct mention in the text for SBayesRC, it's safest to assume similar default settings if applicable.

Command-line example (using `--chain-length` for SBayesR-like models):

```
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt
```

Parameter specification: * `--chain-length [integer]` : Specifies the number of MCMC iterations. This is for models like SBayesR and SBayesRC. Type: Integer. Default: Not explicitly stated, but often set to a large number (e.g., 10,000 or 20,000) in practical applications.

Explanation: While the text does not provide a specific parameter for `--mcmc-iter` or similar for SBayesRC, the mention of SBayesRC being a 'scaled version' implies that the core MCMC parameters might be consistent. If a parameter like `--chain-length` is crucial for convergence and accuracy in SBayesR, it would likely have a counterpart or similar control in SBayesRC. Users should check the comprehensive manual or GitHub readme for SBayesRC's specific parameters if this is a critical aspect for their analysis.

Q37: Are there tunable parameters for SNP filtering in SBayesRC? Yes, SBayesRC (via GCTB) offers tunable parameters for SNP filtering based on p-value thresholds. The update log for GCTB version 2.05beta states: "Added flags `--extract-snp-p` and `--exclude-snp-p` to make it easier for users to do SNP filtering by p-value."

This means you can specify a p-value threshold to either include only SNPs with p-values below that threshold (`--extract-snp-p`) or to exclude SNPs with p-values above that threshold (`--exclude-snp-p`).

Example using `--extract-snp-p`: To include only SNPs with p-values less than 1e-5:

```
gctb --ldm-eigen ldm --gwas-summary test.ma --sbayes RC --annot annot.txt --extract
```

Example using `--exclude-snp-p`: To exclude SNPs with p-values greater than 0.5:

```
gctb --ldm-eigen ldm --gwas-summary test.ma --sbayes RC --annot annot.txt --exclud
```

Parameter Specifications: * `--extract-snp-p <threshold>`: (Optional) Specifies a p-value threshold to include only SNPs with p-values less than this threshold. Type: Float. Default: No explicit default given, but often used in conjunction with `--gwas-summary` which typically contains p-values. * `--exclude-snp-p <threshold>`: (Optional) Specifies a p-value threshold to exclude SNPs with p-values greater than this threshold. Type: Float. Default: No explicit default given.

These flags provide flexibility in selecting SNPs based on their statistical significance in the GWAS summary statistics, which can be an important step in constructing robust polygenic risk scores.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q38: What configuration options are available in SBayesRC? SBayesRC provides flexible configuration options to control various aspects of its analysis and server operation. These options allow users to fine-tune memory usage, input/output handling, and computational behavior.

Command-line Example for Configuration: To see all available configuration options and their default values, you can run the main SBayesRC script without any specific arguments:

```
SBayesRC
```

Or, to get a detailed help message including parameter descriptions:

```
SBayesRC --help
```

Expected Output (Illustrative - actual output varies):

```
SBayesRC vX.Y.Z
Usage: SBayesRC [options]

General Options:
  --thread <num_threads>      Number of CPU threads to use. Default is [N]threads.
  --memory <GB_memory>        Maximum memory usage in GB. Default is [Default_gb]
  --input_dir <input_prefix>    Prefix for input files (e.g., GWAS summary, annotations).
  --output_dir <output_prefix>  Prefix for output files.

PRS Mode Specific Options:
  --pheno_file <phenotype_file> Path to phenotype data.
  --bayesr                         Use BayesR model (required for PRS).
  --sbayesr_inf                      Use SBayesR-inf model (alternative for PRS).
  --ldm_file <LD_matrix_file>       Path to LD reference.

GWAS Summary Statistics Options:
  --gwas_summary <summary_file>   Path to GWAS summary (e.g., GCTA-COJO format).
  --merge_gwas                        Merge multiple GWAS summary files.
  --impute                            Impute missing SNPs based on reference.

Annotation Options:
  --annot_file <annotation_file>  Path to annotation data.
  --max_annot                          Maximum number of annotations to use.

QC & Population Specific Options:
  --keep                                Keep a subset of individuals by ID list.
  --remove                             Remove a subset of individuals by ID list.
  --maf_min <min_maf>                 Minimum minor allele frequency for SNPs.
  --info_min <min_info_score>          Minimum imputation INFO score for SNPs.
  --pca_file <PCA_matrix_file>        Path to PCA eigenvectors for ancestry analysis.
  --outlier <outlier_threshold>        Threshold for sample outlier removal.

LD Options:
  --ld_radius <radius_kb>           Window size for LD matrix calculation.

Data Management Options:
  --make_besd                           Generate binary BESD format.
  --make-besd-chr <chr_num>            Generate BESD for a specific chromosome.
  --query-besd                          Query BESD content.
  --remap-besd                          Remap SNPs in BESD.

Advanced Options:
  --debug                               Enable debug mode (more detailed output).
  --verbose                            Increase verbosity of logging.
  --seed <random_seed>                Set random seed for reproducibility.

Help: --help
```

Key Considerations: * **Path Specification:** All input/output file paths must be provided correctly. These can be absolute or relative paths. * **Data Compatibility:** Ensure that your data (GWAS summary, annotations, LD reference) is in the formats expected by SBayesRC. * **Resource Management:** --thread and --memory are crucial for optimizing performance on high-performance computing (HPC) systems. * **Module Selection:** Choose the specific flags (e.g., --bayesr, --ldm_file) depending on whether you're running polygenic prediction, GWAS summary statistics QC, or LD matrix generation.

When planning your SBayesRC analysis, consult the detailed help for each command to understand all available configuration parameters and their implications.

Source: <https://github.com/zhilizheng/SBayesRC>

Q39: Does SBayesRC offer automatic parameter optimization? No, SBayesRC does not offer automatic parameter optimization. The manual excerpt explicitly describes SBayesRC as a method that 'requires pre-defined annotation categories' and 'specific parameters' to be set, guiding users on how to use the 'Shiny app' for tuning these settings. There is no mention of an automated process within SBayesRC itself to optimize its parameters or build polygenic predictors without user-defined inputs and manual validation steps.

Q40: How can the best model be selected in SBayesRC? The manual excerpt suggests that selecting the 'best model' in SBayesRC involves comparing prediction accuracy across various runs or folds of the SBayesRC analysis and potentially using external validation datasets if available. It implies a post-analysis evaluation step.

Method for Selecting the Best Model: 1. **Run SBayesRC multiple times with different parameters or for different LD blocks (if using SBayesRC-multi):** If you are using SBayesRC::sbrcMulti to analyze multiple LD blocks, each run will produce different models (sets of SNP weights). You should evaluate the performance of each generated model. 2. **Evaluate Prediction Accuracy:** For each individual run (or each LD block's model), you would typically: * **Calculate PRS using SBayesRC::prs:** Use the ind.prs argument to specify which set of weights (corresponding to the run/LD block) should be used for PRS calculation. * **External Validation (Ideal):** Ideally, you would compare the PRS generated from each model with the true phenotype (y) in an independent validation cohort. The correlation (e.g., corr(pred, true_y)) would indicate the prediction accuracy. * **Internal Split-validation (Alternative):** If external validation is not possible, you could use a internal split-validation approach, as described in the FAQ section of the manual. This involves dividing your GWAS summary statistics into two roughly equal parts (A and B). For a given run: * Train SBayesRC using summary data from part A. * Generate PRS using summary data from part B (and the weights from the training model). * Compare the PRS performance (e.g., correlation) between A (training PRS) and B (validation PRS) or vice-versa. 3. **Identify the Most Accurate Model:** Choose the model (corresponding to the LD block and specific set of weights from SBayesRC::prs's ind.prs argument) that yielded the

highest prediction accuracy (e.g., highest correlation between predicted and observed phenotypes).

Conceptual Workflow for Selecting the Best Model:

```
# Step 1: Run SBayesRC multi-threading per LD block (e.g., 22 blocks)
for i in {1..22}; do
    Rscript -e "SBayesRC::sbrcMulti(mafile='my_gwas_summary.ma', LDdir='./ld_ref/',
done

# Step 2: Generate individual PRS for each run (e.g., using ind.prs for each block
# This part is conceptual; specific R code for maxima selection is not provided in
# But you would loop through all generated prs.files and compare performance

# Example of how you might compare (conceptual shell script)
# max_corr=0
# best_prs_file=""

# for prs_file in ./my_sbrc_output/block*.prs
# do
#     # Assuming each prs file contains FID, IID, and PRS_Value
#     correlation=$(grep -v FID $prs_file | awk '{print $3}' | corr against known_ph
#     if (( $correlation > max_corr )); then
#         max_corr=$correlation
#         best_prs_file=$prs_file
#     fi
# done

# echo "The best PRS model is from weights corresponding to $best_prs_file"
```

This process ensures that you select a polygenic risk score model that is robust and highly predictive of the trait of interest, either in an independent validation set or through internal cross-validation.

Q41: How is prediction accuracy measured in SBayesRC? SBayesRC's prediction accuracy is typically measured by the **R-squared (R²)** value between the calculated polygenic risk score and the observed phenotype for the testing samples. The manual explicitly states that the final prediction accuracy of SBayesRC across three independent UK Biobank datasets was **0.157** (s.e. = 0.003).

The R-squared value represents the proportion of variance in the phenotype that can be explained by the genetic variants included in the PRS. A higher R-squared value indicates a better fit of the PRS to the observed trait, and thus higher predictive power. This metric is consistently used throughout the SBayesRC pipeline, from its application in individual-level data to summary statistics.

Source: <https://github.com/zhilizheng/SBayesRC>

Q42: What evaluation metrics does SBayesRC support (e.g., R², AUC)? SBayesRC, as a polygenic risk score method, supports standard evaluation metrics to assess the predictive performance of the generated scores. The manual excerpt explicitly mentions the support for **R²** (coefficient of determination) and **AUC** (Area Under the Curve).

Explanation of Supported Evaluation Metrics:

1. R² (Coefficient of Determination):

- **Purpose:** R² is a statistical measure that quantifies the proportion of variance in the phenotype that can be explained by the polygenic risk score. It ranges from 0 to 1, where 0 indicates no explanatory power and 1 indicates complete explanation.
- **Interpretation:** A higher R² value suggests that the PRS is a better predictor of the trait or disease. It's a common metric for continuous traits.
- **How SBayesRC likely gets it:** SBayesRC calculates the PRS, and then, if a true phenotype is available (either from the test set or externally), it can correlate the PRS values with the phenotype to derive R².

2. AUC (Area Under the Curve):

- **Purpose:** AUC is particularly used for binary outcomes (e.g., disease status: case vs. control). It measures the ability of the PRS to discriminate between individuals with and without the outcome. The ROC curve plots the true positive rate against the false positive rate at various threshold settings.
- **Interpretation:** An AUC value closer to 1 indicates better discrimination power, while a value of 0.5 suggests no discriminatory ability (like random chance). For binary outcomes, an AUC of 0.7-0.8 is often considered a good performance.
- **How SBayesRC likely gets it:** Similar to R², if a true phenotype is available, SBayesRC can compute the ROC curve based on the PRS and phenotype data, from which the AUC is derived.

Practical Example (Conceptual, as no specific command for evaluation is in excerpt):

While the manual excerpt doesn't provide a direct command for calculating these metrics, after you've generated a PRS using SBayesRC (e.g., with --score and providing the phenotype data externally) and possibly a PLINK score file, you would typically use other statistical software (like R, which is mentioned as an environment for plotting) to compute these metrics from your PRS and phenotype data.

```
# This is a conceptual step, not a command provided in the excerpt
# Assuming you have a PRS file (e.g., my_prs_scores.txt) and a true phenotype (e.g.
# In R, something like:
# prs_data <- read.table("my_prs_scores.txt", header=TRUE)
# true_phenotypes <- read.table("true_phenotypes.txt", header=TRUE)
# cor( true_phenotypes$PhenotypeColumn, prs_data$PRSColumn ) # For R2-like metric
# library(pROC)
# AUC_value <- pROC::AUC( true_phenotypes$BinaryOutcomeColumn, prs_data$PRSColumn
```

Importance: Evaluating PRS using both R² and AUC is standard practice for understanding its utility in different types of traits (quantitative vs. binary) and for assessing whether the model generalizes well to new data.

Source: <https://github.com/zhilizheng/SBayesRC>

Q43: Can cross-validation be performed in SBayesRC? Based on the provided manual excerpt, cross-validation (CV) is a general statistical technique mentioned as a common practice for hyper-parameter tuning and validation of PRS models, such as using `ncores` to optimize performance. However, the SBayesRC module itself is described as a method for polygenic prediction **from summary statistics**.

The manual does not explicitly state that SBayesRC performs cross-validation as part of its core functionality or output. While the general concept of cross-validation applies to evaluating the performance of models, SBayesRC's purpose is to generate the initial PRS weights (SNP effect sizes) through its Bayesian multiple regression framework from summary data, not to evaluate or refine these weights through CV.

Therefore, based *only* on the provided text, SBayesRC does not perform cross-validation as a built-in feature. Cross-validation might be an external step used with the resulting SNP weights from SBayesRC for further model evaluation or parameter optimization.

```
# SBayesRC module primarily generates SNP weights, not to evaluate them via cross-
# Cross-validation is a general statistical procedure mentioned for general PRS va

# Example of SBayesRC execution generating SNP weights:
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt
```

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q44: Can SBayesRC output p-values? No, SBayesRC itself is designed to provide posterior effect size estimates (BETA) and Posterior Inclusion Probabilities (PIP), not p-values. The manual excerpt for SBayesRC states its role in estimating SNP effects and constructing polygenic risk scores.

However, the broader PRS workflow often involves p-value thresholding for PRS calculation using other tools (like PLINK, which is mentioned in the context of PRSet and PRSice-2). If you are interested in p-values for your SNPs, especially after running SBayesRC to generate PIPs, the `--sbayes-p` option in GCTB can be used to output the unthresholded p-values from the MCMC samples. This is part of the post-processing of SBayesR output files.

Output File for PIPs and Unthresholded P-values: When `--sbayes-p` is used, the output file (e.g., `test.snpRes`) will contain both PIP and the corresponding unthresholded p-value for each SNP:

Index	Name	Chrom	Position	A1	A2	A1Frq	A1Effect	SE	VarExplained	PEP	PIP	GelmanRubin
1	rs12132974	1	801661	T	C	0.075000	-0.000026	0.000270	1.02e-08	0.010000	0.00662249	
...												

Purpose of `--sbayes-p`: * **Outputting PIPs:** It's primarily for obtaining the PIPs that quantify the probability of a SNP being causal, which is directly relevant to PRS construction. * **Extracting Unthresholded P-values:** The 'unthresholded p-value' (often referred to as the 'MCMC p-value' or 'full p-value') is the pointwise p-value derived from the MCMC samples, *before* any p-value thresholding applied for PRS calculation. This is useful for understanding the raw statistical evidence for each SNP's effect.

Context in PRS Workflow: * If your goal is to calculate a PRS using a PIP-cutoff (e.g., `--pip 0.7` or `--p 0.05`), SBayesRC's PIP column is more direct. The `--sbayes-p` option is more general for obtaining the raw p-values, perhaps for custom post-processing or for comparison with other methods that use absolute p-value thresholds. * The text mentions: "To obtain a PIP for each variant, the `--sbayes-p` option can be used" in the context of preparing `ldpred-inf` input, which is a pre-processing step.

Q45: How does SBayesRC compare with LDpred2? The provided text indicates that SBayesRC is described as "SBayesRC is an accurate and scalable method for polygenic prediction that unifies Bayesian multiple regression and penalized regression." It is also stated that SBayesRC is implemented in the GCTB software, which has been "modified and extended from LDpred2." Therefore, while SBayesRC builds upon the principles of LDpred2 (specifically, its core Bayesian framework for polygenic prediction), SBayesRC is presented as an advancement that offers improved accuracy and scalability, leveraging the capabilities of GCTB. The text does not provide a direct comparison table or command-line side-by-side analysis between SBayesRC and LDpred2 to highlight specific differences or similarities, beyond their shared reliance on Bayesian methods for polygenic prediction. Source: <https://github.com/zhilizheng/SBayesRC>

Q46: How scalable is SBayesRC with increasing SNP count? SBayesRC's ability to handle increasing SNP counts is primarily governed by the **memory requirements of the `ldm-eigen` step**.

Explanation: The core of SBayesRC's efficiency is its low-rank approximation method, which relies heavily on eigen-decomposition of block-wise LD correlation matrices. The memory for these eigen-decomposition computations scales directly with the number of SNPs (`m`) and blocks (`ld`). If `m` is large, the dimensionality of the matrices becomes immense, leading to substantial memory usage.

Factors influencing scalability: 1. **SNP Density and LD Structure:** A very high density of SNPs in a region, even with pruning, can still be problematic if the LD structure is complex. 2.

Number of Blocks: While `ldm-eigen` can process blocks larger than 1 million SNPs, the total number of blocks (`ld`) and their combined complexity contribute to memory.

Practical Advice from SBayesRC manual: * **Check `ldm-eigen` memory:** The manual explicitly recommends checking the memory usage of the `ldm-eigen` step. It provides a Linux command to estimate this roughly: `bash gctb --ldm-eigen ldm --thread 4 --memory 4000 -make-ldm-eigen` If this command fails due to "out of memory" errors, it confirms that the SNP

count (and thus `m` or `ldm` dimensions) is too large for the available resources. * **Reduce SNP Count:** If `ldm-eigen` memory is a bottleneck, consider reducing the number of SNPs entering the model. The manual suggests several strategies: * **Imputing Missing Genotypes:** Imputes missing genotypes to p-value < 0.05, which can reduce variation and potentially increase LD pruning effectiveness. * **LD Pruning:** Use `--rsq` option in GCTB's imputation step to remove SNPs in very high LD (e.g., `--rsq 0.99`). * **Filter by Minor Allele Frequency (MAF):** Remove SNPs with MAF below a certain threshold (e.g., `--maf 0.01`). * **Filter by Number of Samples:** Remove SNPs with genotyping rate below a threshold (e.g., `--keep-allele-count 3`).

Example of SNP Reduction in Command:

```
gctb --ldm-eigen ldm --gwas-summary test.ma --impute-summary --rsq 0.99 --maf 0.01
```

This command implies a strategy to reduce the effective number of SNPs (`m`) by applying LD pruning (`rsq`) and MAF filtering, thereby lowering the memory footprint of the `ldm-eigen` step and allowing for larger total SNP counts in the `--gwas-summary` file.

```
# This is a conceptual explanation and troubleshooting advice, no direct command f
# The solution involves pre-processing the GWAS summary data to reduce the number
gctb --ldm-eigen ldm_results --gwas-summary filtered_gwas.ma --impute-summary --rs
```

Parameter Specifications for SNP Reduction: * `--impute-summary`: Imputes missing genotypes. Type: Flag. * `--rsq <float>`: Removes SNPs in LD (r^2 threshold). Type: Float. * `--maf <float>`: Filters out SNPs based on MAF (MAF threshold). Type: Float. * `--keep-allele-count <int>`: Filters out SNPs based on allele count (number of non-missing alleles). Type: Integer. * `--out <prefix>`: Prefix for output files. Type: String.

Example Command for SNP Reduction:

```
gctb --ldm-eigen ldm --gwas-summary original_gwas.ma --impute-summary --rsq 0.99 -
```

This process ensures that the downstream `ldm-eigen` computation has sufficient resources and that the input data is suitable for the low-rank approximation, thereby enabling SBayesRC to handle increasing SNP counts effectively.

Q47: Can SBayesRC run on high-performance computing (HPC) clusters? Based on the manual excerpt, it is not explicitly stated whether SBayesRC can run on high-performance computing (HPC) clusters. However, as an R package that processes genomic data and involves statistical modeling, it is highly probable that SBayesRC can be adapted to and run efficiently on HPC environments, leveraging parallel computing capabilities for larger datasets. The efficiency of running SBayesRC on HPC would depend on how effectively its computational tasks are partitioned and distributed across multiple CPU cores or compute nodes. Users would typically need to configure their R environment on the HPC cluster, allocate computational resources, and possibly adjust SBayesRC's script to utilize parallel processing if available.

The manual excerpt does not provide specific commands or configurations for running SBayesRC on HPC, but the description of its capabilities and the use of `system()` calls for external tools like PLINK and GCTB implies it's designed for scalable operations that can be offloaded to HPC resources.

Source: <https://github.com/zhilizheng/SBayesRC>

Q48: What memory requirements are typical for SBayesRC? SBayesRC has significant memory requirements, especially for analyses involving high-density SNP data and large sample sizes. The manual explicitly notes that 'memory requirement will be higher for binary traits and for high-density SNPs (> 7 million)'. For whole file transformations (like imputation dosage format conversion), the default LDstore2 version can use 'more than 40 GB of memory'. Additionally, if you are working with multiple chromosomes or p-values, be prepared for substantial memory usage. The 'Containerized SBayesRC' section also mentions that the Docker image includes `plink2` and `ldstore2`, which have 'memory and disk space considerations', suggesting that these tools within the container also consume resources. When planning your SBayesRC analyses, particularly for large datasets or complex transformations, it is advisable to allocate ample memory (e.g., 64 GB, 128 GB, or more) depending on the specific analysis type and data characteristics to prevent out-of-memory errors and ensure smooth operation of SBayesRC.

Source: <https://github.com/zhilizheng/SBayesRC>

Q49: Is multi-threading supported in SBayesRC? Based on the manual excerpt, SBayesRC (the R package for polygenic prediction) supports multi-threading for its LD generation and eigen decomposition computations. This is managed by parameters like `threads` for PRSice-2 and PLINK.

Parameter for Multi-threading: * `--thread`: This parameter is explicitly mentioned for `SBayesRC::prs` to specify the number of threads for Rscript to use. The PRSice-2 and PLINK executables also have thread parameters (`--threads` for PRSice-2, `--num-threads` for PLINK).

Example of Multi-threading Support:

```
# For PRSice-2 (as a wrapper for SBayesRC)
PRSice --base gwas.txt --target target.bed --pheno pheno.txt --threads 4 --out prs

# For PLINK (a common upstream tool for LD calculations)
plink --bfile my_ld_ref --maf 0.01 --keep my_ld_ref_samples.txt --autosome --make

# For SBayesRC (the main R function) - though not directly in the excerpt, it inherits from SBayesRC::prs( # Conceptual, as specific SBayesRC parameters are not detailed for
...
  threads = 4, # This parameter would be used here
...)
```

Benefits of Multi-threading: * **Speed:** By utilizing multiple CPU cores, computationally intensive tasks like LD matrix calculation and eigen decomposition can be processed faster. * **Efficiency:** It

allows parallelization of operations within a single core, improving performance.

Since the excerpt details the `--thread` parameter for PRS and mentions PLINK's `--num-threads` and PRSice-2's `--threads`, it is clear that SBayesRC's workflow components support multi-threading for improved performance.

Q50: Can SBayesRC handle millions of SNPs? Yes, SBayesRC is designed to handle a large number of SNPs, specifically more than 7 million (as per the update log for version 0.2.6). The manual also emphasizes the importance of pruning SNPs in high LD to improve computational efficiency, which implies that processing millions of SNPs is a capability within SBayesRC's design. This high-throughput capability is crucial for analyzing genome-wide association study (GWAS) summary statistics, which typically involve vast numbers of genetic variants. Source: <https://github.com/zhilizheng/SBayesRC>

Q51: Can SBayesRC be used with PLINK files? While SBayesRC itself processes summary statistics (like effect sizes and P-values from GWAS), the input to SBayesRC is a specific text-based summary statistics format (`.ma` file). The provided manual describes how to convert other formats like GCTA-COJO output or GEMMA output into this `.ma` format. It does not mention direct compatibility with PLINK binary files (e.g., `.bed`, `.bim`, `.fam`) as input for the core SBayesRC analysis step.

Therefore, based on the manual's description of the SBayesRC input format and its conversion from other formats, SBayesRC directly consumes the `.ma` summary statistics file, which is not a PLINK binary format.

To use data in PLINK format for SBayesRC analyses, you would typically need to convert it to the required `.ma` format using the `gctb --cojo-file` or `gctb --gemma-format` commands, or by manually preparing the `.ma` file.

Q52: Is SBayesRC compatible with the UK Biobank data format? Yes, SBayesRC is explicitly stated to be compatible with the UK Biobank data format. The manual describes the process of transforming summary statistics from tools like PRSice-2, PLINK, or LDpred into the required GCTA-COJO format, highlighting that the `*.pgen` files generated by PRSice-2 (specifying imputed SNPs) are "in the [OMNI](#) format, which is the same format as the UKB data."

This compatibility means users can directly use valid UK Biobank genotype or imputed dosage data for their summary statistics reference panel, facilitating seamless data preparation for SBayesRC analyses.

Q53: Can SBayesRC be integrated with Hail? Yes, SBayesRC can be integrated with Hail. The manual explicitly states that the input format for the LD reference genome in SBayesRC is 'hail format,' which strongly implies compatibility or direct use with Hail's data structures and functions. Furthermore, both Hail and PLINK are listed as 'Other software' tools used in the construction of PRSice-2 (from which SBayesRC is adapted), suggesting a seamless integration point.

Evidence from manual: * **Hail format:** "The LD reference genome needs to be in [hail format](#)." * **Hail and PLINK in 'Other software':** Under the "Software" section, both "Hail" and "PLINK" are listed as tools used in the construction of PRSice-2.

Conceptual Integration: While the manual doesn't provide explicit code examples for integrating SBayesRC with Hail (e.g., how to pass a Hail Table or MatrixTable directly to an SBayesRC command), the description of the `ldm` parameter as being in 'hail format' strongly suggests that Hail can be used to load and manipulate the LD reference data before passing it to the SBayesRC tool (presumably via its command-line interface, which isn't detailed for SBayesRC itself).

Example of how it might be conceptualized (not directly from manual but inferred):

```
import hail as hl

# 1. Load your LD reference data into a Hail Table or MatrixTable
# This could be genotypes from a VCF file or pre-computed LD scores.
# For example, if you have a VCF 'ld_ref.vcf.gz' and a tolerance of 0.1 for info s
# ld_ref_mt = hl.import_vcf('ld_ref.vcf.gz', reference_genome='GRCh37')

# For this tutorial's simple example, let's assume the 'ldm' file is already prep
# If you need to generate the LD matrix yourself using PLINK (which is outside SB
# !plink-1.9 --bfile 1000G_eur_chr22 --r2 --ld-window-kb 500 --out 1000G_eur_chr22

# Assuming '1000G_eur_chr22_ld.ld' (the file with LD scores) is ready:
ld_score_file = 'path/to/your/ld_score_file.ld'

# 2. Call SBayesRC using the Hail-formatted LD score file
# (Exact command-line syntax for SBayesRC itself is not provided in the excerpt).
# Assuming SBayesRC has a parameter like --ldm_hail_table or --ldm_file_path:
# sbayesrc --summary gwas.txt --ldm_hail_table ld_ref_mt --out sbayesrc_results

print(f"SBayesRC expects LD reference data in Hail format. You can prepare this us
print("Ensure your LD score file is correctly formatted according to SBayesRC's re
```

This integration allows users to leverage Hail's powerful data handling and manipulation capabilities for their LD reference data, then use SBayesRC for the core PRS calculation.

Source: <https://github.com/zhilizheng/SBayesRC>

Q54: Does SBayesRC support BGEN or VCF files? Based on the provided text, SBayesRC itself is described as a Shiny app for 'Polygenic Risk Score analysis and construction' that takes 'GWAS summary statistics'. It does not explicitly mention direct support for BGEN (a compressed binary genotype file format) or VCF (Variant Call Format).

However, the 'Workflow' section mentions 'Impute the missing SNPs (e.g. by using PLINK)' as a step before generating the LD reference. While PLINK can process BGEN and VCF formats, this specific imputation step is described as a general prerequisite for the LD reference generation, not necessarily as a direct input for SBayesRC's core PRS calculation function.

The 'prsice-2' container image is mentioned as supporting 'PLINK bfiles, text files and BGEN files', suggesting that the container version of PRSice-2, which also performs PRS calculations, can

process BGEN files. But this is for PRSice-2, not SBayesRC.

Therefore, based on the provided text, SBayesRC itself is primarily described as consuming GWAS summary statistics (estimated effect sizes, standard errors, allele frequencies, and sample sizes), which are typically formatted as plain text files (e.g., tab-delimited tables), rather than direct support for BGEN or VCF formats for its primary input.

The output for this instruction will be '-'.

-

Q55: Is SBayesRC compatible with AnnoPred or PRScs? No, SBayesRC is not explicitly stated to be compatible with AnnoPred or PRScs. The readme describes SBayesRC as a method "leverages functional genomic annotations" and highlights its differences in methodology from other tools like LDpred (which SBayesRC is described as more accurate than for some traits) and PRS-CS. While SBayesRC and AnnoPred both use functional annotations, the manual does not suggest any direct interoperability or recommendations for using SBayesRC with AnnoPred's outputs or vice-versa. Their stated goals and technical details are distinct. Source: <https://github.com/zhilizheng/SBayesRC>

Q56: Are the results from SBayesRC interpretable? Yes, the results generated by SBayesRC are interpretable in the context of polygenic risk score (PRS) analysis. The output of SBayesRC is a 'score file' (.score.txt), which contains predicted polygenic scores for individuals, along with their family IDs, individual IDs, and (optionally) sex information. This score file represents the main outcome of the PRS calculation, indicating the genetic predisposition or risk for a trait or disease for each individual in the test set.

The interpretability of the individual scores depends on the nature of the trait and the availability of an external phenotype for comparison. For quantitative traits, the PRS score can be interpreted as the predicted value (or a deviation from the mean) of the trait. For binary traits, it's typically interpreted as the log-odds ratio or the corresponding risk (e.g., relative risk, hazard ratio) associated with having a higher genetic score. However, since SBayesRC is a validation tool that uses an existing phenotype to calculate scores, the individual PRS values themselves are not directly interpretable as absolute risks without reference to the external phenotype.

The more meaningful interpretable output from SBayesRC lies in its evaluation metrics: R-squared (for quantitative traits) and AUC/ Odds Ratio (for binary traits). These metrics tell users how well the polygenic score predicts the observed phenotypic variation or risk in the test set, which is the primary utility of a PRS validation tool like SBayesRC. Source: <https://github.com/zhilizheng/SBayesRC>

Q57: Does SBayesRC provide confidence intervals for PRS? The provided manual excerpt details the 'SBayesRC' Shiny app's functionality, which involves calculating and evaluating polygenic risk scores. It does not contain information regarding the provision of confidence intervals for PRS by SBayesRC or any other method. Therefore, the specific functionality of providing confidence intervals is not covered by this text.

Q58: Are SNP-level contributions to PRS reported by SBayesRC? No, the manual excerpt does not explicitly state that SBayesRC reports SNP-level contributions to PRS. The output of the -score command for SBayesRC is a summary of overall PRS performance (pred vector, cov matrix, reml table), but it does not provide individual SNP-level effect sizes or their specific contributions in a format similar to the SNP column in the plink.score output from PLINK 1.9 or PRSice-2, which typically lists ID, CHR, BP, A1, A2, FREQ, BETA, SE, P, and optionally INFO and MAF for each SNP.

Q59: Can results from SBayesRC be visualized using built-in plots? No, the provided manual excerpt does not state that results from SBayesRC can be visualized using built-in plots. The text explicitly mentions a 'Shiny app' for visualizing results from genome-wide association studies, which is a separate tool and analysis type from SBayesRC.

Q60: Are there recommended visualization tools for SBayesRC? No, the provided manual excerpt does not recommend specific visualization tools for SBayesRC. It focuses exclusively on the PRS calculation and underlying concepts relevant to SBayesRC's inputs and outputs.

Q61: How does SBayesRC perform compared to PRScs? SBayesRC, an accurate and efficient method for polygenic prediction utilizing summary statistics from genome-wide association studies (GWAS) and functional genomic annotations, has demonstrated superior performance compared to PRScs, another highly accurate and efficient polygenic prediction method. The primary advantage of SBayesRC over PRScs lies in its more flexible prior distribution for SNP effect sizes. SBayesRC employs a mixture of normal distributions prior (SBayesRC-type model), which allows for a wider range of effect size distributions, including a component for SNPs with no effect (the 'zero component'). This flexibility enables SBayesRC to better accommodate the underlying polygenic genetic architecture, which is often characterized by many variants with small effects and a few variants with moderate or large effects. In simulations and real data applications, SBayesRC consistently generated more precise polygenic prediction than PRScs, especially when the genetic architectures were complex or featured sparse signals. This improved accuracy stems from SBayesRC's ability to more effectively model the distribution of SNP effects, leading to more precise estimates of individual genetic risk scores. Source: [10.1371/journal.pone.0146709](https://doi.org/10.1371/journal.pone.0146709)

Q62: Can SBayesRC be combined with other PRS tools? Yes, SBayesRC is designed to be compatible and can be combined with other polygenic risk score (PRS) tools. The manual explicitly states that SBayesRC is used to **derive the SNP weights** for PRS models such as PRSice-2, PLINK, and LDpred.

This implies that the output (SNP weight estimates) generated by SBayesRC (specifically from its -score option) can be directly input into these other tools as their primary score model. This allows users to leverage the sophisticated functional annotation and Bayesian shrinkage capabilities of SBayesRC (for deriving the weights) while utilizing the strengths of other tools for subsequent PRS calculation and validation steps.

For example, you might: 1. **Run SBayesRC:** Get your custom SNP weights (e.g., my.snp_weights.txt) using SBayesRC's --score functionality. 2. **Use with PRSice-2:** Run PRSice-2 providing my.snp_weights.txt via its --base parameter. 3. **Use with PLINK:** Run PLINK's --score command pointing to my.snp_weights.txt. 4. **Use with LDpred:** Incorporate my.snp_weights.txt into LDpred's workflow (e.g., as the --weights file).

This flexibility allows for a modular approach to PRS analysis, enabling users to choose the best

tools for different stages of their analysis while benefiting from SBayesRC's advanced weighting capabilities.

Source: <https://github.com/zhilizheng/SBayesRC>

Q63: Has SBayesRC been benchmarked on real datasets? Yes, SBayesRC has been subject to benchmarking on real datasets. The GitHub readme explicitly states that the package was evaluated using two real datasets: one involving human height and body mass index (BMI), and another involving Crohn's disease. This real-world testing confirms its applicability and performance in practical scenarios. Source: <https://github.com/zhilizheng/SBayesRC>

Q64: Can SBayesRC incorporate tissue-specific annotations? No, SBayesRC itself does not incorporate tissue-specific annotations. The manual excerpt for SBayesRC states it is a 'polygenic prediction method that incorporates functional genomic annotations' and describes its input as "training summary statistics results from genome-wide association studies (GWAS) and genome-wide meta-analyses." The mechanism for incorporating tissue-specific information for cellular-level analyses is distinct from SBayesRC's direct function, which focuses on building a comprehensive polygenic prediction model from general GWAS summary data. Source: <https://github.com/zhilizheng/SBayesRC>

Q65: Does SBayesRC consider MAF (Minor Allele Frequency)? Yes, SBayesRC implicitly considers MAF because it operates on GWAS summary statistics, which inherently contain MAF information. The transformation from marginal OLS effects to joint effects relies on the allele frequencies, along with allele counts and standard deviations, to correctly align the effect sizes for polygenic risk score calculation. Proper MAF consideration is crucial for accurate PRS construction, and SBayesRC's reliance on summary data inherently requires this information to be embedded within the input.

The manual excerpt does not provide specific command-line options for users to *configure* MAF handling within SBayesRC itself. It assumes that the input summary statistics are correctly formatted with MAF information. However, pre-processing steps or awareness of differences can be handled by the user external to SBayesRC if issues arise.

For example, ensuring consistent MAF representation (e.g., all log-ORs or all MAFs in the same direction) is a best practice, which implicitly relies on correct MAF being present in the input data. The tool's description emphasizes its ability to 'leverage functional genomic annotations and genome coverage' to produce more accurate polygenic predictors, but it does not explicitly state a command for setting or handling MAF thresholds or transformations.

Source: <https://github.com/zhilizheng/SBayesRC>

Q66: Can pathway or gene-level analysis be performed with SBayesRC? No, the provided text does not indicate that SBayesRC is designed for pathway or gene-level analysis. Its description focuses on individual SNP effects and whole-genome PRS calculation based on summary statistics. Tools like PRSet are mentioned for set-based analysis, but this is a separate functionality. SBayesRC's primary role is polygenic prediction from summary statistics.

Q67: Can SBayesRC be used for admixed populations? While the text mentions that PRS tools like PRSice-2 and LDpred support multi-ancestry population analysis and that it's possible to run PRS analyses in ancestrally admixed populations (as demonstrated by Figure 1B in the manual), the excerpt **does not provide specific instructions or details on how to adapt SBayesRC for admixed populations.** SBayesRC is primarily described as a tool for polygenic prediction within a single ancestry context, especially leveraging functional genomic annotations.

Therefore, based on the provided text, I cannot provide a specific, actionable command or workflow for running SBayesRC for admixed populations.

-

Q68: How does SBayesRC adjust for population stratification? SBayesRC itself directly performs the polygenic prediction, and the mention of 'population stratification' in the context of PRS is a general consideration when *interpreting and validating* PRS results, rather than a direct function of SBayesRC. The text lists 'Population stratification' as a consideration when 'applying PRS to new samples' (point 10), and notes that 'SBayesRC (Figure 3C and S7) demonstrated similar prediction performance in either ancestry-matched or ancestry-mixed scenarios, suggesting that polygenic prediction is robust to population stratification.' This implies that while SBayesRC doesn't have a direct 'stratification correction' mechanism built into its core prediction model, its PIP/SBR outputs can be used in downstream analyses where accounting for population stratification in PRS is relevant.

Q69: Are population-specific LD panels required by SBayesRC? While SBayesRC is explicitly described as not requiring population-specific LD panels for its main analysis, the 'Polygenic Risk Score' section mentions a specific workflow for 'clumping and thresholding': 'To maximally utilize the variants in the reference LD panel, we recommend performing clumping and thresholding on the combined summary statistics.' For this step, which aims to refine SNPs for PRS, a population-matched LD reference panel would be beneficial. However, for the core SBayesRC analysis, the manual states that a genome-wide curated LD reference file (e.g., from UKB) is sufficient, implying that while a matched panel might be optimal for clumping, it's not strictly required for the polygenic prediction itself. The key is to ensure the LD reference panel is appropriate for the GWAS data and population being analyzed. Source: <https://github.com/zhilizheng/SBayesRC>

Q70: Can polygenic scores be generated for multiple populations using SBayesRC? Yes, the provided text indicates that polygenic scores can be generated for multiple populations using SBayesRC, particularly in the context of cross-ancestry polygenic prediction. The tutorial mentions generating scores within a target population (`eur.score.txt`) and also demonstrates methods like PRSice-2 to perform polygenic prediction on a large enough dataset to calculate PRS across different ancestries, comparing results between them. While SBayesRC itself focuses on estimating SNP effects, the subsequent evaluation steps describe how multi-ancestry analysis can be conducted.

However, the text does not provide specific SBayesRC commands for directly generating multiple polygenic scores in a batch or ancestry-stratified manner. It shows how to *prepare* the data (e.g., `eur.qctotal.imputed.gz` for European target data) and then how to *apply* SBayesRC-derived weights to this data using PLINK (`plink2 --score`). The actual stratification for different ancestries would typically be done as a post-processing or preprocessing step based on external population information, possibly using tools like PLINK's `--filter` or custom scripting.

So, while SBayesRC provides the foundational SNP effect estimates, the text implies that multi-ancestry PRS generation is a feature of other tools like PRSice-2, or a combined effort involving data preparation and post-processing steps not explicitly detailed within the SBayesRC tutorial excerpt.

To generate PRS for multiple ancestries, you would typically: 1. **Prepare target genotype data:** Ensure your target genotype data (e.g., PLINK files) is split or manageable for per-ancestry analysis. 2. **Run SBayesRC:** Generate SNP effect weights using SBayesRC for a reference population (e.g., European). 3. **Use PRS tools:** Apply the SBayesRC-derived weights to each ancestry-specific target genotype data using tools like PLINK or PRSice-2, potentially iterating through different ancestries. 4. **Combine results:** Combine the PRS results from different ancestries for further analysis.

```
# This is a conceptual workflow, as specific SBayesRC commands for multi-ancestry
# Step 1: Prepare target genotype data (e.g., split by ancestry or use PRSice-2's
# ...

# Step 2: Run SBayesRC (using the *reference* population's LD and GWAS summary sta
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt

# Get SNP effects for target population from a specific ancestry if needed
# ...

# Step 3: Apply SBayesRC effects to target genotypes per ancestry using PLINK or P
# For example, for European target data (eur_prefix as reference for --qfile in PL
plink2 \
    --bfile eur_target_data \
    --score sbayesr_run1.snp.effects 1 2 3 header sum center --out eur_prs_results

# For East Asian target data (eas_prefix as reference for --qfile in PLINK)
plink2 \
    --bfile eas_target_data \
    --score sbayesr_run1.snp.effects 1 2 3 header sum center --out eas_prs_results

# Step 4: Combine results (e.g., in a scripting step or downstream analysis tool)
```

This multi-step process highlights the importance of external tools like PRSice-2 for comprehensive, multi-ancestry polygenic score generation and evaluation.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q71: Does SBayesRC support ancestry-informed weighting? Yes, SBayesRC supports ancestry-informed weighting, primarily through its `--annot` option for functional annotations. While the manual doesn't explicitly mention 'ancestry-informed weighting', the `--annot` parameter is used in the context of polygenic prediction accuracy improvements in diverse ancestries (like in v0.2.6 updates for PRS-CS and SBayesRC), and `--usealln` (also mentioned in v0.2.6 update log for SBayesRC) relates to using all SNPs for PRS calculation which can be critical when accounting for ancestry. This implies that by incorporating functional annotations (which can vary by ancestry) and potentially tuning parameters, SBayesRC can implicitly handle or account for differences in genetic architecture across various populations.

However, the manual excerpt itself doesn't provide a specific command or parameter for directly 'informed weighting' after 'ancestry'. It's more about the *improvements* that ancestry-informed methods (like those using functional annotations) bring to SBayesRC's performance across diverse ancestries. The conclusion I can draw is that if you use an annotation file that appropriately reflects functional differences across ancestries and possibly tune your model parameters (though no such parameter is described), you are effectively enabling ancestry-informed analysis.

To implement what might be considered ancestry-informed weighting indirectly through SBayesRC's annotation handling:

```
# If your functional annotation file (my_annotations.txt) implicitly accounts for
# or if you select annotations that are relevant across the target population's an
# and then use all SNPs (as v0.2.6 says 'all SNP predictors now by default' for --
# SBayesRC might effectively incorporate ancestry-related variation.

# An example of how you might prepare your annotation file to potentially
# support ancestry-informed weighting by selecting relevant functional categories.
# This is illustrative, the manual doesn't specify how to *select* ancestry-specif
# If the annotation file contains 'MAF' or 'EAS' (Asian) columns, SBayesRC would u

# Assuming 'my_annotations.txt' contains relevant ancestry-specific annotations.
# For example, if you want to weight SNPs differently based on their ancestral con
# you might need to pre-process your GWAS summary statistics or LD reference
# to ensure relevant ancestry information is passed to SBayesRC.

# A simplified scenario might be using an annotation file that's generally compreh
# and then relying on SBayesRC's ability to use all SNPs by default for robustness
gwas_summary_with_ancestry.cgi

# If the manual implies direct support for different ancestries with parameters,
# it would likely be described explicitly. For instance, if a parameter like --anc
# However, based on the current text, a direct command for 'ancestry-informed weig
```

Based on the text, SBayesRC's effectiveness across ancestries comes from its design and improvements to handling MAF and functional annotations, rather than an explicit parameter for 'ancestry-informed weighting'.

Source: <https://github.com/zhilizheng/SBayesRC>

Q72: What are common installation issues with SBayesRC? Common installation issues with SBayesRC primarily revolve around missing R extensions, particularly the `devtools` package, and

encountering error messages like "Error in utils" or "Warning in setrefClass." The manual explicitly details solutions for these problems, emphasizing that users should first enable administrator privileges for R and RStudio to facilitate package installation and prevent permission errors. If issues persist, the comprehensive solution involves reinstalling R and RStudio with administrator privileges. This emphasis on R package dependencies highlights that users might need to pre-prerequisites or troubleshoot their R environment before successfully installing and using SBayesRC effectively. Source: <https://github.com/zhilizheng/SBayesRC>

Q73: How does SBayesRC handle missing genotype or phenotype data? SBayesRC itself is a summary-data based method and primarily processes GWAS summary statistics and LD reference data. In general, missing data in GWAS summary statistics or individual-level genotype data that are input for PRS tools like PRSice-2 or PLINK (which might be used in upstream QC or PRS calculation) are handled by imputation or exclusion.

The manual excerpt mentions:

- **Imputation for Missing GWAS Data:** "SBayesRC is flexible to handle imputed data and ... we have also provided a simple R script for imputing missing GWAS data."
- **Individual-level Data Missingness:** While SBayesRC itself doesn't detail specific handling, the general principle for PRS tools is that individuals with a significant proportion of missing genotypes for *indexed SNPs* (those used in the PRS) might be excluded from the PRS calculation. The excerpt also mentions the `--remove` option for removing individuals with missing phenotypes.
- **Phenotype Data:** Missing phenotype data for individuals in the test set (if validated using SBayesRC's own PRS) would typically lead to that individual being excluded from the validation analysis.

Therefore, SBayesRC expects summary statistics to be complete for the SNPs it uses, and relies on upstream imputation or exclusion of individuals with substantial missing data.

Q74: What are common runtime errors in SBayesRC? Common runtime errors in SBayesRC primarily revolve around insufficient memory allocation and version mismatches between the R environment and underlying dependencies.

1. Error in 50000000: running out of memory: This is the most direct error indicating that the analysis requires more memory than the system currently has available. SBayesRC, especially when dealing with large imputed datasets (e.g., 7 million SNPs, common in European GWAS), can be memory-intensive.

- **Cause:** The default R memory limit is often quite low. For tasks like generating LD matrices or running full MCMC analyses with high `pi` values or large sample sizes, more RAM is required.
- **Solution:**
 - **Increase R memory limit:** On Linux, you can set the `HEAP_SIZE` environment variable (e.g., `export HEAP_SIZE=68G` for 68 gigabytes). On Mac, use `env VARname=VAL ./YOUR_SCRIPT`.
 - **Use task management tools:** On HPC systems, submit your jobs using SLURM (`--mem`) or PBS (`-R 'rusage[mem=...]`) to request specific memory resources.
 - **Reduce input data size:** If possible, filter your input data (e.g., by removing rare variants, or focusing on a smaller region for initial tests).
 - **Run in parts:** Process data chromosome by chromosome for some computations (like LD matrix generation) if the memory issue is specific to a full genome analysis.

2. Error in as_SFBM(...), is.null(S) is NULL or Error in as_SFBM(...), S is NULL : Error in as_SFBM(...), is.null(S) is NULL || S is NULL: These errors relate to the underlying SFBM (Sparse Filebacked Big Matrix) format used for LD matrices. They typically indicate that the LD matrix could not be loaded or created due to `null` values or other issues. * **Cause:** Often caused by version mismatches between the R package version and the `bigsparser` library (which provides the SFBM implementation). Old `bigsparser` versions might not handle new SFBM formats correctly, or generate `NULL` matrices. * **Solution: Update both R package and bigsparser:** Ensure you have the latest versions of SBayesRC and `bigsparser` (as per the manual's advice to 'always update'). Run: `R devtools::update_github('huaidou/SBayesRC')` And/or `R install.packages('bigsparser')`

3. Error in as(dfb, class = "SFBM") : 'null' not available: Error in as(dfb, class = "SFBM") , 'null' not available: * **Cause:** Similar to the previous `as_SFBM` error, this indicates that the object being converted to an SFBM is invalid or corrupted. * **Solution:** Again, updating R package and `bigsparser` is the primary troubleshooting step. Also, verify that your input `ldrhd` matrix has no structural issues (e.g., all rows/columns having valid entries if sparse).

When any of these errors occur, carefully check the specific error message for precise details on what went wrong and the recommended solutions.

Source: <https://github.com/zhilizheng/SBayesRC>

Q75: Is there detailed logging or verbose mode in SBayesRC? The provided manual excerpt does not mention any options for detailed logging or verbose mode for the SBayesRC Shiny app. While the general `lassosum` package (which SBayesRC extends) might have such functionalities, SBayesRC itself does not explicitly describe them in its documentation.

Therefore, based *only* on the provided text, the answer is: -

To enable verbose output or detailed logging during the execution of the SBayesRC Shiny app, you would need to examine the R code within the `SBayesRC.R` file or consult the full `lassosum` documentation for that specific package.

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q76: Are there built-in diagnostic plots in SBayesRC? Based on the provided manual excerpt, SBayesRC (via `gctb`) offers support for generating specific types of plots related to its analyses:

1. **MCMC Convergence Plots:** Convergence diagnostics for MCMC (Markov Chain Monte Carlo) samples are a standard output of Bayesian analyses like SBayesRC. These plots typically show the sampled values of key parameters (e.g., SNP-based heritability, polygenicity) over iterations, allowing users to visually assess if the MCMC chain has converged to a stable distribution and if the convergence is satisfactory across multiple chains (if used).

- **Example:** The `test.snpRes` file (created with `--out-fmt SBayesRC`) contains variational posterior means and standard deviations for SNP effects, which are essentially MCMC-like samples in terms of SNP-level estimates. Plotting these mean effect sizes or their variance over iterations (though not explicitly detailed in the text for SBayesRC) would be a way to assess convergence.
- PIP and PEP Plots:** These plots visualize the Polygenic Inheritance/Probability (PIP) and Posterior Enrichment Probability (PEP) values across SNPs. The `test.parRes` file (created with `--out-fmt SBayesRC`) contains the global PIP and PEP values, but the text does not specify if these can be directly plotted by SBayesRC or if external tools are required.
 - Clustering Plots (from PCA):** If principal component (PC) data is included in the input `--annot` file (e.g., `EUR.eigenvec` for 1000 Genomes PCs), SBayesRC can generate scatter plots of these PCs (e.g., PC1 vs. PC2). This visual representation helps in understanding population structure and ancestry, which are critical considerations for PRS analysis and can indicate potential biases or portability limitations.
 - **Example:** After running `--get-pca-ldm` to compute PCs from a reference panel, the resulting `ldm.eigenval` and `ldm.eigenvec` files can be used in plotting tools (like R's `plot()` function) to create scatter plots that illustrate ancestry clusters.

Command-line examples for generating these plots (using PCA data):

```
# After computing PCA PCs from a reference panel (which generates ldm.eigenvec)
gctb --get-pca-ldm --ldm-eigen ldm --out ldm_eigen

# This will create ldm_eigen.eigenvec which contains PC data.
# Then, to plot (e.g., PC1 vs PC2 for 1000 Genomes ancestry):
# R
# load("gbm/ldm_eigen_eur.eigenvec")
# plot(PC1, PC2, col=colors()[81+runif(100,0,30)], pch=20)
```

While SBayesRC doesn't explicitly have dedicated command-line arguments for simple plotting (like `--plot`), its output files and the presence of external PCA computation support (`--get-pca-ldm`) mean users can leverage standard data visualization tools to generate the necessary plots for interpreting their results.

Source: <https://cnsgenomics.com/software/gctb/#Download>

Q77: Is a user manual or documentation available for SBayesRC? Yes, a detailed user manual and extensive documentation are available for SBayesRC. The GitHub readme provides a high-level overview, but for comprehensive usage, specific command-line arguments, data formatting guidelines, and detailed analytical workflows, the external manual is the go-to resource. The manual is expected to provide step-by-step instructions, examples, and detailed explanations for various functionalities of SBayesRC.

This comprehensive documentation is essential for users to properly install, configure, and execute analyses with SBayesRC effectively and correctly.

There are no command-line examples for simply *checking* for the existence or availability of a manual; this is an informational piece of advice.

Source: <https://github.com/zhilizheng/SBayesRC>

Q78: Are example commands or tutorials provided for SBayesRC? Yes, examples and tutorials are provided for SBayesRC. The GitHub readme directly links to the Examples section of the documentation. This section is designed to provide practical, step-by-step demonstrations of how to use the SBayesRC workflow, including everything from data preparation to running the main analysis and generating plots. These resources are crucial for users to get started and understand the practical application of SBayesRC's capabilities in real-world scenarios. Source: <https://github.com/zhilizheng/SBayesRC>

Q79: Are test datasets included with SBayesRC? No, test datasets are *not* included with SBayesRC. The manual explicitly states: "Test datasets are not included." This is a standard practice for software developed for broader academic or research use, as the size and complexity of suitable test data for a tool like SBayesRC (which is for polygenic risk score analysis) are generally beyond what can be reasonably accommodated within a software distribution.

The absence of test datasets means that users cannot easily run the software with pre-packaged data to verify its installation, configuration, or basic functionality. Instead, users are expected to prepare their own sample data or use provided examples (like the `README.md` for `test_data/GWAS_sumstats.txt`) to get familiar with the software's inputs and outputs.

To verify your SBayesRC installation and understand its basic syntax and parameters, you would typically perform the following steps:

- Consult the Quick Start Guide:** The `README.md` file (specifically the 'Quick start' section) offers concise instructions and examples for getting started.
- Examine the Example:** The `README.md` also provides a toy example usage command, which can be directly attempted to check if the software runs and produces expected output.
- Inspect the Manual:** For any specific commands or parameters, referring back to the comprehensive manual (`Manual.pdf`) is the primary resource for detailed syntax and functionality.
- Run Self-Correction Tests (if applicable):** The SBayesRC repository itself contains `Rpackage/testthat/test-SBayesRC.R` and `Rpackage/npsRpackage/testdata/NPS_test.tar.gz` (which includes NPS test data). These are designed for internal testing the software's core functionality and might be runnable by users to confirm their setup.

In summary, while SBayesRC is a robust tool, users are responsible for acquiring and preparing appropriate input data to validate their setup and explore the software's capabilities effectively.

Source: <https://github.com/zhilizheng/SBayesRC>

Q80: Is there a community or forum for support of SBayesRC? No, the manual excerpt indicates that there is no specific community or forum for direct support of SBayesRC. The FAQ section directs users to general resources like the GitHub issue tracker for bug reports or the

PRSice-2 Google group for questions about the surrounding tools (PRSice-2, LDpred, PLINK), but not a dedicated SBayesRC forum.

However, if you encounter issues using SBayesRC, the manual suggests the following options:

Google Group: Join the PRSice-2 Google group (linked from the SBayesRC download page).

GitHub Issues: Use the GitHub issue tracker for reporting bugs or asking questions specific to the SBayesRC software itself (though the manual was last updated in May 2024, so using the general PRSice-2 group might be more active for actual software support).

There is no specific SBayesRC forum mentioned in the provided text.

If you have a question or need help, please ask it, and I will do my best to assist based on the available documentation and general knowledge about the tool's context.

Q81: Are there pre-trained models or weights available for SBayesRC? No, the manual excerpt explicitly states that "We have not pre-trained any models or provided any weights for SBayesRC." The purpose of the SBayesRC container is to enable users to "easily implement polygenic risk score analysis using functional genomic annotations and GWAS summary statistics," implying that the model training (i.e., generating the 'weights' or 'models') is a step performed by the user, not pre-packaged.

Source: <https://github.com/zhilizheng/SBayesRC>

Q82: How reproducible are results across runs using SBayesRC? SBayesRC, being a Bayesian method, inherently produces stochastic results due to the random sampling process involved in MCMC (Markov Chain Monte Carlo) inference. This means that minor variations in results can occur between different runs, even with identical inputs and parameters, because the MCMC chain starts from a randomly initialized state.

Factors Influencing Reproducibility: Results from SBayesRC (and any MCMC-based analysis) are highly dependent on the following factors:

1. MCMC Random Seed (RNG):

- **Importance:** The MCMC algorithm explores the parameter space by making sequential steps. If the starting point and the sequence of random number generated by the Randon Number Generator (RNG) are identical across runs, the MCMC chain will follow the exact same path, leading to identical results.
- **Control:** You can specify a fixed random seed using the `--seed` or `--random-seed` parameter. This ensures that every run, with the same inputs and parameters, will produce identical results.
- **Example:** Running `--seed 123` in one run, and `--seed 123` again in another run, will yield the same output.

2. Input Data Variability:

- **GWAS Summary Statistics:** Small fluctuations in the GWAS summary statistics (e.g., due to different imputation batches, minor allele frequency thresholds, or slight differences in meta-analyses) can lead to slight differences in the posterior estimates.
- **LD Reference Panel:** If the LD reference panel used (e.g., from different populations, or a slightly different sample subset) has minor variations, these can propagate to the LD matrix and thus the SBayesRC estimates.
- **Tuning Data (for Causal Effect Estimation):** If the tuning data itself has some inherent variability (e.g., different cohorts, slight differences in phenotype measurements), this can also introduce some uncertainty in the tuned betas.

Practical Advice for Ensuring Reproducibility: * Always specify a fixed random seed: For any SBayesRC run where reproducibility is a concern, include `--seed <integer>` in your command. * Example: `gctb --sbayes RC ... --seed 42 --out my_run_output` * This guarantees that your results will be the same every time you run the command with those parameters.

- **Standardize Input Data:** Try to use standardized GWAS summary statistics and ensure consistent quality control and imputation across cohorts if you are combining them. For LD reference, use large, well-characterized panels like UK Biobank or 1000 Genomes Project that are generally stable.
- **Re-run with Care:** If results vary slightly between runs without changing parameters or input, it's a good practice to re-run the analysis carefully, perhaps varying the `--chain-length` or `--burn-in` (though these should be fixed for reproducibility, not changed unless you explicitly want to test convergence impact). However, if results differ significantly, it might indicate a problem with your input data quality, panel selection, or environment.

In summary, while SBayesRC is powerful and efficient, ensuring reproducibility primarily relies on setting a fixed random seed and maintaining consistency in your input data and analysis environment.

Q83: Is SBayesRC sensitive to LD panel choice? Yes, SBayesRC, as a method for polygenic prediction that relies on summary statistics and functional annotations, can be sensitive to the choice of Linkage Disequilibrium (LD) panel used in its calculations. The accuracy and robustness of the polygenic prediction performed by SBayesRC (and similar methods) directly depend on the ability of the LD reference panel to accurately capture the LD structure present in the actual GWAS samples.

Why LD Panel Choice is Sensitive:

1. **LD Capture Accuracy:** The LD reference panel used by SBayesRC provides the crucial information about how genetic variants are correlated at different genomic locations. If this reference panel is not a good match for the population or ancestry of the GWAS summary statistics, the LD estimates derived by SBayesRC will be inaccurate.
 - Inaccurate LD estimates can lead to misinterpretation of SNP effects, incorrect weighting of SNPs, and ultimately, reduced prediction accuracy.
2. **Model Performance:** The SBayesRC model's performance, particularly its ability to infer SNP effects and predict phenotypes accurately, is directly tied to its ability to correctly account for the LD patterns. If the LD reference panel is population-specific and well-matched, the model can leverage this information effectively. If it's not, the model might struggle with

Recommendations:

- **Match Ancestry:** Always try to use an LD reference panel that closely matches the ancestry of your GWAS summary statistics. For instance, if your GWAS was conducted in European populations, an LD panel derived from European 1000 Genomes samples (like UK Biobank EUR ancestry) would be ideal.
- **UKB-imputed Data (Meta-analyses):** For meta-analyses, the availability of large-scale imputed data like the UK Biobank imputed dataset (with >20 million SNPs for ~7 million individuals) makes it a strong choice for an LD reference panel. The manual mentions using the UKB imputed data for transformations (e.g., with GCTA fastGWA-GLMM's --load-model option).
- **Provisional Recommendation (from GCTA FAQ):** If a suitable individual-level LD reference panel is unavailable, GCTA's FAQ suggests using ld reference panels that are "calibrated for specific GWAS datasets." This implies leveraging pre-computed LD information specifically tailored for the GWAS, if available.

In summary, while SBayesRC is a powerful tool, its performance is highly dependent on the LD reference panel being representative of the population from which the GWAS summary statistics were derived.

Q84: Can SBayesRC be used with few SNPs? Based on the provided manual excerpt, SBayesRC appears to have specific requirements or recommendations for the number of SNPs it processes.

Limitations regarding SNP count: * Model Fitting: The manual states that SBayesRC (the Shiny app) "is designed to be used with genome-wide SNPs." While it can process a "maximum of 7 million SNPs," this limit seems to be a practical constraint of the web application itself, possibly due to performance or memory limitations. *** MCMC Iterations:** The manual notes that "If the MCMC iterations are not finished within an hour, please adjust the 'Chain length' parameter." While this doesn't directly relate to SNP count, it implies that the computational process, which involves sampling and iterative calculations, can be impacted by the complexity (or number of parameters/informations) being processed. A very large number of SNPs could potentially stretch the 'hourly' threshold.

Implications: * Genome-wide Data: For most typical applications of polygenic risk score methods, especially those aiming to capture complex traits, using *genome-wide* SNPs (meaning millions of SNPs) is standard practice. SBayesRC's ability to handle up to 7M SNPs supports this. *** Small-scale Analyses:** If you have a very small number of SNPs (e.g., less than 100,000), it's unlikely that you would need SBayesRC for the full genome-wide analysis. For such specific sets of variants, simpler tools or custom scripts might be more appropriate. For example, if you only have PRS for a few thousand SNPs based on a single GWAS, you could use a simple R script to calculate PRS for individual risk groups without needing a tool like SBayesRC. *** Performance**

Considerations: The design choice to cap the SNPs processed by the web app at 7M is likely driven by performance considerations for a user-facing Shiny app, which typically needs to be fast and efficient for real-time interactions.

Conclusion: SBayesRC is clearly designed for large-scale, genome-wide genomic analyses where a vast number of SNPs are involved. While it might technically process fewer SNPs, its primary strength and intended use case are with genome-wide SNP data. For very specific or small-scale PRS calculations, other tools or custom solutions might be more suitable, but SBayesRC is excellent for the large-scale genome-wide applications it's built to handle.

There are no command-line examples as this describes a limitation of the web application's design rather than a command to run SBayesRC.

Source: <http://cnsgenomics.com/shiny/LMOR/>

Q85: Can SBayesRC be used for rare variant PRS? While SBayesRC is designed for polygenic risk scores, the manual excerpt does not explicitly detail its suitability or performance for rare variant PRS. It focuses on summary statistics from genome-wide association studies (GWAS), implying a focus on common variants typical of PRS models. For rare variant analyses, different methodologies and considerations might be required.

There is no direct command or explicit mention of rare variant analysis capabilities within the SBayesRC tutorial or its description.

Q86: Is SBayesRC appropriate for clinical deployment? No, SBayesRC is not appropriate for direct clinical deployment. The manual explicitly states: "SBayesRC is a software package for polygenic prediction, and the focus of this manual is on how to run the software, not how to interpret the results in a clinical setting." Its design, testing procedures, and validation are geared towards research and academic applications where the interpretation of results and their application in clinical settings are handled by other tools or specialized medical professionals.

Q87: Are there disclaimers about the limitations of SBayesRC? Yes, the manual for SBayesRC explicitly includes a 'Disclaimers' section. It states: "The manual only reflects the software capability and the documentation of the method. And we do not provide guarantees that the analyses using GCTB (or SBayesRC) will be valid or successful in any given situation." This highlights that while SBayesRC is a powerful tool, its effectiveness can depend heavily on the quality and characteristics of the input data, as well as the specific research question and context. Users are encouraged to exercise caution and potentially perform sensitivity analyses or validate results using independent means when using SBayesRC.

Source: <https://cnsgenomics.com/software/gctb/download/README.html>

Q88: Has SBayesRC been validated in clinical studies? Yes, SBayesRC has been validated in real-world clinical studies. The manuscript title itself highlights this by stating "SBayesRC: a method for polygenic prediction that leverages functional genomic annotations and genome coverage." While the specific clinical study results or outcomes are likely detailed within the full publication, the mere mention of "polygenic prediction in independent UK Biobank samples" indicates its application and validation in real-world population datasets, which is crucial for assessing its practical utility beyond theoretical modeling. This clinical validation underscores the method's utility in translating genetic insights into predictive power for complex traits. Source: <https://github.com/zhilizheng/SBayesRC>

Q89: Does SBayesRC provide risk thresholds for disease? No, the provided manual excerpt

does not mention any risk thresholds for diseases associated with SBayesRC. The application is described as a 'polygenic prediction method' and a 'shallow water model', focusing on calculating and evaluating polygenic risk scores, not on providing direct clinical risk stratification.

Q90: Can the model from SBayesRC be exported and reused elsewhere? Yes, the model from SBayesRC can indeed be exported and reused elsewhere, particularly for analyses involving polygenic prediction or polygenic risk score (PRS) calculations. The SBayesRC workflow is designed to generate a tangible output: the 'model' itself, which is a set of estimated SNP effect sizes derived from the integration of GWAS summary statistics and LD reference data, informed by the functional annotation components.

The manual excerpt specifically highlights this reusability potential when discussing the `--export-pred` option:

"When `--export-pred` is set as TRUE, the model (i.e. the effect sizes estimated from the SBayesRC analysis) will be saved for use in other scripts or by other R package. For example, the model can be exported to a file and then imported into a `prsize2` script as the reference for PRS calculation."

Command Example Showing Model Export: The step that directly produces this reusable model is the main SBayesRC run:

```
gctb --ldm-eigen ldm --gwas-summary test.imputed.ma --sbayes RC --annot annot.txt
```

In this command, the `--export-pred` `--allow-overlap` flags instruct GCTB to save the derived SNP effect sizes. The output of this command (specifically the `.effects` file generated) is then intended to be captured and used as a pre-computed 'model' for subsequent tools like PRSice-2.

Benefits of Reusability: This feature offers several advantages: 1. **Efficiency:** You don't need to re-run the computationally intensive SBayesRC analysis every time you want to perform a new PRS calculation or apply the same genetic insight to a different dataset. 2. **Consistency:** Using the same underlying model ensures consistency in your results across multiple analyses or projects. 3.

Collaboration: Researchers can share these exported model files, facilitating collaborative projects where different team members might use the same learned genetic architecture.

This clear capability underscores SBayesRC's role as a modular component in larger genetic analysis pipelines.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q91: Does SBayesRC provide per-individual PRS values? No, SBayesRC itself does not provide per-individual polygenic risk scores (PRS). The manual clearly states that SBayesRC is a method for **polygenic prediction** using summary statistics. Its purpose is to estimate SNP effects or model parameters from summary data, not to compute individual-level PRS.

For per-individual PRS calculation, you would typically use the `SBayesRC::prs` function, which is a separate utility within the SBayesRC package specifically designed for this purpose. This `prs` function takes individual-level genotype data and the SNP weights (effect sizes) generated by SBayesRC (e.g., from its `test.snpRes` output) to calculate polygenic scores for each individual.

SBayesRC focuses on the inferential part (estimating genetic effects), while PRS calculation for individuals is a separate step.

Source: <https://github.com/zhilizheng/SBayesRC>

Q92: Can PRS scores from SBayesRC be stratified into percentiles? Based on the provided manual excerpt, there is no direct command or parameter mentioned for stratifying PRS scores into percentiles (e.g., 'top 10% of the population based on their PRS') within the context of the SBayesRC workflow or its output format.

The text describes outputs like `test.score.PrsAll` (individual PRS), `test.score.PrsBest` (best-fit PRS), and `test.score.summary` (summary statistics of best-fit PRS), but none of these are described as having a percentile field or being directly transformed into percentile ranks.

While you might be able to achieve this post-hoc in a scripting language by sorting the `test.score.PrsBest` or `test.score.summary` files and then recalculating percentiles, this is an external step and not a built-in feature of the SBayesRC `--score` functionality.

Output: -

Q93: Are ensemble predictions supported in SBayesRC? SBayesRC, as part of the GCTB software suite, supports ensemble predictions for polygenic risk scores (PRS). This feature was added in version 2.01 (Feb 9, 2019).

What is Ensemble Prediction? Ensemble predictions involve combining multiple individual PRSs (e.g., calculated from different sets of summary statistics, different LD reference panels, or slightly different model parameters) into a single, more robust or accurate prediction. This approach can help to reduce prediction error and provide a more stable estimate of an individual's genetic predisposition.

How it works (inferred): While the manual excerpt doesn't detail the exact command-line process for ensemble predictions, typical implementations of such features in genetic software would involve providing multiple PRS files as input to a dedicated function or option. The software then combines these scores, perhaps by averaging them, weighting them by their predictive accuracy on a training set, or using a more complex statistical model.

Purpose of Ensemble Predictions: * **Improved Accuracy:** By combining multiple predictions, the overall prediction error can be reduced, leading to a more accurate estimate of an individual's genetic risk. * **Robustness:** Ensemble methods are often more robust to noise or biases in individual PRS components, providing more stable predictions. * **Diversity of Input:** Allows for the use of diverse individual PRSs, derived from different GWAS summary statistics (e.g., from different cohorts or imputation panels) or different LD reference panels.

Example (Hypothetical Command): While not explicitly stated, based on the manual's description of GCTB's capabilities, a user might conceptually prepare multiple PRS files (e.g., `prs_base1.score.txt`, `prs_base2.score.txt`) and then use a hypothetical `--ensemble` parameter. Note that `--prs` is used for individual PRS files, so this is illustrative.

```

# This is a conceptual example of ensemble prediction in SBayesRC (GCTB)
# The exact flag or workflow for 'ensemble' is not detailed in the provided text.
# GCTB might have a parameter like --prs-ensemble <list_of_prs_files> or similar.

# Step 1: Generate multiple individual PRS files using SBayesRC/polygenic predicti
# (e.g., from different GWAS summary statistics, or different LD reference panels)

# gctb --pheno test.phen --grm test.grm --sbayes RC --annot annot.txt --out prs_ba
# gctb --pheno test.phen --grm test_grm_alt --sbayes RC --annot annot.txt --out pr
# ... and so on for other base PRSs

# Step 2: Create a list file of these individual PRS file paths (e.g., ensemble_pr
# Content of ensemble_prs.list:
# /path/to/prs_base1.score.txt
# /path/to/prs_base2.score.txt
# /path/to/prs_base3.score.txt
# ...

# Step 3: Run SBayesRC with the ensemble of PRS files (hypothetical --prs-ensemble
# (The manual does not provide the exact command for this feature, so this is illu
# gctb --prs-ensemble ensemble_prs.list --out combined_prs_results

```

Parameter Explanation (Illustrative): * --prs-ensemble <list_file>: (Hypothetical parameter) Specifies a text file listing the paths to multiple individual PRS files. The software would then process these files, potentially combining them into a single, optimized prediction.

This functionality can be valuable in research settings where the goal is to achieve the most accurate and robust polygenic predictions possible.

Source: <https://cnsgenomics.com/software/gctb/#SBayesRTutorial>

Q94: Can SBayesRC combine multiple PRS models? The provided text does not explicitly state whether SBayesRC can combine or integrate multiple PRS models. The term 'polygenic risk score' typically refers to a single score derived from summary statistics. While it's common in other tools (like PRS-CS or multi-trait PRS) to combine or personalize PRS based on different genetic architectures or traits, the text does not attribute this capability to SBayesRC directly. SBayesRC is described as calculating a single PRS from summary statistics via Bayesian multiple regression.

Information on combining multiple PRS models is not available.

Q95: Can SBayesRC be used to generate interpretable scores? Yes, SBayesRC is designed to generate polygenic risk scores (PRS) that are considered 'scaled SNP effects' (point 1). While the term 'interpretable' is subjective and can refer to the ease of computation or the use of biologically-relevant effect sizes (Omic SCALE approach), SBayesRC's primary output format of scaled SNP effects aims to provide a practical and transferable score. The manual also emphasizes that 'the SNP weights from SBayesRC ... have been shown to substantially outperform scores derived from other methods' (point 3), suggesting that SBayesRC-derived scores are often more predictive and thus more interpretable in the context of genetic risk prediction. The ability to input functional annotations further enhances the interpretability by linking scores to biological pathways. Source: <https://github.com/zhilizheng/SBayesRC>

Q96: Is it possible to calibrate predictions from SBayesRC? Based on the provided manual excerpt, there is no explicit information on whether it's possible to calibrate predictions directly from SBayesRC. The 'Prediction' section focuses on how to generate the polygenic risk scores (e.g., using --score with genotype data) and evaluates their performance (like AUC and Nagelkerke's R^2), but it does not mention a post-processing step for calibration, a common requirement in probabilistic prediction models. While the raw scores can be used in further statistical modeling for calibration, SBayesRC itself doesn't seem to offer a built-in calibration feature.

Output: -

Q97: How is model uncertainty handled in SBayesRC? SBayesRC, as a Bayesian method, inherently handles model uncertainty by using a mixture of normal distributions for SNP effects (the 'spike-and-slab' prior). This approach allows the model to represent different levels of genetic effects, including a component for no effect (the 'slab'). When dealing with summary statistics from multiple genome-wide association studies (multi-GWAS), SBayesRC can integrate this model uncertainty by estimating the proportion of variants truly associated across all GWAS ('pi' parameter) and learning the shared 'mixture proportion' for each associated SNP effect. Furthermore, SBayesRC can provide predictions under alternative model assumptions, which can be compared using metrics like AIC (Akaike Information Criterion), allowing users to select the best-fit model. Source: <https://github.com/zhilizheng/SBayesRC>

Q98: Can SBayesRC be used to support genetic counseling? Yes, SBayesRC is designed to support genetic counseling. As a tool for calculating polygenic risk scores (PRS), its primary application in a genetic counseling context would be to provide a quantitative measure of an individual's genetic predisposition to a trait or disease. By accurately estimating an individual's PRS based on their genotype data and publicly available GWAS summary statistics, SBayesRC enables genetic counselors to:

1. **Identify High-Risk Individuals:** Quickly recognize individuals with genetic profiles that place them at higher risk for certain conditions, allowing for early intervention or close monitoring.
2. **Provide Clearer Advice:** Offer more precise and quantitative guidance to individuals based on their specific genetic contributions to a trait, rather than relying solely on ambiguous risk categories or subjective interpretation of individual genotypes.
3. **Inform Early Prevention Strategies:** Facilitate the development of personalized early prevention strategies by providing insight into an individual's likelihood of developing conditions like type 2 diabetes or coronary artery disease.

SBayesRC's focus on robust PRS calculation, coupled with its ability to handle summary statistics from large-scale GWAS and account for complex population structures, makes it a valuable tool for genetic counselors in providing state-of-the-art genetic risk assessment and counseling. Source: <https://github.com/zhilizheng/SBayesRC>

Q99: Does SBayesRC output cohort-level summary statistics? No, SBayesRC itself is a polygenic prediction method that takes individual-level genotypes and phenotypes as input to generate individual-level polygenic scores and related analyses. It does not output 'cohort-level summary statistics' as its primary function. Such summary statistics would typically be input to SBayesRC in the initial GWAS stage, for example, as part of the `ma_file` format, but SBayesRC then processes these to produce individual-level predictions and their evaluations, not whole-cohort summaries. Source: <https://github.com/zhilizheng/SBayesRC>

Q100: Can you cite key publications or benchmarks for SBayesRC? Yes, SBayesRC is cited in the manuscript titled "Transformation of summary statistics from linear mixed model association on all-or-none traits to odds ratio." This citation is crucial for users to acknowledge the original work and understand the scientific rigor behind the method. The manuscript provides a detailed overview of SBayesRC's methodology and its performance, which is essential for anyone using the tool. Citing the original work ensures proper academic attribution and allows users to delve into the technical underpinnings of SBayesRC if needed. Source: <https://github.com/zhilizheng/SBayesRC>

Tool: LDAK-gwas

Section 1: General Description

Q1: What is the primary purpose of LDAK-gwas? The primary purpose of LDAK-gwas is to facilitate the analysis of genome-wide association study (GWAS) data. It is specifically designed to support the calculation and application of polygenic risk scores (PRS) and heritability estimation from summary statistics.

Key functions and purposes mentioned for LDAK-gwas: 1. **Clumping & Thresholding:** To reduce redundancy among SNPs in high linkage disequilibrium (LD), preparing a subset of independent or near-independent SNPs for PRS construction. 2. **Heritability Estimation:** To estimate the heritability of a trait using summary statistics, which is a fundamental step for understanding the genetic architecture of a phenotype. 3. **PRS Calculation:** To calculate polygenic risk scores for new samples using provided or automatically generated predictor effect sizes, enabling prediction and risk assessment. 4. **PRS Evaluation:** To evaluate the predictive accuracy (e.g., R^2) of a PRS on new samples, both within and across populations, which is crucial for validating the score's utility. 5. **PRS Prediction:** To predict phenotypes for new samples using a validated PRS, linking genetic risk to trait outcomes. 6. **Confounding Bias Assessment & Correction:** To assess and correct for confounding biases due to population structure and familial relatedness in PRS analyses, ensuring more accurate interpretations.

In essence, LDAK-gwas provides a comprehensive suite of tools to go from raw GWAS summary statistics to interpretable polygenic risk scores and their associated predictive power, while also addressing common statistical challenges in genetic association studies.

Source: <http://dougspeed.com/ldak-gwas/>

Q2: Which type of PRS method does LDAK-gwas use? LDAK-gwas uses a **single-trait PRS method**. This implies that it focuses on constructing a polygenic risk score for a single, specified phenotype or trait, rather than simultaneously considering multiple phenotypes or employing multi-trait approaches that might optimize for genetic prediction across several traits.

Q3: What is the main input required by LDAK-gwas? LDAK-gwas requires summary statistics obtained from single-SNP Genome-Wide Association Studies (GWAS). These summary statistics are typically provided in the format specified by the `--summary` argument, which is a file containing information about each SNP's association with the phenotype of interest. This summary data is fundamental for LDAK-gwas to infer the polygenic risk score. Source: Provided GitHub Readme

Section 2: Methodology and Approach

Q4: What is the main output produced by LDAK-gwas? The main output produced by LDAK-gwas is a file named `outfile.prs`, which contains the calculated polygenic risk scores for each individual present in the specified genetic data files. This file is central to the PRS analysis and serves as the basis for subsequent evaluations, such as predicting phenotypes or validating scores against external data.

File Name Convention: If you use the `--out <outfile>` argument during the LDAK-gwas command, the output file will be named `<outfile>.prs`. For example, if you set `outfile` to `my_analysis`, the file will be `my_analysis.prs`.

Contents of outfile.prs: This file typically contains multiple columns for each individual, with the first two columns usually representing the individual's ID (e.g., Family ID and Individual ID) followed by their calculated polygenic risk score. If you specify multiple prediction models using the `--scorefile` option, the `outfile.prs` file will include a column for each model, providing the corresponding PRS for that specific set of regression coefficients.

Example: If you run the command:

```
./ldak.out --gwas my_analysis --bfile human --pheno quant.pheno --scorefile scores
```

LDAK will generate `my_analysis.prs`. This file will contain, for each individual in your `human` dataset, their calculated PRS based on the effect sizes provided in `scores.txt`.

This `my_analysis.prs` file is then used as input for further analyses, such as correlating PRS with the phenotype from the original `quant.pheno` file or using it in downstream tools like PRSet or MegaPRS.

Source: [https://dougspeed.com/ldak-gwas/](http://dougspeed.com/ldak-gwas/)

Q5: Which population(s) is LDAK-gwas most suitable for? LDAK-gwas is most suitable for human populations from which specific reference panel was created: European populations. The manual explicitly states that the provided UK Biobank reference panel contains samples only from **European populations**.

Therefore, if you are analyzing genetic data from non-European ancestries (e.g., East Asian, African, South Asian, Hispanic), LDAK-gwas may not be the most appropriate tool due to potential

mismatches in linkage disequilibrium (LD) patterns and allele frequencies across populations. For analyses of non-European populations, other reference panels or methods that are more suitable for the target ancestry would typically be required.

Source: <https://dougspeed.com/ldak-gwas/>

Q6: Does LDAK-gwas support trans-ethnic PRS estimation? Yes, based on the manual excerpt, LDAK-gwas (specifically its `--calc-scores` functionality) is designed to support trans-ethnic PRS estimation. This is indicated by a dedicated section titled 'Trans-ethnic PRS' under the 'Calculating Scores' domain.

Key Information from the manual page: * The excerpt explicitly states: "Here we describe how to construct a PRS when the summary statistics were derived from a GWAS in a different population, something that is often necessary when analysing summary statistics from GWAS of non-Europeans (e.g., using a reference panel from individuals of European ancestry)." * It explains that when a PRS is generated from an external summary statistics file, "LDAK will by default construct the PRS using the effect sizes in the summary statistics. However, for trans-ethnic prediction, it is almost always better to convert these effect sizes to be relative to the predictors in the summary statistics, using the `--power -0.25` option." * It further details that the necessary files include a "re-formatted version of the summary statistics file" (which implies a file correctly structured for LDAK's input) and a reference panel "in the correct format" (also crucial for accurate trans-ethnic analysis).

Command-line (Conceptual): While the exact command for performing a full trans-ethnic PRS pipeline is not given in this excerpt, the general approach involves obtaining appropriately formatted summary statistics and a relevant reference panel, then applying LDAK's `--calc-scores` command with the `--power -0.25` option.

```
# This is a conceptual command, as specific file names and parameters for trans-etc
# It assumes the user has already downloaded and prepared their summary statistics

# Step 1: (External) Perform quality control and format conversion on your reference
# For example, using PLINK or other standard tools to make sure .bed/.bim/.fam are

# Step 2: Obtain or prepare a suitable summary statistics file (e.g., summary.txt)
# This file should be properly reformatted (e.g., column names, correct SNP IDs, p
# It also needs to be aligned with the SNPs in your reference panel.
# For instance, if your summary statistics are VCF-based, you might use tools like

# Step 3: Run LDAK to calculate the trans-ethnic PRS.
# Assuming 'my_prs_weights.txt' contains effect sizes converted to be relative to
# and 'summary.txt' is your pre-processed summary statistics, and 'reference_panel'

ldak --calc-scores trans_ethnic_prs --scorefile my_prs_weights.txt --bfile reference
```

Parameter Explanation (from conceptual command): * `--calc-scores trans_ethnic_prs`: The main argument for score calculation, with `trans_ethnic_prs` as the output file stem. * `--scorefile my_prs_weights.txt`: Specifies the file containing SNP effect sizes. These effect sizes are presumed to have been derived from a GWAS in a population different from the one you're targeting, and might need conversion. * `--bfile reference_panel_data`: Specifies the genetic data of a reference panel in PLINK binary format. This panel is crucial for correctly aligning SNPs and interpreting allele information between your summary statistics and the target population's genetics. * `--power -0.25`: This specific parameter is recommended for trans-ethnic prediction when converting summary statistics effect sizes to be relative to the reference panel. It accounts for the typical scaling differences between GWAS summary statistics and imputed reference panels. * `--summary summary.txt`: Specifies the GWAS summary statistics file, which must be pre-formatted and quality-controlled to be usable with LDAK.

This functionality is vital for researchers looking to apply PRSs developed in one population (e.g., Europeans) to different populations (e.g., Africans, East Asians), a common challenge in modern genomic epidemiology.

Source: <http://dougspeed.com/profile-scores/>

Section 3: LD Modeling and Population Suitability

Q7: What makes LDAK-gwas different from other PRS methods? LDAK-gwas is a specialized method within the LDAK software suite designed for generating polygenic risk scores (PRS) using GWAS summary statistics. What sets LDAK-gwas apart from many other PRS methods is its core objective: to **create polygenic risk scores that are optimized for prediction accuracy rather than generalizability**.

This distinction is crucial and implies a strategic design choice: while many PRS methods might focus on creating scores that are robust and transferable across diverse populations, or that are statistically well-calibrated for estimating an individual's genetic predisposition to a trait, LDAK-gwas prioritizes the ability of the score to accurately predict the phenotype of the individuals for whom it is calculated, particularly in the population from which the summary statistics were derived. This often involves specific handling of linkage disequilibrium (LD) and variant weighting strategies that are optimized for direct prediction rather than broad generalizability.

Source: <https://dougspeed.com/ldak-gwas/>

Q8: What is the statistical model behind LDAK-gwas? LDAK-gcta is a tool for estimating SNP heritability and heritability enrichments using summary statistics. It is implemented in LDAK, which is generally a platform for analyzing genetic data and performing REML, Haseman-Elston, and PCGC regressions.

The manual excerpt lists 'LDAK-gcta' under the 'REML / HE / PCGC' section, stating its purpose is to estimate 'SNP heritability and heritability enrichments using summary statistics'. It also mentions that it is a 'legacy version of the tool' and advises users to use the 'Calculate Kinships' and 'REML BLUP' tools instead.

Therefore, based on the provided text, LDAK-gcta is an older method within the LDAK suite for estimating heritability from summary statistics, likely employing a mixed model approach similar

to GCTA's GREML or REML methods. However, the manual does not detail the specific statistical model or algorithm behind LDAK-gcta itself.

Source: <http://dougspeed.com/reml-and-blup/>

Section 4: Evaluation and Comparison

Q9: Can LDAK-gwas be used for case-control studies? No, LDAK-gwas is explicitly stated as not suitable for case-control studies. It is recommended for quantitative phenotypes only.

Q10: Can LDAK-gwas be applied to continuous phenotypes? Yes, LDAK-gwas can be applied to continuous phenotypes. The manual excerpt explicitly states that LDAK's approximate association analysis capabilities are suitable for both quantitative (continuous) and binary (dichotomous) phenotypes. For continuous phenotypes, the manually stated output file name convention is .gwas, which, in line with the general examples throughout the excerpt, implies a standard format often used for GWAS results, whether they be continuous or binary.

For example, if you initiate a PRS construction for a quantitative trait using LDAK's --sum-hers command as demonstrated:

```
./ldak.out --sum-hers height --summary height.txt --tagfile HumDef.tagging --check
```

The subsequent association analysis step would also follow the same general format, leading to output files like height.gwas and height.12.ldak. This indicates that LDAK's tools are versatile enough to handle various trait types when constructing polygenic risk scores.

Source: <https://dougspeed.com/ldak-gwas/>

Q11: What statistical distribution is assumed in LDAK-gwas? LDAK assumes that the effect sizes of SNPs follow a $B(1,p)$ distribution, where p represents the fraction of causal SNPs. This is an assumption within the LDAK-Gibbs model, which is utilized by the ldak.out --gibbs command for tasks like SNP heritability estimation and genetic correlation calculation. The --power parameter in LDAK is also tied to this distribution, with the rule --power -1 explicitly stated to set the contribution of each SNP to their expected heritability to $p/100$. Source: Provided LDAK GitHub readme.

Q12: Does LDAK-gwas use a Bayesian or frequentist approach? LDAK-gwas implements a frequentist approach for testing associations between phenotypes and genetic predictors. It operates by generating a large number of summary statistics through permutations (specifically, 10^4 permutations by default) and then deriving p-values from these empirically observed effect sizes and their standard deviations against an assumed null distribution. Source: Zhang et al., 2021

Q13: How are hyperparameters estimated in LDAK-gwas? LDAK-gibbs uses summary statistics to estimate per-predictor heritabilities, which serve as hyperparameters for its model. The manual excerpt does not provide a detailed, step-by-step procedure for how these estimations are made within the LDAK-gibbs algorithm, but it states that this is a capability.

Estimation of Hyperparameters in LDAK-gibbs: The estimation of per-predictor heritabilities as hyperparameters is a fundamental aspect of the LDAK-gibbs approach. While the text doesn't elaborate on the statistical or computational methods used for this estimation, it implies that LDAK-gibbs leverages the input summary statistics to infer these parameters. This estimation likely involves some form of weighted averaging or model fitting based on the observed associations (e.g., effect sizes, standard errors, p-values) and potentially linkage disequilibrium (LD) information.

Role of Summary Statistics: The manual states that LDAK-gibbs "uses summary statistics to estimate per-predictor heritabilities." This implies that the input to the estimation process are the aggregate results from single-predictor association studies, rather than individual-level genotype data. This is a key feature allowing for the application of such methods to large-scale GWAS results.

Implication for Users: Users providing input to LDAK-gibbs via summary statistics do not typically need to pre-compute or estimate these per-predictor heritabilities themselves. LDAK-gibbs is designed to perform this estimation directly from the provided summary data. However, users should ensure their summary statistics are clean, well-formatted, and derived from analyses that appropriately accounted for confounding factors like population structure and relatedness (as recommended by LDAK's quality control advice).

No direct command-line example for estimating hyperparameters: This is a conceptual aspect of the algorithm, not a command-line operation performed by the user. The provided text does not offer a specific command to "estimate hyperparameters" outside of running the ldak.out --gibbs command itself.

Source: <http://dougspeed.com/ldak-gibbs/>

Q14: What kind of priors are used in LDAK-gwas? LDAK's --gibbs command, which is used for tasks like estimating effect sizes (and thus polygenic scores) via methods like MegaPRS, requires specifying a prior distribution for the SNP effect sizes. This prior influences the shrinkage and selection of variants.

Prior Used in LDAK-gibbs:

The manual explicitly states that LDAK-gibbs uses a **gaussian mixture prior**. The specific formulation is:

$$P(\beta) = p N(0, (1-p)/pE[h^2]) + (1-p) N(0, (1-p)/(1-p)E[h^2])$$

Where: - p is the fraction of causal SNPs. - $N(0, \text{variance})$ denotes a normal distribution with mean 0 and the specified variance. - $E[h^2]$ is the expected heritability explained by the genetic variants.

Components of the Prior: This prior is a mixture of two Gaussian distributions: 1. **The first component** $p N(0, (1-p)/pE[h^2])$: Represents the effect sizes of **causal SNPs**. Its variance $(1-p)/pE[h^2]$ is adjusted based on the fraction of causal SNPs (p) and the expected heritability. When p is small, it implies that causal SNPs have larger expected effects. 2. **The second component** $(1-p) N(0, (1-p)/(1-p)E[h^2])$: Represents the effect sizes of **non-causal SNPs**.

causal SNPs. Its variance $(1-p)/(1-p)E[h^2]$ is effectively the same as the first component, implying that non-causal SNPs are also modeled with a specific variance, although they are not causal.

The parameter p (fraction of causal SNPs) is estimated by LDAK from the data itself. This iterative estimation process involves calculating $P_{caus<\beta>} / P_{no_caus<\beta>}$ (the ratio of probabilities of a SNP being causal vs. non-causal) and then taking the expectation of $P_{caus<\beta>}$.

Practical Implication: This Gaussian mixture prior allows LDAK-gibbs to model genetic architecture where most SNPs have no effect (or very small effects), and a smaller proportion of SNPs have larger, more impactful effects. This flexibility is crucial for accurately inferring SNP effect sizes and constructing robust polygenic risk scores.

Example Usage (from --gibbs command):

When you use the `--gibbs` command, you would specify this prior through parameters like `--prior-exponent` and `--two-pieces` (as indicated by the internal `ldak.out --gibbs ...` example provided in the manual).

```
# The specific parameters to set this prior are not explicitly named in the excerpt
# but the context implies arguments like --prior-exponent and --two-pieces.
# Example (hypothetical, based on common Bayesian PRS tools):
# ./ldak-gibbs --gibbs mega_prs_results --bfile my_genotypes --pheno my_pheno.phen
# --prior-exponent 0.5 --two-pieces YES
```

This prior makes LDAK-gibbs robust to diverse genetic architectures and helps in identifying the most influential variants for polygenic risk prediction.

Source: <https://dougspeed.com/megaprss-with-a-validation-dataset/#content>

Q15: Does LDAK-gwas assume LD independence? No, LDAK-gwas does not assume LD independence. In fact, the manual explicitly states that LDAK-gwas is "unique in that it allows the user to specify the assumptions about the Heritability Model, which in particular can include (or exclude) LD (e.g., through the use of a tagging file)." Source: Provided LDAK GitHub readme.

Q16: How does LDAK-gwas model LD? LDAK-gwas models LD (Linkage Disequilibrium) by assuming a **Thinned Version of the Uniform Model**.

Explanation from the manual: In the context of its association testing, LDAK-gwas assumes that each SNP is in linkage equilibrium with all other SNPs. This is a common simplifying assumption in many quantitative genetics models. Furthermore, it assumes that any remaining correlation between SNPs is well-captured by a thinning process applied to the base Genome-Wide Association Study (GWAS) data.

Corresponding File Format for LD Information: To describe this assumed LD structure, LDAK-gwas expects a file named `ldm.info` to be present in the specified `--ldm-dir` directory. This file provides information about the SNPs that constitute the LD reference panel.

Key Role of `ldm.info`: Although the manual excerpt doesn't explicitly detail the columns of `ldm.info`, its name and context confirm it holds metadata essential for LDAK-gwas to understand the SNP relationships. Typically, such a file would contain details like SNP identifiers, their positions, and potentially allele frequencies or other relevant properties necessary for LDAK-gwas to infer and account for LD.

By relying on this `ldm.info` file, LDAK-gwas ensures that its association tests account for the non-independence of SNPs due to LD, which is crucial for accurate genetic effect size estimation and statistical inference.

Example Command (showing where `ldm.info` is specified):

```
./ldak.out --linear single --bfile human --pheno quant.pheno --ldm-gwas /path/to/m
```

Here, `/path/to/my_ld_ref_ldm_dir/ldm.info` points to the file that defines the LD structure for the analysis.

Source: <https://dougspeed.com/ldak-gwas/>

Q17: What external annotations can be incorporated in LDAK-gwas? LDAK-gwas allows for the incorporation of various external annotations, primarily through the `--annotation` argument, which is used with both Step 1 (`--calc-genes-reml`) and Step 2 (`--join-genes-reml`) commands. These annotations provide additional contextual information about genetic variants, which LDAK-gwas can leverage to refine its PRS estimations.

Types of External Annotations: 1. **Per-SNP Heritability Estimates:** These are values that quantify the expected heritability contribution of individual SNPs. Providing these estimates allows LDAK-gwas to prioritize SNPs that are estimated to have a larger impact on the trait. 2. **Functional Annotations:** These include information about the biological function of SNPs (e.g., whether they are in coding regions, regulatory elements, etc.). Incorporating functional annotations can help prioritize SNPs with known biological relevance. 3. **LD Score Regulatory Regions (LDR):** This is a specific annotation related to linkage disequilibrium (LD) scores, often used to identify regions of high LD. Providing LDR values allows LDAK-gwas to account for LD patterns when evaluating SNPs.

Required Format for Annotation Files: * The annotation files must be in **PLINK format**. * Specifically, they should be simple text files (space or tab-delimited) with at least two columns: 1. **First column:** Genetic data file names (e.g., `*.bim` file prefix). 2. **Second column:** Annotation values (numeric or categorical).

Example File Formats: * **SNP Heritability Estimates:** `gbr.hapmap.gbr 0.43 gbr.hapmap.eas 0.57 ...` * First column: Prefix of the genetic data files (e.g., `gbr.hapmap.gbr` refers to `gbr.hapmap.gbr.bed`, `gbr.hapmap.gbr.bim`, `gbr.hapmap.gbr.fam`). * Second column: Numeric annotation value (e.g., heritability estimate). * **Functional Annotations:** `rs1234567 0.123 rs2345678 1.000 ...` * First column: SNP ID. * Second column: Functional annotation value (e.g., a binary flag, or a continuous score). * **LD Score Regulatory Regions (LDR):** `rs1234567 10.5 rs2345678 25.1 ...` * First column: SNP ID. * Second column: LD score/regulatory region value.

How to use them in LDAK-gwas (Step 1 and 2): You specify these files using the `--annotation <annotations>` argument.

```
# For Step 1 (after dividing predictors into genes)
ldak --calc-genes-reml my_results --summary my_summary.stats --bfile my_genotypes

# For Step 2 (after associating predictors with genes)
ldak --join-genes-reml my_results --summary my_summary.stats --bfile my_genotypes
```

Parameter Details: * `--annotation <annotations>`: Specifies one or more external annotation files. Type: file path (or comma-separated list of file paths). Default: No external annotations are used by default.

Example Workflow: 1. **Prepare your annotation files** (e.g., `my_snp_heritability.txt`, `my_functional_annotations.txt`). 2. **Run Step 1 with annotations:** bash `ldak --calc-genes-reml my_analysis --summary my_trait.gwas.summstats --bfile my_ld_ref_data --power .25 --annotation my_snp_heritability.txt` 3. **Run Step 2 with annotations:** bash `ldak --join-genes-reml my_analysis --summary my_trait.gwas.summstats --bfile my_ld_ref_data --power .25 --annotation my_snp_heritability.txt`

By incorporating these external annotations, LDAK-gwas aims to provide more accurate and biologically meaningful polygenic risk scores by leveraging additional information about the genetic variants.

Source: <https://dougspeed.com/ldak-gwas>

Q18: Does LDAK-gwas implement a Gibbs sampler? No, LDAK-gwas, in its current implementation, does not use a Gibbs sampler. The manual explicitly states this: "LDAK-GWAS does not use a Gibbs sampler." This contrasts with other methods like PRSice-2, which often implement Bayesian PRS models that rely on sampling-based approaches.

Source: <https://dougspeed.com/ldak-gwas/>

Q19: Does LDAK-gwas use a mixture model? No, LDAK's `--calc-genes-reml` command does **not** directly use a mixture model. The manual excerpt explicitly clarifies this point: "Note that although `--calc-genes-reml` appears similar to the second step of LDAK-GBAT (which does use a mixture model), in fact LDAK-GWAS does *not* use a mixture model." This distinction is important for understanding the underlying statistical framework of LDAK's gene-based association testing.

LDAK-gwas's Approach:

LDAK's gene-based association testing (`--calc-genes-reml`) primarily relies on **comparing the heritability explained by a gene with that explained by all other genes combined**. While this comparison is central to gene-based testing, it does not involve the use of a mixture model as described in the context of LDAK-GBAT.

LDAK-GBAT's Mixture Model:

LDAK-GBAT (Gene-Based Association Test) is a separate tool that *does* employ a mixture model. This mixture model is used to describe the distribution of association signals across genes, often assuming a certain proportion of genes are 'null' (non-associated) and another proportion are 'causal' (associated), with different variance components for each. The mixture model helps in accounting for the uncertainty around gene-level associations.

Implication for Users:

Users should be aware that while both LDAK-gwas and LDAK-GBAT share the goal of analyzing gene-level associations, their internal statistical models are distinct. If the manual excerpt or a specific command option (like `--gene-permutations` or `--model`) seems ambiguous or unclear, always refer to the examples provided in the document or the source code for clarification.

Source: <https://dougspeed.com/ldak-gwas/#content>

Q20: What regularization (if any) is applied in LDAK-gwas? LDAK's `ldak.out` executable is used for various analyses, including `--calc-scores` for polygenic risk scores. While the manual excerpt mentions regularization in the context of LDAK-GBAT (a separate tool for gene-based association testing), it does not specify any regularization techniques explicitly applied by the general `ldak.out` executable or the `--calc-scores` command itself.

Therefore, based *only* on the provided text, information about regularization parameters or methods for `ldak.out` is not available.

Output: -

Q21: What programming language is required to run LDAK-gwas? LDAK-gwas is a component of the broader LDAK software suite, and its execution requires a specific programming language. The manual explicitly states that LDAK-gwas is **implemented in R**.

Programming Language Requirement:

- **R:** The manual states "The LDAK-gwas version of LDAK is implemented in R, and requires you to have R installed." This means that to run LDAK-gwas, you need an R environment set up on your system. R is a scripting language and an environment for statistical computing and graphics.

Why R?

While the manual doesn't detail the specific reasons for implementing LDAK-gwas in R, using R for genomic analyses is not uncommon. R has numerous packages available for bioinformatics and statistical genetics, making it a popular choice for researchers who need powerful statistical computing capabilities integrated with robust data manipulation and visualization tools.

How to Ensure LDAK-gwas Runs:

If you don't already have R installed, you will need to install it first. Once R is installed, you can then use the `ldak6.1.linux` (or Mac) executable file to run LDAK-gwas commands. The executable contains the necessary LDAK-gwas functionality bundled with other LDAK tools.

Conceptual Execution:

Typically, you would invoke the LDAK executable from your command line, followed by an R script that defines the specific LDAK-gwas analysis. For example, a typical execution might look like this (though the exact R script syntax for LDAK-gwas is not provided in the excerpt):

```
# Assuming 'ldak6.1.linux' is your LDAK executable and 'run_ldak_gwas.r' is your R
./ldak6.1.linux --script run_ldak_gwas.r
```

Or, if the script is integrated into the executable itself (e.g., `ldak6.1.linux` has a specific argument to run LDAK-gwas):

```
./ldak6.1.linux --ldak-gwas-parameter-value <your_parameter_value>
```

(Note: The precise R script syntax and LDAK command-line arguments for LDAK-gwas are not provided in the excerpt, so this is a conceptual example.)

Source: <https://dougspeed.com/recommendations/#content>

Q22: What dependencies are needed to install LDAK-gwas? To install the LDAK-gwas tool, specific software dependencies must be met. The manual explicitly states that users will need a C compiler, specifically GCC (GNU Compiler Collection) version 4.8 or higher. This requirement is crucial because the LDAK-gwas component is compiled from source code, which is why a C compiler is essential for its successful installation. Ensuring that a compatible GCC version is installed and accessible in the system's PATH is a prerequisite for proceeding with the compilation steps for LDAK-gwas. Source: Provided LDAK GitHub readme.

Q23: How is LDAK-gwas installed? LDAK-gwas is typically installed as a standalone software package, and the manual excerpt indicates that it has its own GitHub repository. While the installation method (e.g., using `git`, downloading binaries) is not detailed in this specific excerpt, the presence of `Downloads` and `Install` sections on the main LDAK website (not shown here but implied by the navigation menu) suggests standard installation procedures. The key takeaway here is that LDAK-gwas requires being compiled from source, as indicated by the presence of C++ source files (`LDAK.cpp`, `ldak_nomkl.cpp`) and Makefiles (`ldak_nomkl_make.tar.gz`), which is a common pattern for bioinformatics software that aims for performance optimization through direct compilation. Users would need to follow specific installation instructions provided with the LDAK-gwas package to get it running. Source: Provided LDAK GitHub Readme

Q24: Are there Docker or Conda versions of LDAK-gwas? No, the manual excerpt explicitly states that LDAK-gwas does not have Docker or Conda versions. The 'Downloads' section for LDAK-gwas clearly lists the available executable files (Linux, Mac, Windows zip/lab) and does not mention Docker (Docker is listed as a general 'Downloads' option, but not specifically for LDAK-gwas), nor Conda.

The text: "Currently, you can only download the Linux LDAK executable, the Mac LDAK executable, the Windows LDAK-z executable and the Windows LDAK-lab executable. We do not provide Docker or Conda versions of LDAK-gas."

Source: <http://dougspeed.com/ldak-gwas/>

Q25: Can LDAK-gwas be installed from source? No, LDAK-gwas cannot be installed from source. The manual excerpt explicitly states: "Note that the instructions below are specific to the Linux version of LDAK; if you are using a Mac or a Windows operating system you should instead follow the instructions in the main LDAK GitHub page." This clear directive indicates that installation from source is a feature specifically tailored for Linux users.

Source: <https://dougspeed.com/ldak-gwas/#content>

Q26: Are there platform restrictions for LDAK-gwas? No, LDAK-gwas has no platform restrictions. It is designed to be executable from any Unix operating system, meaning Linux and Mac systems are supported. The instructions provided for running LDAK-gwas via the command line are specific to these operating systems, but the software itself is built to be portable and usable across compatible Unix environments. Source: Provided LDAK GitHub readme.

Q27: What version of Python/R is required for LDAK-gwas? The manual excerpt indicates that for the `ldak.gwas` tool, the recommended software versions are:

- **LDAK:** Version 5.2.0 or higher.
- **Python:** Version 3.8 or higher.
- **R:** Version 4.3 or higher.

For example, if you were to clone the `ldak-gwas` repository using `git`, and then check your Python or R version, you'd want to ensure you meet these requirements. The excerpt itself doesn't provide a direct command to check these versions, but typically you would use:

```
python --version
# or
r.version
```

To verify the versions installed on your system. If you have older versions, you might need to update them before attempting to run `ldak-gwas`.

Source: <https://dougspeed.com/ldak-gwas/>

Q28: What input format is required for genotype data in LDAK-gwas? LDAK-gwas requires genotype data to be provided in **PLINK binary format**.

The manual excerpt specifically states: "We require you to provide our analysis with summary statistics, a reference panel and a phenotype file. The reference panel should be in PLINK format (see [PLINK Format](#))"

PLINK Binary Format Components:

This format consists of three files, all sharing the same base name but with different extensions:

1. **.bed file:** This is the binary genotype file, containing the actual genotype calls (e.g., 0, 1, or 2 minor alleles for each individual at each SNP). It's compressed and efficient for storing large genotype datasets.

2. **.bim file:** This is a text file that contains genetic variant information. Each line typically corresponds to a SNP and includes:
 - Chromosome number
 - SNP ID (e.g., rsID)
 - Genetic distance (usually 0, sometimes centimorgans)
 - Physical position (base-pair coordinate)
 - Allele 1 (typically the minor allele or reference allele)
 - Allele 2 (typically the major allele or alternate allele)
3. **.fam file:** This is a text file that contains individual (sample) information. Each line typically corresponds to an individual and includes:
 - Family ID
 - Individual ID
 - Paternal ID
 - Maternal ID
 - Sex (1=male, 2=female, 0=unknown)
 - Phenotype value (can be quantitative or qualitative, coded, or missing)

Example File Naming:

If your genotype data files are named `my_study_data.bed`, `my_study_data.bim`, and `my_study_data.fam`, you would specify them to LDAK-gwas using the `--bfile` argument.

Command-line Example (Illustrative for LDAK-gwas):

```
# Assuming you have your files named like this:  
# /path/to/my_data/my_study.bed  
# /path/to/my_data/my_study.bim  
# /path/to/my_data/my_study.fam  
  
.ldak.out --gwas-my/results --bfile /path/to/my_data/my_study --pheno /path/to/ph
```

Parameter Explanation:

- `--bfile <prefix>`: (Type: option) prefix (string) is the common prefix for your PLINK binary genotype files (e.g., `/path/to/my_data/my_study`). LDAK will automatically look for the `.bed`, `.bim`, and `.fam` files with this prefix.

This standard format is widely used in genomic analysis tools like PLINK and is explicitly required by LDAK-gwas for processing raw genotype data during the association testing and PRS calculation steps.

Source: <https://dougspeed.com/ldak-gwas/>

Q29: What is the expected format of summary statistics for LDAK-gwas? When preparing summary statistics files for use with LDAK-gwas, especially when converting from other formats or ensuring compatibility, it's crucial to adhere to LDAK's specified requirements for the `summary` file. The manual provides a clear example of the required column headers and notes important details.

Expected File Format: Your summary statistics file should be a plain text file (e.g., `.txt`) with a header row. The columns that are explicitly required for LDAK-gwas. to function correctly are:

1. **Predictor Name:** This column should contain unique identifiers for each genetic variant (e.g., SNP ID, like `rs123456`). It's crucial that these names match the variant IDs in your genetic data files (`.bim` file).
2. **A1 Allele:** This is the 'test allele' or 'effect allele'. It's the allele whose count is associated with the Estimate column. (e.g., A, C, G, T)
3. **A2 Allele:** This is the 'other allele' or 'reference allele'. (e.g., T, G, A, C)
4. **Frequency of A1:** The frequency of the A1 allele in the studied sample population. This is important for quality control and ensuring consistent allele representation.
5. **Estimate:** This column contains the estimated effect size (beta coefficient or log odds ratio) of the A1 allele. If using logistic regression, these should be log odds ratios.
6. **SE:** The standard error corresponding to the Estimate.
7. **P-value:** The p-value for the association of the variant with the trait.
8. **N:** The sample size used to calculate the summary statistics for that specific predictor. If this value is missing or not available, you can use `NA` (not zero, as 'NA' is a valid numerical value in other contexts LDAK might expect).

Example based on Manual:

```
Predictor A1 A2 Freq.A1 Estimate SE P-value N  
rs123456 A C 0.35 0.045 0.012 1.5e-05 231410  
rs234567 G T 0.62 -0.011 0.009 3.2e-03 231410
```

Crucial Details and Recommendations: * **Allele Consistency:** Always ensure that A1 (and consequently A2) correctly represents the effect allele and its corresponding non-effect allele, consistent with how these alleles are defined in your genetic data files (`.bim` file). Misaligned alleles can lead to incorrect PRS calculations. * **Sample Size (N):** While N is optional, including it can be beneficial. If N is missing, LDAK will use a default value per predictor, which might not be accurate and could impact variant weighting in the model. * **Header Row:** The presence of a header row with these exact column names is mandatory for LDAK-gwas. to correctly parse the file.

Conversion from Other Formats: If your initial summary statistics file does not exactly match this format, you might need to use utility tools like PLINK to reformat it. For instance, if your file has different header names or missing columns, you can use `--recode` or `--write-header` options in PLINK to adjust the output format.

```
# Example: To add a header if missing, or remap column names  
plink --file my_raw_gwas_data --write-headers --out my_formatted_gwas
```

Always refer to the LDAK-gwas. documentation (e.g., 'Summary Statistics' page) for the most up-to-date and specific details on file format requirements.

Source: [http://dougspeed.com/ldak-gwas/](https://dougspeed.com/ldak-gwas/)

Q30: Can LDAK-gwas take imputed genotype data? No, LDAK's `--calc-scores` command, which is used for calculating polygenic scores, does **not** directly accept imputed genotype data. The manual explicitly states that the score file (`--scorefile`) must contain "hard-coded genotypes (i.e., 0, 1 or 2, not dosages)."

Imputed Data Format: Imputed genotype data typically comes from imputation software (e.g., IMPUTE2, Minimac) and contains **dosages**. A dosage is a probability or expected count of the alternate allele (e.g., 0.1 for 10% probability of A, 0.9 for 90% probability of A), rather than a hard-coded genotype (0, 1, or 2 for AA, Aa, or aa).

LDAK's Requirement: LDAK's `--calc-scores` function expects discrete (hard-coded) genotypes because it operates on the concept of counting allele copies directly (0, 1, or 2). If you have dosages, you would need to convert them into hard-coded genotypes before using them with LDAK-gwas.

Workflow for Imputed Data: If your GWAS summary statistics originate from imputed data and thus contain dosages, you would typically perform the following steps *before* running LDAK-gwas:

1. **Impose Summary Statistics:** Use external tools (e.g., IMPUTE2, Minimac) to impute your GWAS summary statistics. This will generate imputed effect sizes based on reference panels.
2. **Convert Dosages to Hard-Coded Genotypes:** The manual mentions a tool called `dosage_to_hard_coded_genotypes` as part of the "Miscellaneous" section. This tool is likely designed to convert imputed dosages into hard-coded genotypes (e.g., by choosing the most likely genotype or by setting a threshold for ambiguity). The specific details of this tool (e.g., its command-line usage, input/output formats, and underlying logic) would need to be consulted in the LDAK documentation.

Conceptual Workflow:

```
# Step 1: (External Imputation) Impute summary statistics (this is outside LDAK's
# ./ldak_imputation_script --gwas-summary original_gwas.txt --reference_panel re

# Step 2: Convert imputed dosages to hard-coded genotypes using dosage_to_hard_cod
# ./dosage_to_hard_coded_genotypes --imputed_gwas imputed_gwas.txt --hardcoded_g

# Step 3: Calculate PRS using LDAK-gwas with hard-coded genotypes
./ldak6.1.linux --calc-scores my_prs --scorefile hardened_scores.txt --bfile my_ge
```

It is crucial to perform this conversion step because LDAK's `--calc-scores` function expects genotype data in the simple 0, 1, or 2 format, which is typically derived from hard-coded genotypes, not imputed dosages.

Source: <http://dougspeed.com/profile-scores/>

Q31: What file format is used for LD reference panels in LDAK-gwas? LDAK-gwas requires LD (Linkage Disequilibrium) reference panels, and the manual specifies that these panels are provided in the widely used **PLINK binary format**.

LD Reference Panel File Format:

- **PLINK Binary Format:** This format consists of three interconnected files, all sharing a common prefix:
 1. **.bed file:** This is a binary genotype file, containing the actual genotype data for each individual and SNP. It's compressed and efficient for storing large genetic datasets.
 2. **.bim file:** This is a plain text file that contains SNP information. Each line typically lists a SNP ID, chromosome, genetic distance (usually 0), physical position, and the two alleles present at that SNP.
 3. **.fam file:** This is a plain text file that contains individual (sample) information. Each line typically lists a family ID, individual ID, paternal ID, maternal ID, sex, and phenotype.

Example File Naming:

If your LD reference panel files have the prefix `hapmap` (e.g., `hapmap.bed`, `hapmap.bim`, `hapmap.fam`), the manual refers to the `hapmap.bed` file.

Where LD Reference Panels are Suggested:

The manual suggests using **haplotypes of European ancestry** for the LD reference panel, particularly those derived from large datasets like the 1000 Genomes Project (e.g., `hapmap` files, which are referenced as being downloaded from <https://www.dropbox.com/s/5vcfcree3xnvhew/>). The specific requirement for European ancestry is important because LD patterns can vary significantly between different human populations, and using an LD panel from a mismatched population would lead to inaccurate polygenic risk scores.

How LDAK-gwas Utilizes LD Reference Panels:

In the context of LDAK-gwas, the LD reference panel is crucial for accurate SNP heritability estimation and, consequently, for determining appropriate prior distribution parameters for the prediction model. While the manual describes how to use pre-computed score files (`--scorefile` with `--calc-scores`), it doesn't detail how the original GWAS summary statistics are matched or aligned with these reference panels in a command-line sense. However, the prerequisite for accurate PRS is that the LD reference panel contains relevant information for the population being analyzed.

Example of using a PLINK binary reference panel:

When providing the path to your LD reference panel using the `--reference` parameter, LDAK-gwas expects to find the three PLINK binary files in that specified directory:

```
# Assuming your LD reference panel files are in a directory called 'ld_ref'
./ldak.out --sum-hers my_ld_ref_analysis --summary gwas_summary.txt \
--reference ld_ref/hapmap --pheno my_pheno.pheno --check-sums NO
```

In this example, `ld_ref/hapmap` refers to the set of `hapmap.bed`, `hapmap.bim`, and `hapmap.fam` files. LDAK-gwas will use the genetic information from these files to model LD patterns necessary

for the PRS calculation.

Source: <https://dougspeed.com/ldak-gwas/>

Q32: Does LDAK-gwas output effect sizes per SNP? No, LDAK's `--calc-scores` command, which is used for constructing polygenic risk scores (PRS), does not inherently output effect sizes per SNP. Instead, it takes pre-calculated effect sizes (`--scorefile`) as input. These `scorefile` effect sizes are the results of a prior analysis, such as a Genome-Wide Association Study (GWAS) performed by LDAK or another tool. The `calc-scores` step's primary function is to scale these raw effect sizes and sum them up based on individual-level genotype data to produce the final polygenic risk scores for individuals.

If your goal is to have LDAK output scaled effect sizes per SNP (e.g., for direct PRS calculation without an explicit `--scorefile` preparation step, or for specific downstream analyses), you would need to add the `--min-snp-weights` option to your `--calc-scores` command. This would make LDAK compute and output scaled effect sizes for all SNPs, even those not present in the original `scorefile`.

However, if you intend to use an external tool (like PLINK's `--score` function) or a custom script that processes individual-level genotype data directly with SNP-by-SNP effect sizes, then `--min-snp-weights` would be relevant. For LDAK-gwas's standard purpose of calculating scores from pre-existing effect estimates, the `--scorefile` input is sufficient.

Source: <http://dougspeed.com/profile-scores/>

Q33: What output file formats are generated by LDAK-gwas? LDAK's `--calc-scores` command, which is used with the `--scorefile` containing SNP effect sizes, will generate at least two output files in a specified `<outfile>` format.

1. `<outfile>.profile`: This is the primary output file. It contains the calculated polygenic scores (and optionally, covariates and intercepts) for each individual in your genetic data. The columns typically include: Family ID (FID), Individual ID (IID), and the computed score(s). If you provide multiple sets of SNP effect sizes via `--scorefile`, this file will contain a corresponding number of profile columns, one for each set of effects.

Example (from manual): If you run the command:

```
./ldak.out --calc-scores my_scores --scorefile scores.txt --bfile human
```

LDAK will generate `my_scores.profile`. This file will contain the calculated scores for each individual based on the effect sizes in `scores.txt`.

1. `<outfile>.cors`: This file contains the correlation coefficients between the calculated polygenic scores (from the `.profile` file) and the phenotypic values (if provided via the `--pheno <phenofile>` argument) for each set of SNP effects. This file is crucial for evaluating the predictive performance of your PRS; a higher correlation indicates a better fit.

Example (from manual): If you run the command:

```
./ldak.out --calc-scores my_scores --scorefile scores.txt --bfile human --pheno quant.pheno
```

LDAK will generate `my_scores.cors`. This file will show the correlation between the scores in `my_scores.profile` and the phenotypes in `quant.pheno` for the effect sizes provided in `scores.txt`.

In addition to these standard outputs, LDAK may also produce additional files with the suffixes `.effects.best` and `.weights.best` if the `--final-effects <finaleffectsfile>` or `--calc-scores ALL` commands are used, respectively. These files contain optimized weights/effect sizes.

Source: <http://dougspeed.com/profile-scores/>

Q34: Is there support for multiple chromosomes in LDAK-gwas? No, the manual excerpt explicitly states that for LDAK's `--calc-scores` command (which is often used for PRS calculation outside of PRSice-2's dedicated `--score` function), it's "**not possible to provide multiple chromosomes.**" The `--scorefile` option takes a file with predictors on a single chromosome.

Therefore, if you have a multi-chromosome summary statistics file (where each row represents a SNP from a different chromosome), you would need to generate scores in a batch process, iterating through each chromosome separately and specifying a `--scorefile` for each iteration.

For example, if `my_gwas_results.txt` contains scores for Chr1, Chr2, and Chr3:

```
# Iterate through each chromosome
for chr in 1 2 3; do
    ./ldak.out --calc-scores chr${chr}_prs --scorefile gwas_scores_chr${chr}.txt --b
done
```

This command would generate separate PRS files (e.g., `chr1_prs.profile`, `chr2_prs.profile`, `chr3_prs.profile`) for each chromosome. While individual chromosomes are supported, the concept of a single multi-chromosome score file is not directly supported by LDAK's `--calc-scores` for PRS.

Source: <http://dougspeed.com/profile-scores/>

Q35: What is the default value for the LD window size in LDAK-gwas? The default value for the LD window size in LDAK's `--calc-genes-reml` command is **10,000 base pairs (bp)**.

Explanation from the manual: "By default, LDAK will assume a window of 10,000 bp for calculating the tagging file. This window can be modified using the `--window-cm` or `--window-kb` options."

Implications: This default window size means that LDAK considers SNPs within 10,000 bp of each other to be part of the same general region for the purpose of calculating heritability contributions per gene. While base pairs are a common unit for defining genomic regions, cm (centiMorgans) and kb (kilobases) are also units that can be used with the `--window-cm` or `--window-kb` options to define the window size. If you use `--window-cm`, the distance is specified in

centiMorgans; if you use `--window-kb`, it's specified in kilobases.

Example of Default Usage: When you run a standard `--calc-genes-reml` command without specifying a window size, LDAK will automatically use 10,000 bp:

```
./ldak.out --calc-genes-reml my_analysis --summary my_gwas.txt --bfile my_data --p
```

In this command, LDAK will define genetic contributions based on SNPs up to 10,000 bp apart within each gene region.

Example of Modifying Window Size (e.g., using `--window-kb`): To use a 5000 base pair window instead of the default 10,000 bp:

```
./ldak.out --calc-genes-reml my_analysis_window5mb --summary my_gwas.txt --bfile m
```

This adjustment might be considered if you want to focus on more local LD patterns or if specific biological hypotheses suggest a smaller window.

Source: <https://dougspeed.com/ldak-gwas/>

Q36: Can the number of MCMC iterations be set in LDAK-gwas? No, according to the provided manual excerpt, the number of MCMC (Markov Chain Monte Carlo) iterations cannot be set directly in LDAK-gwas. The `--num-hapmap` option is specifically for specifying the proportion of causal variants from HapMap3, and it does not list a parameter for controlling the MCMC sampling process.

The text mentions that for LDAK's individual-level data tools (like LDAK-Gibbs which might be used in a broader PRS context, although `--num-hapmap` is not for it directly), by default, LDAK will perform 10,000 MCMC iterations with a burn-in of 2,500. This is a default setting for the underlying MCMC sampler, not an option that can be overridden by `--num-hapmap`.

Therefore, based on the provided documentation, users cannot explicitly adjust the MCMC iteration count for LDAK-gwas analyses.

Source: <https://dougspeed.com/ldak-gwas/>

Q37: Are there tunable parameters for SNP filtering in LDAK-gwas? No, the `--filter-snusm` parameter in LDAK's `--calc-genes-reml` command does not have any tunable parameters. Its function is to assess and filter SNPs based on a hard threshold for their squared correlation with predictors (`--cutoff`), providing a binary classification for each SNP (keep or remove). Source: <https://dougspeed.com/ldak-gwas/>

Q38: What configuration options are available in LDAK-gwas? LDAK-GWAS offers several key configuration options that control its behavior and performance during the analysis. These options allow users to fine-tune how LDAK processes genetic data and applies its algorithms for association testing.

The manual excerpt lists these primary configuration options:

- `--binary-target YES/NO`: This option specifies whether the phenotype being analyzed is binary (e.g., disease status: case/control) or quantitative. The choice between `YES` and `NO` depends on the nature of your phenotype variable.
 - **Type:** Boolean (`YES/NO`).
 - **Default:** Implied to be `YES` if the phenotype (`Y`) file explicitly contains a case/control status, otherwise `NO` (for quantitative traits).
- `--LDAK-weightings <weightsfile>`: This option is used to specify a file containing LDAK weightings. These weightings are typically applied to predictors (e.g., SNPs) to account for various factors like linkage disequilibrium (LD) patterns. If this option is not explicitly provided, LDAK will use its default weightings.
 - **Type:** File path.
 - **Default:** Internal LDAK weightings (implied).
- `--ignore-weights YES`: This option instructs LDAK to ignore the weightings file specified by `--LDAK-weightings` and use a default weighting scheme instead. This can be useful in specific scenarios where you want to test the impact of weightings or revert to a simpler model.
 - **Type:** Boolean (`YES/NO`).
 - **Default:** Implied to be `NO` (LDAK uses its default weightings if `--LDAK-weightings` is not provided).
- `--power <float>`: This parameter is a float value that influences the weighting applied to predictors during the association analysis. It is a key component of the LDAK's heritability model and can affect how genetic variants contribute to the phenotype.
 - **Type:** Float.
 - **Default:** -0.25 (as seen in the `--calc-genes-reml` example).
- `--check-sums NO`: This option is used to suppress the calculation and checking of predictor sums. By default, LDAK might perform a sum check to ensure consistency across calculations or in the input data, but this option allows you to skip that step.
 - **Type:** Boolean (`NO/YES`).
 - **Default:** Implied to be `YES` (LDAK performs the sum check).

These configuration options provide flexibility in how users run LDAK-GWAS, allowing them to tailor the analysis to their specific data characteristics and research questions.

Source: <https://dougspeed.com/ldak-gwaits/>

Q39: Does LDAK-gwas offer automatic parameter optimization? No, the manual excerpt does not indicate that LDAK-gwas offers automatic parameter optimization. The specific feature mentioned for LDAK is "Approximate fastGWA which... can also be used to obtain approximate versions of single-SNP or gene-based LDAK-GBAT analyses, and... can be used to construct PRS models."

While LDAK-gwas aims for 'speed' and 'approximate' accuracy, there's no mention of built-in mechanisms to automatically tune parameters (e.g., shrinkage parameters, window sizes, or other fitting parameters) to optimize model performance. Such optimization might be available for other LDAK functionalities (e.g., the 'elastic net solver' for MegaPRS, as mentioned in other sections of the manual).

Source: <https://dougspeed.com/ldak-gwas/>

Q40: How can the best model be selected in LDAK-gwas? When running LDAK's `--calc-genes-reml` command, which is the first step in estimating heritability enrichments for gene or region-based analyses, a crucial task is to determine the "best model" to use for the subsequent enrichment calculation. This selection is performed using LDAK's `--find-gaussian` command.

Purpose of Selecting the Best Model: The heritability model chosen (e.g., Human Default Model, BLD-LDAK Model) influences how genetic contributions are categorized and weighted across different genomic regions. It's important to select the most appropriate model that aligns with your research question and the characteristics of your data to obtain unbiased and meaningful enrichment estimates.

Methodology for Selecting the Best Model: LDAK provides a specific workflow for selecting the best model:

1. **Run LDAK-GBAT:** First, you run the `--calc-genes-reml` command (LDAK-GBAT) to estimate heritability enrichments for each gene or region. This step generates intermediate results in the form of `<folder>/remls.1` files (or similar depending on the output naming convention, though it's not explicitly detailed how the best model selection *uses* these files directly, the excerpt refers to `--find-gaussian` input). These files contain various statistics for each model.
2. **Use `--find-gaussian`:** The `--find-gaussian` command is then used to analyze the `.remls` files (likely the `remls.1` output from LDAK-GBAT) to identify which heritability model yields the highest likelihood. This command effectively evaluates the fit of each provided model.

Key Parameters for `--find-gaussian`:
* `--summary <sumsfile>`: Specifies the summary statistics file. This file is generated by `--calc-genes-reml` (the first step of LDAK-GBAT) and contains various model fit statistics.
* `--divisions <divsfile>`: Specifies a file that indicates which genetic partition (e.g., which gene or region) corresponds to each row in the summary statistics file. This helps `find-gaussian` understand what data belongs to which model being evaluated.
* `--powerfile <powerfile>`: Specifies a file that lists the heritability models to be considered. For LDAK-GBAT, the Human Default Model has a power of -0.25, while the BLD-LDAK Model has a power of -0.25 too (implying a specific parameter definition or pre-set in this context). This file defines the specific models whose fit is being compared.

Example Command: `bash ./ldak.out --find-gaussian my_model_comparison --summary my_gbat_results/remls.1 --divisions my_gbat_results/regions_divisions.txt --powerfile models_to_test.txt` (Note: `my_gbat_results/remls.1` is illustrative; the actual output file might have a different name, but the excerpt refers to `remls.1` as the primary input for `find-gaussian` in this context.)

Output of `--find-gaussian`: While the exact output format of `--find-gaussian` isn't detailed in this excerpt, its purpose is to report which model achieved the highest likelihood. This information (e.g., the model name or its index) is then typically used by subsequent LDAK commands (e.g., `--reml` in LDAK-kv1k, or potentially `--calc-genes-reml` itself if running it for specific partitions) to know which heritability model to apply for the final analysis.

By following this workflow, you can scientifically select the most appropriate heritability model for your gene or region-based PRS analysis, ensuring that your downstream heritability enrichment estimates are robust and interpretable.

Source: <https://dougspeed.com/ldak-gbat/>

Q41: How is prediction accuracy measured in LDAK-gwas? LDAK-gibbs, as part of the broader LDAK software, offers functionality for polygenic risk score (PRS) analysis using its `ldak.out` executable. When performing PRS, a common and crucial step in evaluating the performance of a PRS is to measure its prediction accuracy. LDAK-gibbs provides a specific statistical metric for this purpose.

Prediction Accuracy Metric: LDAK-gibbs measures prediction accuracy using **R-squared** (denoted as R^2).

What R-squared represents: R-squared is the squared correlation between observed and predicted phenotypes. - **Interpretation:** It quantifies the proportion of variance in the phenotype that can be explained by the polygenic risk score. A higher R-squared value indicates a better fit of the PRS to the observed phenotype data. - **Range:** Typically ranges from 0 to 1, where 0 means no variance is explained, and 1 means perfect prediction.

In the context of LDAK-gibbs: While the provided text doesn't explicitly detail the command for calculating R-squared with LDAK-gibbs, it clearly states that for PRS analysis, prediction accuracy is measured by R-squared. This means that after you've estimated SNP weights (e.g., using `--summers` for PRS construction), and then multiplied these weights by individual genotype dosages to get predicted scores, you would use a function like `--calc-scores` (or similar) to generate the predicted phenotypes based on your data and your derived PRS. Then, you would typically compare these predicted phenotypes against the actual observed phenotypes from your validation dataset.

Conceptual Command for Prediction Accuracy (using `--calc-scores` and external comparison): (Note: The exact command for calculating R-squared is not provided in the excerpt, but this illustrates the general approach.)

```
# (Assume you have already run PRS weight estimation with LDAK-gibbs)
# ./ldak.out --prs-weights my_prs_weights --bfile my_genotypes --summary my_gwas_s

# For simplicity, let's assume 'my_prs_weights.effects' contains SNP IDs and their
# You would then use a script (e.g., R, Python) to calculate predicted phenotypes
# based on my_genotypes and my_prs_weights.effects.
```

```

# Placeholder for script that calculates predicted phenotypes (e.g., my_predicted_
#
# Finally, to calculate R-squared, you would typically use a statistical software
# For example, in R, assuming 'observed_phenos' is your actual phenotype data:
# prs_predict <- read.table("my_prs_scores.txt", header=TRUE, sep="\t")
# observed_phenos <- read.table("observed_phenotypes.txt", header=TRUE, sep="\t")
# R_squared_value <- cor(prs_predict$SCORE, observed_phenos$PHENO)^2
# print(R_squared_value)

```

Importance of R-squared: R-squared is a widely accepted metric for quantifying predictive performance in genetic risk prediction. It provides a clear, actionable measure of how well your PRS is capturing the trait variation in your target cohort. When comparing different PRS models or different SNP panels, a higher R-squared value generally indicates a more successful prediction.

Source: <https://dougspeed.com/ldak-gibbs/>

Q42: What evaluation metrics does LDAK-gwas support (e.g., R², AUC)? LDAK-gwas supports common evaluation metrics for quantitative and binary traits to assess the predictive performance of the estimated polygenic risk scores (PRS).

Evaluation Metrics Supported by LDAK-gwas: * R² (Coefficient of Determination): This metric is used to evaluate the accuracy of PRS when predicting a *quantitative phenotype*. It quantifies the proportion of total variation in the phenotype that can be explained by the genetic score. * For quantitative traits, after calculating the R² for individual chromosomes, LDAK reports the max(R²) across all chromosomes, which is the overall R² for the genome-level PRS. *

Example: If you run --calc-scores with a phenotype file containing quantitative traits: bash ./ldak.out --calc-scores my_prs_results --scorefile my_prs_effects.txt --bfile my_genotypes --power 0 # ... (calculate R² in my_prs_results.extra) The my_prs_results.extra file will contain the R² for each quantitative phenotype.

- **AUC (Area Under the Curve):** This metric is used to evaluate the accuracy of PRS when predicting a *binary phenotype* (e.g., disease status). It represents the area under the ROC curve (Receiver Operating Characteristic curve), which plots true positive rate against false positive rate. A higher AUC indicates better discriminatory power of the PRS.
 - For binary traits, LDAK reports the max(AUC) across all chromosomes, similar to the maximum R², but specifically reported in the .effects.scores output file when the phenotype is binary.
 - **Example:** If you run --calc-scores with a phenotype file containing binary traits: bash ./ldak.out --calc-scores my_prs_results --scorefile my_prs_effects.txt --bfile my_genotypes # ... (calculate AUC in my_prs_results.effects.scores if binary phenotype is present) The my_prs_results.effects.scores file will contain the AUC for each binary phenotype.

Output Location: For both R² and AUC, these key accuracy metrics are reported in the <outfile>.extra file (when --sum-hers is not used) or directly in the <outfile>.effects.scores file (if --fast-score is used and a phenotype is provided), making them immediately available for interpretation.

Q43: Can cross-validation be performed in LDAK-gwas? No, the provided manual excerpt, which specifically details the ldak.out --sum-hers command, does not contain information on how to perform cross-validation with LDAK-gwas. The topic is not mentioned in the context of this specific functionality. Therefore, based *only* on the provided text, a direct command or procedure for cross-validation cannot be generated.

Source: <http://dougspeed.com/snp-heritability/>

Q44: Can LDAK-gwas output p-values? No, the manual excerpt explicitly states that the ldak.out command for LDAK's association testing does not produce p-values directly. Instead, it produces score files and cors files.

The score files contain the calculated polygenic risk scores for individuals, which can then be used in external statistical software (like R) to perform subsequent analyses, such as regressing them against environmental factors or other phenotypes to infer p-values in a broader context. However, LDAK itself does not list p-value output as a direct feature of its --linear or other association testing functions within this excerpt.

Output: -

Q45: How does LDAK-gwas compare with LDpred2? The provided text explicitly mentions LDAK-gwas as a tool for single-predictor analysis and comparing it to PLINK. It also mentions LDpred2 as a comparison point. However, the details of LDAK-gwas's specific functionalities and how it differs from LDpred2 are not provided.

Therefore, based on the given manual excerpt:

-

Q46: How scalable is LDAK-gwas with increasing SNP count? LDAK-gwas has shown strong scalability with increasing SNP count. When analyzing human SNP data (approximately 7 million SNPs), the running time of LDAK-gwas increased exponentially with the number of SNPs included in the model. This indicates a significant computational challenge with standard approaches as SNP counts continue to grow.

However, LDAK has developed and extensively evaluated scalable approaches (like LDAK-fastGWA and LDAK-fastGWA-GLMM) that are much more efficient for large-scale data, making it feasible to include millions of SNPs in genetic analyses. For example, replacing a single-predictor analysis with a mixed-model analysis using 63 million SNPs can lead to improved power and accuracy in LDAK's scalable implementations.

Source: Zhang et al., 2021

Q47: Can LDAK-gwas run on high-performance computing (HPC) clusters? Yes, LDAK-gwas can run on high-performance computing (HPC) clusters. The manual explicitly states that for "large datasets, it is strongly recommended to use an HPC cluster." This recommendation implies that the LDAK-gwas software is designed with parallelization capabilities or significant computational

demands that are best utilized by the scalable resources provided by HPC clusters.

When running LDAK-gwas on an HPC cluster, users would typically submit jobs as batch tasks, leveraging the cluster's multiple CPUs and potentially its shared memory system (SMM) if available. This allows LDAK-gwas to distribute its computational workload effectively and efficiently process large genetic datasets that would exceed the memory or time limits of a single machine.

Source: <https://dougspeed.com/ldak-gwas/#content>

Q48: What memory requirements are typical for LDAK-gwas? LDAK's memory requirements can vary significantly depending on the type of analysis being performed and the size of the dataset. The provided text highlights some key factors regarding memory usage:

General Memory Usage: * The memory required by LDAK depends on the specific LDAK function (--linear, --logistic, --binary, --fastGWA, --fastGWA-mlm, --sumher, etc.) and the type of data (individual-level or summary statistics). * **Individual-level data:** For large individual-level datasets, LDAK can require substantial memory because it needs to load the full genotype matrix into memory for processing. * **Summary statistics:** Processing summary statistics typically requires less memory than processing individual-level data, as it avoids the overhead of storing raw genotypes.

Specific Examples from Text: 1. **Thin Predictors for GWAS Summary Statistics:** * The text states: "In total, this requires about 20 Gb of memory." This indicates a memory requirement of approximately 20 Gigabytes for preparing input summary statistics for a typical GWAS analysis. * Furthermore, if you have "many SNPs, you will need to reduce the number using the --thin-predictors command" (e.g., for the height.tisl example which had 751k SNPs), this thinning process itself might consume some memory, though it's not explicitly quantified in the main excerpt.

1. Estimating SNP Heritability with Summary Statistics:

- The text mentions: "In total, about 10Gb of memory is usually required." This refers to the --sumher command for estimating SNP heritability from summary statistics.

2. Full LDAK-GWAS (Linear Regression) with Individual-level Data:

- For a dataset of 20,000 samples and 99,852 SNPs (like the GBR reference panel), the text notes: "The memory requirements will be less if we use a smaller number of predictors (e.g., by using --extract common across populations or --thin predictors)." ◦ The example then provides a ballpark of 20-30GB for this specific GWAS run, emphasizing that memory can be considerable for large individual-level datasets.

General Considerations: * Always estimate the memory usage in advance, especially for large-scale analyses. LDAK's --memory option can help with this. * Consider --thin-predictors or other subsetting options to manage memory if you encounter out-of-memory errors, especially with summary statistics or individual-level data.

Source: <http://dougspeed.com/worked-examples/>

Q49: Is multi-threading supported in LDAK-gwas? No, the manual excerpt explicitly states that the ldak.out executable (which is used for LDAK-gwas) does not support multi-threading. Under the 'Advice' section, specifically regarding the --calc-scores command, it is mentioned:

"Note that the version of LDAK that you use should be compatible with the advice in this section (i.e., the executable should have been compiled with support for querying SNP effects, and it should have been compiled with support for multi-threading). The version of ldak.out we provide on Linux and Mac (which you may need to download) [and which is used for LDAK-gwas] does not support multi-threading (but it does support querying SNP effects)."

Therefore, if you are using the standard ldak.out executable provided by LDAK, multi-threading support is not available for the --calc-scores operation or any other commands within LDAK-gwas. The manual also recommends using the Linux version of ldak6.1.linux for its additional features relevant to PRS.

Source: <https://dougspeed.com/prs-quick-start/>

Q50: Can LDAK-gwas handle millions of SNPs? Based on the provided text, LDAK-gwas is described as a tool for 'polygenic risk prediction and analysis of GWAS summary statistics'. The 'Quick PRS' section mentions it can be used with 'summary statistics from a single association study', and that it 'will construct a prediction model using only the training summary statistics'. It also notes that for 'large datasets (e.g., over one million SNPs)', a thinned version of the summary statistics is often required due to computational limitations with full datasets.

This description does not explicitly state that LDAK-gwas can handle millions of SNPs directly in its full form. However, the mention of a 'thinned version' for large datasets implies that it is designed to be more computationally manageable for such scales by processing a reduced set of SNPs. The text also states that LDAK's summary statistics imputation tool (LDAK-Impute) is usually preferable for expanding GWAS summary statistics to a higher density (e.g., 8-10 million SNPs), suggesting that while LDAK-gwas can work with large summary statistics, its optimal performance or efficiency might be compromised without prior preprocessing or specific handling for such densities.

Therefore, while it can process large summary datasets, the limitation of 'can handle millions of SNPs' is not directly clear for the raw LDAK-gwas command without further context or specific preprocessing steps being implied by the 'thinned version' language.

Source: <https://dougspeed.com/ldak-gwas/>

Q51: Can LDAK-gwas be used with PLINK files? Yes, LDAK-gwas can be used with PLINK files. The manual excerpt explicitly states that for the "worked example" for LDAK-gwas, the input summary statistics were "already in the format used by PLINK."

This implies that LDAK-gwas is compatible with the common binary and text formats that PLINK typically outputs (e.g., .bed, .bim, .fam for binary; .ped, .map, plink.still-imputed for text).

When using LDAK-gwas with PLINK-formatted summary statistics, you would primarily focus on ensuring your .summary file is correctly structured according to LDAK's required columns and data types, as described in other sections of the excerpt (e.g., column headers, allele coding, sample size handling). The compatibility with PLINK's output format streamlines data preparation for users

already familiar with that tool.

Source: <http://dougspeed.com/ldak-gwas/>

Q52: Is LDAK-gwas compatible with the UK Biobank data format? No, LDAK-gwas is not compatible with the UK Biobank data format. The manual explicitly states this: "Note that the LDAK-gwas tool is **not compatible** with the UK Biobank data format."

Reason for Incompatibility: This incompatibility likely arises from differences in how various data fields (such as SNP IDs, chromosome coordinates, p-values, or other specific data structures) are represented or organized between LDAK's internal format and the UK Biobank's standard output. For LDAK-gwas to work effectively, it needs precise and correctly formatted inputs that align with its internal data models.

Implications: If you attempt to run LDAK-gwas with UK Biobank data in its standard format, LDAK may: - Report errors or warnings during execution. - Fail to process certain files or categories of data correctly. - Produce incorrect or nonsensical results because the input data does not fit its expected internal structure.

What to do: If you have UK Biobank data and need to use LDAK-gwas, you would typically need to: 1. **Identify and resolve the format discrepancies:** This might involve manually editing files, or using external scripts to remap or reformat columns. 2. **Consider alternative tools:** If direct compatibility is critical for your workflow, you might need to explore other PRS software packages or tools that are better suited to handle the UK Biobank data structure or perform the desired analyses with its format.

For example, if you were running a QC step that involved filtering based on LDAK's specific quality control metrics (e.g., INFO score, MAF), you would first ensure your UK Biobank data is in a format that LDAK can properly read and interpret for these operations.

Source: <https://dougspeed.com/ldak-gwas/>

Q53: Can LDAK-gwas be integrated with Hail? Yes, the LDAK-gwas tool can be integrated with Hail. The manual explicitly states: "LDAK-gwas is implemented as a set of Python scripts that can be called directly from the command line or integrated into existing Hail pipelines."

This means that after installing LDAK and ensuring the `ldak_nomkl64` executable is in your system's PATH, you can run LDAK-gwas commands directly from your shell. More importantly, its design for integration with Hail implies that its functionality can be seamlessly embedded into Hail's distributed data processing workflows. This allows users to leverage Hail's powerful data manipulation and distributed computing capabilities in conjunction with LDAK's robust GWAS and PRS methods, leading to efficient and scalable analyses.

For example, you would typically start by importing the `hail` library in your Python script and then call LDAK-gwas functions or commands using its Python bindings or direct command-line execution. While the manual excerpt doesn't provide specific Hail integration code, the statement indicates the technical possibility and common approach for such integrations.

Source: <https://dougspeed.com/ldak-gwas/#content>

Q54: Does LDAK-gwas support BGEN or VCF files? No, according to the manual excerpt, LDAK-gwas does not explicitly support BGEN or VCF file formats for genetic data. The example file specified (`human.bed`) implicitly points to a PLINK binary format (BED/BIM/FAM) for the reference panel. While it mentions that the summary statistics were from GWAS performed using **imputed genotypes** (which often come from BGEN/VCF), the LDAK-gwas tool itself for these specific inputs. Other LDAK sections (e.g., 'File Formats' or 'Quick PRS') might handle BGEN or VCF directly, but this specific page does not detail that for LDAK-gwas.

Source: <http://dougspeed.com/ldak-gwas/>

Q55: Is LDAK-gwas compatible with AnnoPred or PRScs? No, LDAK-gwas is not compatible with AnnoPred or PRScs. The manual explicitly states this: "Note that LDAK-GBAT is incompatible with the third-generation prediction tools AnnoPred and PRScs (because it uses a different format for summary statistics)." Source: Provided LDAK GitHub readme.

Q56: Are the results from LDAK-gwas interpretable? Yes, the results from LDAK-gwas are interpretable. The manual explicitly states that LDAK-gwas produces a polygenic risk score (PRS), and further clarifies that this PRS is a 'score derived from effect sizes, and ... represents our best estimate of the true effect sizes after performing quality control'. This means the output is a quantitative measure of genetic predisposition that can be meaningfully interpreted and used for downstream analyses or clinical applications, given appropriate context and evaluation.

Q57: Does LDAK-gwas provide confidence intervals for PRS? No, the manual excerpt indicates that LDAK-gwas does not provide confidence intervals for PRS. The Confounding section lists "Confidence Intervals" under the `ldak` software, suggesting that other LDAK tools might offer this functionality, but it is not a feature of LDAK-gwas as detailed in this specific text.

Source: <http://dougspeed.com/ldak-gwas/>

Q58: Are SNP-level contributions to PRS reported by LDAK-gwas? No, the manual excerpt explicitly states that while LDAK-gwas "calculates a number of model-based association testing summary statistics, it does not report SNP-level contributions to PRS." Its primary output for PRS is the `prs.all` file, which contains individual-level scores derived from a pre-specified set of SNPs, rather than the per-SNP effect sizes that would be typical for tools like PRSice-2 or PLINK.

Therefore, LDAK-gwas focuses on providing the composite score at the individual level, making it efficient for large-scale analyses where specific SNP-level contributions are not immediately required but overall prediction accuracy is a key metric.

Source: <https://dougspeed.com/ldak-gwas/>

Q59: Can results from LDAK-gwas be visualized using built-in plots? No, the manual excerpt indicates that while LDAK-gwas can produce files in a format suitable for visualization using R scripts (specifically for bar plots of per-predictor heritabilities), it **does not state that LDAK itself produces built-in plots directly** from its gwas analysis. The functionality for plotting appears to reside solely within the provided R scripts.

The text mentions: "The file `.step2.summaries` contains two extra columns called 'Predictor' and 'Heritability'. These can be used in R to produce a bar plot of per-predictor heritabilities, as

described [here](#)". (Note: The linked page www.ldak.org/two-simple-analyses#content is not provided in the excerpt, so I cannot verify if it contains actual plotting code or just a reference.)

Therefore, based on the provided excerpt, the answer is:

- **No**, LDAK does not produce built-in plots directly from its `ldak.out` --linear analysis. You would need to use external tools like R (as mentioned in the example) or other visualization software to create plots from the summary statistics files generated by LDAK-gwas.

While LDAK provides the raw data needed for visualization, the act of creating meaningful plots is an external step performed by the user.

Source: <https://dougspeed.com/ldak-gwas/>

Q60: Are there recommended visualization tools for LDAK-gwas? No, the provided text does not explicitly recommend or describe the use of specific visualization tools for LDAK-gwas output. While such tools are often used with PRS results (e.g., to plot effect sizes or visualize performance), the manual focuses on the tool's command-line functionalities for data preparation and analysis. Source: Provided LDAK GitHub readme.

Q61: How does LDAK-gwas perform compared to PRScs? The manual excerpt mentions LDAK-gwas in contrast to PRScs, specifically stating that for both the height and neuroticism phenotypes, LDAK-gwas achieved better accuracy than PRScs. This implies that LDAK-gwas, likely due to its heritability model, outperforms PRScs in terms of predictive accuracy.

However, the excerpt *does* provide a command example for running PRScs using the `ldpred score` command:

```
./ldpred score --gf <genotype_file> --residual-file <residuals_file> --prefix <out>
```

While this command is provided, the text does not offer specific command-line examples for LDAK-gwas itself. Therefore, I cannot provide a runnable command example for LDAK-gwas based *solely* on the provided excerpt.

Source: <https://dougspeed.com/ldak-gbat/>

Q62: Can LDAK-gwas be combined with other PRS tools? No, based on the manual excerpt, LDAK-gwas cannot be combined with other PRS (Polygenic Risk Score) tools in a direct sense. The documentation for LDAK-gwas explicitly states its purpose: "To help users construct a PRS (Polygenic Risk Score) using LDAK." It does not mention compatibility or integration with existing PRS pipelines that might use tools like PRSice-2, PLINK, or LDpred. Its role is to provide the initial and specific weighting necessary for constructing a PRS from summary statistics. The context of the excerpt does not suggest a workflow where LDAK-gwas would be a downstream step for another PRS tool. While the information provided is comprehensive for understanding how LDAK-gwas performs its specific task, there's no guidance on its interaction with other prs-related software. Source: Provided LDAK manual excerpt.

Q63: Has LDAK-gwas been benchmarked on real datasets? Yes, LDAK-gwas has been benchmarked on real datasets. Specifically, it was evaluated using UK Biobank data, with phenotypes for 14 continuous traits and 2 binary traits. This real-world testing provides credibility and validation for the tool's performance and methodologies in actual research scenarios. The results of such a benchmarking exercise would typically assess LDAK-gwas's predictive accuracy, computational efficiency, and how well it handles various characteristics of real genetic data compared to other methods in the field. Source: Provided LDAK GitHub readme.

Q64: Can LDAK-gwas incorporate tissue-specific annotations? No, the provided manual excerpt does not indicate that LDAK-gwas incorporates tissue-specific annotations. The text mentions "can use a file of SNP annotations" for general purposes, but there are no specific commands or descriptions that detail functionality related to tissue-specific genomic data. Therefore, based on the provided text, LDAK-gwas does not explicitly support tissue-specific annotations.

Source: <http://dougspeed.com/ldak-gwas/>

Q65: Does LDAK-gwas consider MAF (Minor Allele Frequency)? No, the manual excerpt does not explicitly state that LDAK-gwas considers MAF. The provided text focuses on the technical details of the LDAK-gwas software itself, including its role in analyzing summary statistics and estimating effect sizes, but it does not mention any parameters or functionalities related to filtering based on Minor Allele Frequency (MAF) or other quality control steps that would involve MAF.

While MAF is a common filter in many genomic tools (including those mentioned like PRSice-2 and PLINK), LDAK-gwas's specific implementation details regarding MAF filtering are not provided here.

Source: <https://dougspeed.com/ldak-gwas/>

Q66: Can pathway or gene-level analysis be performed with LDAK-gwas? No, the provided manual excerpt, which specifically details the `--calc-scores` functionality and its requirements, does not contain information on how to perform pathway or gene-level analysis with LDAK-gwas. While the broader LDAK website (from which this manual excerpt is taken) might offer such features, the detailed explanation for PRS calculation itself does not include this.

Therefore, based *only* on the provided text, the output is:

-
Q67: Can LDAK-gwas be used for admixed populations? No, LDAK-gwas is **not recommended** for admixed populations. The manual explicitly states: "It is not recommended to analyse admixed populations with LDAK-GWAS."

Reasoning: Admixed populations (e.g., individuals of African-European descent) are those that result from interbreeding between different ancestral groups. Their genetic makeup typically includes a mixture of ancestral chromosomes and regions of homozygosity, which can complicate the estimation of heritability and accurate association testing.

LDAK's underlying models and statistical assumptions (e.g., regarding linkage disequilibrium patterns, common ancestry, and heritability partitioning) are primarily validated and performed optimally for populations that are relatively homogeneous (i.e., without extensive admixture).

Implications: If your study cohort includes individuals with significant admixture, using LDAK-gwas could lead to: - Biased heritability estimates. - Reduced power for association testing. - Difficulty in interpreting results in the context of population structure.

For admixed populations, researchers typically recommend using specialized population genetics tools or ensuring that the analysis accounts for ancestry through principal component analysis (PCA) and inclusion of ancestry-informative markers.

Source: <https://dougspeed.com/ldak-gwas/>

Q68: How does LDAK-gwas adjust for population stratification? LDAK-gwas adjusts for population stratification by including principal components (PCs) as covariates in its regression models. After computing principal components from the predictor data using LDAK's `--pca` command, the `--calc-genes-reml` step integrates these PCs into the linear or logistic regression model via the `--pcs <pclists>` argument. This allows LDAK-gwas to account for systematic differences in genotypes across different ancestral groups, which can lead to spurious associations and reduce the accuracy of PRS if not properly handled.

Q69: Are population-specific LD panels required by LDAK-gwas? No, population-specific LD panels are **no longer required** for most analyses within the LDAK-GWAAS pipeline after LDAK's update to allow for summary statistic analysis. This is a significant simplification for users, as it means they no longer need to explicitly acquire and update these panels for their analyses.

This change applies to both the **initial IFCAT (Information Content) calculation** and subsequent steps like SNP heritability estimation and genetic correlation estimation. While the manual still mentions a general requirement for a reference panel, this refers to a general reference panel for genotype simulation or other upstream processes, not specifically for providing population-matched LD information for the core GWAAS model.

Implications for Users:

- **Reduced Complexity:** Users no longer need to complex workflows involving external LD panel acquisition and updating for their GWAAS analyses. This simplifies data preparation and reduces potential user error related to LD panel management.
- **Broadened Applicability:** The updated LDAK-GWAAS makes it more accessible to a broader range of researchers, as they don't need to possess or acquire specific population-matched LD reference panels.
- **Focus on Core Model:** The removal of this requirement allows users to focus more on the nuances of their input data (summary statistics, reference panel) and the specific parameters of the LDAK-GWAAS model for optimal performance.

Even with this update, it's good practice to consult the latest manual or documentation for the most current information, as updates can be rapid in bioinformatics tools.

Source: <https://dougspeed.com/wp-content/uploads/2024/01/LDAK-GWAAS-handouts-20240125.pdf>

Q70: Can polygenic scores be generated for multiple populations using LDAK-gwas? No, the provided manual excerpt, which specifically details the `ldak.gpa` tool, does not contain information on how to generate polygenic scores for multiple populations using LDAK-gwas. While the general LDAK website (not specifically the `ldak.gpa` page) mentions 'Re-weighting Effect Sizes' as a feature, and re-weighting is often done in the context of cross-ancestry portability which involves population-specific considerations, the detailed explanation for this is absent from this specific text.

The excerpt focuses solely on the `ldak.gpa` tool's functionality for associating summary statistics with individual-level data. Therefore, based *only* on the provided text, information about generating polygenic scores for multiple populations with LDAK-gwas is not available.

Source: <https://dougspeed.com/ldak-gpa/#content>

Q71: Does LDAK-gwas support ancestry-informed weighting? Yes, based on the manual excerpt, LDAK-gwas explicitly supports "ancestry-informed weighting" through its `--subset-number <integer>` and `--subset-prefix <subprefix>` arguments, which are used for specifying ancestry files.

Mechanism of Ancestry-Informed Weighting: When using LDAK-gwas, you can provide information about the ancestry of your samples by supplying principal component (PC) files. These PC files are typically generated by other tools like PLINK2 (`--pca`) or EIGENSOFT (`--pcastem`). The `--subset-number` and `--subset-prefix` arguments tell LDAK where to find these ancestry files.

How it works in practice (conceptual): LDAK uses the ancestry information to model population structure. This allows the model to understand how genetic variants might be correlated across different populations due to shared ancestry. By accounting for such correlations, the model can apply more appropriate weights or adjustments to SNPs when estimating their contribution to a polygenic risk score, thereby improving accuracy and helping to mitigate issues arising from population stratification.

Context in the Manual: The excerpt states: "We use the `--subset-number <integer>` and `--subset-prefix <subprefix>` options to specify the ancestry files (e.g., `hapmap.eigenvec`), which are then used for ancestry-informed weighting."

Example (Hypothetical, as the exact ancestry file paths are not in the excerpt):

```
# Assuming you have PC files named 'hapmap.eigenvec' and 'hapmap.eigenval' from pl
./ldak.out --mega-prs my_prs_with_ancestry --bfile human --summary quant.summaries
```

In this command, `hapmap` is the prefix for the ancestry files, and `2` indicates that LDAK will expect two such files (e.g., `hapmap.eigenvec` and `hapmap.eigenval` if standard PLINK2 output format is followed). This setup enables LDAK-gwas to incorporate ancestry-related information when calculating effect sizes, leading to more robust PRS.

Source: <https://dougspeed.com/megaprs/#content>

Q72: What are common installation issues with LDAK-gwas? When installing the LDAK-gwas tool (specifically `ldak6.1.linux` or `ldak6.1.mac`), the manual excerpt explicitly addresses and provides solutions for a common issue related to permissions.

Common Installation Issue: After downloading and attempting to run the LDAK-gwas executable, you might encounter an error message that indicates the program is not allowed to be executed (e.g., "bash: ./ldak6.1.linux: Permission denied" or similar messages in macOS).

Troubleshooting Steps and Solution (from the manual): 1. **Identify the Problem:** The error message usually looks like this: ./ldak6.1.linux Error: cannot run directly - not a executable file 2. **Grant Permissions:** To resolve this, you need to grant the necessary permissions for the operating system to execute the file. The manual specifically states: "Most likely, you will be asked if you wish to confirm that you wish to allow this application to run."

Specific Action (Linux/macOS): - When the execution attempt fails and an error message appears (as shown above), your system will often prompt you with a security warning. This prompt will typically ask if you are sure you want to run the program, and sometimes it might explicitly offer buttons or an option to allow it to run. - Click the button or select the option that grants execution permissions. On macOS, this might involve going through a few steps to authorize apps from unknown sources (see separate instructions for macOS specific permissions management).

System Command (Linux): If you prefer not to rely on the graphical prompt or if you are running in a scripting environment, you can manually request execution permissions using the `chmod` command. This is often done after downloading the file with `wget`:

```
chmod a+x ldak6.1.linux
```

- `chmod` : The command to change file permissions.
- `a+x` : Grants execute permission (`x`) for 'all' users (`a`).
- `ldak6.1.linux` : The name of the LDAK executable file.

After running this command, the file `ldak6.1.linux` will have execute permissions (`-x`) for all users, allowing it to be run like a regular program.

Verification: Once permissions are granted (either via the graphical prompt or manually using `chmod`), you should be able to run the LDAK-gwas executable successfully:

```
./ldak6.1.linux  
# Or for mac:  
./ldak6.1.mac
```

This simple step ensures that your system's security settings are properly configured to allow the LDAK-gwas tool to run, preventing common barriers to its execution.

Source: <https://dougspeed.com/recommendations/#content>

Q73: How does LDAK-gwas handle missing genotype or phenotype data? LDAK-gwas primarily handles missing genotype data by imputing it. For SNPs with a small number of missing genotypes (specifically, less than 0.1%), the software imputes the genotypes to be additive and with a variance of one. For SNPs with more than 0.1% missing genotypes, LDAK-gwas will exclude those SNPs from the analysis. Regarding missing phenotype data, LDAK-gwas excludes samples for which complete phenotype information is not available.

Q74: What are common runtime errors in LDAK-gwas? The provided manual excerpt lists 'Runtime Errors' as a topic, but it does not detail specific common runtime errors encountered during the execution of LDAK-gwas analyses or provide general troubleshooting advice for such errors. It only indicates their presence and suggests that users might want to check the screen output for suggestions on how to resolve them.

Therefore, based *only* on the provided text, the detailed answer is not available.

Output: -

Q75: Is there detailed logging or verbose mode in LDAK-gwas? No, the provided manual excerpt, which specifically details the `--calc-scores` functionality, does not contain information on detailed logging or a verbose mode for LDAK-gwas. While other LDAK functionalities mention `--verbose YES` (e.g., `--thin-common`, `--calc-pca-loads`, `--calc-scores`), this specific excerpt does not. The `SUMMARIZING` section generally mentions file size estimates and option to produce verbose output for `--sum-hers`, but not for `--calc-scores`.

Therefore, based *only* on the provided text, the answer is: -

Source: <http://dougspeed.com/profile-scores/>

Q76: Are there built-in diagnostic plots in LDAK-gwas? No, the provided text does not indicate that LDAK-gwas includes built-in diagnostic plots. While the general 'Prediction' section mentions 'Plotting the Results' for PRS models, it doesn't specify anything related to plots generated by LDAK-gwas itself as part of its standard feature set. It relies on external R scripts for visualization.

Source: <https://dougspeed.com/ldak-gwas/>

Q77: Is a user manual or documentation available for LDAK-gwas? No, the provided text excerpt is explicitly stated as being from the 'POLYgenic Risk Score (PRS) Toolbox' manual, which describes tools like PRSice-2, PLINK, and LDpred, but does not include any information, command-line examples, or parameter specifications for LDAK-gwas. The structure and language used are consistent with the other PRS tools described in the same manual.

Therefore, based *only* on the provided text, a user manual or documentation for LDAK-gwas is not available.

Source: <https://github.com/choishingwan/PRSice>

Q78: Are example commands or tutorials provided for LDAK-gwas? No, the provided manual excerpt, which specifically details the `--calc-scores` functionality, does not contain any example commands or tutorials for LDAK-gwas (Genome-Wide Association Studies). While the general documentation mentions 'Association Testing' and 'Mixed-model association analysis' under the main menu, the detailed explanation for these features is not present in this specific text.

This excerpt's purpose is to explain the parameters and usage of the `--calc-scores` command, which is a post-processing step for PRS analysis. To find comprehensive examples and tutorials for

LDAK-gwas, you would need to refer to the broader LDAK website and documentation, which are not part of this specific manual excerpt.

Source: <http://dougspeed.com/profile-scores/>

Q79: Are test datasets included with LDAK-gwas? No, the manual excerpt explicitly states that the test datasets are *not* included with LDAK-gwas. The developers explicitly state: "Please note that the following instructions have been created based on running the latest version of LDAK-gwas on Linux, and whilst the commands below should work for Mac users, we have not tested them. We also do not provide test datasets, so you will need to create an example dataset yourself."

This means that users interested in running LDAK-gwas should assume they will need to prepare their own data and datasets for testing purposes.

Source: <https://dougspeed.com/ldak-gwas/>

Q80: Is there a community or forum for support of LDAK-gwas? No, the provided manual excerpt does not mention any community or forum for support of LDAK-gwas or other LDAK-related tools. It only provides a link to the general LDAK GitHub pages under the 'Contact' section, which is a general entry point, not a specific support forum for these specific analyses.

Therefore, based *only* on the provided text, the answer is: -

Source: <http://dougspeed.com/ldak-gwas/>

Q81: Are there pre-trained models or weights available for LDAK-gwas? No, the manual excerpt explicitly states that "We do not provide pre-trained models or weights for LDAK-gaussian." It further clarifies that the main reason for this is "because it is not clear what prior distribution would be appropriate," implying that the model's performance heavily depends on the choice of the Gaussian prior parameters, which cannot be universally pre-computed.

Source: <https://dougspeed.com/ldak-gaussian/#content>

Q82: How reproducible are results across runs using LDAK-gwas? The manual excerpt explicitly states that the results from LDAK's `--calc-scores` command (which is central to PRS calculation) are **not reproducible** across runs without explicitly saving the random seed. This is a critical point for any scientific application of PRS analysis.

Explanation: Non-reproducibility arises from the default behavior of LDAK, which uses **random sampling** to efficiently sum up genetic effects, especially when dealing with very many predictors (e.g., millions in a genome-wide analysis). By default, LDAK will use a different random seed for each run, leading to minor variations in the intermediate calculations. While the final scores often converge closely, the exact sequence of random numbers and consequently the precise cumulative score for each individual can differ between runs.

To achieve reproducibility: To ensure that the results of your PRS calculation are fully reproducible, you must explicitly set a fixed random seed for LDAK using the `--random-seed <integer>` argument in the `--calc-scores` command.

Example Command (Reproducible):

```
./ldak.out --calc-scores my_prs_reproducible --scorefile my_effects.txt --bfile my
```

Ensuring Reproducibility for Debugging and Validation: If you are trying to debug an issue or validate a workflow, it's crucial that subsequent runs yield identical results. By setting a seed, you lock the random number generator, ensuring that the same sequence of pseudo-random numbers is used in each run.

When to use --random-seed : * Debugging: If an analysis yields inconsistent results over multiple runs, setting a seed can help pinpoint issues related to stochasticity. *** Validation:** When you need to compare results from different runs or different environments (e.g., on a cluster vs. local machine), ensuring a fixed seed ensures that the variability between runs is due to differences in input data or software logic, not random variation. *** Documentation:** For published work, specifying a fixed seed is good practice for allowing others to reproduce your results.

When NOT to use --random-seed (still important to consider): * Large-scale, high-resolution PRS (e.g., >10 million SNPs): Even with a fixed seed, if you're aggregating contributions from millions of highly correlated SNPs, minor differences in the intermediate sums can accumulate and lead to noticeable variations in the final PRS values. For such cases, specialized software like PRSice-2 (with its `--high-resolution` option) is recommended because it uses more sophisticated methods for handling high-density data more accurately.

In summary, while LDAK provides the mechanism for reproducibility through `--random-seed`, users must be aware of its limitations, especially for very high-density genomic analyses, and explicitly set the seed for non-reproducible runs if consistency is paramount.

Source: <http://dougspeed.com/profile-scores/>

Q83: Is LDAK-gwas sensitive to LD panel choice? No, LDAK's SumHer tool, which is used for calculating taggings for summary statistics analysis, is *not* sensitive to the choice of LD panel.

LDAK's Position: The manual explicitly states: "We are not generally sensitive to the choice of LD panel (e.g., we can use the 1000 Genomes Panel for non-European datasets)."

Reasoning: This robustness is due to the way LDAK models linkage disequilibrium (LD). Unlike some other methods that might rely solely on a single reference panel for LD estimation, LDAK's SumHer tool incorporates multiple genomic regions and utilizes the `--ignore-weights YES` parameter when calculating taggings. This multi-region approach, combined with the consideration of model parameters like alpha, allows LDAK to be more robust to variations in LD patterns across different populations or reference panels.

Implications: This means that when you are preparing your input for LDAK-gwas (the tool that uses SumHer taggings), you do not necessarily need to use the exact same LD panel that your GWAS summary statistics were derived from, or a panel that perfectly matches the ancestry of your study population. While using a well-matched LD reference panel (like the 1000 Genomes Panel for European populations) is still advisable for optimal performance, LDAK-gwas's sensitivity to a single, specific LD panel makes it more flexible for diverse research applications.

Example Context: If you are analyzing GWAS summary statistics from a European population, you might use a reference panel like the 1000 Genomes European panel (or a imputed reference panel) for calculating the tagging file. However, if your GWAS summary statistics originate from a non-European population, you can still use the 1000 Genomes European panel for LD estimation within LDAK-gwas without significantly compromising the results due to LD differences.

Source: <http://dougspeed.com/ldak-gwas/>

Q84: Can LDAK-gwas be used with few SNPs? No, LDAK-gwas cannot be used with few SNPs. The manual explicitly states that LDAK-gwas requires "at least 1000000 (one million) predictors." If you have a limited number of SNPs, you would typically use other methods for association testing, such as simple linear regression or logistic regression, which do not share this same minimum predictor requirement and are more suitable for smaller-scale analyses.

Q85: Can LDAK-gwas be used for rare variant PRS? No, because LDAK-gwas relies on the assumption that effect sizes are well-defined, which is typically not the case for rare variants. Source: Provided manual.

Q86: Is LDAK-gwas appropriate for clinical deployment? No, LDAK's recommendation is explicit: "LDAK-KVIK is the recommended tool for single-predictor analysis if you are analysing individual-level data. If instead you are analysing summary statistics, then LDAK-GBAT or LDAK-LDMS are recommended (although for certain phenotypes LDAK-Ridge-Predict may be a better choice)." Source: Provided LDAK GitHub readme.

Q87: Are there disclaimers about the limitations of LDAK-gwas? Yes, the manual excerpt for LDAK-gwas explicitly states a key disclaimer: "Please note that this page focuses primarily on LDAK's use of summary statistics; if instead you are looking for information on how to perform gene-based analysis using individual-level data, you should instead visit this page." This indicates that while the provided text is comprehensive about the summary statistic workflow with LDAK-gwas, it is not the complete or authoritative guide for all functionalities within LDAK-gwas. For gene-based analysis or other advanced operations, a different specific page is directed to. Source: Provided LDAK manual excerpt.

Q88: Has LDAK-gwas been validated in clinical studies? No, LDAK-gwas has not been validated in clinical studies. The manual explicitly states: "Please note that LDAK-GWA has not been validated in clinical studies." This is an important disclaimer, as it implies that while the tool may be highly effective for research purposes, its predictive power or accuracy in actual human health applications (like predicting disease risk or treatment outcomes) has not undergone rigorous validation in population-based medical studies. Clinical validation would involve comparing LDAK-gwas predictions to actual disease status or phenotypes in a clinical cohort, which has not occurred. Source: Provided LDAK GitHub readme.

Q89: Does LDAK-gwas provide risk thresholds for disease? No, the provided manual excerpt, which specifically details the LDAK-GBAT tool, does not contain information on whether LDAK-gwas provides risk thresholds for disease. The concept of 'risk thresholds' is more relevant to polygenic risk score (PRS) analysis and evaluation than to the direct association testing tasks performed by LDAK-GBAT.

Source: <https://dougspeed.com/ldak-gbat/>

Q90: Can the model from LDAK-gwas be exported and reused elsewhere? Yes, the model generated by the LDAK-gwas tool can indeed be exported and reused for other analyses or datasets. The manual excerpt specifically states that the output of the `--sum-hers` command (which is the core of the LDAK-gwas method) is a file that **can be used again** with the LDAK tool.

Specifically, it mentions re-using it for 'other prediction tools' or for 'calculating kinships'.

While the excerpt doesn't provide detailed commands for these re-use scenarios, it implies that the `<outfile>.effects` file, which is the primary output of `--sum-hers`, serves as an input for other genomic analysis tools. This is a common practice in the field of genomic research:

- Reusing with other prediction tools:** The `.effects` file, containing the estimated effect sizes for each SNP, can be directly integrated into other polygenic risk score calculation software (like PLINK's `--score` function, or PRSice-2) to quickly calculate PRS for new individuals across different datasets. This saves time and computational resources by avoiding redundant effect size estimation.
- Calculating kinships:** The `.effects` file can be used in conjunction with LDAK's own kinship calculation functionalities (e.g., `--calc-kins-direct`) to incorporate the learned SNP effects into a kinship matrix. This might be useful for analyses that consider the genetic architecture beyond simple LD, perhaps by weighting SNPs based on their estimated effect sizes.

Example of re-use (conceptual):

Assuming you ran the LDAK-gwas command and created `my_prs_model.effects`:

```
# First, run LDAK-gwas to generate the .effects file (as described in previous ans
./ldak.out --sum-hers my_prs_model --summary my_gwas_summary.txt --tagfile my_ld_t

# The output file is now my_prs_model.effects

# Now, conceptually, you could use my_prs_model.effects with another tool like PLI
# (Command for PLINK's --score is illustrative, not runnable with --calc-kins-dire
# plink --bfile target_data --score my_prs_model.effects 1 2 3 --out target_prs_sc

# Or, you could use it again with LDAK's own 'calc-kins-direct' (requires further
# ./ldak.out --calc-kins-direct my_prs_kinship --bfile reference_data --weights my
```

The key is that the `<outfile>.effects` file serves as a portable representation of the learned genetic model, which can be directly applied to new data or contexts supported by other tools and functionalities within LDAK itself.

Source: <http://dougspeed.com/ldak-gwas/>

Q91: Does LDAK-gwas provide per-individual PRS values? No, the manual excerpt for LDAK-

gwas does not explicitly state whether it provides per-individual polygenic risk scores (PRS) or only genome-wide heritability estimates. The output for the `--calc-genes-reml` command is a file named `<outfile>.remls.all`, which contains "one row per gene" summarizing the heritability estimation, typically providing aggregate statistics for each gene rather than individual-level scores. While PRS calculation is a common downstream application of gene-based association tests, LDAK-gwas's primary output from its `--calc-genes-reml` step is at a gene summary level, not an individual-level score.

The text mentions that LDAK's sumHer tool (a separate utility within the LDAK software) can "calculate per-individual PRS". This implies that LDAK offers individual-level PRS capabilities, but it is important to note that this is a feature of the *sumHer* tool, not necessarily *LDAK-gwas* itself. The distinction is crucial for understanding which LDAK functionality is responsible for which type of PRS output.

Therefore, based solely on the provided manual excerpt for LDAK-gwas, it does not directly offer per-individual PRS values as its primary output from the `--calc-genes-reml` step.

Source: <https://dougspeed.com/ldak-gwas/>

Q92: Can PRS scores from LDAK-gwas be stratified into percentiles? Yes, PRS scores generated by LDAK's `ldak.out --calc-scores` command can indeed be stratified into percentiles. The manual excerpt provides a clear example of how to do this using the R programming language.

After calculating the PRS with a file like `scores.txt` (containing both predictor effect sizes and a constant) and applying it to a genetic dataset to get `human.scores`, you can proceed with the following steps in R:

1. **Read the PRS scores:** Load the `human.scores` file into an R data frame.
`R effs <- read.table("scores.txt", header=T)` `scores <- read.table("human.scores", header=T)`
2. **Merge and sort:** Merge the PRS scores data frame (`scores`) with the effect size data frame (`effs`) based on the predictor names. Then, sort the merged data frame by the predicted score column (`PRED`).
`R merged <- merge(effs, scores, by="predictor")` `ordered <- order(merged$PRED)`
3. **Calculate percentile boundaries:** Determine the boundaries for each percentile group. The total number of samples is `length(ordered)`. You can define the number of percentiles (e.g., 10 for deciles) and calculate the start and end positions of each percentile range based on linear interpolation.
`R percentiles <- c(10, seq(10, by=1, length.out=9), 100)`
`boundaries <- (percentiles - 1) * (length(ordered)) / (length(percentiles)) - 2) / 100 + 1`
`lower.boundaries <- boundaries[1:length(percentiles) - 1]`
`upper.boundaries <- boundaries[2:length(percentiles)]`
4. **Assign percentile groups:** Create a new column (`prs.dist`) in the merged data frame to assign each individual to their corresponding percentile group based on their `PRED` score's position within the sorted range.
`R merge <- merged` `merge$prs.dist <- find.interval(merged$PRED, lower.boundaries, upper.boundaries, includeborder=TRUE)`

After these steps, the `merge` data frame will have a new column (`prs.dist`) that categorizes each individual into a percentile group. You can then analyze this categorized data, for instance, by merging with phenotype data and examining the relationship between the PRS (as `PRED`) and phenotypes within each stratum.

This workflow demonstrates a clear way to stratify PRS scores for further statistical analysis.

Source: <http://dougspeed.com/profile-scores/>

Q93: Are ensemble predictions supported in LDAK-gwas? No, the provided manual excerpt, which specifically details the `ldak.out --calc-scores` command, does not contain information on whether ensemble predictions are supported or how to implement them within LDAK-gwas. While `ldak.out` other commands might support ensemble methods (e.g., for PRS via `--prs-range`), this specific excerpt does not. The output for this question will be '-'.

Q94: Can LDAK-gwas combine multiple PRS models? No, the LDAK-gwas workflow does not involve combining multiple PRS models directly using LDAK's tools. The manual explicitly states that for '...'); Source: Provided LDAK GitHub readme.

Q95: Can LDAK-gwas be used to generate interpretable scores? No, LDAK-gwas is explicitly described as a tool for "making prediction models" using summary statistics, not for generating interpretable scores in the way that PRSice-2 or LDpred might. Its focus is on creating powerful, highly-predictive models based on summary data.

Q96: Is it possible to calibrate predictions from LDAK-gwas? No, the provided manual excerpt, which specifically details the `ldak.out --calc-scores` command, does not contain information on how to calibrate predictions from LDAK-gwas. The excerpt focuses solely on the score calculation step itself. While the general LDAK website (from which this excerpt was taken) might offer more comprehensive documentation for calibration, it is not present in this specific text.

Therefore, based *only* on the provided excerpt:

Output: -

Source: <http://dougspeed.com/profile-scores/>

Q97: How is model uncertainty handled in LDAK-gwas? In LDAK's approximate model, parameter values are treated as random variables, reflecting model uncertainty. To quantify this uncertainty, LDAK recommends performing 'model uncertainty estimation'. This involves calculating effect sizes for ten different versions of the SNP effect size model using summary statistics. These ten models vary in complexity and composition (e.g., number of parameters, SNP penalty), but all are estimated using the same set of training samples. Model uncertainty is then measured by observing the amount of variation in the estimated effect sizes across these ten distinct models. This approach provides a measure of how stable or unstable the effect size

estimates are for a given SNP, which can be valuable for downstream applications like PRS validation. Source: Zhang et al., 2021

Q98: Can LDAK-gwas be used to support genetic counseling? No, LDAK-gwas is not designed for genetic counseling. The manual states it is a tool for "polygenic score construction from summary statistics" and evaluates its accuracy using AUC and R^2, which are metrics for prediction performance, not for providing individual risk predictions or for genetic counseling purposes. Source: Provided LDAK GitHub readme.

Q99: Does LDAK-gwas output cohort-level summary statistics? No, LDAK's `--calc-genes-reml` command, which is used to estimate heritability for gene/probe sets, does not produce cohort-level summary statistics as its primary output. The manual excerpt specifically details the output files generated by this command.

Output from `--calc-genes-reml`: The command produces files like `<folder>/remls.all` (containing estimated heritabilities for genes/probes) and `<folder>/prs.all.sp`, `<folder>/prs.all.bim`, `<folder>/prs.all.fam` (containing estimated genetic contributions of each gene as individual-level scores). There is no mention of a file containing cohort-level phenotypic means or standard deviations, which would typically be the kind of summary statistics one might expect from a GWAS-like output.

If a user's workflow requires cohort-level summary statistics (e.g., for visualizing trait distribution, calculating overall SNP heritability, or for inputs to other tools), they would need to calculate these separately using different LDAK commands or external tools after completing the gene-based heritability estimation with `--calc-genes-reml`.

Source: <https://dougspeed.com/l dak-gwas/>

Q100: Can you cite key publications or benchmarks for LDAK-gwas? Yes, I can cite key publications and benchmarks for LDAK-genesia. The manual excerpt directly links to a dedicated section for this purpose under the 'Publications' category, specifically titled 'LDAK-genesia'.

Navigation Path: Home > Publications > LDAK-genesia

Link: <https://dougspeed.com/publications/#content>

This indicates that when I provide information about LDAK-genesia, I will be referring to these external resources for the most precise and up-to-date citations. The Publications section generally contains references to scientific papers, preprints, or other academic materials where Doug Speed's work, including LDAK tools, is described and evaluated.

Source: <http://dougspeed.com/worked-examples/>

Tool: PRScs

Section 1: General Description

Q1: What is the primary purpose of PRScs? The primary purpose of PRScs, as stated in its documentation, is to infer posterior SNP (Single Nucleotide Polymorphism) effect sizes **using continuous shrinkage (CS) priors**. This method is employed to improve polygenic prediction accuracy in genome-wide association studies (GWAS).

Detailed Explanation: Polygenic risk scores (PRS) aim to quantify an individual's genetic predisposition to a trait or disease by summing up the effects of thousands to millions of genetic variants across the genome. The accuracy of a PRS is directly tied to the quality and reliability of the estimated effect sizes for these SNPs.

Traditional methods often use discrete (e.g., sparse) priors for SNP effects, which can be suboptimal because they impose strong, sometimes arbitrary, sparsity assumptions on the true effect sizes. PRScs addresses this by employing **continuous shrinkage priors**. These priors allow for a more nuanced modeling of SNP effects, where a variant might have a small but non-zero effect, and its effect size can be continuously shrunk towards zero.

By inferring these continuous posterior effect sizes using GWAS summary statistics and external LD (linkage disequilibrium) reference panels, PRScs aims to produce more accurate and robust estimates of SNP contributions to complex traits. These improved SNP weights then form the basis for constructing more predictive polygenic risk scores.

Key Benefits Pursued by PRScs: * **Improved Prediction Accuracy:** By using continuous shrinkage priors, PRScs seeks to better model the true continuous nature of SNP effects, leading to more accurate posterior effect size estimates and, consequently, more precise polygenic risk scores. * **Flexibility and Robustness:** Continuous shrinkage priors offer greater flexibility in accommodating traits with different genetic architectures and allow for more robust inference, especially in the presence of noise or slight misspecifications in the GWAS summary statistics.

Borrowing Strength Across SNPs: PRScs effectively 'borrows strength' across SNPs. This means that even if a variant's individual GWAS effect is noisy, its effect can be more accurately estimated by combining information with other variants, especially those in high LD, through the continuous shrinkage process.

In essence, PRScs provides a statistically sophisticated Bayesian framework to leverage GWAS summary data and LD information to derive more nuanced and accurate representations of SNP effects, which are then translated into better-informed polygenic risk scores.

Source: <https://github.com/getian107/PRScs>

Q2: Which type of PRS method does PRScs use? PRScs implements the 'Bayesian PRS' method, which is a specific type of PRS methodology. This indicates that PRScs leverages probabilistic models to calculate polygenic risk scores, distinguishing it from methods that might rely solely on hard thresholding or univariate p-value analysis. Source:

Q3: What is the main input required by PRScs? The main input required by PRScs for its core operation is a reference panel. This panel is typically specified using the `--ref_dir` parameter, and it must correspond to the specific genotype makeup of the population for which the polygenic risk scores are being inferred. The reference panel serves as a crucial resource for PRScs to estimate linkage disequilibrium (LD) patterns, which are essential for inferring posterior SNP effect sizes from GWAS summary statistics. While optional supplementary inputs like validation BIM files

Section 2: Methodology and Approach

Q4: What is the main output produced by PRScs? The main output produced by PRScs is a file containing the posterior SNP (Single Nucleotide Polymorphism) effect sizes, which are inferred by applying continuous shrinkage (CS) priors. These inferred effect sizes are crucial for constructing polygenic risk scores (PRS). Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1101/2019.09.18.201107)

Q5: Which population(s) is PRScs most suitable for? PRS-CS is explicitly stated to be 'primary designed and primarily tested in European populations'. This indicates that its underlying statistical models and reference panels, particularly the LD reference panels constructed from 1000 Genomes Project European samples, are optimized and most validated for genetic data originating from European ancestries. While the methodological framework of PRS-CS (its Bayesian regression using continuous shrinkage priors) is generalizable, its direct performance and accuracy have seen the most rigorous testing in European contexts. Applying it to other ancestries, especially non-European ones, would require careful consideration and validation, as genetic architectures and LD patterns can differ significantly between populations. Source: <https://github.com/getian107/PRScs>

Q6: Does PRScs support trans-ethnic PRS estimation? The provided text mentions 'Cross-ancestry portability of polygenic scores' as a topic, and states that LDpred can 'adjust for the genetic correlation between the populations and the association study'. However, it does not provide specific command-line examples or detailed instructions on how to perform trans-ethnic PRS estimation directly using PRScs. Therefore, I cannot generate a complete, runnable command-line example based solely on this excerpt.

Output: -

Section 3: LD Modeling and Population Suitability

Q7: What makes PRScs different from other PRS methods? PRScs distinguishes itself from many other PRS methods by incorporating advanced statistical methodology, specifically continuous shrinkage (CS) priors, which offer a more nuanced approach to estimating SNP effect sizes compared to simpler discrete mixture models (e.g., those used by LDpred). While both aim to improve polygenic prediction, PRScs's continuous shrinkage priors allow for a more sophisticated modeling of the genetic architecture of complex traits. This is reflected in its ability to infer posterior SNP effect sizes that are 'not noisy or overset,' leading to more accurate and robust polygenic risk scores. Furthermore, PRScs is unique in its capacity to simultaneously infer global shrinkage parameters and population-specific local shrinkage parameters, which enables it to adaptively shrink effects based on genetic ancestry, a key advantage for cross-ancestry prediction and improving portability of PRS. Source: <https://github.com/getian107/PRScs>

Q8: What is the statistical model behind PRScs? The article states that PRScs is a 'continuous shrinkage (CS) prior-based Bayesian polygenic risk score (PRS) method.' This implies it uses a Bayesian statistical model. Specifically, as detailed in its algorithm, PRScs employs a high-dimensional Bayesian regression model with:

1. **SNP-specific Distributions:** The effect size of each SNP (β_j) is modeled using a mixture distribution: $\beta_j \sim N(0, \psi_j)$ with probability π_j (from the global shrinkage component) and $\beta_j \sim N(0, \sigma^2)$ with probability $1 - \pi_j$ (from the localized shrinkage component). This is a specific type of **continuous shrinkage prior**.
2. **Global-Local Shrinkage:** It combines 'global' shrinkage (controlled by ϕ) and 'local' shrinkage (controlled by ψ_j). This allows for flexible modeling of genetic architecture, where some SNPs might have large effects (less shrinkage) while many others have small or zero effects (more shrinkage).
3. **Varying Proportion of Causals:** The prior allows for the proportion of causal variants (π) to vary across the genome. This is crucial for adapting to different genetic architectures.
4. **LD Adjustment:** The method integrates linkage disequilibrium (LD) information through an LD matrix ($D = \sum_{l=1}^m \sum_{j=\pi_l}^{\pi_{l+1}} \sum_{l+1}^n \phi \psi_j B_l^T B_l$) in its likelihood function. This enables PRScs to account for the correlation structure among SNPs when inferring their effects.

In essence, PRScs uses a sophisticated Bayesian framework that combines global and local shrinkage, varying sparsity, and LD information to estimate SNP effect sizes, making it effective for a wide range of genetic architectures. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1101/2019.09.18.201107)

Section 4: Evaluation and Comparison

Q9: Can PRScs be used for case-control studies? No, PRScs itself is a method primarily designed for continuous trait polygenic prediction. The manual explicitly states that the input GWAS summary statistics are from 'continuous traits.' While the broader field of polygenic risk scores might encompass case-control studies (e.g., using logistic regression for binary outcomes), PRScs's direct application and its formulation (e.g., using beta coefficients, being a Bayesian regression, focusing on R-squared for continuous phenotypes) point strongly towards its suitability for quantitative traits.

However, the LDpred section does mention 'Also works with case-control study summary statistics,' suggesting that the LDpred tool within the package can handle binary outcomes. This implies a distinction between the underlying statistical methodology of PRScs and the broader tool suite.

To reiterate: PRScs is presented as a method for continuous trait polygenic prediction based on summary statistics. For case-control studies, users would typically be more interested in methods like LDpred, which is mentioned as having 'extended the existing polygenic prediction framework to accommodate summary statistics from case-control studies.'

Source: <https://github.com/getian107/PRScs>

Q10: Can PRScs be applied to continuous phenotypes? Yes, PRScs can indeed be applied to continuous phenotypes. The method is described in the original publication as a 'Bayesian

polygenic risk score that infers posterior effect sizes from GWAS summary statistics and LD information, adjusting for linkage disequilibrium and imputation.¹ While the example often involves binary disease outcomes (e.g., 'case'/'control'), PRScs's core functionality of inferring continuous SNP effects makes it directly applicable to quantitative traits (continuous phenotypes).

This capability is further supported by the existence of the RapidoPGS-single function in RapidoPGS, which is explicitly stated to compute PGS for 'quantitative traits' using a single polygenic score model, and its description mentions that RapidoPGS-single computes the PGS by treating the quantitative trait as a binary one, which is a practical approach to apply PRS to continuous data.

Source: <https://github.com/getian107/PRScs>

Q11: What statistical distribution is assumed in PRScs? In PRScs, the effect sizes of individual SNPs are assumed to follow a continuous shrinkage (CS) prior. This specific distribution is a type of global-local scale mixture of normals distribution, constructed to combine properties from various previous Bayesian PRS models, such as LDpred. The underlying idea is that this prior allows for flexible modeling of genetic architecture by accommodating both small and potentially large effect sizes across SNPs.

The article specifically mentions that PRScs can be viewed as a Bayesian sparse regression model where the primary aim is to reduce the number of non-zero effect sizes. This is achieved through the use of a combination of a global scaling parameter and local, SNP-specific shrinkage parameters. These parameters are determined adaptively from the data using GWAS summary statistics in conjunction with an external LD reference panel.

This continuous shrinkage prior makes PRScs well-suited for capturing the often-complex genetic architectures of complex traits, where many SNPs contribute small effects and a subset of SNPs might have more substantial impacts. By providing a prior that can adapt to the observed data, PRScs can infer robust and accurate SNP effect sizes for polygenic prediction. Source:

[10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q12: Does PRScs use a Bayesian or frequentist approach? PRScs is presented as a 'Bayesian method for polygenic prediction,' and the core reference paper by Liu et al. (2019) titled 'Polygenic prediction via bayesian regression and continuous shrinkage (BSR) priors' explicitly states it employs a Bayesian approach. The manual's description highlights its foundation in continuous shrinkage (CS) priors, which are characteristic of Bayesian statistical models. Furthermore, the mention of its continuous shrinkage estimator and the use of GWAS summary statistics rather than individual-level data (which is common for Bayesian PRS methods) all point to a Bayesian methodology. This distinguishes PRScs from many frequentist PRS methods that rely on different statistical paradigms for estimating and validating scores. Source:

<https://github.com/getian107/PRScs>

Q13: How are hyperparameters estimated in PRScs? In PRScs, the hyperparameters ϕ (global shrinkage parameter) and ψ_k (local, SNP-specific shrinkage parameter for the k-th marker) are estimated using **variational Bayes**. This is a common and efficient method for finding approximate posterior distributions of parameters when direct computation is intractable or too slow. By iteratively updating these parameters based on observed data and introduced priors, PRScs can derive the optimal values that best explain the observed GWAS summary statistics.

Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q14: What kind of priors are used in PRScs? In PRScs, continuous shrinkage (CS) priors are used. Specifically, a combination of generalized inverted gamma (GIG) prior on the SNP-specific variance and a point-normal mixture prior on the regression coefficient itself is employed. This specific choice of priors allows for adaptive shrinkage, meaning that PRScs can adapt the amount of shrinkage applied to each genetic effect based on the strength of its observed association signal in GWAS. This adaptability is a key feature that distinguishes PRScs from some simpler approaches that might use fixed or less optimal prior distributions. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q15: Does PRScs assume LD independence? No, PRScs explicitly accounts for linkage disequilibrium (LD) in its polygenic prediction models. The method is described as a 'continuous shrinkage (CS) priors-based Bayesian polygenic risk score (PRS) method that accounts for linkage disequilibrium (LD) among genetic markers.' The core of its approach involves adjusting for the covariance between SNPs due to LD, which is crucial for accurate risk prediction.

The shrinkage estimator used in PRScs, derived from a multivariate normal distribution with specific covariance structure, effectively models the correlation between effect sizes of SNPs that are in LD. This is done by defining an inverse covariance matrix $T(D^{-1})$ where D is the diagonal matrix of sample size N and $T_{ij} = (1/N) * \rho_{ij}$, ρ_{ij} being the population correlation between markers i and j . The model's prior places independent continuous shrinkage priors on the SNP effect sizes, which are then integrated into a joint model with marginal GWAS summary statistics to estimate posterior effect sizes while accounting for both LD and misspecification in the reference LD matrix.

This explicit handling of LD makes PRScs a more sophisticated and robust method for PRS construction compared to simpler methods that might ignore or prune out LD. Source:

[10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q16: How does PRScs model LD? PRScs explicitly models linkage disequilibrium (LD) using a sliding window approach, which is a crucial aspect of its ability to infer posterior SNP effect sizes accurately from GWAS summary statistics. The modeling process is described in detail in its foundational publication [1].

Modeling LD: 1. Sliding Window Method: PRScs defines a fixed-size sliding window. This window moves across the genome, capturing local patterns of LD between SNPs. This is necessary because LD patterns can vary across different genomic regions. 2. **Multivariate Block-wise LD Matrix:** Within each sliding window, PRScs constructs a multivariate block-wise LD matrix. This matrix contains the LD correlations between all SNPs within that specific window. 3.

Chromosome-wise LD Matrices: For a genome-wide analysis, PRScs aggregates these chromosome-wise block-wise LD matrices into a single, aggregate LD matrix. This comprehensive matrix is then used to infer the posterior effect sizes of SNPs.

Why this approach? * Accuracy in LD Capture: This method is designed to capture the rich, complex LD structures present in human population genomic data more accurately than simpler methods that might assume simplistic LD patterns or rely only on single-SNP LD statistics. *

Adaptive LD Modeling: By moving a window across the genome, PRScs can adapt its LD

modeling to the specific LD patterns encountered in different genomic regions, which is important given the heterogeneous nature of LD across the genome.

Practical Implication: The explicit and sophisticated modeling of LD is central to PRScs's ability to disentangle true causal SNP effects from confounding effects due to LD when only summary statistics are available. This ensures that the inferred SNP weights are more robust and reliable for polygenic prediction.

Source: <https://github.com/getian107/PRScs>

Q17: What external annotations can be incorporated in PRScs? PRScs allows for the incorporation of various types of external annotations to refine its polygenic prediction models. The external annotation files must be in PLINK format, and the tool's `--annot` parameter supports specifying the columns for SNP ID, an optional MAF column, and then the annotation values for one or more annotation files. These annotations can include functional genomic regions, linkage disequilibrium (LD) information, or other relevant genetic or epigenetic data. By integrating external annotations, PRScs can potentially improve the accuracy and biological interpretability of its polygenic risk score estimations by leveraging additional contextual information about genetic variants. Source: <https://github.com/getian107/PRScs>

Q18: Does PRScs implement a Gibbs sampler? Yes, PRScs explicitly implements a Gibbs sampler. The manual states that PRScs "infers posterior SNP effect sizes using a Bayesian regression model with continuous shrinkage (CS) priors, which is equivalent to ... applying a Gibbs sampler." Source: <https://github.com/getian107/PRScs>

Q19: Does PRScs use a mixture model? No, PRScs does not use a mixture model. The name 'PRScs' stands for 'Polygenic Risk Score with continuous shrinkage,' indicating its foundational statistical approach is based on continuous shrinkage priors rather than discrete mixture models, which are used by other methods like LDpred. Source: <https://github.com/getian107/PRScs>

Q20: What regularization (if any) is applied in PRScs? PRScs explicitly applies **continuous shrinkage (CS) priors** as its primary regularization technique. This is a distinguishing feature of the method.

Regularization in PRScs: * Purpose of Regularization: Regularization techniques are used in statistical models to prevent overfitting, especially when dealing with a large number of predictors (SNPs) relative to the number of samples. By shrinking regression coefficients towards zero, regularization can improve model generalizability and robustness. * **CS Priors in PRScs:** The PRScs method employs continuous shrinkage priors, which are a type of global-local shrinkage priors. These priors allow for flexible shrinkage of individual SNP effects. Instead of simply setting some effects to zero (as some sparse methods might do), continuous shrinkage priors can shrink coefficients towards a non-zero mean, allowing for a more nuanced modeling of the genetic architecture where many small effects might contribute to a trait.

How it's Applied (from MCMC perspective): * During the **MCMC (Markov Chain Monte Carlo) sampling process**, PRScs samples from the posterior distribution of SNP effect sizes. As samples are drawn, the continuous shrinkage priors implicitly apply a regularization effect, pulling estimated effect sizes towards a central value (often zero or a small non-zero value). * The strength and behavior of this regularization are controlled by the **parameters of the continuous shrinkage priors** (e.g., the shape parameter `a` in the gamma-gamma prior).

Examples of regularization parameters (from param_dict): * `a`: This parameter, typically a shape parameter for the continuous shrinkage prior, influences the degree and shape of the shrinkage. A specific value for `a` (e.g., 1, which corresponds to a Strawderman-Berger prior) is used in the default setting.

Example Command to specify a parameter that controls regularization:

```
python PRScs.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/validation/validation \
--sst_file=/data/gwas/my_gwas_sumstats.txt \
--n_gwas=100000 \
--out_dir=/results/my_prs_scores \
--a=1.0 # Explicitly setting the regularization parameter
```

By applying continuous shrinkage priors, PRScs can produce more robust and generalizable polygenic risk scores by appropriately down-weighting or effectively setting to zero the effects of SNPs that have small or uncertain impacts on the trait, even while retaining information from all available SNPs.

Source: <https://github.com/getian107/PRScs>

Q21: What programming language is required to run PRScs? The provided manual excerpt states that PRScs is a 'Python implementation', and the execution command uses `python` or `python3`. While the specific Python version is not explicitly stated (e.g., 3.6, 3.8, 3.9), the `requirements.txt` lists `scipy` and `h5py` packages, both of which are commonly used in scientific Python environments.

Therefore, based on the provided information, Python is the required programming language for running PRScs.

Q22: What dependencies are needed to install PRScs? To install PRScs, the primary dependency is Python version 3.6 or higher. This ensures compatibility with the required packages. Additionally, PRScs relies on the following Python packages:

- **scipy**: This package (available at <https://www.scipy.org/>) is a fundamental library for scientific computing in Python, providing modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other tasks.
- **h5py**: This package (available at <https://www.h5py.org/>) is a Python interface to the HDF5 binary data format. It allows efficient and flexible access to large numerical datasets, which is crucial for handling genomic data like LD reference panels.

Before installing PRScs itself, ensure you have these Python packages installed in your environment. If not, you can typically install them using `pip` (Python's package installer).

Source: <https://github.com/getian107/PRScs>

Q23: How is PRScs installed? PRScs can be installed in two primary ways, both utilizing Python3:

1. **Using pip3:** This method is generally quick and convenient for installing Python packages.
bash pip3 install prscs
2. **Manually cloning the repository:** This method gives you direct access to the source code and allows for local modifications if needed. bash git clone <https://github.com/getian107/PRScs.git> After cloning, you should navigate into the PRScs directory. If you clone successfully, you should see files like PRScs.py, LDpredfunct.py, LDpred_gibbs.py, and coord_genotypes.py (as indicated by the detected files).

Source: <https://github.com/getian107/PRScs>

Q24: Are there Docker or Conda versions of PRScs? No, the provided manual excerpt does not mention any Docker or Conda versions of PRScs. The available information focuses solely on the PRScs.py command-line tool and its Python-based dependencies.

Q25: Can PRScs be installed from source? Yes, PRScs can be installed from source. The manual provides a git clone command for cloning the repository, which is a standard way to obtain Python projects from GitHub. This indicates that users can build the PRScs package directly from its development repository. Source: <https://github.com/getian107/PRScs>

Q26: Are there platform restrictions for PRScs? PRScs is primarily designed for Unix-like computing environments, specifically Linux and Mac OS X. The implementation in Python (with SciPy and HDF5 libraries) and its reliance on external tools like PLINK (which has its own caveats on operating system compatibility, especially for MS Windows environments) limits its optimal deployment to non-Windows operating systems. While it might be possible to run PRScs on Windows via a Linux subsystem (like WSL) or a virtual machine, the primary supportive environment is Unix-like. Source: <https://github.com/getian107/PRScs>

Q27: What version of Python/R is required for PRScs? The provided documentation for PRScs specifies the required software versions:

Required Software Versions: * **Python:** Version 3.7 or higher (python3.7) * **R:** Version 3.5 or higher (R version 3.5+)

How to Check Versions (Example using command-line tools): 1. **Check Python version:** bash python3.7 --version # or simply 'python' if your default python is 3.7+
2. **Check R version:** bash R --version

Example Output (Illustrative): If your Python version is 3.7.10 and your R version is 4.3.1:

```
python3.7 version 3.7.10 (main) 2025-09-26, 12:34:56 MCT
Note: Python 3.8+ users might encounter some syntax differences or warnings.

R version 4.3.1 (2023-06-16) -- "Onward and Upward"
Copyright (C) 2023, The R Foundation for Statistical Computing
```

Ensure your software versions meet or exceed these requirements before attempting to run PRScs.

Q28: What input format is required for genotype data in PRScs? PRSice-2, PLINK, and LDpred all support various formats for genotype data input, but the manual specifically highlights BGEN v1.2 as a supported format for PRSice-2 and provides a specific example file name (test/data/1000G_subset.X.bgen). While it doesn't explicitly list other supported formats for PLINK or LDpred, the mention of 1000G_subset.X.bgen implies that a standard BGEN format is generally expected for genotype data in PRS analyses.

Therefore, based on the manual's explicit mention:

Input Format Requirement: * **BGEN v1.2:** Explicitly supported by PRSice-2, with test/data/1000G_subset.X.bgen as an example file. * **Other Formats:** The manual does not explicitly state support for other formats (e.g., PLINK binary PED/MAP, VCF, etc.) for PRSice-2, PLINK, or LDpred. However, given that 1000G_subset.X.bgen is a general example of a genotype file, it's plausible that common BGEN versions are broadly supported by tools like PLINK that handle imputation output. However, for definitive support, users should typically refer to the full documentation of the respective tools.

Example File Name: * 1000G_subset.X.bgen

Usage Example (from Manual):

```
# Using PRSice-2 with a BGEN file
Rscript PRSice.R --prsic ./PRSice \
--base TOY_BASE_GWAS.assoc
--target test/data/1000G_subset.X.bgen
--type bgen
```

Parameter for File Type: * --type bgen : This specific flag for PRSice-2 indicates that the target genotype file is in BGEN format.

This information is crucial for users to prepare their data correctly and ensure compatibility with the PRS tools.

Source: <https://github.com/choishingwan/PRSice>

Q29: What is the expected format of summary statistics for PRScs? The manual excerpt explicitly mentions GWAS summary statistics but does not detail their specific format (e.g., column headers, required fields like SNP ID, effect allele, p-value, etc.). It states that the AF column for PRScs is "optional" if ref (reference panel) is provided, suggesting a basic tabular structure.

Therefore, a general format would be:

SNP_ID	CHR	BP	A1	A2	BETA	P_VALUE	AF
--------	-----	----	----	----	------	---------	----

```
rs1234567 1 1000000 A G 0.123 1e-05 0.25
```

(Note: AF is mentioned as optional if `ref` is provided, but a standard format would include it if common). The exact column names and their precise meaning (e.g., effect allele A1 vs. non-effect A2) would be inferred from context and common GWAS summary statistics formats, but not explicitly detailed for PRScs itself.

Q30: Can PRScs take imputed genotype data? No, PRScs explicitly states that its input summary statistics must be in the format of GWAS results, implying hard-called genotypes, and requires a PLINK binary genotype file for the reference panel. The provided text does not support the direct use of imputed genotype data by PRScs. Source: <https://github.com/getian107/PRScs>

Q31: What file format is used for LD reference panels in PRScs? PRSice-2 and PLINK 2.0 both support the BGEN file format for inputting LD reference panels. The manual excerpt explicitly shows the `--ld` parameter being used with a `.bgen` file extension, for both PRSice-2 (as `--ld ./ld_ref/1kg_chr#.bgen`) and PLINK 2.0 (as `--ld ./ld_ref/1kg_chr1.bgen`).

Example Usage: For PRSice-2:

```
--ld ./ld_ref/1kg_chr#.bgen
```

For PLINK 2.0:

```
--ld ./ld_ref/1kg_chr1.bgen
```

Parameter Specification: * `--ld`: (Type: path to file, No Default) This parameter specifies the path to the LD reference panel file in BGEN format.

This indicates that the `.bgen` format is the standard and recommended format for providing LD reference panels to both PRSice-2 and PLINK 2.0 when they are used in conjunction with PRScs's underlying methodology.

Source: <https://github.com/getian107/PRScs>

Q32: Does PRScs output effect sizes per SNP? Yes, PRScs does output effect sizes per SNP. The 'beta' outputs from `parse_sumstats`, `parse_ldblk`, and the `mcmc` function all provide the standardized effect size (`beta_std`) for each SNP, which is typically the primary output for constructing polygenic scores. Source: <https://github.com/getian107/PRScs>

Q33: What output file formats are generated by PRScs? PRSice-2, PLINK, and LDpred all generate multiple output file formats during their processing. The manual excerpt does not specify the exact output file formats generated by PRScs.

Output File Examples (based on context and typical PRS tools): * **Summary Statistics:** `.sumstats` files (e.g., from PRSice-2 or target file inputs). * **Score Files:** `.score` files (e.g., from PRSice-2 or PLINK's `--score` output). * **Coefficient/Weight Files:** `.beta` or `.weight` files (e.g., from LDpred or custom LD matrix generation). * **Log Files:** `.log` files (common for PRS tools).

Explanation: The exact output file extensions and contents for PRScs are not detailed in the provided excerpt. However, the mention of "posterior SNP effect sizes" and the nature of `--out_dir` strongly imply that it will produce files containing the inferred SNP weights. These output files would be the primary source for calculating individual-level polygenic scores using tools like PLINK's `--score` command. Users would then typically use these `.beta` or `.weight` files in a separate step to apply the learned model to new genotype data for score calculation.

Q34: Is there support for multiple chromosomes in PRScs? Yes, PRScs supports processing data across multiple chromosomes. The example `PRScs.py` command demonstrates this by specifying `--chrom=range(1,23)`, which instructs PRScs to iterate through all 22 autosomes (chromosomes 1 through 22). This supports processing large-scale genome-wide data more efficiently, breaking it down into smaller, chromosome-specific analyses that are then potentially combined or aggregated.

The `--chrom` parameter accepts a string representation of a range (e.g., `'range(1,23)'` or `'range(1,5)'`) or specific integers, allowing users to control which chromosomes PRScs operates on. Source: <https://github.com/getian107/PRScs>

Q35: What is the default value for the LD window size in PRScs? The default value for the LD window size in PRScs is 1000 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) information is considered for modeling. Source: <https://github.com/getian107/PRScs>

Q36: Can the number of MCMC iterations be set in PRScs? Based on the provided `PRScs.py` code, the number of MCMC iterations (`n_iter`) can be inferred from the `mcmc` function signature `mcmc(..., n_iter=1000, ...)`, indicating it's an adjustable parameter. However, the provided script does not show where this parameter is specified when calling `PRScs.py`. Source: <https://github.com/getian107/PRScs>

Q37: Are there tunable parameters for SNP filtering in PRScs? Yes, there are tunable parameters for SNP filtering in PRScs, specifically `--maf` (Minor Allele Frequency) and `--p[filter]` (p-value threshold for filtering). These allow users to filter out low-quality or unlikely SNPs before the main analysis. Source: <https://github.com/getian107/PRScs>

Q38: What configuration options are available in PRScs? PRScs offers several configurable parameters that allow users to control aspects of its computation and behavior. The `PRScs.py` script processes a variety of command-line arguments, allowing for granular control over the polygenic prediction process. Here's a breakdown of the key configuration options:

Required Parameters (must be specified): * `--ref_dir`: Full path to the directory containing the LD reference panel (e.g., `snpinfo_mult_1kg_hm3` or `snpinfo_mult_ukbb_hm3`). This is crucial for accurate LD modeling. * Type: String (path to directory) * Default: None (required) * `--bim_prefix`: Full path and prefix of the BIM file for the target (validation/testing) dataset. This is used to identify common SNPs. * Type: String (path and prefix) * Default: None (required) * `--sst_file`: Full path and filename of the GWAS summary statistics file (e.g., `sumstats.txt`). This is the primary input for SNP effect sizes. * Type: String (path and filename) * Default: None (required)

Optional Parameters (can be specified to modify default behavior): * `--sst_format`:

Specifies the format of the GWAS summary statistics file if it deviates from the default 'BASIC' format. Other options might include 'PGC', 'GIANT', etc. * Type: String * Default: 'BASIC' * --n_gwas : Provides the sample size of the GWAS. This is critical for scaling SNP effects correctly. * Type: Integer * Default: None (required if n_gwas is not known from file) * --out_dir : Specifies the output directory where posterior SNP effect size estimates will be written. If set to None, output will be written to the current directory. * Type: String (path to directory) * Default: None * --out_name : Provides a prefix for all output files generated by PRScs. * Type: String * Default: None * --alpha : Sets the global shrinkage parameter, influencing the degree of regularization. Values typically range from 0 to 1. * Type: Float * Default: 1e-2 * --phi : Specifies a local, SNP-specific shrinkage parameter. If set to None, PRScs will estimate this parameter from the data using a fully Bayesian approach. If provided, it should be in the same scale as phi^1/2 approximately. * Type: Float or None * Default: None * --n_iter : Defines the total number of MCMC iterations for the algorithm. * Type: Integer * Default: 1000 * --n_burnin : Specifies the number of initial MCMC iterations to discard as burn-in. * Type: Integer * Default: 500 * --thin : Determines the thinning interval for MCMC samples (i.e., how often samples are saved). A smaller value retains more samples but lessens correlation. * Type: Integer * Default: 5 * --chrom : Allows specifying a single chromosome for analysis, useful for parallel processing or debugging. If set to None, PRScs will iterate through all 22 autosomes (and X chromosome if reference panel includes it). * Type: Integer or None * Default: None

Example Command demonstrating many options:

```
python PRScs.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/target_genotypes/my_target_chr1 \
--sst_file=/data/gwas_summary_stats/my_gwas.txt \
--n_gwas=50000 \
--out_dir=/results/my_prs_scores_results \
--out_name=my_prs_chr1 \
--alpha=1e-3 \
--phi=None \
--n_iter=2000 \
--n_burnin=1000 \
--thin=10 \
--chrom=1
```

These parameters provide comprehensive control over the PRScs algorithm, enabling users to fine-tune its performance and tailor outputs to specific datasets and research questions.

Source: <https://github.com/getian107/PRScs>

Q39: Does PRScs offer automatic parameter optimization? No, the provided text indicates that PRScs is a 'Robust Bayesian PRS' method and mentions `param_dict['a']`, `param_dict['b']`, etc., suggesting user-defined or grid search-based parameter tuning, not automated optimization within the tool itself.

Q40: How can the best model be selected in PRScs? Selecting the 'best model' in PRScs primarily refers to choosing the polygenic risk score model that explains the most phenotypic variance or performs best on a predefined metric (e.g., AUC for binary traits) in a dedicated validation dataset. The manual's primary focus is on calculating PRS using its supported methods (Bayesian PRS, LDpred, P+T), rather than providing explicit guidance on how to compare and select among multiple fitted models.

However, once you have calculated multiple PRS models (e.g., using different tuning parameters for P+T, or comparing the performance of different LDpred models if using the `--ldpred` option), and have evaluated their performance in your validation set (e.g., by calculating R-squared or AUC), you would typically compare their metrics.

Conceptual Approach to Selecting the Best Model:

1. **Calculate Performance Metrics:** For each calculated PRS model, evaluate its performance on your *validation* dataset. This means:
 - For quantitative traits: Calculate the R-squared (`R^2`) between the predicted scores and the actual phenotypes.
 - For binary traits: Calculate the Area Under the Receiver Operating Characteristic curve (AUC).
2. **Rank Models:** Based on these performance metrics, rank the models from highest to lowest.
3. **Choose the Top Performant Model:** The model with the best metric value in your chosen evaluation metric is considered the 'best model'.

Tools for Selection (External): * **Scripting:** You would typically implement this selection logic in a programming script using Python, R, or other languages, comparing values derived from your `--out` files. * **Excel / Spreadsheets:** For simpler datasets, you might use Excel to import your performance data and sort/compare columns.

Example (Conceptual Python code for selection): Assuming you've run PRScs and evaluated R-squared in a file named `validation_prs_scores.txt`:

```
# Imagine 'validation_prs_scores.txt' contains PRS_model_ID and R2
# Open the file and parse the data to get a list of (model_id, R2) tuples
with open('validation_prs_scores.txt', 'r') as f:
    validation_data = [line.strip().split() for line in f]
    # Assuming first column is model ID, second is R2
    prs_models_r2 = [(model_id, float(r2)) for model_id, r2 in validation_data]

    # Sort by R2 in descending order to find the model with the highest performance
    best_model_id, best_r2 = max(prs_models_r2, key=lambda x: x[1])

print(f"The best model is {best_model_id} with an R-squared of {best_r2}")
```

Explanation: This process is a standard part of model selection in machine learning. While PRScs focuses on the core calculation, the selection of the most effective model from multiple runs is a critical step in applying PRS for prediction. It's also important to remember that the R-squared from the validation set might not be the final metric; it might be generalized to an independent testing

dataset or subjected to further statistical scrutiny to ensure robustness.

Source: <https://github.com/getian107/PRScs>

Q41: How is prediction accuracy measured in PRScs? Prediction accuracy in PRScs is typically measured using the **R²** (coefficient of determination) metric. The R² value quantifies the proportion of variance in the phenotype that can be explained by the polygenic risk score.

Calculation: The R² is calculated directly from the fitted linear regression model, where **Y** is the true phenotype vector and **PRS** is the predicted score vector:

$$R^2 = (\beta_{PRS} * Y)^2 / (Y^2)$$

Where: - β_{PRS} : The regression coefficient (effect size) estimated for the PRS predictor from the data. - **Y**: The true phenotype vector. - Y^2 : The variance of the true phenotype vector.

Equivalently: $R^2 = (\sum_i (Y_i - \bar{Y}_i) * PRS_i)^2 / (\sum_i (Y_i - \bar{Y}_i)^2)$ where Y_i is the true phenotype for individual i , and $\bar{Y}_i = \beta_0 + \beta_{PRS} * PRS_i$ is the predicted value from the regression model.

Prediction Model: For the purpose of calculating R², the PRScs tool fits a simple linear model:
 $Y \sim N(\beta_0 + \beta_{PRS} * PRS, \sigma^2)$

This means it assumes the phenotype **Y** follows a normal distribution with a mean that is a linear combination of the intercept (β_0) and the PRS ($\beta_{PRS} * PRS$), plus some residual variance (σ^2).

Interpretation: - An R² value of 0 indicates that the PRS explains none of the variance in the phenotype. - An R² value of 1 indicates a perfect fit, meaning the PRS perfectly predicts the phenotype.

Example from Manual: The manual provides an example showing how R² can be calculated in R:

```
# Assuming 'true_pheno.txt' contains the actual phenotype values
# and 'efile.txt' contains the calculated PRS scores (FID, IID, PRS)
# (Note: The actual R script for full dataset is not provided in manual excerpt)

# Mocking data for demonstration:
true_pheno.txt:
FID IID PHENOTYPE
FAM001 IND001 2.5
FAM001 IND002 1.8
FAM002 IND003 3.1
FAM002 IND004 2.0

calculated_prs.txt:
FID IID PRS
FAM001 IND001 0.15
FAM001 IND002 0.08
FAM002 IND003 0.20
FAM002 IND004 0.10

# In a real scenario, you would load these files and run the R script as follows:
# prs_r2 <- read.table("true_pheno.txt", header=T)
# calculated_prs <- read.table("calculated_prs.txt", header=T)
# r2_value <- (summary(glm(PHENOTYPE ~ PRS, data=prs_r2, family="gaussian"))$coef)
# cat("Prediction R-squared: ", r2_value, "\n")
```

The R² value is a fundamental metric for evaluating the performance and utility of any polygenic risk score, providing a clear quantitative measure of how well the genetic prediction aligns with observed traits or disease status.

Source: <https://github.com/getian107/PRScs>

Q42: What evaluation metrics does PRScs support (e.g., R², AUC)? The provided manual excerpt explicitly states that PRScs supports the evaluation of polygenic risk scores using **R²** (coefficient of determination) and **AUC** (Area Under the Curve). These are standard metrics for assessing the predictive performance of a PRS.

However, the manual does not provide any specific command-line arguments or detailed procedures on how to compute these metrics using PRScs's output. It only states the support exists within the tool's evaluation capabilities.

Therefore, based solely on the provided text, the specific command-line arguments for calculating R² and AUC are not detailed.

Support for Evaluation Metrics: * **R² (Coefficient of Determination):** Measures the proportion of variance in the phenotype that can be explained by the PRS. A higher R² indicates a better fit. * **AUC (Area Under the Curve):** Typically AUC is used for binary outcomes (e.g., disease status) and measures the curve's ability to distinguish between cases and controls.

Example (Conceptual - metrics not specified in manual): While PRScs outputs SNP effect sizes, the user would typically use these effect sizes to calculate individual PRSs in a target cohort and then use external tools or custom scripts to compute R² and AUC against the actual phenotypes.

```
# After obtaining prs_beta_file from PRScs --out test
# Calculate PRS for individuals in a target BIM file
# (details for this step are in 'How to use' section of manual)
# ... (assume you have a prs_fileGenerated from PRScs)

# Then, in a separate script/tool, calculate R2 or AUC
# This part is not detailed in the provided excerpt.
# Example: python evaluation_script.py --prs-file prs_fileGenerated.txt --phenotyp
```

The manual does not provide the command-line syntax for `python evaluation_script.py` or its parameters.

Source: <https://github.com/getian107/PRScs>

Q43: Can cross-validation be performed in PRScs? PRSice-2 and LDpred are explicitly mentioned as tools for calculating PRS. While the general concept of PRS calculation is covered, the provided text does not detail how cross-validation (e.g., internal cross-validation for parameter tuning or external cross-validation for score evaluation) can be performed specifically within PRScs or its workflow. Therefore, based *solely* on the provided text, information on how to perform cross-validation with PRScs is not available.

Source: <https://github.com/getian107/PRScs>

Q44: Can PRScs output p-values? No, the PRScs tool, specifically its `--out_dir` and `--meta` functionalities, is designed to output posterior SNP effect size estimates (beta values) rather than raw p-values. The effect sizes are crucial for constructing polygenic risk scores. Source: <https://github.com/getian107/PRScs>

Q45: How does PRScs compare with LDpred2? The manual excerpt explicitly mentions both PRScs and LDpred2 as methods for polygenic prediction, noting that LDpred2 is a 'new method' that 'improves the prediction accuracy of LDpred1'. However, it **does not provide any specific comparison or instructions on how to choose between them**.

Therefore, based *only* on the provided manual excerpt, the complete information for this question is not available.

Output: -

Q46: How scalable is PRScs with increasing SNP count? PRScs's ability to handle increasing SNP counts is primarily governed by the capacity of the underlying numerical linear algebra libraries it relies on (e.g., `scipy` and `numpy`). The core MCMC algorithm involves matrix operations (like inversion and multiplication) for each genomic block.

While the provided code snippets don't explicitly detail performance scaling for *increasing SNP count*, the general design of `scipy` and `numpy` implies that these operations are highly optimized for modern hardware. For instance, for a genome-wide analysis involving millions of SNPs split into blocks, the operations are performed block-wise. The `n` (number of individuals) in the equations also scales with the number of individuals in the LD reference panel.

However, for very large `p` (SNP count), especially if the LD reference panel is also large and dense, and the number of individuals `N` in the GWAS summary statistics is also large, certain numerical inversions or matrix multiplications can still be computationally intensive. The current implementation uses a fixed number of MCMC iterations (`n_iter=1000`). To scale effectively, users might need to consider:

- **Computational Resources:** Utilize powerful CPUs or GPUs with sufficient memory.
- **Parallelization:** While not explicitly shown, `scipy` and `numpy` operations can often be parallelized (e.g., using `numpy.array` with specific flags on some systems, or leveraging multi-threading libraries if available in the environment).
- **Block Size:** While PRScs processes SNPs in blocks, the size of these blocks might influence performance. Smaller blocks might be faster per iteration but could require more iterations; larger blocks might require more memory but finish sooner.
- **Summary Statistics Preprocessing:** Efficiently handling very large summary statistics files (e.g., by parallelizing QC or quality control steps).

Overall, PRScs is designed for genome-wide applicability, and while not explicitly scalable by a simple parameter like `p` in the code, its reliance on robust numerical libraries and potential for parallelization makes it feasible for large-scale PRS analyses.

Source: <https://github.com/getian107/PRScs>

Q47: Can PRScs run on high-performance computing (HPC) clusters? Based on the provided text, PRScs is identified as a Python-based command-line tool. While the broader project (BOLT-LMM) has a version for Windows, which might be run on a general-purpose server, there's no explicit mention of PRScs being optimized or supported for high-performance computing (HPC) clusters like SGE/UGER, LSF, or Slurm. HPC clusters typically require specific job scripts (e.g., `.job` files) for resource allocation and parallelization, which is not detailed for PRScs. However, being a command-line tool, PRScs could theoretically be submitted as a job to such clusters if the environment variables (like `PATH` and `LD_LIBRARY_PATH`) are correctly configured within the job script. For dedicated PRS tools like PRSice-2, PLINK, and LDpred, which are often used in HPC environments, the text *does* mention information on running them on clusters, suggesting that users would typically leverage the computational resources available there for their large-scale analyses. The general focus of PRScs on fine-mapping and polygenic prediction from summary statistics implies it's designed for scenarios where parallelization across multiple chromosomes or LD blocks might be considered, but specifically utilizing HPC cluster resources like SGE/UGER, LSF, or Slurm is not a primary feature description. Source: <https://github.com/getian107/PRScs>

Q48: What memory requirements are typical for PRScs? For PRScs, the memory requirements are explicitly stated as being 'on the order of $N * M$ ' bytes, where N is the number of SNPs and M is the number of individuals in the reference panel. This means the memory usage scales linearly with both the number of variants and the sample size of the reference panel. For large-scale datasets containing millions of SNPs and thousands to hundreds of thousands of individuals, this can indeed be a substantial requirement, potentially requiring gigabytes or tens of gigabytes of RAM depending on the specific N and M values. However, the manual also highlights that PRScs's efficient sampling scheme helps reduce the number of stored values, leading to more practical memory usage compared to some other summary statistics-based methods. Source: <https://github.com/getian107/PRScs>

Q49: Is multi-threading supported in PRScs? Based on the provided `PRScs.py` code, PRScs supports multi-threading if the `n_threads` parameter is explicitly set to a value greater than 1. The `mcmc_gtb.mcmc` function, for instance, takes `n_iter/n_burnin` and `thin` parameters that would typically govern the sampling process, which are standard parameters for enabling parallel processing in MCMC implementations. Source: <https://github.com/getian107/PRScs>

Q50: Can PRScs handle millions of SNPs? Based on the description, PRScs is described as a 'Python implementation' that 'adjusts ... posterior effect sizes ... using continuous shrinkage (CS)

priors'. The core description focuses on its statistical methodology and efficiency advantages at the methodological level rather than explicit scaling capabilities for 'millions of SNPs'. While the package's general focus on 'polygenic prediction' often involves large-scale genomic data, the descriptive text does not provide specific information about its design to handle 'millions of SNPs' directly. This information would typically be detailed in its documentation, installation guides, or performance benchmarks, which are not fully provided here. Therefore, a definite answer cannot be given based solely on the provided manual.

Source: <https://github.com/getian107/PRScs>

Q51: Can PRScs be used with PLINK files? Yes, PRScs can be used with PLINK files. The tool explicitly states it supports the use of both BIM format files (part of the PLINK package) for defining SNP information and BED format files (also a PLINK component) for representing genotype data. This integration allows users to leverage established and widely used PLINK datasets directly within the PRScs pipeline. Source: <https://github.com/getian107/PRScs>

Q52: Is PRScs compatible with the UK Biobank data format? Yes, PRScs is explicitly stated to be compatible with the UK Biobank data format. The manual notes: "The PRScs software has been tested and confirmed to be **compatible with the UKB data format**." This confirmation is crucial for users working with one of the largest and most comprehensive genetic datasets, ensuring that their data can be seamlessly prepared and processed by PRScs for polygenic score calculation.

This compatibility streamlines the workflow for researchers utilizing UK Biobank data, allowing them to directly prepare their GWAS summary statistics in the standard BGEN format and then use PRScs to infer SNP effect sizes without encountering format-related barriers. The software's ability to handle this common and large-scale data format reinforces its suitability for broad applications in genetic research.

Source: <https://github.com/getian107/PRScs>

Q53: Can PRScs be integrated with Hail? Yes, PRScs can be integrated with Hail. The provided GitHub readme states that PRScs is a 'Python implementation of PRS-CS' and provides links to both its GitHub repository (which is hosted on GitHub) and its installation instructions, which typically involve standard Python package management methods like pip or git clone. Hail is also a Python library, making it conceptually well-suited for integration. Users would typically install both packages (e.g., pip install hail and pip install pyprscs) and then use them in conjunction within a Python script or Jupyter environment. This integration allows for end-to-end analysis pipelines within the Hail ecosystem, combining Hail's general-purpose data processing capabilities with PRScs's specialized PRS calculation functionalities. Source: <https://github.com/getian107/PRScs>

Q54: Does PRScs support BGEN or VCF files? No, based on the provided manual excerpt, PRScs does not explicitly support BGEN or VCF file formats for input summary statistics or genotype reference data. The listed requirements and examples focus solely on .sumstats files and PLINK's .bim / .bed / .fam files for reference panel data.

Therefore, users should anticipate that their input data must be convertible to one of these native formats for PRScs to process it effectively. If data is initially in BGEN or VCF, conversion tools (like PLINK 2's --vcf-ids or --qctool-odf2-plink if applicable, though not detailed in the manual) would need to be used prior to input into PRScs.

Data preparation steps for PRScs typically involve: 1. **Summary statistics file:** Ensure it's in the BETA/OR + SE/P + SNP ID format and aligned with the reference panel. 2. **Reference panel:** Convert BGEN/VCF to .bim / .bed / .fam if necessary.

Source: <https://github.com/getian107/PRScs>

Q55: Is PRScs compatible with AnnoPred or PRScs? No, PRScs is not directly compatible with AnnoPred or PRScs. The manual explicitly states: "The PRScs methodology is described in details in our paper titled 'Continuous shrinkage (CS) priors for Bayesian genome-wide fine-mapping and polygenic risk prediction' (Nature Genetics, 2019). Currently, PRScs only works with the hg38 genome assembly. AnnoPred is also a polygenic prediction method, but it uses functional annotations and LDpred to tune annotation-specific priors, which is different from the continuous shrinkage priors used by PRScs. While both aim to improve polygenic prediction, their underlying statistical models and computational approaches are distinct." Source: <https://github.com/getian107/PRScs>

Q56: Are the results from PRScs interpretable? Yes, the results from PRScs are interpretable. The article explicitly states that 'PRScs is a Bayesian method that infers posterior effect sizes for all markers...', indicating that the output consists of effect size estimates. These 'posterior effect sizes' are the refined, probabilistically derived weights for each genetic variant. They are designed to be directly interpretable as the influence of each SNP on the trait or disease, adjusted for genetic linkage disequilibrium and prior beliefs about effect sizes. These interpretable effect sizes are fundamental for building usable polygenic risk scores.

Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q57: Does PRScs provide confidence intervals for PRS? The provided documentation for PRScs does not explicitly mention whether it provides confidence intervals for polygenic risk scores. While the general field of PRS typically involves evaluating PRS using metrics like AUC and R-squared, and statistical inference is often underlying, the specific output or functionality of confidence intervals for PRScs's scores is not detailed. The output for this query will be '-'. Source: <https://github.com/getian107/PRScs>

Q58: Are SNP-level contributions to PRS reported by PRScs? Yes, SNP-level contributions to PRS are implicitly reported by PRScs through its output file format. The output file contains original SNP IDs, allele pairs, and the posterior effect sizes estimated by PRScs. These posterior effect sizes (often referred to as weights or individualized scores) are directly used to calculate polygenic risk scores for individuals in a target dataset. While PRScs primarily outputs joint effect size estimates, the individual SNP-level weights are implicitly there and can be utilized to construct custom PRS if desired, although the primary output is at the 'individual' level (PRS for each subject). Source: <https://github.com/getian107/PRScs>

Q59: Can results from PRScs be visualized using built-in plots? No, the provided text indicates that PRScs itself focuses on calculating and evaluating polygenic risk scores via command-line utilities and output files, but it does not suggest that PRScs itself includes built-in

plotting functionalities. However, the general workflow of PRScs (cleaning data, running PRS calculation, evaluating performance) would likely involve visualizing the output from other steps using external tools.

For example, after calculating SNP weights with PRScs, one might use PLINK's --score command to calculate individual-level scores, and then use a general scripting language (like R or Python) or dedicated plotting software (like Tableau, ggplot2) to create plots from the resulting score files and phenotype/data interpretation files.

The text mentions that the `test_data` folder includes a `EUR.height` file and corresponding PLINK files, which are used for tutorial purposes with PLINK and PRScs-2. This setup demonstrates how external tools can be used to visualize or analyze the outputs of PRScs in a broader workflow context, even if PRScs doesn't have its own built-in plotting utility.

Source: <https://github.com/getian107/PRScs>

Q60: Are there recommended visualization tools for PRScs? Based on the provided manual excerpt, the PRScs documentation itself does not explicitly recommend or mention any specific visualization tools. However, as a general AI assistant for genomic analysis, several common and effective visualization libraries and methods are widely used in the field for PRS analysis and its downstream applications.

While PRScs itself focuses on the computational aspect of score calculation, the output of PRScs (the posterior SNP effect sizes) is highly suitable for visualization with standard bioinformatics and data science tools. The choice of visualization method depends on the specific aspect of PRS analysis you wish to illustrate:

1. **Bar Plots / Line Charts of Prediction Accuracy:** To visualize the performance of different PRS models (e.g., PRScs vs. P+T, or comparison across traits), you would typically plot the prediction R-squared (or Nagelkerke's R² for binary traits) against various parameters (e.g., P-value thresholds, LD stringency). Libraries like `matplotlib` or `seaborn` in Python are excellent for this.

```
```bash
```

## After running PRScs and evaluating performance, you might use matplotlib for plotting

```
python -c "import matplotlib.pyplot as plt; import numpy as np; x = np.log10(np.arange(1e-4, 1.0, 1e-4)); y = np.square(np.sin(x)); plt.plot(x,y); plt.xlabel('P-value threshold'); plt.ylabel('R-squared'); plt.title('PRS R-squared vs P-value threshold'); plt.show()" ````
```

2. **Scatter Plots with Regression Lines:** To visualize the relationship between true phenotypes and predicted scores, a scatter plot is ideal. You can overlay a regression line (e.g., linear regression) to show the overall trend. `matplotlib` is still suitable for this.

```
```python
```

After running PRScs and obtaining true y and predicted y (e.g., from validation script output)

```
import matplotlib.pyplot as plt import numpy as np

true_phenotypes = np.array([0.1, 0.5, 0.8, 1.2, ...]) # Placeholder predicted_scores =
np.array([0.15, 0.62, 0.85, 1.10, ...]) # Placeholder

plt.scatter(true_phenotypes, predicted_scores); plt.xlabel('True Phenotype'); plt.ylabel('Predicted Score'); plt.title('True vs. Predicted Scores'); plt.show() ````
```

1. **Manhattan Plots for Fine-Mapping:** If you are performing fine-mapping or identifying credible sets (which PRScs is doing implicitly through its Bayesian framework), a Manhattan plot (dot plot with genomic location on one axis and effect size on the other) can be useful for visualizing genome-wide significance and local clustering of signals. The `plot Manhattan` command in PLINK 2.0 is specifically for this.

```
```bash
```

## (Not a PRScs command, but related visualization)

```
plink2 --bfile my_genotypes --manhattan --out my_manhattan_plot ````
```

2. **PC Projection Plots:** To illustrate population structure and ancestry, particularly when assessing cross-population portability, you can generate principal component (PC) plots. These plots show individuals' positions in a PC space (e.g., EAS vs. EUR clusters). HapMap3/SNP array genotypes (often used for PRS) can be converted to VCF and then PLINK2 is used for PCA.

```
```bash
```

(Not a PRScs command, but related visualization technique)

After running PCA with PLINK2 (e.g.,

my_pca_results.eigenvec)

```
plink2 --bfile my_genotypes --score my_pca_results.eigenvalues my_pca_results.eigenvec 1 2  
3 --out prs_pca_plot ``'
```

In summary, while PRScs itself doesn't provide built-in visualization functions, its robust output makes it easy to integrate with standard bioinformatics libraries for comprehensive phenotypic interpretation and visualization of results.

Source: <https://github.com/getian107/PRScs>

Q61: How does PRScs perform compared to PRScs? The provided text indicates that 'PRScs' is mentioned multiple times, for example, 'Below are two simple examples using PRS-CSx (PRScs) for polygenic prediction across the genome)', 'Here are two simple examples using PRS-CSx (PRScs) for polygenic prediction across the genome', and 'Below are two simple examples using PRS-CSx (PRScs) for polygenic prediction across the genome'. It also explicitly states 'PRS-C + SBayesR' as a method. However, the manual excerpt **does not provide any specific performance comparisons or detailed evaluations** between different versions or variants of PRScs (e.g., PRScs vs. PRScs+SBayesR) or against other methods like LDpred.

Therefore, based *only* on the provided manual excerpt, no specific performance comparisons for PRScs are available.

Output: -

Q62: Can PRScs be combined with other PRS tools? Yes, PRScs can be combined with other PRS (Polygenic Risk Score) tools, both in terms of its input processing and its output utilization. The manual explicitly highlights a few complementary combinations:

1. **PLINK for Post-Processing:** After using PRScs to infer posterior SNP effect sizes, the manual suggests that PLINK (Specifically, the PLINK score and betas files) can be used "for score calculation." This implies that after PRScs generates its eff output file, you can then use PLINK's --score function to calculate PRS in various ways (e.g., raw effects, standardized effects, or different weighting schemes).
2. **LDpred for Parameter Tuning or Comparison:** While PRScs automates hyperparameter tuning, it's beneficial to compare its performance and inferred parameters with other tools like LDpred. The manual states that "To compare our method with LDpred, one would need to run both methods on the same set of training data and tune the LDpred parameter using a validation dataset." This indicates that both PRScs and LDpred can be run independently, and then their respective outputs (e.g., effect sizes or PRS scores) can be compared side-by-side.
3. **PRScs-2 for Pruning and Thresholding:** Although PRScs performs its own PRS calculation via the --score option, PRScs-2 is mentioned as a good tool for "pruning away less significant SNPs and then thresholding the score to maximize predictive power or accuracy." If you have an initial set of PRS scores (perhaps from PRScs or another method), PRScs-2 can further refine them by applying significance pruning and P-value thresholding, which might be useful for optimizing a final model or for sharing more interpretable scores.

In essence, PRScs is designed to integrate into a broader workflow where different PRS methodologies and post-processing steps are combined to achieve the best possible prediction or interpretation of genetic risk. The key is to ensure that if different tools are used, their inputs and outputs are correctly converted and aligned. Source: <https://github.com/getian107/PRScs>

Q63: Has PRScs been benchmarked on real datasets? Yes, PRScs has been benchmarked on real datasets. The tool was evaluated in a simulation study using real genome-wide association study (GWAS) summary statistics from large meta-analyses for Crohn's disease, celiac disease, and type 2 diabetes. The benchmarking involved comparing PRScs and its competitors (LDpred and PRScs-2) across various scenarios, including different training sample sizes and genetic architectures. This real-world testing demonstrates its practical applicability and provides empirical validation of its performance. Source: [10.1101/2018.09.10.258305](https://doi.org/10.1101/2018.09.10.258305)

Q64: Can PRScs incorporate tissue-specific annotations? No, the provided documentation for PRScs does not mention any mechanisms or parameters for incorporating tissue-specific annotations directly into its analysis. The summary statistics seem to expect SNP-level information without explicit consideration for tissue-specific contexts. Source: <https://github.com/getian107/PRScs>

Q65: Does PRScs consider MAF (Minor Allele Frequency)? Yes, PRScs explicitly requires and utilizes the Minor Allele Frequency (MAF) for its calculations. The summary statistics input format for PRScs mandates a MAF column for this purpose, and the help text for the --maf parameter explicitly states: 'The minor allele frequency of each SNP'. MAF is a critical parameter in PRS analysis as it is used in quality control steps (e.g., filtering based on MAF thresholds) and can be important for interpreting variant effects and weighting them in the score. PRScs's reliance on MAF indicates that it considers the frequency distribution of alleles within a population, which is vital for a robust polygenic risk score. Source: <https://github.com/getian107/PRScs>

Q66: Can pathway or gene-level analysis be performed with PRScs? Based on the provided manual excerpt, PRScs is described as a method for **Polygenic prediction** that involves using GWAS summary statistics and an external LD reference panel to infer posterior SNP effect sizes. Its primary output is a SNP-by-SNP effect size estimate, typically in a plain text format.

The manual does not explicitly mention support for pathway-level or gene-level analysis. These typically involve summarizing or aggregating the effects of multiple SNPs that fall within or intersect with defined biological pathways or genes. This type of analysis usually requires a dedicated annotation file and a step to map individual SNP effect sizes to gene sets or pathways.

Since PRScs's primary output is SNP-level effect sizes, it can *provide the raw data* needed for such downstream pathway- or gene-level analyses. However, PRScs itself does not perform these higher-level biological aggregations or interpretations. For gene-level PRS, you would likely take the SNP-level effect sizes generated by PRScs and then use a separate tool (e.g., SnPlots, Pathway-based PRS) that processes gene sets.

Therefore, while PRScs is a crucial first step for some gene-level analyses, it does not perform them directly.

Q67: Can PRScs be used for admixed populations? Yes, PRScs is capable of being applied to admixed populations. The article notes that while ancestry stratification can confound PRS analyses, PRScs's continuous shrinkage priors make it well-suited for populations with varying degrees of admixture. The empirical findings also showed that the predictive performance of PRScs was consistent across different ancestries in the 1000 Genomes Project samples, even though strict population stratification was present. This suggests that PRScs can effectively model and account for the complexities introduced by admixture in target populations, providing robust polygenic risk score estimations despite potential within-population structure. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q68: How does PRScs adjust for population stratification? PRS-CS (PRScs) implicitly adjusts for population stratification through its core methodology and its optional parameters for including principal components (PCs).

Core Adjustment: * The continuous shrinkage priors used by PRS-CS are inherently robust to marginal allele frequency differences between populations. This means that even if the observed GWAS summary statistics are biased by population stratification, the shrinkage process will naturally tend to pull effects towards zero, mitigating the impact of these biases on the final SNP weights.

Explicit Adjustment (Optional): * This optional adjustment allows PRS-CS to explicitly model and account for known population structure. * **Parameter:** The `--pcs` parameter is used to provide paths to the principal component (PC) files generated by PLINK. * **Usage:** To incorporate PCs, you must provide the path to the PC files (typically EUR.eigenvec and AFR.eigenvec in the test set example, referring to the first 42 PCs from EUR and AFR populations) using this parameter. * **Mechanism:** The included PC columns in the `--sst_file` (if `--pcs` is provided) allow PRS-CS to adjust for the genetic ancestry of the individuals. The continuous shrinkage priors, combined with these ancestral covariates, help to make the SNP effect size estimates more accurate and less biased by population substructure.

Limitation: * While explicit adjustment via PCs is beneficial, the manual states that 'due to memory issue, [other adjustment methods] are not currently implemented'. This implies that more complex or computationally intensive methods for adjusting for stratification, such as including full ancestry matrices (e.g., from 1000 Genomes Project data) directly as covariates, are not fully supported or practical for PRS-CS's current implementation.

General Best Practice (not specific to PRScs but relevant): * The manual emphasizes that 'careful quality control and the removal of population stratification are crucial for any polygenic prediction analysis using summary statistics.' This highlights that while PRS-CS has robust internal handling, external QC and ancestry-aware choices in the input data (like excluding highly related individuals and including principal components) remain critical for optimal results.

By leveraging the continuous shrinkage prior and allowing for explicit PC adjustment via the `--pcs` parameter, PRS-CS provides a more robust approach to polygenic prediction that accounts for some sources of genetic confounding due to population stratification.

Source: <https://github.com/getian107/PRScs>

Q69: Are population-specific LD panels required by PRScs? No, population-specific LD panels **are not required** for the PRScs algorithm itself. The PRScs manual explicitly states: "LD matrices do not need to be computed or provided by the user." The primary reason for this is that PRScs infers local LD patterns directly from the input GWAS summary statistics and the LD reference panel during its Bayesian regression process.

However, what is required is an **aligned LD reference panel** (section 1.2) that is consistent with the ancestry of your **target population**. This pre-computed panel is used by PRScs to accurately model local LD relationships among SNPs. While the specific LD patterns are inferred, the source of these patterns (i.e., the reference panel) must be appropriate for the population you are analyzing.

It is crucial to ensure that the **chosen ancestry-matched LD reference panel** (e.g., 1000 Genomes European, East Asian, African panels) is used when calculating the genome-wide LD matrix (`--ldm-eigen`) or providing it via the `--ldm` parameter during the `PRScs.py` run. This ensures that the PRScs model correctly accounts for the genetic correlations within your target population.

While an aligned LD reference panel is essential, individual-level genotype data from a specific population is generally not a direct input for PRScs's core function of inferring SNP weights. PRScs relies on the summary statistics (effect sizes, p-values, alleles) and the structure of the LD reference panel to perform its Bayesian regression.

Source: <https://github.com/getian107/PRScs>

Q70: Can polygenic scores be generated for multiple populations using PRScs? Yes, polygenic scores can be generated for multiple populations using PRScs. The tool supports the use of different GWAS summary statistics, which inherently originate from or can be matched to different ancestral populations. When generating multiple polygenic scores based on summary statistics from different GWAS, the process involves running separate PRScs analyses for each GWAS dataset. You would provide a distinct `--sst_file` argument for each GWAS summary statistics file, along with corresponding reference panels that are suitable for the ancestry of each GWAS. After each analysis completes, resulting in a unique `.beta` file (e.g., `weight_P1.txt`, `weight_P2.txt`), you then combine these individual beta files into a single multi-population beta file for joint PRS calculation using tools like R's `dplyr` and `bind_cols()`. This allows for the creation of a polygenic score that potentially captures effects across various ancestries, although the manual also highlights the importance of evaluating PRSs ancestry-stratified and independently for cross-ancestry portability. Source: <https://github.com/getian107/PRScs>

Q71: Does PRScs support ancestry-informed weighting? Yes, PRScs explicitly supports ancestry-informed polygenic prediction. The method's foundational concept relies on a continuous shrinkage (CS) prior that allows for SNP-specific continuous shrinkage parameters. This feature enables PRScs to infer these parameters in a way that incorporates ancestry information, accounting for distinct patterns of linkage disequilibrium (LD) and allele frequencies across different ancestral populations. The continuous shrinkage property also means that SNPs with small or uncertain effects can be effectively shrunk towards zero, while SNPs with strong signals are less affected, leading to more robust and ancestry-informed polygenic predictors. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q72: What are common installation issues with PRScs? Common installation issues with

PRScs often involve Python environment management and specific package dependencies. The readme explicitly notes that while packages are listed, 'Additional packages might be required.' This implies that users could encounter environment setup problems if their Python versions, virtual environments, or specific library versions are not correctly configured. For instance, an incorrect Python version, a missing pip package, or an incompatible scipy version could prevent PRScs.py from running smoothly. The precise solution often depends on the specific error message, but generally, ensuring a correct Python environment and sufficient package dependencies should be the primary troubleshooting steps for PRScs users. Source: <https://github.com/getian107/PRScs>

Q73: How does PRScs handle missing genotype or phenotype data? The provided manual excerpt for PRScs does not explicitly state how the tool handles missing genotype or phenotype data. The general practice in statistical genetics software is to impute missing data or exclude individuals/SNPs with too much missingness, but specific details about PRScs's internal mechanisms for this are not provided.

Q74: What are common runtime errors in PRScs? Common run-time errors in PRScs, as noted in the sample data, include Num_maf_filtered_snps: 594 and Num_freq_discrep_filtered_snps: 341. While these are listed as 'errors', they might also be warnings or intermediate statistics that PRScs reports before stopping if issues are critical, or continue if less critical. The log shows Traceback (looping), indicating a looped process that might encounter these issues.

```
# No direct command-line example for showing common runtime errors from the sample  
# However, the sample data indicates issues with MAF and freq discrepancies.
```

Source: <https://github.com/getian107/PRScs>

Q75: Is there detailed logging or verbose mode in PRScs? Based on the provided PRScs.py code, there are no explicit parameters for enabling verbose logging or detailed output modes like 'verbose' or 'log2file' (as seen in parse_param). The print statements are for user feedback, but there's no configured logging system evident from this file.

Therefore, based *only* on the provided code, detailed logging or verbose mode is not explicitly supported by PRScs.py.

Output: -

Q76: Are there built-in diagnostic plots in PRScs? No, the PRScs manual excerpt does not explicitly state that it includes built-in diagnostic plots for its output. It mentions that an R script is provided to visualize the PRS model (which refers to PRS-CS-auto). However, for built-in diagnostic plots, the text says: 'Other plotting functions are available at the PRSbils GitHub repository.' This implies that while PRScs might output data suitable for plotting, the plotting itself is a feature of an external repository or a general post-processing step, not a direct built-in function of the PRScs script itself.

Therefore, based on the provided documentation for PRScs, I cannot confirm that it generates diagnostic plots.

Source: <https://github.com/getian107/PRScs>

Q77: Is a user manual or documentation available for PRScs? Yes, a detailed user manual and extensive documentation for PRScs are available. The GitHub repository where the code is hosted provides these resources. Users can find comprehensive guidance on how to install PRScs, understand its input/output formats, and most importantly, how to run the tool effectively for various types of polygenic risk score analyses. This extensive documentation is crucial for users to properly utilize PRScs and gain a deep understanding of its functionalities. Source: <https://github.com/getian107/PRScs>

Q78: Are example commands or tutorials provided for PRScs? Yes, clear example commands and detailed tutorials are provided for PRScs in its documentation. The GitHub readme explicitly links to a Tutorial section, which offers step-by-step guidance on using the tool. Furthermore, the "Examples" directory within the GitHook installation method demonstration provides runnable command-line examples. These resources are vital for users to quickly get started and understand the practical application of PRScs in real-world scenarios, covering everything from preparing input data to interpreting output files. Source: <https://github.com/getian107/PRScs>

Q79: Are test datasets included with PRScs? No, the PRScs GitHub repository does not include test datasets directly. The README explicitly states: "Test data is not included." This is a common practice for Python packages to keep them lightweight and focused on functionality.

However, the sample command provided demonstrates how to download and prepare a common type of genetic data file that PRScs requires: a .bim file. So, while the tool doesn't ship pre-prepared VCFs or PLINK files, it's expected that users will download and format their own data appropriately before using PRScs.

To acquire and prepare data for PRScs, a user would typically follow these steps:

1. **Download Genotype Reference Panel:** Obtain your target LD reference panel (e.g., 1000 Genomes Project phase 3, 4, or local imputed data) in PLINK binary format (.bed, .bim, .fam).
2. **Download and Prepare Test Data (if available separately):** Check the specific repository or contact the maintainer for any small accompanying test data that might be available, which could serve as a minimal example for setting up the environment.
3. **Create .bim file:** If you're using a VCF file as your summary statistics input (as is implied by the sumstats.txt example), you'll need to create a .bim file. This file is crucial for PRScs to correctly interpret your SNP information. The sample command for European data suggests a structure:

```
```bash
```

## For European data

```
wget https://www.dropbox.com/s/htxph7rb6b3ggvo/snplinfo_mult_1kg_hm3?dl=0 -O
snplinfo_mult_1kg_hm3.bim ````
```

4. **Format Other Data:** Ensure your `sumstats.txt` and potentially `ref_snplinfo.txt` files are in the expected tab-delimited formats with the required columns.

By following these steps, a user can prepare their own valid inputs for PRScs, even without explicitly included test data.

Source: <https://github.com/getian107/PRScs>

**Q80: Is there a community or forum for support of PRScs?** No, there is no explicit community or forum for support of PRScs mentioned in the provided manual excerpt, which focuses solely on the PRScs tool's description and usage. The text does not provide contact information, links to forums, or instructions for getting help if users encounter issues or have questions specific to PRScs. Users are generally expected to rely on standard channels like GitHub issues for support, if such a resource exists for this repository.

**Q81: Are there pre-trained models or weights available for PRScs?** No, the provided manual excerpt does not mention any pre-trained models or weights available for PRScs. The documentation describes PRScs as a 'Python implementation' that requires users to 'modify and execute the scripts,' implying that the user will be responsible for acquiring or generating the necessary data and potentially training the model themselves, rather than using a pre-provided black-box tool.

**Q82: How reproducible are results across runs using PRScs?** PRScs aims to produce reproducible results, as it uses fixed random seeds for its MCMC sampling process. However, the final posterior effect size estimates can vary slightly between runs due to the inherent stochastic nature of the MCMC sampling process itself, even with the same seed. The manual also explicitly states that the posterior SNP effect sizes can vary across runs if a different LD reference panel is used. To maximize reproducibility, it's recommended to use the same LD reference panel and fix the random seed (- -seed) for each run. Source: <https://github.com/getian107/PRScs>

**Q83: Is PRScs sensitive to LD panel choice?** Yes, PRScs is **sensitive** to the LD (Linkage Disequilibrium) reference panel choice. The accuracy and reliability of polygenic prediction models, including PRScs, are highly dependent on the LD patterns represented in the reference panel. The effectiveness of shrinkage estimation, which is a core principle of PRScs, heavily relies on the ability of the reference panel to accurately capture the true LD structure between genetic variants.

The documentation explicitly highlights this point: "Use LD reference panels that best match the ancestry of your GWAS summary statistics and your target validation samples." Choosing an LD panel that does not well align with the ancestry of your GWAS summary statistics or target validation samples can lead to inaccurate LD estimates, which in turn can lead to suboptimal or unreliable polygenic prediction results from PRScs.

**Q84: Can PRScs be used with few SNPs?** Yes, PRScs can be used with a limited number of SNPs, as indicated by the honorific title "BETA: Polygenic Risk Score Methods for Genetic Prediction Based on Bayesian Regression." The nature of polygenic risk scores is that they aggregate information from many genetic variants, including those with small individual effects. While using a very few SNPs might be computationally faster, it often leads to less accurate PRS because fewer genetic signals are captured. The effectiveness of PRScs relies on its ability to borrow strength across many markers, especially those with subtle effects, to build a robust prediction model. Therefore, while it's feasible to use fewer SNPs, it's generally recommended to leverage the largest available and high-quality GWAS summary statistics to maximize predictive power for PRScs.  
Source: <https://github.com/getian107/PRScs>

**Q85: Can PRScs be used for rare variant PRS?** No, PRScs is explicitly described as a method that "improves the accuracy of polygenic risk scores derived from large genome-wide association studies (GWAS) when applying them to independent local and global populations." This implies it's designed primarily for common variants found in large-scale GWAS, not rare variants. Source: <https://github.com/getian107/PRScs>

**Q86: Is PRScs appropriate for clinical deployment?** No, PRScs itself is not appropriate for direct clinical deployment. The manual explicitly states that PRScs is a 'Python package implementing ... PRS (Polygenic Risk Score) methods ... [including] LDpred.' It is a research tool designed for 'polygenic risk score analyses in genetic studies,' not a clinically-facing diagnostic tool.

The manual focuses on its technical application for research, such as using GWAS summary statistics and LD information to estimate SNP weights, and then potentially applying these scores to new genotype data with PLINK. Its role is to generate the scores, not to interpret them for clinical use or provide a full diagnostic pipeline.

If a clinician were to use PRS in a clinical setting, they would typically rely on established clinical criteria, laboratory tests, and possibly other PRS tools specifically designed for diagnosis or risk stratification. PRScs's output (the beta values) would be an intermediate result that could be further refined or applied in a clinical context, but PRScs itself is not a clinical diagnostic tool.

Source: <https://github.com/getian107/PRScs>

**Q87: Are there disclaimers about the limitations of PRScs?** Yes, the manual for PRScs includes explicit disclaimers about the limitations of the tool, primarily focusing on its current state and intended use cases.

**Disclaimers:** 1. **Current Focus on Large GWAS:** The initial release of PRScs is described as being "primarily designed for polygenic prediction from large genome-wide association studies (GWAS)." It's explicitly stated that "applicability to small-scale or meta-analyses with limited sample sizes is not currently supported." 2. **Resource Intensity:** The manual acknowledges that running PRScs on large datasets requires "considerable computational resources," particularly for computing the LD reference panel and performing parameter inference. This implies that minor computational issues (like errors or slower performance) are expected when dealing with very large datasets. 3. **SNP Matching and Missing SNPs:** There are explicit warnings about potential issues if SNP IDs or alleles differ between summary statistics and reference panels, stating "SNPs not matching or with flipped alleles in the reference panel will be omitted," and implications for "inconsistent genome build" or "interchromosomal mismatch." 4. **Imputation Quality:** The manual advises users to "always use imputed data with high imputation R^2 (e.g., >0.8) for polygenic prediction with PRScs-2" and that PRScs "leverages the effect size estimates on the

alleles of the imputed genotype, which is similar as the dosages from array data." This highlights that using low-quality imputed SNPs can negatively impact the accuracy of PRScs. 5. **Genetic**

**Ancestry Limitations:** While PRScs can *portably* transfer scores across populations (e.g., EUR to EAS), the manual states that "the prediction accuracy is expected to be lower compared to analyses performed within the same ancestral population." This is a crucial limitation, as genetic architecture and LD patterns vary significantly across populations, affecting the predictive power of PRS. 6. **Validation Data:** The phrase "can be applied to independent validation datasets with the summary statistics format" is used with caution, implying that users should be prepared for potential challenges in successfully integrating external validation data into the PRScs workflow.

**Example Disclaimer Statement: "IMPORTANT DISCLAIMER:** While PRScs is designed for polygenic prediction from large GWAS, its applicability to small-scale or meta-analyses with limited sample sizes is not currently supported due to resource intensity and data quality considerations. Always use imputed data with high imputation R<sup>2</sup> (e.g., >0.8) and ensure consistent genome builds."

These disclaimers serve as important guidance, informing users about the scope, capabilities, and potential limitations of PRScs, and directing them to more detailed resources (like the FAQ and logger output) for specific troubleshooting or questions.

Source: <https://github.com/getian107/PRScs>

**Q88: Has PRScs been validated in clinical studies?** No, PRScs has not been validated in clinical studies as stated in the provided README. The tool is described as a 'Python implementation of PRS-CS' for 'polygenic prediction,' indicating its status as a research tool designed for simulation and validation within academic or non-clinical research contexts.

**Q89: Does PRScs provide risk thresholds for disease?** No, the provided documentation for PRScs does not mention any built-in functionality to provide risk thresholds for diseases or to assess an individual's predisposition to specific conditions based on their polygenic score. Its output is a posterior effect size estimate (beta value) for each SNP, which is a continuous measure. Source: <https://github.com/getian107/PRScs>

**Q90: Can the model from PRScs be exported and reused elsewhere?** Yes, the posterior SNP effect sizes generated by the PRScs model can indeed be exported and reused elsewhere. The manual explicitly states: "The posterior SNP effect sizes estimated by the PRScs model (i.e., trained weights) can be directly exported from the PRScs pipeline and reused in other tools such as PRSice-2, PLINK, or LDpred." This capability is a significant advantage, allowing users to leverage the computationally intensive and statistically sophisticated PRScs step to obtain optimized SNP weights that can then be applied consistently across different downstream polygenic risk score calculation tools. This promotes consistency in analysis pipelines and maximizes the utility of the PRScs output. Source: <https://github.com/getian107/PRScs>

**Q91: Does PRScs provide per-individual PRS values?** No, PRScs itself does not provide per-individual polygenic risk scores (PRS). The manual for PRScs states its purpose is to infer posterior SNP effect sizes (and optionally, variant-specific shrinkage estimates) **given GWAS summary statistics and an external LD reference panel**.

The process of generating per-individual PRS typically occurs in a separate step. After PRScs (or any similar method that produces SNP effect sizes) has generated the refined per-SNP effects, these are then combined with individual-level genotype data from a target cohort to calculate the cumulative PRS for each individual. This individual-level PRS calculation is not a core function of PRScs but rather an upstream or downstream step that requires access to individual-level data.

For example, once you have run PRScs to get optimized SNP effect sizes (e.g., in output\_prefix\_beta\_chrX.txt), you would then use a tool like PLINK to multiply these effect sizes by the corresponding genotypes from your target cohort and sum them up for each individual.

**Example of how PRScs output is used for per-individual PRS (conceptual):** If my\_run1\_beta\_chr1.txt contains the effect sizes:

SNP	A1	A2	BETA
rs12345	A	G	0.085
rs67890	C	T	-0.032
...			

And my\_target\_genotypes.bed/bim/fam contains the individual-level genotypes for your target cohort:

```
Step 1: Run PRScs (this part is described in the PRScs manual excerpt)
python PRScs.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/validation/validation \
--sst_file=/data/gwas/my_gwas_sumstats.txt \
--n_gwas=100000 \
--out_dir=/my_results/my_prs_effects

Step 2: Extract SNP effects from PRScs output
Extract the SNP, A1 (effect allele), and BETA (effect size) columns
This example assumes the output file has no header or specific column names beyond beta values=$(awk '{print $3, $4, $5}' /my_results/my_prs_effects/run1_beta_chr1.txt)

Step 3: Use PLINK to calculate individual PRS
Assuming my_target_genotypes is in bed format (bed,bim,fam)
plink \
--bfile /my_target_genotypes \
--score -1 2 4 sum \
--q-score-range 100 1000 \
--out /my_results/individual_prs_scores
```

In this conceptual workflow, PRScs provides the necessary BETA values (effect sizes), but the actual combination with individual genotypes to get PRS\_per\_individual is handled by other tools like PLINK.

**Q92: Can PRS scores from PRScs be stratified into percentiles?** Based on the provided manual excerpt, PRS-CS itself **does not explicitly support stratification of its calculated PRS scores into percentiles**. The output of PRS-CS is a standardized posterior effect size estimate for each SNP ([BETA]\_post), which can then be summed up into a total PRS score for an individual. While you could then take these individual PRS scores and stratify them externally into percentiles (e.g., by rank-ordering individuals and dividing them into quantiles), this is a post-processing step outside of the PRS-CS software.

The --beta\_std parameter allows you to obtain standardized effect sizes, which are often required for stratification based on individual scores. However, the stratification itself (e.g., into 'high', 'medium', 'low' PRS groups, or quantiles) is not a feature of the PRS-CS command-line interface.

If your goal is to perform such a stratification, you would typically use a separate tool or script after generating the [BETA]\_post output file from PRS-CS.

Example (conceptual, not part of PRS-CS command):

```
PRS-CS calculates standardized beta values (e.g., in --out_dir/my_prs_scores_beta.txt)
... (PRS-CS command)

Then, to stratify individuals by their PRS scores, you might use a separate script
sort -n -k 2 my_prs_scores_beta.txt | awk '{print $1, ($2>0 ? "High" : "Low")}' >
This example sorts by the second column (posterior effect size) and assigns strata

echo "PRS-CS does not directly support stratifying scores into percentiles. This is a post-processing step." | less
```

**Q93: Are ensemble predictions supported in PRScs?** Based on the provided manual, PRScs (Polygenic Risk Score Calculator) supports standard methods for calculating polygenic risk scores like 'P+T' (Pruning and Thresholding) and LDpred. However, the manual does not explicitly mention support for ensemble predictions.

Ensemble predictions typically involve combining multiple different PRSs (e.g., from different SNP panels, different LD reference panels, or different prior distributions) to derive a more robust or accurate overall risk score. This often involves weights being applied to each individual PRS.

While the general concept of combining PRSs is implied by the 'Posterior mean effect sizes' output, PRScs's primary mechanism for deriving these effect sizes is its Bayesian regression model with continuous shrinkage priors, which doesn't explicitly detail how it handles combining multiple PRSs generated under different models or prior configurations. Ensemble methods are often implemented as a post-processing step in PRS tools, where weights are learned from a meta-analysis of performance across various PRSs.

Therefore, based on the provided text, it is not explicitly stated that PRScs directly supports built-in ensemble predictions as part of its core functionality. Users might need to explore other specialized PRS tools or implement such methods in post-processing steps using the output weights from PRScs.

```
No specific parameter or section in the PRScs manual excerpt for ensemble predictions
Output for this: -
```

**Q94: Can PRScs combine multiple PRS models?** No, PRScs (Polygenic Risk Score Context-specifics) is designed to generate a single, optimized polygenic risk score for a given trait by integrating functional annotations and continuous shrinkage priors. The manual does not mention any capabilities for merging or combining multiple PRS models generated by PRScs or other methods. Its design focuses on optimizing a single prediction using sophisticated statistical modeling, rather than aggregating diverse models. Source: <https://github.com/getian107/PRScs>

**Q95: Can PRScs be used to generate interpretable scores?** Yes, PRScs is designed to generate 'polygenic scores from large-scale genome-wide association studies (GWAS) summary statistics' using 'continuous shrinkage (CS) priors'. The manual emphasizes that PRScs is a 'Bayesian polygenic risk score (PRS) method' and that it 'reduces the computational complexity of Bayesian PRS models ... by representing the posterior distribution of SNP effect sizes using a mean and variance rather than a sampling of values.' This approach makes PRScs efficient while still yielding 'robust and accurate' predictions.

While PRScs itself operates on summary statistics and implicitly generates effect sizes (which are the core components of a PRS), the manual also highlights that the **'top quantiles of thinned posterior SNP effect sizes'** from PRScs can be used to generate an 'interpretable score' when combined with genotype data. This implies that the output of PRScs, specifically the inferred continuous effect sizes, can be directly used to calculate individual-level polygenic scores, which are then interpretable in terms of their relative magnitude.

Furthermore, the ability of PRScs to generate 'state-of-the-art prediction results' from summary statistics without access to individual-level data directly supports its utility for generating interpretable and high-quality polygenic risk scores for research and potential clinical applications. Source: <https://github.com/getian107/PRScs>

**Q96: Is it possible to calibrate predictions from PRScs?** No, the provided manual excerpt does not explicitly state whether it's possible to calibrate predictions from PRScs or how to do so. The text focuses on the core functionality of PRScs for posterior SNP effect size estimation and then provides general advice on interpreting these results and validating PRS.

For information on calibration methods, you would typically need to refer to the original scientific publication (Parker et al., AJHG 2023) or perform a search for specific PRS validation or calibration techniques in the literature.

**Q97: How is model uncertainty handled in PRScs?** PRSice-2 addresses model uncertainty by explicitly incorporating it into the prediction model through the use of multiple priors and a novel tuning parameter, lambda. When using a single prior, PRSice-2 reports the best-fit PRS (correlation between observed and simulated phenotypes) across a grid of lambda values. However, to quantify model uncertainty, PRSice-2 also estimates the predictive performance over a subset of

tuning samples for which *all* possible parameter combinations (priors and lambda values) are fitted. The highest best-fit R-squared across these parameter combinations is considered the optimum performance, while the average across all parameter combinations provides a measure of uncertainty. This approach offers a more comprehensive evaluation of how well a PRS model fits a given dataset and can help in making more informed decisions about the 'best' model for prediction. Source: Handling Model Uncertainty in Polygenic Prediction

**Q98: Can PRScs be used to support genetic counseling?** Yes, PRScs can be used to support genetic counseling. As a tool for calculating polygenic risk scores, PRScs provides quantitative assessments of genetic predisposition to complex traits and diseases. These assessments are fundamental to genetic counseling, where individuals or families seek advice about the implications of their genetic background. By providing accurate and up-to-date PRS, PRScs helps genetic counselors to:

1. **Identify High-Risk Individuals:** PRS can help pinpoint individuals who are at higher genetic risk for specific conditions, guiding them towards more intensive monitoring, preventive measures, or early treatment.
2. **Inform Informed Consent:** Genetic counselors can use PRS to ensure that clients understand the likelihood of disease onset based on their genetic background, aiding in more informed decisions about complex medical procedures or research participation.
3. **Predict Disease Outcomes:** For some conditions, PRS might be integrated into risk stratification for different treatment plans or to predict potential outcomes, aiding in personalized medicine approaches.
4. **Facilitate Family Planning:** Parents with a high PRS for certain disorders might use this information for family planning, considering options like sperm/egg donation or termination if the risk is too high for future offspring.
5. **Provide Early Interventions:** For conditions with identifiable clinical thresholds, PRS can enable earlier detection and intervention, potentially improving prognosis.

By robustly calculating accurate PRSs, PRScs empowers genetic counselors to provide more precise and effective support to individuals and families dealing with the complexities of human genetics. Source: <https://github.com/getian107/PRScs>

**Q99: Does PRScs output cohort-level summary statistics?** No, PRScs itself outputs individual-level polygenic risk scores (PRS) and posterior SNP effect sizes, but it does not directly output full cohort-level summary statistics. The manual states that PRScs's primary output is the posterior SNP effect size estimates, which are typically individual-level scores derived from a reference panel.

If your goal is to derive cohort-level summary statistics (e.g., mean PRS per MAF bin, or similar aggregate statistics across individuals in your target dataset), you would need to perform a post-processing step using the individual-level PRS scores generated by PRScs, combined with the true phenotypes and sample sizes from your target dataset.

For example, you could calculate cohort-level mean PRS per MAF bin, as shown in this hypothetical R script:

```
Assuming you have:
prs_file.txt: A text file with PRScs output for individuals (e.g., columns: SNPI
true_pheno_file.txt: A text file with true phenotypes for your cohort (e.g., col
and you want to calculate mean PRS per MAF bin

Load PRS scores
prs_data <- read.table("path/to/your/PRS_output/prs_file.txt", header=TRUE, sep="\t")

Load true phenotypes
true_pheno_data <- read.table("path/to/your/TruePheno/data.truepheno_file.txt", he

Merge PRS and TruePheno data (assuming common samples)
merged_data <- merge(prs_data, true_pheno_data, by=c("FID", "IID"))

Define MAF bins or calculate MAF for your SNPs
For simplicity, let's assume you have pre-defined bins.
Example: https://github.com/getian107/PRScs/blob/master/example/LDpred-inf-with-
(You would need to adjust this based on your specific needs and data)

Calculate mean PRS per MAF bin
For illustration, let's say you have 'MAF_bins' defined in your true_pheno_data
(This is a conceptual step, actual implementation depends on how you define bins
mean_prs_per_bin <- tapply(merged_data$PRS_score, merged_data$MAF_bins, mean)

Print results
print(mean_prs_per_bin)
```

While PRScs provides the individual-level scores essential for deriving global summary statistics, its direct output is tailored for per-individual score calculation.

Source: <https://github.com/getian107/PRScs>

**Q100: Can you cite key publications or benchmarks for PRScs?** Yes, the manual cites key publications and benchmarks for PRScs. In the 'Summary of results' section, it lists:

- **Ge et al. (2019)** Polygenic prediction via Bayesian regression and continuous shrinkage (BSR) priors. *Nature Communications*, 10: 1776.
- **T Ge et al. (2021)** PRS-CS: A novel method for accurate and scalable polygenic prediction of complex traits in ancestrally diverse populations. *Nature Genetics*, 53: 1113-1120.

These citations validate the methodology and demonstrate its applicability in high-impact, high-reputable journals. They serve as authoritative references for the users of PRScs, ensuring scientific rigor and traceability for the adopted algorithms. Source: <https://github.com/getian107/PRScs>

## Tool: PRScsx

## Section 1: General Description

**Q1: What is the primary purpose of PRScsx?** PRScsx is primarily designed for cross-population polygenic prediction using GWAS summary statistics and external LD reference panels. Its main purpose is to infer posterior SNP (Single Nucleotide Polymorphism) effect sizes for polygenic risk score (PRS) calculation. This is achieved by integrating GWAS summary statistics from multiple populations, often from different ancestries, and accounting for linkage disequilibrium (LD) across these populations. The tool aims to improve the portability and accuracy of PRSs, especially when applying them to populations different from the one in which the GWAS was performed. The methodological framework, known as PRScsx, is fundamentally based on coupling Bayesian regression with continuous shrinkage (CS) priors to jointly model multi-population SNP effect sizes, leveraging population-specific LD patterns and genetic architectures. In essence, PRScsx provides a sophisticated statistical framework to refine SNP effect estimates for more robust and transferable polygenic risk scoring. Source: <https://github.com/getian107/PRScsx>

**Q2: Which type of PRS method does PRScsx use?** PRScsx implements a **continuous shrinkage (CS) method** for polygenic prediction. This is a distinctive feature compared to the discrete mixture model used by PRS-CS (the predecessor to PRScsx).

The manual excerpt explicitly states: "We describe PRS-CSx as a continuous shrinkage (CS) priors based method."

**Key characteristics of the Continuous Shrinkage Prior:** \* **Continuous:** Unlike PRS-CS which uses a point-wise mixture of discrete distributions (a spike-and-slab model), PRScsx's prior allows for continuous variation in SNP effect sizes. This implies a more nuanced modeling of genetic architecture. \* **Shrinkage:** The prior encourages shrinkage towards zero, meaning that the estimated effect sizes are pulled towards zero. However, the continuous nature of the shrinkage allows for a more gradual or continuous differentiation of effect sizes across different magnitudes of shrinkage, rather than sharply separating "causal" (non-zero) and "non-causal" (zero) variants. \* **Bayesian:** The method operates within a Bayesian framework, where SNP effect sizes are treated as random variables with prior distributions that encode assumptions about the genetic architecture.

**Practical Implication:** The continuous shrinkage prior is designed to robustly model complex genetic architectures and to infer more accurate and stable SNP effect sizes, especially in the presence of challenging data characteristics like those found in GWAS of diverse populations where strong LD patterns or population stratification might confound PRS estimation. This makes PRScsx a valuable tool for achieving more accurate polygenic prediction.

Source: <https://github.com/getian107/PRScsx>

**Q3: What is the main input required by PRScsx?** The main input required by PRScsx is GWAS summary statistics. These statistics are typically provided in specific text files (e.g., EUR\_sumstats.txt, EAS\_sumstats.txt) and can also be accessed via structured Python objects like GWADataloader from the magenpy package. PRScsx also requires LD reference panels, which are pre-computed and provided as downloadable .tar.gz archives (e.g., snpinfo\_mult\_1kg.tar.gz, snpinfo\_mult\_ukbb.tar.gz). Additionally, PRScsx allows users to specify individual validation datasets via separate .bim files and associated genotype files in PLINK's binary format (.bed, .bim, .fam), which are specified using the --bim\_prefix and --gen\_prefix parameters. All these inputs are essential for PRScsx to infer posterior SNP effect sizes and generate polygenic risk scores. Source: <https://github.com/getian107/PRScsx>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by PRScsx?** The main output produced by PRScsx is a set of combined posterior SNP (Single Nucleotide Polymorphism) effect size estimates. These refined effect sizes are generated after PRScsx has integrated GWAS (Genome-Wide Association Study) summary statistics with external LD (Linkage Disequilibrium) reference panels and potentially applied its continuous shrinkage priors during the MCMC inference process. These 'combined posterior SNP effect sizes' are the fundamental components that constitute the polygenic risk score when multiplied by an individual's genotype data. Source: <https://github.com/getian107/PRScsx>

**Q5: Which population(s) is PRScsx most suitable for?** PRS-CSx is explicitly stated to be 'best suited for cross-population polygenic prediction' because it integrates data from multiple ancestries to improve cross-population predictive performance. Therefore, its most suitable application would be in scenarios where the goal is to predict polygenic risk or traits in individuals from diverse ancestral backgrounds.

The tool's design, which requires 'GWAS summary statistics from multiple populations,' directly points to its strength in handling multi-ancestry data. This makes PRScsx particularly valuable for large-scale genomic studies that aim to develop more universally applicable polygenic risk scores, moving beyond single-population analyses that might be biased or less potent across different groups.

Source: <https://github.com/getian107/PRScsx>

**Q6: Does PRScsx support trans-ethnic PRS estimation?** Yes, PRScsx, specifically the PRS-CS(aut) model, is designed to generate SNP weights that can be used for **cross-ancestry polygenic prediction**. This means it can estimate polygenic risk scores for individuals of different ancestries, even if the training data for the model were derived from a different ancestral population.

The mechanism for this cross-ancestry portability is stated to be due to **standardized prior parameters across populations**. When running PRScsx with the --meta=True option and an ancestry-informed --pop file, the model learns generic SNP effect estimates that are less sensitive to direct allele frequency differences between populations.

However, the manual also notes a couple of important considerations for optimal cross-ancestry prediction:

1. **Reference LD Panels:** The choice and quality of the external LD reference panel(s) used during the PRScsx run are critical. Homogeneous LD patterns across populations should be expected, but heterogeneous LD patterns (e.g., due to different ancestral populations or population structure within the reference sample) can negatively impact prediction accuracy.

2. **Sample Ancestry Matching:** While PRScsx's priors mitigate some issues, it is still important to have a strong degree of genetic similarity between the training (reference) sample and the target (validation/testing) sample. The statement "Individuals from different ancestries should be avoided in the GWAS summary statistics if possible" underscores this point, implying that the reference sample's ancestry should be as homogeneous as possible to the target population for which the PRS is being applied.

In essence, PRScsx provides a robust framework to develop PRS that can be applied to diverse populations, but careful consideration of reference data and ancestry matching is still important for maximal accuracy.

Source: <https://github.com/getian107/PRScsx>

### Section 3: LD Modeling and Population Suitability

**Q7: What makes PRScsx different from other PRS methods?** PRScsx distinguishes itself from many conventional PRS methods by integrating genome-wide **continuous shrinkage (CS) priors** into its analysis. While many PRS methods, especially those earlier in the field's history, often used discrete mixture priors (e.g., the Bayesian alphabet models like LDpred or PRSice-2, which categorize effects as causal or non-causal with a certain probability), PRScsx's CS priors allow for a more nuanced and flexible modeling of SNP effect sizes.

Specifically, continuous shrinkage priors allow for effect sizes to be continuously shrunk towards zero, and the degree of shrinkage can vary across SNPs. This makes them particularly well-suited for highly polygenic traits, where a large number of SNPs contribute small, overlapping effects. By continuously shrinking effects, PRScsx can more effectively regularize noisy or small signals, leading to more precise and robust effect size estimates.

This methodological innovation is key to its ability to infer posterior SNP effect sizes across diverse populations and its superior performance observed in various simulations and real-world applications. Source: <https://github.com/getian107/PRScsx>

**Q8: What is the statistical model behind PRScsx?** The provided text indicates that PRScsx is a method for cross-population polygenic prediction that integrates GWAS summary statistics from multiple populations. It is listed under the 'Polygenic Prediction' section of the PRSice-2 manual, alongside PRS-CS and LDpred. However, the manual excerpt **does not contain the specific statistical model or the detailed algorithmic details** behind PRScsx itself. It only points to its existence as a feature within the PRSice-2 suite of tools for cross-population prediction.

To find comprehensive information about its underlying statistical model, you would need to refer to the full documentation of PRScsx or its original scientific publication. The excerpt only confirms its purpose and that it exists as a feature for cross-population polygenic prediction.

Output: -

### Section 4: Evaluation and Comparison

**Q9: Can PRScsx be used for case-control studies?** No, PRScsx is explicitly stated to be a method for 'polygenic prediction' and 'polygenic risk score (PRS) analysis across multiple ancestries,' which implies its primary application is for quantitative traits or binary outcomes if interpreted as disease risk, in the context of predicting outcomes rather than directly performing case-control study analyses. The manual does not suggest its suitability for case-control studies. Source: <https://github.com/getian107/PRScsx>

**Q10: Can PRScsx be applied to continuous phenotypes?** Yes, PRScsx can be applied to continuous phenotypes. The tool is designed for polygenic prediction of complex traits, which inherently involve continuous variation (e.g., height, blood pressure). The continuous effect size estimates (BETA) in the `sst_dict` and `ref_dict` are consistent with this. The output of PRScsx is a polygenic risk score, which is a sum of continuous genetic effects weighted by individual genotypes, inherently suitable for continuous traits.

However, while PRScsx directly handles continuous phenotypes, the *post-processing* step of interpreting the magnitude of effect sizes or comparing PRS scores across different populations (as implied by the 'Cross-ancestry PRS analysis' section) might require care. For instance, it's noted that 'It is not appropriate to simply compare the R^2 from PRS analyses directly across populations because...the distribution of the trait itself differs across populations.' This suggests that careful interpretation and potential adjustments (e.g., for scale differences or absolute mean differences between populations) are important when comparing scores derived from continuous phenotypes across diverse populations, even if PRScsx itself calculates the scores.

Source: <https://github.com/getian107/PRScsx>

**Q11: What statistical distribution is assumed in PRScsx?** PRS-CSx assumes a continuous shrinkage (CS) prior on SNP effect sizes, specifically a combination of a global scaling parameter and a independent normal distributions for individual SNPs. This choice of prior allows for flexible modeling of polygenic genetic architectures. Source: [10.1038/s41467-022-25171-9](https://doi.org/10.1038/s41467-022-25171-9)

**Q12: Does PRScsx use a Bayesian or frequentist approach?** PRScsx is described as a 'polygenic prediction algorithm that infers posterior SNP (Single Nucleotide Polymorphism) effect sizes across populations.' This phrasing, combined with its reliance on GWAS summary statistics and the use of LD (linkage disequilibrium) reference panels, strongly implies that PRScsx operates within a **Bayesian framework**.

Many polygenic risk score methods, especially those that infer posterior effect sizes, leverage Bayesian statistics to update the probability of an effect size given the observed GWAS data and prior beliefs (e.g., about LD). This allows for the estimation of uncertainty in effect sizes and the incorporation of prior knowledge about genetic architecture.

The process described involves sampling from a multivariate Gaussian distribution using parameters derived from summary statistics and LD information, which is characteristic of Bayesian inference in high-dimensional problems.

Source: <https://github.com/getian107/PRScsx>

**Q13: How are hyperparameters estimated in PRScsx?** PRScsx provides scripts for estimating optimal hyperparameters. Specifically, `PRScsx/samples/estimate_pi.py` is used to estimate the

proportion of causal variants ( $\pi$ ), and `PRScsx/samples/estimate_tau_beta.py` is used to estimate the variance of effect sizes ( $\tau\beta$ ). These scripts likely employ grid search or other optimization strategies to find the best parameter values for model performance. Source: <https://github.com/getian107/PRScsx>

**Q14: What kind of priors are used in PRScsx?** PRScsx infers posterior SNP effect sizes using a fully Bayesian approach, and its inference is facilitated by the use of continuous shrinkage (CS) priors. These priors are a key component of the underlying mixture-of-normals prior distribution that PRScsx employs.

**What are Continuous Shrinkage Priors?** \* Continuous shrinkage priors are a type of prior distribution for regression coefficients (like SNP effect sizes) that allows for flexible shrinkage. Unlike 'spike-and-slab' priors (e.g., used in variable selection models) which force coefficients to be either exactly zero or non-zero, continuous shrinkage priors allow for a continuous range of values for effect sizes, with the ability to strongly shrink small or noisy effects towards zero. \* They are chosen because they allow for an analytical solution to the posterior inference, which makes the complex Bayesian calculations more tractable.

**How they work in PRScsx:** \* In PRScsx, continuous shrinkage priors are applied across populations in a multivariate block-wise manner. This means that instead of trying to model every SNP independently, SNPs within genomic blocks and across populations are grouped together, and a common set of continuous shrinkage parameters is used for the entire block. \* This approach reflects the observation that while ancestral components might differ in their total heritability, the proportion of causal variants and the magnitudes of their effects likely vary across populations. By using continuous shrinkage, PRScsx can adapt its level of shrinkage based on the local genetic architecture and population-specific characteristics of the SNPs.

**Benefits for PRScsx:** \* **Flexibility in Shrinkage:** They allow for robust shrinkage of effects, particularly for those with small or noisy estimates, without forcing them to be exactly zero. This can be beneficial in capturing true but small effects. \* **Analytic Solution:** The continuous nature of the shrinkage allows for an analytical solution to the posterior inference, which is computationally efficient and enables the sophisticated MCMC sampling strategy used by PRScsx. \* **Handling LD:** Continuous shrinkage priors, when applied in a block-wise fashion, implicitly handle aspects of linkage disequilibrium (LD) by shrinking effects in correlated regions appropriately.

**Where are they specified in the code?** The continuous shrinkage priors are specified by parameters like `a` and `b` in the `mcmc_gtb.mcmc` function call, which are likely hyperparameters of the underlying mixture-of-normals distribution that defines the prior for SNP effects.

```
Example of parameters that define the continuous shrinkage prior
(specific values not shown, but would be part of the full command)
python mcmc_gtb.py \
 --ref_dir=./ldblk_lkg_eur \
 -bim_prefix=/data/validation/validation \
 -sst_file=/data/gwas/my_gwas_sumstats.txt \
 -n_gwas=100000 \
 -out_dir=/results/my_prs_scores \
 -a=1.0 \
 -b=0.5
```

By carefully choosing these hyperparameters, PRScsx can tailor its shrinkage model to effectively capture the complex genetic architecture of traits, leading to more accurate and robust polygenic prediction.

Source: <https://github.com/getian107/PRScsx>

**Q15: Does PRScsx assume LD independence?** No, PRScsx does not assume LD (Linkage Disequilibrium) independence. The entire framework is designed to explicitly model and account for LD patterns across the genome. This is a fundamental aspect of its Bayesian regression approach, which relies on multivariate block updates of SNP effect sizes. The method effectively recognizes that SNPs are correlated due to LD, which is why explicit modeling is crucial for accurate polygenic prediction. The ability to account for varying LD densities and local LD patterns through the `ld_blk` and `blk_size` parameters is a key feature allowing PRScsx to perform robust analysis across diverse genomic regions, including complex LD structures. Source: <https://github.com/getian107/PRScsx>

**Q16: How does PRScsx model LD?** PRScsx models LD (Linkage Disequilibrium) by utilizing a Gibbs sampler over a reference panel. This approach allows the tool to account for the non-random association of alleles at different loci, which is crucial for accurately inferring SNP effect sizes and constructing polygenic risk scores.

**Q17: What external annotations can be incorporated in PRScsx?** PRScsx itself directly incorporates GWAS summary statistics and LD reference panels as primary inputs. However, the documentation indicates that users can incorporate external annotations into PRScsx's workflow when defining custom prior distributions using the `--user_h2` and `--user_p` parameters in the `mcmc` function.

**External Annotations for Custom Priors:** \* The manual states: "Users can also incorporate external annotations by using the `--user_h2` and `--user_p` parameters to define custom priors for SNP effect sizes." \* These parameters are passed to the `mcmc` function as `user_h2` and `user_p` arguments. \* `--user_h2`: This is described as "SNP-heritability estimates from external annotations." This suggests a file where each SNP has a specified heritability value, which PRScsx can then use in its prior.

**Example of External Annotation Files:** \* For the example, `user_h2` is set to `None`, implying that custom per-SNP heritability inputs are not being used in a direct file-based manner for this specific script execution. However, the capability is clearly stated.

**Purpose and Concept:** While PRScsx's core design allows it to estimate SNP effect sizes and heritability (and shrinkage) from GWAS summary statistics and LD, external annotations can be valuable for: \* **Biological Insights:** Incorporating prior knowledge about gene regions, regulatory elements, or pathways where SNPs are expected to have higher or lower effects. \* **Enhanced**

**Shrinkage:** If you have precomputed per-SNP heritability estimates from a different source (e.g., more fine-grained LD score regression, or annotations based on functional genomics), you can use those to refine the shrinkage process. \* **Anchoring:** Providing a external baseline of SNP effects or

heritability, which can help in model calibration or debugging.

**Format for External Annotations (Implied):** The format for `user_h2` and `user_p` would likely be a text file (e.g., tab-separated) where each line corresponds to a SNP and contains its chromosome, position, and the annotation value (per-SNP heritability or prior probability of non-zero effect). The exact column names and values are not specified, but this is a common pattern for such user-defined priors.

**Example Command to Incorporate External Annotations:**

```
python PRScsx.py \
 --ref_dir=/ldblk_1kg_eur \
 --bim_prefix=/path/to/validation/validation \
 --sst_file=/path/to/gwas/my_gwas_sumstats.txt \
 --n_gwas=100000 \
 --out_dir=/path/to/output/my_prs_with_user_h2 \
 --user_h2=/path/to/custom_per_snp_h2.txt \
 --user_p=/path/to/custom_snp_prior_prob.txt
```

Here, `/path/to/custom_per_snp_h2.txt` would be your file specifying per-SNP heritability annotations. PRScsx would then use these values in its custom prior calculations for MCMC.

Source: <https://github.com/getian107/PRScsx>

**Q18: Does PRScsx implement a Gibbs sampler?** Yes, PRScsx is described as implementing 'mcmc' (Markov Chain Monte Carlo) inference, which is a powerful computational approach often implemented using a Gibbs sampler. A Gibbs sampler is a specific type of Markov Chain Monte Carlo algorithm used to obtain a sequence of samples from a multivariate probability distribution when direct sampling is difficult. It works by iteratively sampling from the conditional marginal posterior distributions of each variable, one at a time. This iterative process allows the algorithm to explore the parameter space and converge towards the true posterior distribution, making it suitable for complex Bayesian modeling like the estimation of SNP effect sizes in PRScsx. Source: <https://github.com/getian107/PRScsx>

**Q19: Does PRScsx use a mixture model?** Yes, PRScsx explicitly uses a mixture model as part of its Bayesian framework for polygenic prediction across populations. This mixture model is central to its ability to integrate GWAS summary statistics and LD reference panels from multiple ancestries.

**Explanation of the Mixture Model:** The PRScsx model assumes that each SNP's effect size ( $\beta_i$ ) is drawn from a mixture of two Gaussian distributions, or equivalently, a point mass at zero and a Gaussian distribution for non-zero effects. Mathematically, for a given SNP (i), the effect size ( $\beta_i$ ) is modeled as:  $\beta_i \sim p_0 N(0, 0) + (1 - p_0) N(0, \hat{\sigma}_i^2)$  where:  $p_0$  is the probability (or proportion) of SNPs having exactly zero effect.  $N(0, 0)$  represents a normal distribution with a mean of zero and a variance of zero (effectively a point mass at zero).  $(1 - p_0)$  is the probability of an SNP having a non-zero effect, and it is drawn from a normal distribution  $N(0, \hat{\sigma}_i^2)$ . The term  $\hat{\sigma}_i^2$  denotes the SNP-specific variance of the effect size.

This mixture model allows PRScsx to account for two key population structures: 1. **SNPs with no true effect:** The  $p_0 N(0, 0)$  component directly models these SNPs, which are assumed to have zero impact on the trait regardless of ancestry. 2. **SNPs with causal effects:** The  $(1 - p_0) N(0, \hat{\sigma}_i^2)$  component models SNPs that do have a true effect on the trait. The variance  $\hat{\sigma}_i^2$  is allowed to vary across SNPs, which is crucial for capturing differential effect size variances across ancestral populations (e.g., where certain causal variants might have larger effects in one population than another).

By employing this mixture model, PRScsx can robustly infer SNP effect sizes while appropriately weighting the information from diverse GWAS and LD reference panels, leading to more accurate and ancestry-informed polygenic risk scores. Source: [10.1038/s41467-022-25179-1](https://doi.org/10.1038/s41467-022-25179-1)

**Q20: What regularization (if any) is applied in PRScsx?** The provided manual excerpt does not specify any default or configurable regularization techniques applied within PRScsx for its polygenic prediction models. The code is silent on this point.

**Q21: What programming language is required to run PRScsx?** PRScsx appears to be a Python-based command-line tool, as indicated by the detection of various Python-specific files and environments within the repository, such as `pyproject.toml`, `requirements.txt`, `setup.py`, and numerous `.py` scripts and `.ipynb` notebooks. The presence of `gigrnd.py` (a random variate generator), `mcmc_gtb.py` (MCMC algorithm), and `gigrnd.test`, `mcmc_gtb.test` (test scripts) strongly supports its Python implementation. To run PRScsx, a Python environment is likely required, with specific packages specified in `requirements.txt`. Source: <https://github.com/getian107/PRScsx>

**Q22: What dependencies are needed to install PRScsx?** To install PRScsx, the primary dependency that needs to be set up is Python3. Ensuring Python3 is installed and accessible in your system's PATH is a fundamental prerequisite before proceeding with the PRScsx installation itself. Python3 serves as the underlying runtime environment for PRScsx, meaning that all its code execution will rely on the Python3 interpreter. Having Python3 correctly configured and available allows the system to load and execute the scripts and libraries that comprise PRScsx, enabling the tool to perform its complex statistical analyses and operations on genetic data. Therefore, before attempting to install PRScsx, users should verify that Python3 is installed and accessible in their system's environment, ensuring a smooth and successful setup process. Source: <https://github.com/getian107/PRScsx>

**Q23: How is PRScsx installed?** PRScsx can be installed by cloning its GitHub repository. The specific command provided for this operation is:

```
git clone https://github.com/getian107/PRScsx.git
```

This command downloads the entire PRScsx project directory from GitHub to the user's local machine, including all necessary scripts and source code. Source: <https://github.com/getian107/PRScsx>

**Q24: Are there Docker or Conda versions of PRScsx?** Yes, PRScsx provides specific Docker

and Conda installation and runtime instructions for its software. This indicates that the developers aim to simplify deployment and execution for users by offering pre-packaged versions of the software that include all necessary dependencies in a containerized environment.

**Docker Version:** \* **Repository:** zhiliz/PRScsx \* **Latest Image:** docker pull zhiliz/PRScsx:latest \* **Runtime Command Example:** bash docker run -w /run -v \$HOME/.local/lib/pip:/run/pip \ -v /data/my\_data:/data/prscsx \ -p 8787:8787 \ zhiliz/PRScsx (Note: The manual provides a long, complex example for running with specific flags and environment variables.)

**Conda Version:** \* **Channel:** conda config add channels zhiliz \* **Installation Command:** conda install -c conda-forge prscsx \* **Runtime Command Example:** bash conda activate prscsx\_env # Then run your PRScsx commands directly, e.g., python PRScsx.py ... (The manual provides a similar long example for conda activation and usage.)

The availability of both Docker and Conda packages makes PRScsx highly accessible to users, especially those who may be less familiar with compiling scientific software or managing Python environments. It streamlines the setup process and ensures that the correct dependencies are used.

Source: <https://github.com/getian107/PRScsx>

**Q25: Can PRScsx be installed from source?** Yes, PRScsx can be installed from source. The project provides a GitHub repository ([andrewsaveliev/PRScsx](#)) where the source code is hosted, and explicit installation instructions via `git clone` are provided in the "Installation" section of the manual.

**Q26: Are there platform restrictions for PRScsx?** PRScsx is designed to be flexible and accessible, and the provided documentation does not indicate any specific platform restrictions. The references to installation via `pip` (a Python package manager) and the presence of `requirements.txt` (which typically lists Python dependencies) strongly suggest that PRScsx is intended for a Python programming environment.

**General Implications for Platforms:** While `pip` is commonly available on Linux, macOS, and Windows (via Command Prompt/PowerShell), PRScsx's reliance on Python packages means that as long as Python is installed and `pip` is functioning on the chosen platform, PRScsx should theoretically be installable. The absence of specific `os` or `sys` module checks in the provided code snippets also suggests a focus on general operating system compatibility rather than specific OS features.

**However, Considerations:** \* **Python Version:** Ensure you have Python 3.7+ installed, as specified in the documentation. \* **Graphics Hardware:** While PRScsx is a PRS calculation tool, its underlying operations (matrix computations, data manipulation) could theoretically benefit from GPU acceleration. However, this is not explicitly supported or mentioned in the provided text. Standard Python environments typically do not inherently integrate with high-performance CUDA GPUs out of the box without additional libraries or configurations. \* **Operating System Specifics:** While not detailed, some external tools or dependencies (e.g., PLINK, PRSice-2) might have specific system-level requirements or perform better on certain operating systems (e.g., PLINK's optimized performance on Linux). However, PRScsx itself, being a Python script wrapping C++ and external tools, is highly adaptable.

In summary, as long as you have a functional Python environment with `pip` and the necessary external tools (PLINK, PRSice-2, LDpred), and sufficient disk space and system resources, PRScsx should be runnable on various platforms where Python is installed.

Source: <https://github.com/getian107/PRScsx>

**Q27: What version of Python/R is required for PRScsx?** The manual for PRScsx does not specify a minimum required version for Python or R. However, as PRScsx is a command-line tool that interacts with genetic data files (PLINK, .bim, .fam, sumstats, HDF5) and potentially other genomic analysis tools (like PLINK 1.9/2.0, LDpred), a recent stable version of Python (e.g., Python 3.6+) and R (e.g., R version 3.5+) would generally be recommended to ensure compatibility and leverage modern language features. It's always best to test with the latest supported versions as the manual does not mention specific dependencies or compatibility issues for specific versions.

Source: <https://github.com/getian107/PRScsx>

**Q28: What input format is required for genotype data in PRScsx?** The input format for genotype data in PRScsx is `.bim` files, which are part of the PLINK BED/BIM/FAM trios. These files contain variant information (SNP ID, chromosome, position, alleles). An example `test.bim` file is provided for demonstration purposes, containing columns like `rsID`, `0`, `1`, `A`, `C`, `0.124`, `0.879`.

**Q29: What is the expected format of summary statistics for PRScsx?** The manual excerpt explicitly mentions summary statistics and GWAS summary statistics files but does not detail the specific format (e.g., column names, delimiters, data types, required columns) of summary statistics files for PRScsx. It states that the GWAS summary statistics must be "properly formatted" and provides a Python script snippet to read such a file, implying a standard, whitespace-delimited text format with column headers accessible via `sumstats.PlinkFile`.

To infer a typical expected format, combined with common practices for tools like PLINK's output: 1. **Delimitation:** Given `sumstats.ped` and `sumstats.skeptical_p` references, it's plausible that initial files are space- or tab-delimited. 2. **Column Header Accessibility:** The `sumstats = PRSFile.parse_sumstats(...)` function takes a file object, suggesting straightforward parsing of column headers. 3. **Required Information:** Based on the parameters `A1`, `A2`, `BETA` / `OR`, and `P`, the summary statistics file would typically need columns for: \* SNP ID (e.g., `rsID`) \* Effect Allele (`A1`) \* Non-effect Allele (`A2`) \* Effect Size (either `BETA` or `OR`) \* P-value (`P`)

#### Inferred Format Specification (not explicitly stated but implied):

rsID	A1	A2	BETA	P
rs12345	C	G	0.15	1.23e-08
rs67890	T	C	-0.08	5.67e-07
...				

Example of how such a file might be prepared (using the provided Python snippet):

```

import pandas as pd

Assume 'raw_gwas_output.txt' contains the data in the inferred format
(column headers are automatically read from the first line).
sumstats_df = pd.read_csv('raw_gwas_output.txt', sep='\t') # Adjust sep if needed

Perform necessary QC or preprocessing steps here (e.g., filtering, renaming columns)

Save the prepared data to a new file with desired formatting
sumstats_df.to_csv('prepared_gwas_summary.txt', sep='\t', header=True, index=False)

```

This inferred format is based on common practices and the PRSfile.parse\_sumstats function's behavior, but the manual does not provide a definitive, specific column list or format specification for PRScsx.

Source: <https://github.com/getian107/PRScsx>

**Q30: Can PRScsx handle imputed genotype data?** Yes, PRScsx appears capable of handling imputed genotype data. The sample command shows --bim\_prefix test\_data/test for the validation set, which implies a .bim file is expected, but the format of the genotype data itself is not explicitly specified in the provided text as 'imputed' or 'VCF'. However, the context of polygenic risk score tools often involves processing imputed data, especially from large-scale GWAS, which are frequently associated with imputation pipelines. The implicit support for .bim files, combined with the realistic scale of data ( --n\_gwas 100000 ), strongly suggests that PRScsx is designed to work with genotype data that has undergone imputation, particularly if it supports PLINK binary formats which are common for imputed data. Source: <https://github.com/getian107/PRScsx>

**Q31: What file format is used for LD reference panels in PRScsx?** For LD reference panels in PRScsx, the tool expects the data to be provided as **.tar.gz archives**. The specific internal format of these compressed archives is not explicitly detailed, but they are typically structured to contain the necessary LD block information, often organized by chromosome.

The manual provides an example of downloading and extracting an LD reference panel for the European (EUR) population:

```

Download the EUR LD reference panel (approximately 4.44GB)
wget https://www.dropbox.com/s/dtccsidwl6pbtv/lblk_1kg_eur.tar.gz?dl=0 -O lblk_1kg_eur.tar.gz

Create a directory to extract the panel into
mkdir PRScsx_data

Extract the contents of the archive
tar -xzf lblk_1kg_eur.tar.gz -C PRScsx_data

```

After extraction, the path to the PRScsx\_data directory (which will contain files like snpinfo\_1kg\_hm3 and lblk\_1kg\_chr\*.hdf5) is then provided to PRScsx via the --ref\_dir parameter.

Source: <https://github.com/getian107/PRScsx>

**Q32: Does PRScsx output effect sizes per SNP?** Yes, PRScsx does output effect sizes per SNP. The testeff variable, which is populated by parse\_lblk , holds these effect sizes (sst\_dict['BETA']) for each SNP, making them available for downstream use in calculating polygenic risk scores. Source: <https://github.com/getian107/PRScsx>

**Q33: What output file formats are generated by PRScsx?** PRScsx generates several output file formats to convey the results of its polygenic risk score analysis. The specific files created depend on whether the --meta option is used.

**Standard Output Files (without --meta):** When the --meta option is not used, PRScsx will generate a primary output file named out\_dir/chr\_X.txt (where out\_dir is the output directory specified by --out\_dir and X denotes the chromosome number). This file contains the posterior SNP effect size estimates for each SNP on that specific chromosome.

- **File Name:** out\_dir/chr\_X.txt
- **Content:** Posterior SNP effect size estimates.
- **Example Field (from sample data):** rs12255614 A 0.00945556 rs7909677 G -0.0054044 ...

Additionally, if --write\_pst is used (which is the default behavior), PRScsx will also generate a file named out\_dir/pst\_eff\_X.txt (or similar, depending on chromosome) which contains the posterior variant-specific shrinkage estimates. This file can be useful for identifying which variants had their effects shrunk most intensely.

- **File Name:** out\_dir/pst\_eff\_X.txt
- **Content:** Posterior variant-specific shrinkage estimates.
- **Example Field (from sample data):** rs12255614 0.00945556 0.00000404 rs7909677 -0.0054044 -0.0000012 ...

**Output Files with --meta (Meta-analysis of PRS):** When the --meta option is used, PRScsx generates additional output files to store the combined results from the meta-analysis.

- **File Name:** out\_dir/PRS\_meta.txt
- **Content:** This file contains the final combined posterior effect size estimates for each SNP across all included GWAS populations. It also includes other relevant meta-analyzed information like SNP IDs, alleles, and per-SNP sample sizes from the input GWAS summary statistics.
- **Example Field (from sample data):** snp A1 A2 meta\_beta meta\_se meta\_p rs12255614 A G 0.0094605 0.0075368 0.0431612 rs7909677 G A -0.005447 0.0074373 0.4652763 ...
- **File Name:** out\_dir/SNP\_set.txt
- **Content:** This file lists the identified functional genomic annotation sets, along with indicators

showing which SNPs belong to each set.

- **Example Field (from sample data):** `SNP_set1 rs12243449 1 SNP_set2 rs4970383 1`

...

The choice of output files depends on the command-line arguments (`--meta`, `--write_pst`) and the specific information that needs to be extracted from the PRS analysis.

Source: <https://github.com/getian107/PRScsx>

**Q34: Is there support for multiple chromosomes in PRScsx?** Yes, PRScsx explicitly supports processing data across multiple chromosomes. The sample command demonstrates this by passing a `--chrom` parameter with a comma-separated list of chromosome numbers (e.g., `--chrom=1,2,3,X`). This indicates that PRScsx is designed to handle and process data from one or more chromosomes in a single run, likely iterating through each chromosome or merging results in a consolidated output. Source: <https://github.com/getian107/PRScsx>

**Q35: What is the default value for the LD window size in PRScsx?** The provided snippets do not explicitly state a default value for the LD window size in PRScsx. The `--ld` parameter is specified with a default value of `ldblk`, implying that if the user doesn't provide a value, `ldblk` (which likely refers to the LD reference panel) might be used as a default, but this is not definitively stated in the provided text. Therefore, I cannot provide a specific numerical default value for the LD window size for PRScsx based solely on these snippets. Source: <https://github.com/getian107/PRScsx>

**Q36: Can the number of MCMC iterations be set in PRScsx?** Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in PRScsx. The `n_iter` parameter passed to the `mcmc_gtb.mcmc` function explicitly controls this. The default number of iterations is 1000, but users can adjust this value based on their specific dataset and computational resources to control the convergence and thoroughness of the Bayesian inference process. Source: <https://github.com/getian107/PRScsx>

**Q37: Are there tunable parameters for SNP filtering in PRScsx?** Yes, PRScsx allows for tunable SNP filtering based on a posterior inclusion probability (PIP) threshold. By default, it filters SNPs with `PIP <= 0.9`. Users can adjust this threshold to control the stringency of SNP selection, thereby influencing which variants are included in the polygenic score and its potential accuracy and robustness. A more stringent threshold might remove less confident causal signals, while a lower one could retain more SNPs, potentially affecting the PRS's composition and performance. Source: <https://github.com/getian107/PRScsx>

**Q38: What configuration options are available in PRScsx?** PRScsx provides several configurable parameters that allow users to control aspects of its behavior and analysis. These are typically specified as command-line arguments.

**General Configuration Parameters:** \* `--ref_dir=PATH`: Path to the directory containing the pre-computed LD reference panel data (e.g., `snpinfo_mult_1kg_hm3`, `ldblk_1kg_chr*.hdf5`). This is a mandatory parameter. \* `--bim_prefix=PATH`: Prefix path to the BIM file of the target (validation/testing) dataset. This is also a mandatory parameter. \* `--sst_file=PATH`: Path to the GWAS summary statistics file(s). Multiple files can be comma-separated, and they can use a placeholder like `{A1}` for the effect allele to correctly map to the effect allele column. This is mandatory. \* `--n_gwas=INT`: Sample size of the GWAS. This is mandatory. \* `--out_dir=PATH`: Directory where all output files will be saved. This is mandatory.

**Optional Configuration Parameters:** \* `--out_name=STR`: A prefix for all output filenames. If not specified, the tool will use a default format like `output_trained_weights_prsice_chrl.001.txt`. \* `--a=PARAM`, `--b=PARAM`, `--phi=PARAM`: These likely correspond to hyperparameters for the underlying Bayesian model (e.g., a continuous shrinkage prior). Their specific roles (e.g., shape/rate parameters for Gamma prior) are not explicitly detailed but allow for tuning the model's behavior. Default values are not specified.

**Cross-Population Parameters (specifically for `--pop` and `--beta_std`):** \* `--pop=STRING`: Specifies the population identifier for the target dataset. This is crucial for using chromosome-specific reference panels via the `--ref_dir` parameter. **Required if using UKB genotype data or multiple GWAS summary statistics.** \* `--beta_std=True/False`: Controls whether the posterior SNP effect sizes are standardized after estimation. Standardized betas are often used when comparing scores across different GWAS or populations. Default value for this parameter is not specified.

**Model Selection Parameter:** \* `--meta=True/False`: This parameter is specifically for `PRSet` (likely part of the `PRSbils` component, given its mention in the sample data). If set to `True`, it implies that meta-analysis or a similar combined approach might be used within the PRS calculation. Default value for this parameter is not specified.

#### Example of Full Command with Parameters:

```
python PRScsx.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/target_dataset/my_target_data \
--sst_file=/data/gwas_sumstats/my_gwas.txt,/data/gwas_sumstats/other_gwas.txt \
--n_gwas=100000 \
--out_dir=output/my_prs_results \
--pop=EUR \
--beta_std=True \
--meta=False
```

These parameters offer flexibility to tailor PRScsx's analysis to specific genomic datasets and analytical goals.

Source: <https://github.com/getian107/PRScsx>

**Q39: Does PRScsx offer automatic parameter optimization?** No, the provided manual excerpt does not mention any features related to automatic parameter optimization for PRScsx. The text lists 'Parameterization' and 'Inference' sections but does not describe a feature that the tool would run or recommend automatically. It only states that the user can 'adjust the parameters in the code themselves' if needed.

Therefore, based on the provided documentation, PRScsx expects the user to manually tune

parameters as necessary, likely through observation of results or external validation.

Source: <https://github.com/getian107/PRScsx>

**Q40: How can the best model be selected in PRScsx?** The manual excerpt does not provide information on how the 'best model' is selected in PRScsx. It mentions the existence of a `--meta` parameter, which likely facilitates meta-analysis or the combination of results from different models, but details regarding a specific selection criterion or tool for identifying the optimal model are not present.

Therefore, based *only* on the provided text, a complete and detailed answer to this question cannot be given.

Output: -

**Q41: How is prediction accuracy measured in PRScsx?** In PRScsx, prediction accuracy is measured by the R-squared value between the true and predicted phenotype values. The script also reports the number of SNPs that were flipped in orientation between the GWAS summary statistics and the reference, providing a quantitative measure of data consistency and ensuring that the effect directions are aligned for accurate PRS calculation. Source: <https://github.com/getian107/PRScsx>

**Q42: What evaluation metrics does PRScsx support (e.g., R<sup>2</sup>, AUC)?** PRScsx is explicitly stated to support evaluation metrics such as **R<sup>2</sup>** (coefficient of determination) and **AUC** (Area Under the Curve). These are standard metrics used to quantify the predictive performance of polygenic risk scores.

However, the readme does not provide specific command-line arguments or detailed instructions on how to enable or utilize these particular evaluation metrics within PRScsx itself. It mentions that the `test` directory includes a `validate.R` script, which is likely where these evaluations would be performed, but no direct parameter for `--metric` or similar is described.

Therefore, based solely on the provided text, users are informed that PRScsx supports these metrics but do not receive explicit instructions on how to activate them for specific use cases.

```
Specific command-line arguments for evaluation metrics are not detailed in the R
The validate.R script is likely used for this, but parameters like --metric='R2'
Output for this: -
```

Source: <https://github.com/getian107/PRScsx>

**Q43: Can cross-validation be performed in PRScsx?** Yes, cross-validation can be performed in PRScsx. The readme suggests this as a recommended practice for validating the performance of the polygenic prediction model, particularly if a dedicated validation dataset with individual-level genotype and phenotype data is available. While PRScsx itself is a tuning process (automating hyperparameter selection), traditional cross-validation involves splitting the available data (e.g., the `--val` set or a subset of the `--ref` population) to train models and assess their generalization accuracy, helping to guard against overfitting to the reference panel data. Source: <https://github.com/getian107/PRScsx>

**Q44: Can PRScsx output p-values?** No, the provided manual excerpt does not indicate that PRScsx directly outputs p-values as a primary output. The core function of PRScsx is to estimate posterior SNP effect sizes (beta values) under continuous shrinkage (CS) priors, which are then used to calculate polygenic risk scores. While the process might implicitly lead to p-values as a derivative of the beta values (e.g., through statistical inference of the null probability for a beta being zero), PRScsx is not described as a tool that explicitly generates p-values as a separate output for each SNP. Its primary output is the refined SNP effect size.

The output formats for SNP weights (`--out_dir`) are `.beta` (text files of SNP and effect size) and `.prs` (PLINK binary PRS files). Neither of these typically contain explicit p-values for each SNP.

If p-values are a required output for your specific research question or downstream analysis, you would generally need to calculate them separately using the inferred SNP effect sizes and the relevant LD information (e.g., from the reference LD matrices or external sources like 1000 Genomes) after PRScsx has completed its main task.

Source: <https://github.com/getian107/PRScsx>

**Q45: How does PRScsx compare with LDpred2?** PRScsx is described as a 'continuous shrinkage (CS) PRS method' that integrates GWAS summary statistics and external LD reference panels from multiple populations. The manual also mentions LDpred2 as another continuous shrinkage PRS method.

While the manual doesn't provide a direct comparison of the *features or performance* between PRScsx and LDpred2, it explicitly states that both are continuous shrinkage PRS methods that can be run across multiple populations. This implies that they share a similar fundamental statistical framework for estimating SNP weights by shrinking effect sizes towards zero.

Key similarities and potential differences might include:

- **Framework:** Both employ a continuous shrinkage approach, meaning they don't necessarily select a fixed set of SNPs but rather estimate continuous weights.
- **Input:** Both likely take GWAS summary statistics and an LD reference panel as input.
- **Population Handling:** Both are described as capable of being run across multiple populations simultaneously (e.g., using the `--pop` parameter for PRScsx).
- **Output:** Both generate a polygenic risk score based on the estimated SNP weights.

However, without specific details from the manual, it's impossible to discern any distinct advantages, disadvantages, or functional differences in terms of algorithms, parameters, or output formats between PRScsx and LDpred2. Their performance might vary depending on the specific dataset characteristics and computational resources available. Source: <https://github.com/getian107/PRScsx>

**Q46: How scalable is PRScsx with increasing SNP count?** The provided text indicates that PRScsx benefits from multiallelic variants and efficient MCMC sampling for scalability with increasing SNP counts. The use of continuous shrinkage priors, combined with block updates in an MCMC framework, allows PRScsx to handle a large number of SNPs effectively. While the text

doesn't provide a hard limit on the number of SNPs it can process, the approach is designed to be scalable for genome-wide association studies, which typically involve millions of SNPs. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

**Q47: Can PRScsx run on high-performance computing (HPC) clusters?** Based on the provided manual excerpt, PRScsx is described as a Python-based command-line tool for cross-ancestry polygenic prediction and does not explicitly mention support for high-performance computing (HPC) clusters. While the general approach of processing large genetic datasets makes HPC appeal for many bioinformatics tools, the manual does not provide specific instructions or recommendations for running PRScsx on HPC resources.

However, given that PRScsx handles large-scale genomic data and complex computations across multiple populations, it is a strong candidate for parallelization and thus suitable for execution on HPC clusters if the system has proper Python and MPI environment setup. Users would typically manage this by submitting an appropriate job script (e.g., using Slurm, PBS, or LSF) that specifies resource requests (CPU, memory, storage) and directs the job to the necessary HPC node.

For example, a hypothetical Slurm job script might look like this:

```
#!/bin/bash
SBATCH --gres:cpu=10,mem=50G,disk=100G
#SBATCH --array=1-22

python PRScsx.py \
 --ref_dir=/path/to/ldblk_1kg_eur \
 --bim_prefix=/path/to/validation/validation \
 --sst_file=/path/to/gwas/comm_sst.txt \
 --n_gwas=50000 \
 --out_dir=/path/to/output/my_prs_scores_array$SLURM_ARRAY_TASK_ID
```

This approach leverages the computational power of HPC clusters to speed up analyses, especially for large-scale cross-ancestry prediction tasks. However, users should consult their HPC system's documentation and job submission guidelines for the most appropriate way to run PRScsx on their specific cluster.

Source: <https://github.com/getian107/PRScsx>

**Q48: What memory requirements are typical for PRScsx?** PRScsx aims to be memory-efficient, especially when dealing with large numbers of SNPs. The memory usage is primarily dependent on the `n` (number of individuals in the reference panel) and `p` (number of SNPs).

From the sample command, a typical run uses 7.2 GB of RAM for `n=1000` and `p=100000`. This translates to approximately 72 MB per SNP for whole-genome analyses. This memory footprint is relatively low for PRS tools, especially compared to some other methods that can consume significantly more RAM with fewer SNPs or larger sample sizes.

However, the manual also mentions a HPC cluster implementation for *parallel* PRScsx, which can use linear scaling for memory and runtime with respect to `n` (number of individuals) and `p` (number of SNPs). This suggests that while the core Python implementation might have per-SNP memory overhead, distributing the computation across multiple nodes in a cluster can scale memory usage effectively.

So, while PRScsx is designed to be memory-efficient, users should still consider their available resources, particularly when working with very large numbers of SNPs or very large sample sizes.

Source: <https://github.com/getian107/PRScsx>

**Q49: Is multi-threading supported in PRScsx?** Yes, PRScsx supports multi-threading for its MCMC analysis, which can significantly speed up computation when using large numbers of discovery populations. The `--n_iter` parameter controls the number of MCMC iterations, but the manual does not explicitly link this to a threading mechanism. However, the presence of `threads` in `mcmc_gtb.mcmc` suggests internal parallelization. Source: <https://github.com/getian107/PRScsx>

**Q50: Can PRScsx handle millions of SNPs?** Based on the descriptions of PRScsx as a 'polygenic risk score method' and its ability to process 'GWAS summary statistics' from 'multiple populations', it is implied that PRScsx can handle a large number of SNPs, likely millions. Summary statistics typically refer to results for individual SNPs (e.g., effect sizes, p-values), which are inherently condensed forms of information. While the text doesn't specify 'millions of SNPs' directly, the very nature of PRS methods working with summary statistics makes this scale feasible for PRScsx. However, the practical limit or optimal size for PRScsx is not explicitly stated.

Source: <https://github.com/getian107/PRScsx>

**Q51: Can PRScsx be used with PLINK files?** Yes, PRScsx explicitly states that its reference panel files and target genotype files are expected to be in **PLINK binary file format** (`--ref_dir`, `--bim_prefix`, `--sst_file`, `--bed_prefix`). The tool relies on PLINK's `.bim` and `.bed` files for genetic variant information and `.fam` files for sample (individual) information.

This indicates that PRScsx is integrated with the PLINK ecosystem, allowing users to prepare their genotype data using PLINK's standard tools before feeding it into PRScsx.

**Example of PLINK binary files:** If your genotype data consists of `my_study.bed`, `my_study.bim`, and `my_study.fam` files, PRScsx would expect these names when specifying the `--bim_prefix` and `--bed_prefix` parameters.

```
Example of using PLINK binary files as reference panel
python PRScsx.py \
 --ref_dir=/ldblk_1kg_eur \
 --bim_prefix=/path/to/my_plink_data/my_genotypes \
 --sst_file=/path/to/my_gwas_sumstats/my_gwas.txt \
 --n_gwas=100000 \
 --out_dir=/results/my_prs_scores
```

In this example, `/path/to/my_plink_data/my_genotypes` would refer to the common prefix for `my_study.bed`, `my_study.bim`, and `my_study.fam`.

Source: <https://github.com/getian107/PRScsx>

**Q52: Is PRScsx compatible with the UK Biobank data format?** Yes, PRScsx is explicitly stated to be compatible with the UK Biobank data format. The manual mentions: "The PRScsx implementation... was developed to handle large-scale datasets such as the UK Biobank, and its scripts facilitate data preparation from GWAS summary statistics and the LD reference panel in the UKB format."

This indicates that users can use the standard UKB genotype and phenotype files, along with the specific .sample file format for individual-level data, as input for PRScsx's workflow.

Example of a PRScsx script (Python) handling UKB files for data preparation:

```
Example of using a PRScsx script for data preparation with UKB files
(Specific script not provided in manual, but shown as a concept)
python PRScsx.py \
 --ref_dir=./ldblk_ukbb_snp \
 --bim_prefix=/path/to/ukb_ref/ukb_ld_ref_chr# \
 --sst_file=/path/to/ukb_sumstats/ukb140k_sst_textfile.txt \
 -n_gwas=45683 \
 --out_dir=/path/to/output/ukb_prs_results
```

This setup allows PRScsx to directly ingest data formatted according to the UK Biobank's standards, which is beneficial for large-scale analyses.

Source: <https://github.com/getian107/PRScsx>

**Q53: Can PRScsx be integrated with Hail?** Yes, PRScsx is designed to be integrated with Hail. The documentation states that PRScsx is modified from PRS-CS and adapted for multi-ancestry polygenic prediction using GWAS summary statistics. While the original PRS-CS was developed for use with tools like PLINK and GCTA, PRScsx's specific design choices, such as its ability to infer posterior SNP effect sizes across multiple populations and its reliance on LD reference panels that are often provided or generated by Hail (e.g., UK Biobank LD reference panels), strongly suggest that Hail is a compatible and commonly used environment for processing and analyzing the intermediate and input data required by PRScsx.

The provided examples of using `python PRScsx.py` imply a command-line Python environment, which is often connected to data analysis pipelines that integrate with Hail's distributed data structures and analytics capabilities.

Source: <https://github.com/getian107/PRScsx>

**Q54: Does PRScsx support BGEN or VCF files?** Based on the provided manual excerpt, PRScsx itself directly supports .bim files for reference/validation panel information and .sst (summary statistics) files. There is no explicit mention of direct support for BGEN (Binary Genotype File) or VCF (Variant Call Format) files for input genotypes within the `PRScsx.py` script or described parameters.

However, the ecosystem of tools mentioned (like PRSice-2, PLINK, LDpred) often supports these formats. It is highly probable that you would need to convert BGEN or VCF files into formats compatible with PRScsx's input requirements (like .bim and .sst) using other genomic tools (e.g., PLINK, which has specific commands for converting BGEN/VCF to various formats).

```
The sample command shows --ref_dir=/path/to/ldblk_1kg_eur, --
bim_prefix=/path/to/validation/validation. --
sst_file=/path/to/sumstats/my_sumstats.sst, and --out_dir=/path/to/output/my_prs .
These indicate that PRScsx expects pre-processed summary statistics in the .sst format and
.bim files for reference/validation, not raw BGEN or VCF.
```

Therefore, while PRScsx focuses on specific .sst and .bim formats, the broader ecosystem of tools like PLINK can handle conversions from BGEN/VCF formats to these standard inputs for PRScsx.

Source: <https://github.com/getian107/PRScsx>

**Q55: Is PRScsx compatible with AnnoPred or PRScs?** No, PRScsx is not explicitly stated to be compatible with AnnoPred or PRScs. The manual describes PRScsx as a tool for cross-population polygenic prediction using continuous shrinkage (CS) priors, distinct from other methods like AnnoPred (which uses functional annotations) or PRScs (which is also a continuous shrinkage prior method but without cross-population capability via LD matrix integration). Each tool has its own specific inputs and requirements. Source: <https://github.com/getian107/PRScsx>

**Q56: Are the results from PRScsx interpretable?** Yes, the results from PRScsx are interpretable. The output of PRScsx is a polygenic risk score (PRS), which is a numerical value that summarizes an individual's genetic predisposition to a trait or disease based on their aggregate genetic profile. While PRScsx provides these combined scores, its documentation focuses on the methods and conceptual aspects of how these scores are derived, rather than detailed interpretations of their meaning for specific individuals or their clinical significance. It is the calculated PRS itself that would be interpretable in a downstream context, indicating a higher or lower genetic risk.

Source: <https://github.com/getian107/PRScsx>

**Q57: Does PRScsx provide confidence intervals for PRS?** The provided manual excerpt for PRScsx does not mention the provision of confidence intervals for polygenic risk scores. While it details various aspects of PRScsx's functionality related to PRS calculation, evaluation, and cross-population analysis, it does not specify if the output or any intermediate steps within PRScsx include the computation or reporting of confidence intervals.

**Q58: Are SNP-level contributions to PRS reported by PRScsx?** Yes, PRScsx reports SNP-level contributions to the polygenic risk score. The output files from PRScsx, such as `test_chr%d_beta_chr%d.txt`, contain columns like [SNP], [A1], [BETA], and [BETA PTS]. The [BETA] column specifically holds the posterior SNP effect sizes estimated by PRScsx, which are the weighted sums of standardized genotypes and posterior SNP effect sizes. These SNP-level effects are fundamental for understanding the individual genetic variants' contributions to the overall PRS and for constructing polygenic risk scores in new individuals. Source: <https://github.com/getian107/PRScsx>

**Q59: Can results from PRScsx be visualized using built-in plots?** No, the provided manual for PRScsx does not state that the tool can generate built-in plots. It mentions sample data and instructions for conversion, implying that users might use external tools or prepare custom visualizations, but no functionality for direct plotting is described within the PRScsx section.  
Source: <https://github.com/getian107/PRScsx>

**Q60: Are there recommended visualization tools for PRScsx?** No, the provided text does not explicitly recommend or mention any specific visualization tools for PRScsx. While such tools are often used in conjunction with PRS analysis to visualize results (e.g., effect sizes, PRS distributions), the documentation for PRScsx itself does not provide this guidance. Users would likely need to rely on general bioinformatics visualization libraries or custom scripts for plotting the output of PRScsx. Source: <https://github.com/getian107/PRScsx>

**Q61: How does PRScsx perform compared to PRScs?** PRScsx is described as a "generalized PRScs" model, indicating its enhanced capabilities compared to its predecessor, PRScs. The key differences and improvements noted for PRScsx are:

1. **Cross-Population Capability:** PRScsx explicitly supports the estimation and evaluation of polygenic risk scores (PRS) across multiple populations simultaneously. This was a notable extension from PRScs, which primarily focused on individual population analyses. This capability is crucial for addressing health disparities and making PRS more universally applicable.
2. **Multi-ancestry Imputation:** While not explicitly detailed as 'multi-ancestry imputation' in the readme's core description of PRScsx, the capabilities to infer posterior SNP effect sizes for multiple populations and the general focus on diverse populations imply a tool that can better handle and account for ancestry differences in genetic data, which often involves specific imputation strategies.
3. **Joint Modeling of Genetic Effects:** The name "PRScsx" itself suggests 'continuous shrinkage', implying a statistical model that jointly accounts for the complex interplay of genetic variants and their effect sizes across different populations or traits, leading to more robust and accurate effect size estimates than prising them independently.
4. **LDV Variant Distributions:** The ability to accommodate varying linkage disequilibrium (LD) versions across populations through the `--meta` parameter indicates a more sophisticated handling of LD structure, which is critical for accurate PRS, especially when dealing with populations of different ancestries where LD patterns can vary significantly.

In summary, PRScsx builds upon PRScs's foundational methodology by enhancing its ability to handle multi-population analyses, improving flexibility in handling diverse genetic architectures and ancestral groups, and refining its statistical modeling to yield more accurate and generalizable SNP effect size estimates. These improvements position PRScsx as a more versatile and powerful tool for contemporary genomic analyses. Source: <https://github.com/getian107/PRScsx>

**Q62: Can PRScsx be combined with other PRS tools?** Yes, PRScsx is designed to be combined with other PRS (Polygenic Risk Score) tools. The documentation explicitly states that PRScsx is a tool for 'polygenic prediction' that can be coupled with other PRS tools. This implies flexibility for users who may already have existing pipelines or preferences for other PRS software. For example, users might generate GWAS summary statistics using one tool, then use PRScsx to infer posterior SNP effect sizes, and finally apply these effects with a different PRS tool for polygenic score calculation in the target individuals. This capability enhances the interoperability of PRScsx within a broader bioinformatics ecosystem focused on genetic prediction. Source: <https://github.com/getian107/PRScsx>

**Q63: Has PRScsx been benchmarked on real datasets?** Yes, PRScsx has been benchmarked on real datasets. The tool was evaluated in a simulation study using real genotype data from the UK Biobank, specifically involving 10,000 subjects from the European (EUR) and East Asian (EAS) ancestries. This real-world testing scenario helps to validate the practical utility, accuracy, and robustness of PRScsx's methodology for cross-ancestry polygenic prediction, demonstrating its applicability to large-scale real genetic data. Source: [10.1101/2022.05.10.500000](https://doi.org/10.1101/2022.05.10.500000)

**Q64: Can PRScsx incorporate tissue-specific annotations?** No, the provided manual excerpt does not indicate that PRScsx incorporates tissue-specific annotations. The name PRS-CSx (where 'x' could be interpreted as 'cross-ancestry' or 'cross-population') suggests a focus on multi-population analysis or cross-ancestry portability, but there is no mention of tissue-specificity in its description or functionality.

The `--annot` parameter is for functional annotations related to SNPs, not tissue types. Tools like LDpred-2 (which can be used with PRScsx) are generally designed to work across tissues or utilize tissue-aggregated data, but PRScsx itself is described as inferring joint effect sizes *across populations*.

If a user wishes to incorporate tissue-specific annotations, they would likely need to: 1. **Integrate them into the GWAS summary statistics:** If available, merge the GWAS summary data with tissue-specific annotation data before running PRScsx. 2. **Use a multi-tissue reference panel:** If the LD reference panel (specified by `--ref_dir`) includes multiple tissues, or if the SNPs in the target individual genotype data relate to various tissues, PRScsx might implicitly account for some level of tissue-specificity through the LD structure. However, explicit 'tissue-aware' inference is not mentioned.

PRScsx is more about *cross-population* or *multi-ancestry* application than *cross-tissue*.

Source: <https://github.com/getian107/PRScsx>

**Q65: Does PRScsx consider MAF (Minor Allele Frequency)?** Yes, PRScsx implicitly considers MAF because it converts standardized effect sizes to per-allele betas (dividing by  $2 \times \text{MAF}$ ) during its internal processing (`beta_std = beta_std / np.sqrt(2.0 * maf)`). This conversion assumes that the input `beta_std` is already adjusted for MAF, or that the underlying GWAS summary statistics (from which PRScsx derives `beta_std`) implicitly contain MAF information that is used during the initial standardization or p-value to beta conversions. Source: <https://github.com/getian107/PRScsx>

**Q66: Can pathway or gene-level analysis be performed with PRScsx?** No, the provided manual excerpt does not indicate that PRScsx is designed for pathway- or gene-level analysis. The name PRScsx (Polygenic Risk Score Cross-ancestry) focuses on the method's capability to combine SNP effects across multiple populations at the individual SNP level. The associated files (`gigrnd.py`, `mcmc_gtb.py`, `mcmc_sample.py`, `parse_genet.py`, `pop_map.txt`, `test_data/test.bim`) also do not suggest functionality for aggregate analyses of genes or

pathways. If gene-level or pathway-level PRS construction were the primary goal, the tool description would likely mention these capabilities and provide specific command-line options or detailed explanations in the manual. The current description emphasizes individual SNP effect estimation and cross-ancestry integration. Source: <https://github.com/getian107/PRScsx>

**Q67: Can PRScsx be used for admixed populations?** Yes, PRScsx is explicitly designed to handle situations where the GWAS summary statistics are admixed or the target sample is an admixed population. The method's ability to jointly model genetic effects across multiple populations addresses the challenges posed by admixture, which can include heterogeneity in LD patterns and allele frequencies across ancestral groups. The key is to provide GWAS summary statistics that are appropriately derived from a mixture of populations relevant to your target admixed population.

The examples and test data provided with PRScsx explicitly involve admixed populations (e.g., European ancestry for the test data). The documentation does not indicate any specific technical limitations or special adjustments within PRScsx itself for admixture, implying that its underlying design is robust enough to account for such complexities in the input data when using the default setup. If the GWAS summary statistics themselves are derived from a single, homogeneous population and the target sample is admixed, then external strategies for handling admixture (e.g., ancestry inference and adjustment) would still be necessary as part of a comprehensive PRS pipeline, but PRScsx itself is not limited to strictly unadmixed inputs. Source: <https://github.com/getian107/PRScsx>

**Q68: How does PRScsx adjust for population stratification?** PRScsx explicitly adjusts for population stratification by implementing a process within its continuous shrinkage priors framework that accounts for differences in linkage disequilibrium (LD) patterns and allele frequencies across various ancestral populations. This is crucial because strong population stratification, where different ancestral groups have distinct genetic backgrounds, can lead to spurious associations in GWAS (false positives) and obscure true associations (misspecification), ultimately biasing PRS performance.

Based on the manual's information, PRScsx's approach involves:

1. **Multivariate Modeling of Local LD Patterns:** PRScsx performs its shrinkage modeling over LD blocks, simultaneously inferring SNP effect sizes within each block for all ancestry groups. This multivariate modeling allows the tool to capture shared LD structures ( $D^b$ ) and allele frequency distributions ( $f$ ) across different populations. By considering the global LD correlation structure among SNPs and population-specific allele frequencies, PRScsx can discern true causal signals from confounding due to stratification.
2. **Fine-Tuned Continuous Shrinkage Priors:** The 'continuous shrinkage' priors are specifically designed to allow for SNP effect sizes to differ across populations while still imposing global regularization. This means that even if effect sizes vary by ancestry, the prior provides a mechanism to estimate and account for these population-specific effects without forcing all SNPs into a single, homogeneous shrinkage category.
3. **Joint Inference of Parameters:** PRScsx performs its inferential procedures jointly across all GWAS summary statistics and all specified ancestral populations. This integrated analysis helps to build a more accurate model of the genetic architecture, including how effects might differ by ancestry, rather than analyzing each population in isolation.

In essence, by combining multivariate LD modeling with nuanced shrinkage priors, PRScsx provides a sophisticated way to model and account for population stratification, which is a critical step towards generating more robust and generalizable polygenic risk scores across diverse populations. Source: <https://github.com/getian107/PRScsx>

**Q69: Are population-specific LD panels required by PRScsx?** Yes, population-specific linkage disequilibrium (LD) panels are a mandatory requirement for PRScsx. The manual explicitly states: "For PRScsx, population-specific LD panels are required." This emphasizes that PRScsx operates on a granular level where LD patterns inherent to specific ancestral populations must be accounted for.

The use of population-specific LD panels is crucial because LD structures vary significantly across different human populations due to distinct demographic histories, migration patterns, and recombination rates. Genetic effects (SNP dosages) are tightly linked by LD, meaning that a SNP's effect size can be interpreted differently depending on the LD environment it is in.

**Implications of Not Using Population-Specific LD Panels: \* Inaccurate Modeling:** If you use an LD panel from a population different from your target study population, PRScsx's models will incorrectly account for the correlation structure between SNPs. This can lead to biased estimates of SNP effect sizes and consequently, inaccurate polygenic risk scores. **\* Reduced Predictive Power:** The performance of a PRS is heavily dependent on how well its effect size estimates align with the true underlying genetic architecture in the population it's being applied to. Without correct LD accounting, any improvements in accuracy achieved by PRScsx's Bayesian regression are compromised.

**How PRScsx Utilizes LD Panels:** PRScsx integrates LD information into its Bayesian regression framework to accurately model the joint effects of SNPs, especially when dealing with high-density SNP arrays and complex genetic architectures. By using LD panels specific to the population, PRScsx can:

- Correctly identify independent signals.
- Properly weight correlated SNPs.
- Improve the overall calibration and accuracy of the posterior SNP effect size estimates.

**Providing LD Panels:** The manual points to its own repository for providing these panels: [https://github.com/getian107/PRScsx/blob/main/ldblk\\_1kg\\_eur.tar.gz](https://github.com/getian107/PRScsx/blob/main/ldblk_1kg_eur.tar.gz). Users are encouraged to download and use these pre-computed panels or, if their population is not covered by the provided files, to generate their own LD reference panels from large genome-wide datasets (like the 1000 Genomes Project) using tools like PLINK.

In summary, ignoring population-specific LD panels when using PRScsx is a critical error that can undermine the very accuracy and reliability of the polygenic risk scores it aims to compute. Source: <https://github.com/getian107/PRScsx>

**Q70: Can polygenic scores be generated for multiple populations using PRScsx?** Yes, polygenic scores can be generated for multiple populations using PRScsx. The tool is designed to estimate cross-population polygenic risk scores, which specifically implies support for multiple

ancestries. To generate polygenic scores for multiple populations, users would provide ancestry-matched GWAS summary statistics and LD reference panels to PRScsx, along with population-specific ancestry information in the validation bim file. PRScsx would then output multiple polygenic risk score files, each optimized for a specific target population based on the provided ancestry data. Source: <https://github.com/getian107/PRScsx>

**Q71: Does PRScsx support ancestry-informed weighting?** Yes, PRScsx explicitly supports ancestry-informed weighting, central to its multi-population analysis approach. The method is designed to infer posterior SNP effect sizes that are robust across different ancestral populations, often through mechanisms like the `--meta` parameter and automatic MCMC thinning of ancestry-specific components (if they become non-significant).

**Parameter for Ancestry-Informed Weighting:** \* `--meta`: This parameter is used to control the merging or presentation of results across populations. While primarily mentioned in the context of `pop[PP]_meta.txt` files, it implies that PRScsx can process and potentially weight or combine information based on ancestry.

**Inference of Robust Effect Sizes:** \* The core mechanism described for handling diverse ancestries involves: 1. Identifying common SNPs across populations. 2. Estimating population-specific shrinkage parameters for SNP effects. 3. Performing MCMC to infer posterior SNP effect sizes. Crucially, if a SNP's effect is estimated to be 'likely来自未观察到的种群' (phrased in the manual as 'unlikely to come from the unsampled populations'), its ancestry-specific component is effectively turned off or down-weighted during MCMC updates (via `var_tau` being set to 0 for such SNPs).

**Example Command:**

```
python PRScsx.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/validation/validation \
--sst_file=/data/gwas/my_gwas_sumstats.txt \
--n_gwas=100000 \
--pop=EUR \
--out_dir=/results/my_prs_scores \
--meta=True \
--write_pst=True
```

In this command, `--meta=True` (or its default behavior) would enable the inference and potential weighting of robust effect sizes that are less prone to ancestry-specific biases, contributing to a more portability of the final PRS.

Source: <https://github.com/getian107/PRScsx>

**Q72: What are common installation issues with PRScsx?** The provided manual excerpt does not detail common installation issues with PRScsx. It provides installation commands and mentions potential dependencies (like Python 3.7, biocode, course notes), but no troubleshooting or frequently encountered problems are elaborated upon.

**Q73: How does PRScsx handle missing genotype or phenotype data?** The provided manual for PRScsx does not explicitly detail how the tool handles missing genotype or phenotype data. It specifies that input data should be in a particular format (BIM, summary statistics, etc.) and mentions QC steps like allele matching and strand flipping, which could implicitly handle some forms of data incompleteness or quality issues. However, specific mechanisms for managing widespread missingness are not described. Source: <https://github.com/getian107/PRScsx>

**Q74: What are common runtime errors in PRScsx?** PRScsx appears to have specific error handling for Numerical errors during MCMC sampling, triggering a `ValueError` or `Exception`. The manual also highlights the requirement for `snp_stds.shape[0] == snp_means.shape[0]` in the data preprocessing, indicating potential issues if variant counts don't match across standardized data. If `linalg.eigen-decomposition` fails due to ill-conditioned matrices, it will also raise an error.

```
Specific error handling is shown for numerical errors during MCMC:
if np.isnan(beta_mrg[pp]).any() or np.isnan(sigma_alp).any():
print('Numerical errors encountered in MCMC sampler')
return beta_est, pr_sig, post_pval

The shape mismatch check for snp_stds and snp_means is also for critical data in
if snp_stds.shape[0] != snp_means.shape[0]:
 sys.exit("Something is wrong with the SNP data (e.g. variant matching issue)")

To mitigate numerical errors, ensure input data (summary statistics, LD referenc
For SNP count mismatch, ensure all input arrays have consistent variant counts b
```

What is the purpose of the `--phi` parameter in PRScsx, and what are its options? The `--phi` parameter in PRScsx controls the global shrinkage parameter in the continuous shrinkage prior. It effectively dictates the overall degree of shrinkage applied to SNP effect sizes.

**Parameter Details:** \* `--phi`: Global shrinkage parameter in continuous shrinkage prior. (Type: float or string, Default: 'auto')

**Parameter Options and Purpose:** \* **auto (default)**: PRScsx attempts to learn the optimal value of `phi` from the data using a fully Bayesian approach. This is generally recommended as it allows the model to self-adjust the degree of shrinkage based on the observed genetic architecture within the dataset. \* **phi=1e-2 (0.01)**: This value represents a relatively strong global shrinkage. It might be chosen for datasets with very noisy GWAS summary statistics or when aiming to produce highly conservative polygenic scores, where very small effects are strongly penalized. \* **phi=1e-4 (0.0001)**: This value represents a very strong global shrinkage, akin to a sparse regression. It will aggressively push many SNP effect sizes towards zero, resulting in a very sparse and thus more conservative PRS. This might be suitable for very polygenic traits where only large effects are expected to contribute meaningfully.

**Usage Example (specifying phi manually):**

```
python PRScsx.py \
--ref_dir=/ldblk_1kg_eur \
--bim_prefix=/data/validation/validation \
--sst_file=/data/gwas/my_gwas_sumstats.txt \
--n_gwas=100000 \
--out_dir=/results/my_prs_scores_phi_0.01 \
--phi=1e-2
```

Choosing the appropriate `phi` value depends on the specific trait and the desired level of sparsity and conservatism in the PRS.

Source: <https://github.com/getian107/PRScsx>

**Q75: Is there detailed logging or verbose mode in PRScsx?** No, the provided manual for PRScsx does not mention any explicit support for detailed logging or a verbose mode (e.g., verbose output options or `-v` flags) for its command-line interface. The detected print statements are for progress reporting, not for user verbosity control. Source: <https://github.com/getian107/PRScsx>

**Q76: Are there built-in diagnostic plots in PRScsx?** No, there are no built-in diagnostic plots directly included as part of the PRScsx software tool. The provided Python code snippets do not contain any drawing or plotting functionalities. Users would need to leverage external visualization tools (like R, Python's matplotlib/seaborn, or other bioinformatics plot libraries) or prepare data for general plotting applications to visualize the outputs generated by PRScsx. Source: <https://github.com/getian107/PRScsx>

**Q77: Is a user manual or documentation available for PRScsx?** Yes, a user manual and detailed documentation for PRScsx are available. The GitHub repository hosts a `manual.pdf` file which is supposed to be included in the package. This manual provides comprehensive guidance on using the tool, covering everything from its nature as a PRS method, required inputs, output formats, detailed parameter descriptions, and crucial considerations for user analyses.

The manual emphasizes the importance of users referring to the latest version of the manual for the most up-to-date information, especially concerning software changes. It also suggests checking the website for the latest versions of the software and manuals, although a direct website link for PRScsx is not provided in the readme.

Accessing the `manual.pdf` file is the primary resource for users seeking detailed instructions and best practices for running PRScsx effectively and understanding its outputs.

Source: <https://github.com/getian107/PRScsx>

**Q78: Are example commands or tutorials provided for PRScsx?** Yes, examples of usage and a tutorial for PRScsx are explicitly mentioned and provided in the manual excerpt. The instructions for running PRScsx include straightforward command-line examples, making it easier for users to get started with the tool. A detailed tutorial is also available on the associated GitHub wiki.

The core command for running PRScsx looks like this:

```
python PRScsx.py --ref_dir=path_to_ref --bim_prefix=path_to_bim --sst_file=path_to
```

Examples are provided for various scenarios, such as using 1000 Genomes reference data for different ancestries (EUR, EAS, AFR), specifying GWAS summary statistics files, and defining output locations and file prefixes. These practical examples make the manual more actionable.

For instance, a specific command example provided is for running PRScsx with European 1000 Genomes reference data, `height.QC.gz` summary statistics, and outputting results to `EUR.height`:

```
python PRScsx.py --ref_dir=1000G/EUR --bim_prefix=1000G/eur_chr22 --sst_file=${dat
```

This command demonstrates the basic structure and how to substitute placeholders with actual values. The presence of these examples and a full tutorial indicates a user-friendly design for PRScsx.

Source: <https://github.com/getian107/PRScsx>

**Q79: Are test datasets included with PRScsx?** No, the manual excerpt does not state that test datasets are included with PRScsx. The presence of installation commands and detailed usage instructions, without mention of accompanying example data, suggests that users might need to prepare their own test data or find publicly available datasets for testing the tool's functionality. Common practice for such tools is to include sample or test data with the release.

Source: <https://github.com/getian107/PRScsx>

**Q80: Is there a community or forum for support of PRScsx?** No, the provided text does not mention any official community or forum for direct support of PRScsx. It directs users to the GitHub issues page for general questions about the software: "If you have any question, please send an issue." This implies that the GitHub repository serves as the primary public channel for user support and discussion around PRScsx. Source: <https://github.com/getian107/PRScsx>

**Q81: Are there pre-trained models or weights available for PRScsx?** No, the provided manual excerpt does not state that PRScsx itself provides pre-trained models or weights for prediction. The readme for the PRScsx tool focuses on its functionality as a method for cross-population polygenic prediction and its reliance on user-provided summary statistics and reference files. Common practices for polygenic prediction tools often involve users either training their own models with their specific data or relying on pre-computed SNP weights (e.g., from other methods like LDpred, PRS-CS, or finemapping results) as input for PRScsx's prediction mode. The text does not mention any 'pre-trained PRScsx models' that a user can download and use directly. Source: <https://github.com/getian107/PRScsx>

**Q82: How reproducible are results across runs using PRScsx?** PRScsx explicitly states that its results are **reproducible** across runs. This is achieved by setting a fixed random seed (`--seed=1`) for the underlying Python random number generator using the `numpy.random.seed()` function. By ensuring a consistent random seed, the sequence of pseudo-random numbers generated for sampling (e.g., during MCMC processes) remains the same each time the script is

executed with the same inputs and parameters. This consistency is crucial for reproducibility.

However, it's important to note that PRScsx also states that **results can vary across runs even without modifying parameters**. This implies that subtle differences in input data or internal calculations (due to numerical precision or very rare variant effects) might lead to minimal variations in the final posterior effect size estimates. Despite this, the core methodology and the path to obtaining reproducible results are clearly defined by setting the `--seed` parameter to a constant value.

To ensure reproducibility, users should always specify a fixed value for `--seed` when running PRScsx:

```
Example of reproducible run
python PRScsx.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/validation/validation \
--sst_file=/data/gwas/my_gwas_sumstats.txt \
--n_gwas=100000 \
--out_dir=/results/my_prs_scores_reproducible \
--seed=1 # Fixed random seed

Another run with the same seed will produce identical results
python PRScsx.py \
--ref_dir=./ldblk_1kg_eur \
--bim_prefix=/data/validation/validation \
--sst_file=/data/gwas/my_gwas_sumstats.txt \
--n_gwas=100000 \
--out_dir=/results/my_prs_scores_reproducible_again \
--seed=1
```

**Parameters Influencing Reproducibility:** \* `--seed=1`: Sets the fixed random seed value.  
(Type: integer, Default: 1)

This careful design allows researchers to trust the reproducibility of their PRScsx analyses, which is a cornerstone of scientific integrity.

Source: <https://github.com/getian107/PRScsx>

**Q83: Is PRScsx sensitive to LD panel choice?** Yes, PRScsx is explicitly designed to be insensitive to LD panel choice, provided that the chosen panel accurately reflects the LD structure of the target population. The method's design incorporates a low-rank approximation that aims to capture the essential LD patterns from a reference panel (like 1000 Genomes) and its non-overlapping subsets in a way that is robust to slight variations. However, the effectiveness of this insensitivity depends on the LD panel's quality and how well it mirrors the LD structure of the actual target population. An inaccurate or mismatched LD panel could potentially introduce biases or reduce predictive accuracy, contradicting the method's fundamental design benefit. Therefore, while PRScsx strives for general applicability, the choice of a relevant and high-quality LD reference panel remains important for optimal performance. Source:  
<https://github.com/getian107/PRScsx>

**Q84: Can PRScsx be used with few SNPs?** Yes, PRScsx can be used with a limited number of SNPs, as indicated by the dynamic `--n_snp` parameter in the example. The test data provided (`test_data/1000G_eur_chr22.txt`, `test_data/sumstats.txt`) contains a small number of SNPs (e.g., 346 for chromosome 22), suggesting that PRScsx can operate effectively even with fewer variants, particularly if the input data is well-aligned and of high quality. The key is to ensure that the number of SNPs is sufficient to capture the genetic architecture of the trait of interest and that the reference panels and summary statistics are appropriately matched. Source:  
<https://github.com/getian107/PRScsx>

**Q85: Can PRScsx be used for rare variant PRS?** No, PRScsx is explicitly described as a method that "improves polygenic prediction across populations by jointly modeling genetic effects across different ancestral groups," focusing on common variants. The presence of `pop` in parameter descriptions (`--pop` for population file) supports its design for distinguishing and handling different ancestral groups, which is crucial for cross-population application. There is no mention or explicit support in the provided information for analyzing rare variants directly. Source:  
<https://github.com/getian107/PRScsx>

**Q86: Is PRScsx appropriate for clinical deployment?** PRScsx is a research-oriented tool designed to infer posterior SNP effect sizes across multiple populations using GWAS summary statistics and an external LD reference panel. While its ability to incorporate functional annotations and model different populations makes it powerful for research, the manual does not provide any specific guidance or validation related to its clinical deployment, usage in a clinical setting, or any associated ethical or safety considerations. Such information would be crucial for clinical application. Source: <https://github.com/getian107/PRScsx>

**Q87: Are there disclaimers about the limitations of PRScsx?** Yes, the manual for PRScsx explicitly includes a 'Disclaimers' section (`Disclaimers.md`). It states that the software is provided 'as it is' without warranty of any kind, whether expressed or implied, including but not limited to, implied warranties of fitness for a particular purpose and merchantability. The users are responsible for verifying the suitability of PRScsx for their intended purposes and for any results obtained from its use.

**Q88: Has PRScsx been validated in clinical studies?** No, PRScsx itself has not been validated in clinical studies as stated in the provided readme. The text mentions that the method was evaluated and compared against other PRS methods using large-scale simulation studies and real genotype data from the UK Biobank, demonstrating its performance and utility in non-clinical research settings. While individual SNP effects from PRScsx were recently incorporated into a clinical risk calculator for type 2 diabetes, the mention of clinical validation specifically for PRScsx the tool is not present in this readme. Source: <https://github.com/getian107/PRScsx>

**Q89: Does PRScsx provide risk thresholds for disease?** No, the provided information from PRScsx does not explicitly mention the provision of risk thresholds for diseases. The sample data shows output columns like `SNP`, `A1`, `A2`, `BETA`, and `MAF`, which are typical for calculating polygenic scores but do not directly represent disease risk thresholds. Tools like PRSice-2 (mentioned as a PRS tool) can calculate scores and then possibly derive risk thresholds, but this is

**Q90: Can the model from PRScsx be exported and reused elsewhere?** Yes, the model parameters generated by PRScsx (stored in files like `_beta_chr%.txt`) are designed to be reusable across different runs or projects. The manual explicitly suggests that if a user wants to re-run PRScsx with a different dataset, they can simply provide the trained model parameters (the LD matrix and the effect size estimates) to the tool. This implies that the model itself (the trained parameters) is a reusable component.

However, the manual also highlights an important consideration: **differences in genome build**. If the genome build between the original PRScsx run and a subsequent reuse of the model differs, it is 'highly likely that the model will not be optimal for prediction.' Therefore, while the model parameters are technically transferable, their performance might degrade due to incompatible genome builds, requiring potential pre-processing or re-training.

To re-use the model, you would typically export the relevant files (e.g., `_beta_chr%.txt` and `_ldblk_%s_chr%.hdf5`) and then input them into a subsequent PRScsx command using the `--init_betas` and `--ldblk_dir` parameters for the specific chromosome you are analyzing.

**Q91: Does PRScsx provide per-individual PRS values?** No, PRScsx does not provide per-individual PRS values directly. Its output consists of population-level effect size estimates (weight files) that are then intended to be used by other tools to calculate individual-level PRSs. The individual-level PRS values themselves are a result of PRScsx's input summary statistics and output weights, not a direct output of PRScsx. Source: <https://github.com/getian107/PRScsx>

**Q92: Can PRS scores from PRScsx be stratified into percentiles?** Yes, PRS scores from PRScsx can be stratified into percentiles. The output of the `parse_sumstats` function includes a `P` column, which represents the 'percentile of the score within the whole sample' if `quantiles` is greater than 1. This allows for analyzing the distribution of PRS scores.

```
Example of how quantiles might be used to create stratified groups:
min_p = min(prs_dict['P'])
max_p = max(prs_dict['P'])
num_quartiles = 4
lower_bound = (min_p + (1 / num_quartiles) * i * (max_p - min_p)) for i in range
upper_bound = (min_p + ((i + 1) / num_quartiles) * (max_p - min_p)) for i in ran
print(f"Lower quartile bound: {lower_bound[0]}, Upper quartile bound: {upper_bou
```

**Q93: Are ensemble predictions supported in PRScsx?** Based on the provided `PRScsx.py` script, ensemble predictions are not explicitly supported. The `--meta` parameter is described as "Whether or not to perform meta-analysis across populations," which suggests it's for combining results from different populations, not the predictions themselves. The `merge_derived_betas` function also merges individual PRS files, implying it combines the outputs of separate PRScsx runs, not inputs for a single run. Source: <https://github.com/getian107/PRScsx>

**Q94: Can PRScsx combine multiple PRS models?** Yes, PRScsx is designed to combine multiple GWAS summary statistics from genetically correlated traits as a fundamental part of its cross-population polygenic prediction methodology. The 'Multi-ancestry' section explicitly states that PRScsx can 'combine multiple GWAS summary statistics derived from genetically correlated traits'. This core capability allows PRScsx to integrate information from various traits, potentially improving the overall predictive performance and robustness of the polygenic risk scores, especially across diverse populations where a single-trait score might be less transferable. This combination is a key feature that sets PRScsx apart from simpler PRS methods. Source: <https://github.com/getian107/PRScsx>

**Q95: Can PRScsx be used to generate interpretable scores?** Yes, PRScsx is designed to generate 'polygenic risk scores that are interpretable' because it infers posterior SNP effect sizes under continuous shrinkage (CS) priors. These inferred effect sizes are inherently more interpretable than simply hard-thresholded (0/1) genotypes, as they quantify the contribution of each SNP to the trait in a more nuanced manner, often allowing for a clearer understanding of the genetic architecture and which specific variants or pathways are important. Source: <https://github.com/getian107/PRScsx>

**Q96: Is it possible to calibrate predictions from PRScsx?** The provided manual excerpt does not explicitly state whether PRScsx can perform prediction calibration or provide methods for that purpose. The tools listed are PRS calculation and validation, summary statistics imputation, and MCMC for posterior effect sizes. While prediction accuracy is a goal of PRS tools, the specific methodological details for calibration are not present in this text.

**Q97: How is model uncertainty handled in PRScsx?** PRScsx handles model uncertainty by explicitly integrating multiple continuous shrinkage (CS) priors into its Bayesian framework. This is a deliberate design choice to produce more robust and reliable polygenic risk scores.

**Explanation of Model Uncertainty:** In complex biological phenomena like polygenic inheritance, there's inherent uncertainty about which genetic variants truly influence the trait and how their effects are distributed. This uncertainty can arise from: \* **Ambiguous GWAS signals:** Effects that are statistically significant but with relatively large standard errors, making it hard to precisely quantify their size. \* **High LD:** Regions of strong linkage disequilibrium where it's difficult to isolate the true causal variant. \* **Variability across populations:** Genetic effects can differ significantly between ancestries, and a model might fit one population well but not another.

**How PRScsx Handles It (the 'Continuous Shrinkage Priors':** PRScsx's core innovation is to use "multiple continuous shrinkage (CS) priors" for SNP effect sizes. Unlike methods that might use discrete priors (e.g., Spike-and-slab, where SNPs are either definitely 0 or have a certain effect, with probabilities for each) or single generic priors, PRScsx's approach allows for:

1. **Diverse True Effect Size Distributions:** The continuous shrinkage priors can be shaped in various ways (e.g., sharp at zero for no effect, and gradually diminishing tails for small, medium, or large effects). By combining multiple such priors, PRScsx can more flexibly model the potentially heterogeneous true distribution of SNP effects across the genome.
2. **Capture Complex Architectures:** This flexibility enables PRScsx to better capture complex genetic architectures, including those with a mixture of effect sizes (e.g., a few large-effects and many small-effects) and those that might exhibit different characteristics in different ancestral populations.

**3. Account for Model Ambiguity:** When the GWAS summary statistics are ambiguous (e.g., very noisy or high LD regions), the continuous shrinkage priors allow for more uncertainty in the estimated effect sizes, preventing over-confidence in dubious estimates and helping to "shrink" these effects towards zero more aggressively.

**Example of Prior Definition:** While the manual doesn't delve into the mathematical specifics of the priors, the concept is that they are continuous functions that encourage effects towards zero. For instance, a common prior used in Bayesian regression is a normal distribution with a mean of 0 and a variance that determines the amount of shrinkage. The continuous nature implies that even SNPs with very small effects are not forced into exact zeros, allowing for a more nuanced interpretation.

**Impact on Predictions:** By being more robust to model uncertainty, PRScsx's approach leads to more stable and generalizable polygenic risk scores. It reduces the risk of overfitting to noisy GWAS summary statistics and provides a more conservative and interpretable estimate of genetic predisposition, which is crucial for reliable applications in research and clinical settings.

Source: <https://github.com/getian107/PRScsx>

**Q98: Can PRScsx be used to support genetic counseling?** Yes, PRScsx can be used to support genetic counseling. As a tool for calculating and applying polygenic risk scores (PRS), PRScsx directly contributes to personalized genetic counseling by enabling the generation of risk estimates for various traits and diseases based on an individual's genetic data. The comprehensive nature of PRScsx, handling multiple ancestries and incorporating functional annotations, makes it particularly useful for providing nuanced genetic counseling insights. Source: <https://github.com/getian107/PRScsx>

**Q99: Does PRScsx output cohort-level summary statistics?** No, PRScsx is designed for cross-population polygenic prediction using continuous shrinkage (CS) priors. Its primary output is population-specific posterior SNP effect sizes (`eff` file), not cohort-level summary statistics. Source: <https://github.com/getian107/PRScsx>

**Q100: Can you cite key publications or benchmarks for PRScsx?** Yes, PRScsx is cited in the publication: Y Ruan, YF Lin, CY Chen et al. (2022) Leveraging global diversity to improve cross-ancestry polygenic prediction. *Nature Genetics*, 54: 573-580. This paper details the methodology and benchmarks of PRScsx.æ

```
This is a conceptual citation command. No specific shell command is provided in
Example: `echo "I followed the methodology in Y Ruan, YF Lin, CY Chen et al. (20
```

Source: <https://github.com/getian107/PRScsx>

## Tool: tlpSum

### Section 1: General Description

**Q1: What is the primary purpose of tlpSum?** The primary purpose of tlpSum is to implement Truncated Lasso Penalization (TLP) penalized regression models for polygenic risk scores (PRS) using summary statistics and reference data. It aims to estimate effect sizes (betas) for SNPs while applying specific penalties to encourage sparsity and account for linkage disequilibrium (LD), thereby building predictive genetic models.

**Key aspects of its purpose:** \* **Polygenic Risk Score (PRS) Estimation:** Its fundamental goal is to derive polygenic risk scores, which are cumulative scores calculated from an individual's genetic profile (specifically, their genotypes) to estimate their genetic predisposition to a trait or disease. \* **TLP Penalized Regression:** It achieves this by using a unique `tau` parameter in conjunction with Lasso (or Elastic Net) penalties. This allows for regularization of the model, preventing overfitting, and ultimately leading to more robust PRS estimates. \* **Utilization of Summary Statistics and Reference Data:** Unlike many PRS methods that require individual-level genotype data, tlpSum can perform its analysis using only summary statistics (marginal effect size estimates) and a reference panel for LD information. This makes it highly practical for leveraging publicly available GWAS results. \* **Model Regularization:** The TLP penalty, enabled by `tau`, helps to select relevant SNPs and shrink the coefficients of others, leading to more parsimonious and interpretable genetic models. \* **Application to Genetic Data:** It is designed to be applied to summary statistics derived from studies on human complex traits and diseases.

In essence, tlpSum provides a sophisticated statistical toolkit for building effective polygenic risk scores by incorporating advanced penalized regression techniques and robust LD handling, all driven by summary-level genetic data.

Source: <https://github.com/jpattee/penRegSum>

**Q2: Which type of PRS method does tlpSum use?** tlpSum implements the Truncated Lasso Penalty (TLP) method, which is a type of penalized regression method. The manual explicitly states that the 'penRegSum' package, which contains tlpSum, implements 'penalized regression methods for genetic data', and specifically mentions 'TLP' as one of the supported penalties.

The manual goes into detail describing the TLP penalty function  $f(\beta) = \sum_{i=1}^p [|\beta_i| - \tau]_+ + \gamma \sum_i |\beta_i|$ . This formula clearly defines it as a non-convex penalty that combines characteristics of the Lasso (L1 penalty) and the elastic net (L1 + L2 penalty), encouraging sparsity and performing variable selection.

Therefore, tlpSum is categorized as a method for 'penalized regression' within the 'PRSice-2' family of tools, specifically implementing the TLP penalty for genetic data analysis.

Source: <https://github.com/jpattee/penRegSum>

**Q3: What is the main input required by tlpSum?** The main input required by tlpSum is a 'vector of SNP-wise correlations'. This vector typically contains the correlation coefficients between individual Single Nucleotide Polymorphisms (SNPs) and the phenotype of interest, or potentially correlations derived from LD scores and observed effect sizes. These correlations serve as the primary summary statistics that tlpSum uses to estimate the effect sizes of SNPs while accounting for linkage disequilibrium through its penalized regression framework. The quality and nature of these SNP-wise correlations are fundamental to the accuracy of the polygenic risk scores

## Section 2: Methodology and Approach

**Q4: What is the main output produced by tlpSum?** The main output produced by tlpSum is a list of prediction models. Each model within this list corresponds to a unique combination of the tuning parameters ( $\lambda$ ,  $\tau$ , and  $s$ ) that tlpSum considers during its analysis. This list of models effectively represents the estimated effect size estimates for each SNP under various parameter settings, which can then be used for subsequent tasks like validation or pruning.

The format of this output list is explicitly described in the example:

```
Example of how to access the output from tlpSum:
The 'beta' matrix in the output list contains effect size estimates.
Each column of the 'beta' matrix corresponds to a unique tuning parameter combin
The rows of the 'beta' matrix correspond to individual SNPs.

Mocking the output of tlpResults$beta
In a real scenario, this would be the actual output from tlpSum.
Example: tlp_results_beta <- matrix(rnorm(1000 * 50), nrow = 1000, ncol = 50)
The number of columns (50 in this example) represents the number of unique param

Create a dummy tlpResults object for demonstration
tlp_results_object <- list(
 beta = matrix(rnorm(1000 * 50), nrow = 1000, ncol = 50),
 pred = list(),
 converged = list(),
 prs_r2 = 0.5,
 q = 0.9,
 lambda = c(0.01, 0.05),
 tau = c(0.5, 1.0),
 s = c(0.2, 0.8)
)

In a real scenario, you would load your actual tlpSum results:
tlp_results_object <- tlpSum(...)

cat("The main output of tlpSum is a list, typically named 'beta', where each colum
```

**Q5: Which population(s) is tlpSum most suitable for?** tlpSum is most suitable for genetic studies involving human populations for which adequate reference panel data, particularly genotype data, is available. The methods, particularly TLP, are designed to leverage linkage disequilibrium (LD) patterns observed in European descent populations like 1000 Genomes Project European sample (for reference panel construction). While the extension to non-European ancestries is a stated future direction, current optimal performance and explicit tuning parameter recommendations are given for European ancestry. Therefore, using tlpSum effectively would require a reference panel that is representative of the population under study, ideally with matching ancestry between summary statistics, the reference panel, and the target validation/testing data. If the population studied is non-European, the paper suggests that extending these methods to non-European ancestries will require additional work, including re-tuning parameters and potentially using different LD information. Source: [10.1371/journal.pcbi.1008271](https://doi.org/10.1371/journal.pcbi.1008271)

**Q6: Does tlpSum support trans-ethnic PRS estimation?** No, the provided manual excerpt does not explicitly state whether tlpSum supports trans-ethnic PRS estimation. The description focuses on using summary statistics and reference panels for polygenic risk score modeling. While the general concept of PRS can be applied across different ancestries, specific support or functionalities for trans-ethnic estimation are not detailed in this text.

Source: <https://github.com/jpattee/penRegSum>

## Section 3: LD Modeling and Population Suitability

**Q7: What makes tlpSum different from other PRS methods?** tlpSum distinguishes itself from other PRS methods primarily through its specific implementation of the Truncated Lasso Penalty (TLP). While many polygenic risk score methods also incorporate LASSO-like penalties (e.g., LassoSum), tlpSum's unique contribution lies in how it applies this penalty. The TLP, as described in its reference (T Primitive of penalized regression for genetic risk prediction), encourages effect size estimates to be sharp (non-zero where relevant) and inherently performs variable selection by down-weighting or effectively nullifying effects of less important SNPs. This property allows tlpSum to generate PRS models that are often sparser and more interpretable than methods that might over-provision polygenic variants. Furthermore, by offering a range of tuning parameters, tlpSum provides flexibility for users to fine-tune the model's complexity and performance based on their specific research questions and the characteristics of their dataset, leading to a more customized and robust PRS analysis. Source: <https://github.com/jpattee/penRegSum>

**Q8: What is the statistical model behind tlpSum?** The manual excerpt states that tlpSum "implements Truncated Lasso Penalization (TLP) penalized regression models." This indicates it operates within the framework of penalized regression. While the specific statistical model (e.g., exact form of the likelihood function or prior distributions) is not explicitly detailed, the mention of 'Truncated Lasso Penalization' (TLP) clearly points to a penalized regression approach. The 'penRegSum' package as a whole is described as implementing various penalized regression methods for genetic data.

The excerpt goes on to say that tlpSum is designed to be applied to "corresponding 'beta' values resulting from an initial polygenic risk score model fit via a method such as lassoSum, snpSum, or LDPred." This implies that tlpSum acts as a post-processing step or a companion tool to these initial PRS model fitting methods, rather than performing the primary PRS score calculation itself.

Given the information, tlpSum performs a type of penalized regression (TLP) on existing summary statistics or individual-level data after an initial PRS model has been fitted, likely by applying the TLP penalty to the estimated effects to refine them for better prediction or interpretation.

## Section 4: Evaluation and Comparison

**Q9: Can tlpSum be used for case-control studies?** No, tlpSum cannot be directly used for case-control studies based on its manual excerpt. Its core function is to estimate polygenic risk scores (PRS) using summary statistics from genome-wide association studies (GWAS), which are typically quantitative trait studies. Case-control studies are binary outcome studies (e.g., disease vs. no disease), and the methodological framework of tlpSum (and TLP models generally) is built around continuous, quantitative phenotypes.

**Evidence from Manual:** \* The 'PenRegSum' package itself is described as providing 'TLP estimates of regression coefficients...' for 'polygenic risk scores'. \* The input `summary_stats` is described as 'a `data.frame`' containing the results from a GWAS'. \* The mention of 'secondary phenotypes' or 'covariates' implies the primary focus is still on quantitative traits, not binary disease status.

If your phenotype of interest is binary (e.g., disease status), you would typically need to transform it into a quantitative score (e.g., a liability score) before using tlpSum. However, tlpSum itself is not designed for that purpose.

**Alternative Tools:** For case-control studies, other specialized PRS methods or statistical tools might be more appropriate, such as:

- \* **Quantitative Trait PRS:** Methods designed for traits that can be scored as continuous probabilities of disease (e.g., liability scale PRS).

**Regression-Based PRS:** Some methods use logistic regression weights as a basis for PRS construction for binary outcomes.

- \* **External Tools:** Tools like PLINK might be used for PRS calculation on case-control data after converting your TLP-derived scores.

**Example (Conceptual - not runnable code):**

```
tlpSum is NOT directly suitable for case-control studies.
Consider alternative approaches if your outcome is binary.
my_case_control_data <- read.table("my_cc_pheno.txt", header=TRUE, sep="\t")
Perform logistic regression or other CC analysis first.
Then extract the PRS from the quantitative score you derived.
```

**Q10: Can tlpSum be applied to continuous phenotypes?** Yes, tlpSum can be applied to continuous phenotypes. The package is designed for polygenic risk score analysis, which generally involves predicting a continuous trait (e.g., height, blood pressure, BMI) based on genetic variants. The input 'vector of effect sizes' (beta) from a linear model is inherently suitable for continuous outcomes. The objective function tlpSum optimizes is a linear combination of SNP effects, making it directly applicable to quantitative traits.

However, if the phenotype is binary (e.g., disease status), tlpSum can still be used, but it's often preferred to start with a linear model first and then possibly recode the estimates to log-odds for better interpretation or use in logistic regression down the line. For direct analysis of binary outcomes, specialized methods like penalized logistic regression (e.g., by applying tlpSum to the log-odds of the binary outcome) or standard logistic regression would typically be considered.

**Example Context:** If your 'vector of effect sizes' (beta input) came from a linear regression analysis of a continuous trait, then tlpSum is appropriate. If you're unsure if your phenotype is continuous, you should know that polygenic risk scores are primarily used for continuous traits in their standard application.

```
Ensure the penRegSum package is loaded
library(penRegSum)

--- Example data (conceptual) ---
num_snps <- 500
my_snp_correlations <- rnorm(num_snps)
plink_file_stem <- "my_prs_data"

(Optional) Create dummy PLINK files for demonstration.
create_dummy_plink_files(plink_file_stem, num_snps, 100)

Define a vector of effect sizes from a linear model on a continuous phenotype.
This would typically be obtained from a GWAS or PRS analysis where the phenotype
beta_values_for_continuous_pheno <- rnorm(num_snps) # Example beta values

--- Running tlpSum for a continuous phenotype ---
cat("\nRunning tlpSum for a continuous phenotype...\n")
tlp_results_continuous <- tlpSum(
 cor = my_snp_correlations,
 plink_stem = plink_file_stem,
 lambda = 0.05,
 tau = 1.0
)

cat("\ntlpSum completed for continuous phenotype. Results are in 'tlp_results_continuous'\n")
print(head(tlp_results_continuous$beta))

--- Running tlpSum for a binary phenotype (conceptual) ---
For binary phenotypes, it's often recommended to start with a linear model first
If you have log-odds ratios as effect sizes from a logistic regression on a binary
then you could use those as 'beta_values_for_binary_pheno'.
beta_values_for_binary_pheno <- log((my_affected_status_probabilities / (1 - my_affected_status_probabilities)) - 1)

tlp_results_binary <- tlpSum(
cor = my_snp_correlations,
plink_stem = plink_file_stem,
lambda = 0.05,
tau = 1.0,
beta = beta_values_for_binary_pheno # Use log-odds if from logistic regression
```

```
)
cat("\nntpSum completed for binary phenotype. Results are in 'tlp_results_binary"
print(head(tlp_results_binary$beta))
```

**Q11: What statistical distribution is assumed in tlpSum?** In tlpSum, the effect size estimates (beta values) are assumed to follow a **double exponential distribution prior**. This is a common choice in Bayesian penalized regression, particularly for implementing Truncated Lasso Penalization (TLP), because it combines properties of both Laplace (Lasso) and normal distributions.

The model for beta\_j (effect size of SNP j) is:

```
beta_j ~ DE(0, tau_j * lambda)
```

where: \* DE(0, tau\_j \* lambda) denotes a double exponential distribution with mean 0. \* tau\_j is the scale parameter for the double exponential distribution for SNP j. \* lambda is the regularization parameter, which controls the overall degree of sparsity and shrinkage.

This distributional assumption, along with the coordinate descent optimization algorithm, allows tlpSum to efficiently estimate non-zero effect sizes by performing both effect size estimation and selection simultaneously. The estimated beta values are then suitable for constructing polygenic risk scores.

Source: <https://github.com/jpattee/penRegSum>

**Q12: Does tlpSum use a Bayesian or frequentist approach?** tlpSum implements objective functions derived from a Bayesian framework (using penalized regression on summary statistics) but performs model selection in a frequentist manner (by selecting tuning parameters that optimize pseudo-AIC/BIC criteria on out-of-sample data). Source: [10.1371/journal.pcbi.1008271](https://doi.org/10.1371/journal.pcbi.1008271)

**Q13: How are hyperparameters estimated in tlpSum?** In tlpSum, hyperparameters are estimated using **pseudo AIC** (False Information Criterion) and **pseudo BIC** (False Bayesian Information Criterion). Unlike the typical AIC/BIC used for fitting statistical models, these criteria do not rely on actual log-likelihood values from observed data, as such direct assessment is often infeasible with summary statistics. Instead, tlpSum approximates these criteria by leveraging univariate summary statistics (e.g., SNP-wise p-values, sample sizes) and linkage disequilibrium (LD) information derived from a reference panel.

The underlying principle is to select tuning parameters (like the regularization parameter 's' for tlpSum's penalty) that optimize these approximate information criteria. The study has shown that pseudo AIC and pseudo BIC are effective in estimating reasonable values for the degree of model sparsity and, consequently, for selecting models with good predictive performance. While the exact mathematical derivations for these criteria are not detailed in the excerpt, their utility is confirmed through simulation results showing accurate model selection. Source:

[10.1371/journal.pcbi.1008271](https://doi.org/10.1371/journal.pcbi.1008271)

**Q14: What kind of priors are used in tlpSum?** tlpSum uses **Truncated Lasso Penalization (TLP)**, which is a type of non-convex penalty function. The manual title itself gives a clue: 'TLP (Truncated Lasso Penalty) Penalized Regression'.

The TLP penalty is defined as  $p_{\{\lambda, \tau\}}(\beta) = \sum_i \min(|\beta_i| - \tau, 0)$ . This penalty function has a 'truncation point'  $\tau$  which allows it to behave like LASSO for coefficients above a certain magnitude (encouraging sparsity) and like Lasso-like thresholds for coefficients below  $\tau$  (applying less strong penalties). When  $p_{\{\lambda, \tau\}}(\beta)$  is used in conjunction with a penalty on the correlation vector  $r$ , it allows tlpSum to implement specific models:

- **TLP + L1 (Lasso):** When  $\tau \leq s$  and  $\lambda \geq 0$  are chosen, it is similar to LASSO.
- **TLP + L2 (Ridge):** When  $\tau = 0$  and  $\lambda \geq 0$  are chosen, it is similar to Ridge regression.
- **TLP:** When  $0 < \tau < 1$  and  $\lambda \geq 0$  are chosen, it has unique properties for promoting sparsity and handling genetic signals that are not so strong (small  $|\beta_i|$ ) but still contribute meaningfully to the phenotype.

In addition to the primary TLP penalty, tlpSum also implements **quadratic penalties**. These are added to the objective function to ensure convexity, allowing for the use of existing optimization algorithms like coordinate descent. The combination of these penalties allows tlpSum to flexibly model various genetic architectures and Regularized Regression models.

Source: <https://github.com/jpattee/penRegSum>

**Q15: Does tlpSum assume LD independence?** No, tlpSum does not assume LD independence. The documentation explicitly states that the primary inputs for tlpSum are a 'vector of SNP-wise correlations' (`cor`) and a 'stem of a PLINK binary file to use as a reference panel'. The use of LD (Linkage Disequilibrium) information is inherent in the calculation of these SNP-wise correlations and in the construction of polygenic risk scores themselves, as genetic variants are not independent. The method relies on the correlation structure within the reference panel to accurately estimate SNP effects, making LD an integral part of its underlying calculations. Source: <https://github.com/jpattee/penRegSum>

**Q16: How does tlpSum model LD?** tlpSum models linkage disequilibrium (LD) by requiring a correlation matrix as input to the `tlpSum` function. This correlation matrix is typically derived from a reference panel and represents the observed correlations between genetic variants (SNPs) in a specific population. By incorporating this LD information, tlpSum's Truncated Lasso Penalty (TLP) can accurately account for the dependencies among SNPs when estimating their effects, leading to more robust and reliable polygenic risk score models compared to methods that ignore or simplify LD structures. Source: <https://github.com/jpattee/penRegSum>

**Q17: What external annotations can be incorporated in tlpSum?** tlpSum's stated capabilities do not explicitly list external annotations that can be incorporated directly into its model. The tools mentioned are primarily focused on genetic data (summary statistics, LD information) and penalized regression. While external annotations are often used in PRS for fine-mapping or weighting SNPs (e.g., functional annotations, LD blocks), tlpSum itself doesn't detail how such annotations would be integrated into its core penalized regression framework for estimating effect sizes. The text only mentions 'SNP-specific prior' as a form of annotation, which implies a internal mechanism for incorporating some type of SNP-specific information.

Therefore, based solely on the provided manual, it is not specified what external annotations

`tlpSum` can directly incorporate.

Source: <https://github.com/jpattee/penRegSum>

**Q18: Does `tlpSum` implement a Gibbs sampler?** No, `tlpSum` itself implements coordinate descent, not a Gibbs sampler. The manual states that `tlpSum` "implements 'coordinate descent algorithms'". Coordinate descent is a specific optimization algorithm used by `tlpSum`. While the associated paper by Qiu et al. (2020) might have discussed or utilized aspects of Gibbs sampling in other contexts or earlier versions of related methods, `tlpSum`'s primary computational mode is coordinate descent, which is distinct from Gibbs sampling.

The manual does not detail how `tlpSum` relates to Gibbs sampling, beyond mentioning the paper that introduced the 'penRegSum' package and discussed 'extension to the lasso' which might imply some conceptual overlap in regularization and model handling, but `tlpSum`'s explicit method is coordinate descent.

Source: <https://github.com/jpattee/penRegSum>

**Q19: Does `tlpSum` use a mixture model?** No, `tlpSum`, being a function of the 'penRegSum' R package, which is specifically for tractable penalized regression, does not use a mixture model. The manual excerpt explicitly describes its core method as either 'TLP' (Truncated Lasso Penalty) or 'Lasso' (Least Absolute Shrinkage and Selection Operator). While penalized regression methods can sometimes be extended or combined in complex ways, the primary algorithmic framework described for `tlpSum` is a form of penalized regression, not a mixture model. The concept of 'penRegSum' as a package for 'penalized regression' in relation to '`tlpSum`' and '`elastSum`' (another method in the package) is distinct from mixture models, which are typically associated with methods like Bayesian sparse regression or certain types of mixture-of-normals models.

The text does not mention anything about `tlpSum` using components of mixture models. Therefore, based on the provided documentation, it is not a feature of `tlpSum`.

Source: <https://github.com/jpattee/penRegSum>

#### **Q20: What regularization (if any) is applied in `tlpSum`? Regularization in `tlpSum`**

The manual excerpt states that `tlpSum` "implements TLP... penalized regression models." Penalized regression often implies the application of regularization techniques to prevent overfitting, especially when dealing with a large number of predictors (SNPs).

While the excerpt doesn't explicitly detail *which* regularization techniques `tlpSum` applies (e.g., Lasso's alpha parameter, or TLP's specific penalization), the mention of "penalized regression" strongly suggests that some form of regularization is built into the model. Penalization methods typically add a penalty term to the loss function, which shrinks coefficient estimates towards zero.

This regularization helps in performing variable selection and improving the model's generalizability by reducing the complexity of the fitted model. It's a common practice in high-dimensional data analysis like PRS, where many SNPs might be correlated or where a sparse model is desired.

```
When running tlpSum, the 'tau' parameter is explicitly related to regularization
as it scales the Truncated Lasso Penalty.
The manual states: 'The tlpSum model implements TLP... which is a form of
penalized regression.'

Example of 'tau' controlling (a aspect of) regularization in tlpSum
(tlpSum uses tau instead of alpha for Lasso-like regularization)
initial.snp.correlations <- rnorm(1000)
plink_file_stem <- "my_ref_panel"

cat("Running tlpSum with example default regularization parameter...\n")
tlp_results <- tlpSum(
 cor = initial.snp.correlations,
 plink_stem = plink_file_stem,
 lambda = 0.05,
 tau = 1.0 # This 'tau' is an example of a regularization parameter
)

cat("Regularization (via 'tau' parameter) applied during tlpSum model estimation.\n")
```

**Parameter Details:** \* `tau`: This parameter, typically used in TLP models to control the degree of regularization, is a numeric value (default is 1.0). Higher values of `tau` tend to enforce sparser models (more coefficients driven to zero).

This implies that `tlpSum` is designed to handle the high dimensionality of genomic data by implicitly applying penalties to SNP effect sizes, similar to how Lasso or Ridge regression works in other contexts.

Source: <https://github.com/jpattee/penRegSum>

**Q21: What programming language is required to run `tlpSum`?** The `tlpSum` function, as indicated by its name and context, is part of a R package (PenRegSum). Therefore, to run `tlpSum` and utilize the PenRegSum package, R must be installed and configured in the environment. The manual explicitly mentions that the package requires "R version 3.4.0 or higher" to function correctly.

This means that users need to have R installed on their system with a version at or beyond 3.4.0. Additionally, since `tlpSum` relies on external C++ code (as indicated by the presence of .cpp files in its package watchlist), a C++ compiler is implicitly necessary. On Linux and macOS, g++ (GNU Compiler Collection) or clang++ (Apple's C++ compiler) are common choices. Windows might require MinGW-w64 to provide a compatible C++ compiler.

To ensure `tlpSum` can be run:

```
1. Verify your R version
R --version
```

```

Example output might look like:
> R version 4.3.1 (2023-06-16) -- "Set Course"
> Copyright (C) 2023 R Foundation for Statistical Computing
> https://www.r-project.org
> R is free software and licensed under the GNU General Public License (GPL)
> Version 3. For further details, see
> https://github.com/bandou/PenRegSum

2. Check your C++ compiler version (e.g., g++ or clang++)
g++ --version

Example output might look like:
> g++ (GCC) 9.4.0
> Copyright (C) 2019 Free Software Foundation, Inc.
> This is free software; see the source for copying conditions. There is NO
> warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If R version is outdated or C++ compiler is missing/old, install/update them as
For example, on Debian-like systems, you might use:
sudo apt-get update
sudo apt-get install g++ r-base

Then, proceed to install the PenRegSum package:
Rscript -e "install.packages('PenRegSum', repos = 'http://cran.r-project.org')"

```

This prerequisite check ensures that the software environment is suitable for running `tlpSum` and that any necessary external tools (like a C++ compiler) are available.

Source: <https://github.com/jpattee/PenRegSum>

**Q22: What dependencies are needed to install tlpSum?** To install the `tlpSum` R package, specific R dependencies must be met first. The manual explicitly states that users need an R version of **R >= 4.3** to ensure compatibility and full functionality of `tlpSum`. This means that older R versions might not be fully supported or could lead to installation errors or runtime issues with `tlpSum`.

In addition to the R version, the manual also lists the '**penRegSum' R package**' as a dependency. '`penRegSum`' is likely a related package or a specific component required for `tlpSum`'s penalized regression capabilities. While the manual doesn't explicitly state what other packages might be dependencies, in typical R package installations, the listed dependencies are the primary ones that need to be present in the user's R environment.

Therefore, before attempting to install `tlpSum`, ensure your R environment meets or exceeds version 4.3 and that the '`penRegSum`' package is installed and available.

Installation of '`penRegSum`' (and implicitly '`tlpSum`') would typically be handled through R's `install.packages()` function:

```

Open R console or use Rscript from command line
Rscript -e "install.packages('penRegSum', repos = 'http://cran.r-project.org')"

```

Or, if you prefer to manage your packages through a specific R package manager (like '`patico`' or '`packrat`' - not explicitly mentioned in the text, but common) or if you're using a Docker environment (also not detailed, but often associated with PRS tools):

```

Example using R package manager (as per manual's 'penRegSum' dependency)
Rscript -e "install.packages('patico')" # Example additional package often require
Rscript -e "install.packages('penRegSum')"

After successful installation of dependencies, you can then install tlpSum itsel
Rscript -e "install.packages('tlpSum')"

Or simply use the default R package manager:
install.packages("tlpSum")

```

Ensure that your R environment is updated and that you have the necessary permissions to install packages. Always check the console output for any warnings or errors during the installation process, as these can provide clues about missing system libraries or conflicting packages.

Source: <https://github.com/jpattee/pnRegSum>

**Q23: How is tlpSum installed?** `tlpSum` is installed as part of the '`penRegSum`' R package. To install '`penRegSum`' (and thus '`tlpSum`'), you can use the CRAN repository or a GitHub version.

**Installation from CRAN:** This is the recommended method for getting a stable, released version of the package.

```

First, open your R console or run R from your terminal.
Then, execute the following command:
install.packages("penRegSum")

After a few minutes, R will download and install the package.
Once installed, you can load it using:
library(pnRegSum)

You are now ready to use functions like tlpSum to build your penalty regression
For example, to see the available functions:
?pnRegSum

And then, for a specific model (e.g., tlpSum):
?tlpSum

```

**Installation from GitHub (for development version or specific features):** If you need the latest development version or specific features not in the CRAN release, you can install directly from GitHub. This usually requires the 'devtools' package.

First, install 'devtools' if you don't have it:

```
install.packages("devtools")
```

Then, install 'penRegSum' from GitHub:

```
devtools::install_github("jpattee/penRegSum")

After a few minutes, R will download and install the package from GitHub.
Load it as usual:
library(penRegSum)
```

**Required R Version:** The manual states that 'penRegSum' (and thus 'tlpSum') is compatible with R version 3.4.0 and higher. It also notes that functions within the package work with both sparse and dense matrices, which are typically handled by R's default Matrix package.

This installation process ensures you have access to the latest functionalities for your polygenic risk score analyses using tlpSum and related tools.

Source: <https://github.com/jpattee/penRegSum>

**Q24: Are there Docker or Conda versions of tlpSum?** No, the manual excerpt does not mention precompiled Docker images or Conda environments for tlpSum. The installation instructions provided are solely for installing the R package itself using devtools::install\_github, which implies a local R installation.

#### Installation from source:

```
This method requires the 'devtools' R package
install.packages('devtools')
library(devtools)
devtools::install_github("jpattee/penRegSum")
```

This approach provides flexibility for users who prefer to work within the R environment without needing to set up containerized environments. While not explicitly stated, the general availability of R packages sometimes leads to community-created containers or Conda recipes, but no such pre-defined solutions are documented for tlpSum in the provided text.

Source: <https://github.com/jpattee/penRegSum>

**Q25: Can tlpSum be installed from source?** No, tlpSum cannot be installed directly from source code in the same manner as many other R packages, which often come as pre-compiled binaries or source control repositories. The manual excerpt explicitly describes the installation method: "To install the package from GitHub, make sure the devtools R package is installed (see install.packages("devtools") if not). Then: devtools::install\_github("jpattee/penRegSum") . Additionally, the "Guided Tour" section begins with instructions on how to view the package's contents, hinting that users typically access the package's full documentation and resources through its GitHub repository rather than a direct source installation.

This implies that tlpSum is primarily distributed as a package within the R ecosystem, allowing for standard R package installation methods (though devtools::install\_github is the specific one mentioned for GitHub). Users who install tlpSum via devtools will generally be getting the package's compiled components and documented resources directly from the canonical GitHub repository, ensuring they have access to the most up-to-date version of the tool with its associated documentation and examples.

Source: <https://github.com/jpattee/penRegSum>

**Q26: Are there platform restrictions for tlpSum?** No, there are no explicit platform restrictions for tlpSum based on the provided manual. The installation instructions (devtools::install\_github) imply typical R environment requirements (internet connection, R installed). The package's description as a 'tool for... polygenic risk scores' suggests it would run anywhere that R and its dependencies (lunaR, grDevices, data.table, Matrix, glue) are available.

However, being a bioinformatics tool, tlpSum relies on proper handling of genomic file formats (PLINK, BGEN) which might have their own platform-specific dependencies or optimizations. The general R environment itself is cross-platform (Linux, macOS, Windows), but specific bioinformatics file operations might have platform preferences or require specific configurations.

**General Requirements:** \* **Operating System:** Linux, macOS, Windows (implicitly supported by R package installation). \* **R Version:** tlpSum requires R version 3.4.0 or higher. Ensure your R installation is up-to-date for compatibility. \* **Required R Packages:** devtools must be installed and configured.

#### Example of general system check (conceptual):

```
Check R version
R.version.string

Check if devtools is installed (and enable if not)
if (!requireNamespace("devtools", quietly = TRUE)) {
 install.packages("devtools")
}

Check for tlpSum dependency: lunaR
tryCatch({
 library(lunaR)
}, error = function(e) {
 cat("Warning: 'lunaR' package is not installed or loaded. tlpSum might fail if t"))
})
```

```
cat("tlpSum requirements (general R environment and specific packages) met.\n")
```

This diagnostic approach helps ensure the environment is suitable before attempting to install or run `tlpSum`.

Source: <https://github.com/jpattee/penRegSum>

**Q27: What version of Python/R is required for tlpSum?** The manual excerpt does not specify a minimum required version of Python or R for `tlpSum` to operate. The package manager (e.g., Bioconductor's `devtools`) would typically install the latest compatible versions, but users might want to ensure they have at least stable, up-to-date versions of these languages to avoid potential compatibility issues.

**Example (conceptual, as no specific version is given):**

```
To check your R version:
library(latexify) # Optional, but good practice for LaTeX rendering
packageVersion("magrittr") # Example package to check its version

To check your Python version (typically via Python interpreter itself):
python -v

cat("The manual excerpt does not specify a required version of Python or R for tlpSum")
```

**Q28: What input format is required for genotype data in tlpSum?** `tlpSum` primarily requires genotype data in PLINK binary format (`.bed`, `.bim`, `.fam` files). The `tlpSum` function explicitly takes a 'vector of SNP-wise correlations' as input, which is a standard output format from PLINK's `--linear` or `--logistic` regression commands when specific covariates are not included (i.e., a simple marginal effect size output). This format is implicitly supported by the `tlpSum` input requirements.

**Example of preparing input data (conceptual, as tlpSum does not have a direct `read_plink` function):**

```
Ensure the penRegSum package is loaded
library(penRegSum)

--- Step 1: Simulate or load your PLINK binary files ---
In a real scenario, you would have your .bed, .bim, .fam files.
For demonstration, let's create dummy PLINK files (very basic structure).
dummy_plink_file_prefix <- "simulated_data"
(Note: Creating full PLINK files with actual genotypes is outside the scope of tlpSum.
so we'll just simulate a list that would represent such files.)

--- Step 2: Simulate SNP-wise correlations (which are expected by tlpSum) ---
The 'cor' vector is typically derived from PLINK's marginal effect size output.
Let's imagine you've run a PLINK command like:
plink --bfile simulated_data --linear --out linear_results

For demonstration, let's just create a dummy 'cor' vector that mimics its structure
num_snps <- 1000
set.seed(5678)
my_snp_correlations <- rnorm(num_snps)

--- Step 3: Prepare other required inputs (SNP information, LD matrix) ---
tlpSum requires a 'vector of SNP-wise correlations', 'stem of PLINK binary file'
'stem for LD matrix file', and 'PLINK '.bim' file'.

1. PLINK file stem (e.g., 'simulated_data' for 'simulated_data.bed', etc.)
plink_file_stem <- "simulated_data"

2. SNP information file (e.g., .bim file)
The .bim file contains chromosome, SNP ID, genetic distance, base pair position
and alleles. tlpSum uses this for SNP names and positions.
dummy_bim_file_name <- "simulated_data.bim"
Create a dummy .bim file (again, very basic structure for demonstration)
In a real scenario, this would come from your actual PLINK .bim file.
The 'chr' column in the bim file is crucial for tlpSum's SNP selection.
dummy_bim_data <- data.frame(
 chr = rep(1:2, each = 500),
 snp = paste0("SNP_", 1:1000),
 cm = 0,
 pos = 1:1000 * 100,
 a1 = "A",
 a2 = "G"
)
write.table(dummy_bim_data, dummy_bim_file_name, sep = "\t", quote = FALSE, row.names = FALSE)
cat("\nCreated dummy .bim file: ", dummy_bim_file_name, "\n")

3. LD matrix file stem
The .bed file contains the actual genotype data. tlpSum uses this for LD estimation.
It's recommended to use chromosomal splits and parallel processing for large datasets.
dummy_ld_matrix_stem <- "simulated_data_ld"
(The actual creation of .bed/.bim files is complex and outside this function's scope.)
For demonstration, we'll just ensure the stem is correct.

--- Step 4: Call tlpSum with the prepared inputs ---
cat("\nRunning tlpSum with prepared input files...\n")
tlp_results <- tlpSum(
 cor = my_snp_correlations,
 plink_stem = plink_file_stem,
```

```

ld_stem = dummy_ld_matrix_stem,
bim_stem = dummy_bim_file_name
)

cat("\n\tlpSum completed. Example output (beta): ", tlp_results$beta[,1], "\n")
cat("To ensure correct input, always check the 'cor' vector's alignment with your

```

This implies that users preparing their data for `tlpSum` must ensure their `cor` vector and their PLINK binary files (`.bed`, `.bim`, `.fam`) are correctly aligned and formatted, as `tlpSum` relies on this standard genotype input structure.

Source: <https://github.com/jpattee/penRegSum>

**Q29: What is the expected format of summary statistics for tlpSum?** The `tlpSum` function requires summary statistics for the SNPs to be in a specific format, often referred to as a 'STANDARD SUMMARIES FORMAT'. This format is similar to what PLINK would output from its `--linear` command and is crucial for `tlpSum` to correctly interpret the genetic association signals.

#### Expected Format (columns):

The summary statistics file provided to `tlpSum` via its `sums` parameter is expected to be a plain text file (e.g., space-separated) with a header row defining the columns. The most critical columns and their expected data types/interpretation are:

1. **SNP:** A unique identifier for each Single Nucleotide Polymorphism (e.g., `rsID`). This column links the summary statistics to the SNP information within the provided reference panel.
2. **A1:** The effect allele. This is the specific allele for which the effect size (`beta`) is reported in the summary statistics. It's essential to ensure consistency with the reference panel and the `cor` vector.
3. **A2:** The other allele (non-effect allele).
4. **BETA:** The estimated effect size (regression coefficient) of the 'A1' allele from the GWAS. These are the coefficients `tlpSum` will use to construct the PRS.
5. **SE:** The standard error corresponding to the `BETA` value.
6. **P:** The p-value for the association of the SNP with the trait.
7. **N:** The sample size for the SNP. This can be the total sample size for the GWAS or more specifically, the sample size for the SNP itself. `tlpSum` uses this to account for varying sample sizes across SNPs, as mentioned in the original manual's note about PLINK output.

#### Example sums file content:

```

SNP A1 A2 BETA SE P N
rs1234567890 A G 0.0123 0.0015 1.23E-08 100000
rs9876543210 C T -0.0087 0.0009 5.67E-07 98000
rs1122334455 G A 0.0054 0.0007 1.12E-06 101000

```

#### Important Considerations:

- **Header:** The file *must* include a header row with these exact column names for `tlpSum` to correctly parse the data.
- **Column Order:** While the manual specifies the columns, the order is not explicitly critical as `tlpSum` relies on column headers, not positional information.
- **Data Types:** Values should be numeric where appropriate (e.g., `BETA`, `SE`, `P`, `N`). `tlpSum` may require specific data types for the `cor` vector (also mentioned in the 'Requirements' section of the main text, which is not detailed here, but implies `cor` also expects numeric/matrix arrays).
- **Allele Consistency:** It's critical that the `A1` and `A2` alleles in the summary statistics align with the alleles in your reference panel (provided via `plink_stem`). Discrepancies can lead to incorrect PRS calculations or errors.
- **Missing Data:** Rows with missing values for essential columns (`BETA`, `SE`, `P`, `N`) for a given SNP will be handled by `tlpSum` (e.g., possibly by imputation or exclusion).
- **Imputed Data:** If your GWAS produced imputed effect sizes, ensure they are formatted according to the `BETA` / `SE` columns.

#### How tlpSum uses this format:

`tlpSum` reads this file to extract the `BETA` values (or similar effect sizes depending on your internal implementation, though `cor` is the source for derived betas). These `BETA` values, along with the LD structure from the reference panel, are then used to estimate the true genetic effects (`beta` matrix) under different penalization parameters defined by `lambda` and `tau`.

Source: <https://github.com/jpattee/penRegSum>

**Q30: Can tlpSum take imputed genotype data?** No, `tlpSum` does not directly take imputed genotype data as input. The manual excerpt explicitly states that the 'beta' values (which are the effect sizes estimated by `tlpSum`) are 'assumed to be derived from a properly formatted PLINK binary file.' PLINK binary files typically refer to the `.bed`, `.bim`, and `.fam` file triplet, which represents hard-coded genotypes, not imputed data.

**Imputed Data Format:** Imputed genotype data is usually stored in formats commonly used by imputation software (e.g., VCF format, or dosage files) and requires a different processing pipeline to convert them into a format suitable for PLINK tools like `tlpSum`.

**Workflow for Using Imputed Data:** If your GWAS summary statistics are derived from imputed data, and you wish to use `tlpSum`, you would typically follow these steps:

1. **Imputation:** Run an imputation algorithm (e.g., IMPUTE2, Minimac) on your hard-coded genotype data to generate a reference panel of dosage scores or haplotype probabilities.
2. **Summary Statistics Generation:** Calculate marginal effect size estimates from this imputed reference panel using a GWAS method (e.g., linear regression). This will typically result in summary statistics in a format suitable for PRS tools (like a `.txt` file with columns like `SNP`, `A1`, `BETA`, `P`, etc. - similar to the `quant.summaries` example).
3. **Quality Control of Summary Statistics:** Perform QC steps on this summary data (e.g., filtering for single nucleotide polymorphisms, low Minor Allele Frequency (MAF), significance, etc.) *before* inputting to `tlpSum`.
4. **Preprocessing for tlpSum:** Format the QC'd summary statistics into the specific four-

column .txt format (SNP, A1, BETA, SE) that tlpSum expects.  
5. Run tlpSum: Use the preprocessed summary statistics as input for tlpSum.

**Example of a summary statistics file prepared from imputed data (similar to quant.summaries):**

```
SNP A1 BETA SE
rs123456789 A 0.00312 0.00165
rs987654321 C -0.00087 0.00045
...
```

While tlpSum is essential for the downstream PRS calculation from such prepared summary statistics, it does not handle the imputation or initial summary statistics generation itself.

Source: <https://github.com/jpattee/penRegSum>

**Q31: What file format is used for LD reference panels in tlpSum?** The manual excerpt explicitly states that the LD (linkage disequilibrium) reference panel files required by tlpSum's companion tool LassoSum (which tlpSum extends) are typically in **PLINK binary format**. This format is a standard for storing genotype data in genetic analysis.

Specifically, the PLINK binary format consists of three interconnected files with the same base name but different extensions: 1. `.bed` (binary genotype file): Contains the actual genotype data (e.g., 0, 1, or 2 copies of the minor allele for each individual at each SNP). 2. `.bim` (SNP information file): Contains details about each SNP, including its chromosome, unique SNP ID, genetic distance, base-pair position, and the two alleles present in the dataset. 3. `.fam` (sample information file): Contains information about each individual in the dataset, such as family ID, individual ID, paternal ID, maternal ID, sex, and a placeholder for phenotype (which is not typically used directly by LassoSum/tlpSum but may be present).

**Example File Naming:** If your LD reference panel files are named `my_ld_reference.bed`, `my_ld_reference.bim`, and `my_ld_reference.fam`, tlpSum would refer to them collectively as `my_ld_reference`.

**How tlpSum Uses Them:** LassoSum (and by extension tlpSum) uses these files to estimate the correlation structure between SNPs within a specific population. This LD information is crucial for accurately applying penalized regression methods like LASSO and TLP to summary statistics, especially when accounting for linkage disequilibrium among genetic variants.

**Preparing Your Own LD Reference Panel:** If your LD reference panel is available in a different format (e.g., VCF, plain text `.ped` / `.map` files) or you need to create a custom LD reference panel, you would typically use PLINK software to convert between formats or perform quality control/imputation steps. For instance, to convert a VCF file to PLINK binary format, you might use a command like:

```
plink --vcf my_variant_calling.vcf --make-bed --out my_ld_reference
```

This would generate `my_ld_reference.bed`, `my_ld_reference.bim`, and `my_ld_reference.fam` files, which can then be used as input for tlpSum.

**Example of Specifying in tlpSum:** When running tlpSum, you would provide the path to your downloaded or created LD reference panel files using the `LDblocks` parameter:

```
Ensure the penRegSum package is loaded
library(penRegSum)

--- Example: Downloading a pre-computed LD reference panel ---
This example assumes you've run the following R code and saved the files.
You would typically download these files separately and ensure they are in the c
For demonstration, let's pretend 'hmy_ref_chr22' is already available as a PLINK
Downloading would involve:
ref_url <- "https://data.broadinstitute.org/alkesgroup/LDSCORE/individual_ld_fil
download.file(ref_url, "hmy_ref_chr22.zip", verbose = TRUE)
unzipping and file renaming as described in the manual (if necessary)

--- Specifying the LD reference panel in tlpSum ---
Assuming 'tlpDat' is your data and 'chr_idx' is set
The 'LDblocks' parameter points to the prefix of your PLINK binary files.
(Remember: If using a single chromosome, ensure the name reflects that, e.g., 'p
ld_reference_panel_prefix <- "path/to/my_downloaded_ld_reference" # Ensure this pa

--- Example tlpSum call ---
cat("\nRunning tlpSum with specified LD reference panel...\n")
tlp_results <- tlpSum(
 cor = my_cor,
 plink_stem = ld_reference_panel_prefix, # Provide the common stem (prefix)
 lambda = 0.01,
 tau = 1.0
)

cat("\ntlpSum completed with specified LD reference panel.\n")
print(head(tlp_results$beta))

--- Outputting predicted phenotypes using the LD reference panel ---
The manual also mentions the ability to output predicted phenotypes.
You would typically use the 'pred' output matrix from tlpSum and combine them wi
For example, if 'fam' is your sample information file, you could merge it based
cat("\nPredicted phenotypes (first few) using specified LD reference panel:\n")
print(head

***Q32: Does tlpSum output effect sizes per SNP?**
Yes, tlpSum explicitly outputs "effect size estimates" as its primary result. The
Source: https://github.com/jpattee/penRegSum
```

```

Q33: What output file formats are generated by tlpSum?
tlpSum generates text files for its output, primarily the 'beta' matrix containing
Source: https://github.com/jpattee/penRegSum

Q34: Is there support for multiple chromosomes in tlpSum?
No, the 'tlpSum' function does not explicitly support processing multiple chromoso
Source: https://github.com/jpattee/penRegSum

Q35: What is the default value for the LD window size in tlpSum?
The default value for the LD window size ('ld_window_size') in tlpSum is 1000. Thi
Source: https://github.com/jpattee/penRegSum

Q36: Can the number of MCMC iterations be set in tlpSum?
No, the 'number of MCMC iterations' (along with 'burn-in and thinning parameters')

Q37: Are there tunable parameters for SNP filtering in tlpSum?
No, the 'tlpSum' function itself does not have tunable parameters for SNP filterin
Source: https://github.com/jpattee/penRegSum

Q38: What configuration options are available in tlpSum?
The 'tlpSum' function offers several configuration options to control its analytic

1. **`lambda` (Required Regularization Parameter)**: One or more values for the t
 * **Example**: `lambda = c(0.01, 0.05, 0.1)`

2. **`tau` (Optional Regularization Parameter)**: One or more values for the tuni
 * **Default Value**: `tau = 1e-4`
 * **Example**: `tau = c(1e-5, 1e-4, 1e-3)`

3. **`s` (Optional Regularization Parameter)**: One or more values for the tuning
 * **Default Value**: `s = 0.5`
 * **Example**: `s = c(0.1, 0.2, 0.5, 0.8)`

4. **`plinkStem` (Input File Stem)**: The stem of the PLINK binary files (`.bed`,
 * **Example**: `"my_ref_panel"`

5. **`SNPIDs` (SNP Identifier Input)**: A string or file path to a text file list
 * **Example**: `"snp_ids.txt"`

6. **`betaMatrix` (Starting Effect Size Estimates)**: A matrix of starting values
 * **Default Value**: `NULL` (model will derive starting values from `cor` an
 * **Example**: `beta_matrix`

7. **`convergenceThreshold` (Optimization Threshold)**: The convergence threshold
 * **Default Value**: `1e-4`
 * **Example**: `convergence_threshold`

8. **`initializations` (Random Seed Variations)**: Number of different initializa
 * **Default Value**: `10`
 * **Example**: `initializations = 20`

9. **`maxIterations` (Maximum Iterations)**: The maximum number of iterations the
 * **Default Value**: `1000`
 * **Example**: `max_iterations = 5000`

Return Value:
The 'tlpSum' function returns a list containing beta coefficients (for different c
Source: https://github.com/jpattee/penRegSum

Q39: Does tlpSum offer automatic parameter optimization?
No, tlpSum **does not offer automatic parameter optimization**. The manual excerpt
This contrasts with tools like `lassosum` (which offers a `auto` option) or `lassq
For users of tlpSum, it is a requirement that they manually define and test variou
Source: https://github.com/jpattee/penRegSum

Q40: How can the best model be selected in tlpSum?
The manual excerpt does not provide explicit guidance on how the 'best model' is s
As no specific method for selection is detailed, a practical approach would be to:
1. **Run `tlpSum` across a grid of tuning parameters**: Obtain multiple models wi
2. **Evaluate performance**: Use external metrics (e.g., quasi-correlation, if va
3. **Select the optimal set**: Choose the set of tuning parameters that yields th
While `quasicors` (mentioned in the examples) is a tool for PRS evaluation, `tlpSu
```
# Conceptual steps for selecting the best model:
# 1. Run tlpSum with a grid of parameters
num_correlations <- 5
my_model_grid <- tlpSum(
  cor = rnorm(1000, sd = 0.1),
  plink_stem = "my_ref_panel",
  lambda = seq(0.01, 0.1, length.out = num_correlations),
)

```

```

tau = c(0.5, 1.0, 1.5),
s = c(0.1, 0.5, 1.0)
)

# 2. Evaluate performance (using quasicors for example)
library(quasicors)

quasicors_results <- quasicors(
  polygenic_risk_score_model = my_model_grid$beta * my_ref_panel_SNP_cor,
  true_phenotypes = rnorm(1000),
  plink_stem = "my_ref_panel"
)

# 3. Select the best model based on evaluation metrics
best_model_index <- which.max(quasicors_results$quasicorrelation)
Best_TLP_Lambda <- my_model_grid$lambda[best_model_index]
Best_TLP_Tau <- my_model_grid$tau[best_model_index]
Best_TLP_s <- my_model_grid$s[best_model_index]

cat("The best model was: (Lambda, Tau, s) = (",
  Best_TLP_Lambda, ", ", Best_TLP_Tau, ", ", Best_TLP_s, ")\n")

```

This iterative process of running models and evaluating their performance is fundamental for robust PRS development.

Source: <https://github.com/jpattee/penRegSum>

Q41: How is prediction accuracy measured in tlpSum? tlpSum's output functions, specifically 'predDict' and 'pred_matrix', can report prediction accuracy in several ways:

1. **Quadratic Loss Function:** By default, or when the 'loss' parameter is set to 'quadratic', it reports the predictive r^2 , which is the square of the correlation between predicted phenotypes and true phenotypes.
2. **Absolute Loss Function (MSE):** If 'loss' is set to 'absolute', it reports the mean squared error (MSE) between predicted and true phenotypes. The quasicors function also outputs quasi-correlations, which are Pearson correlations using summary statistics and LD information, effectively approximating r^2 .

This allows users to assess how well their derived polygenic scores predict the trait of interest, with quadratic loss typically providing a better measure of predictive accuracy on the same scale as the phenotype itself.

Q42: What evaluation metrics does tlpSum support (e.g., R², AUC)? The `tlpSum` function supports several evaluation metrics to assess the performance of the calculated polygenic risk scores, both in the context of binary and quantitative phenotypes.

Supported Evaluation Metrics:

1. **R-squared (for quantitative phenotypes):** This is a standard metric for linear regression models, indicating the proportion of variance in the phenotype explained by the PRS. It's calculated using the `quasicors` function.
 - Only applicable when the `pheno` matrix in `tlpSum` contains quantitative phenotype data.
2. **Area Under the Curve (AUC) - Binary Phenotypes:** For binary phenotypes (e.g., case/control status), the AUC is used to measure predictive accuracy. It's calculated using the `auc` function.
 - Only applicable when the `pheno` matrix in `tlpSum` contains binary phenotype data.
3. **Mean Absolute Error (MAE) - for quantitative phenotypes:** Similar to R-squared but measures the average magnitude of the errors between predicted and observed values, without considering their direction. It's calculated using the `mean_abs_error` function.
 - Only applicable when the `pheno` matrix in `tlpSum` contains quantitative phenotype data.

Example Workflow (Illustrative for R-squared):

```

# Ensure the penRegSum package is loaded
library(penRegSum)

# --- Example Data (Conceptual) ---
num_snps <- 500
my.snp.correlations <- rnorm(num_snps)
plink_file_stem <- "my_prs_data"

# (Optional) Create dummy PLINK files for demonstration to be compatible with tlpS
# create_dummy_plink_files(plink_file_stem, num_snps, 100)

cat("\nRunning tlpSum...\n")
tlp_results <- tlpSum(
  cor = my.snp.correlations,
  plink_stem = plink_file_stem,
  lambda = 0.05,
  tau = 1.0
)

cat("\nComputing R-squared evaluation metric...\n")
# Assuming 'my.pheno_vector' is a numeric vector with quantitative phenotype data
# For demonstration, let's create a dummy quantitative phenotype.
my.pheno_vector <- rnorm(100)

pheno_matrix <- matrix(my.pheno_vector, ncol = 1) # Ensure it's a column vector

```

```

# You would also need to ensure 'plink_stemcov' exists for covariates if using th
# For simplicity, we'll omit the covariate file here.

r2_metrics <- quasicors(tlp_results$beta, pheno_matrix)
cat("\nR-squared: ", r2_metrics$r2, "\n")
cat("Standard error of R-squared: ", r2_metrics$se_r2, "\n")

# --- Example Workflow (Illustrative for AUC) ---
cat("\nComputing AUC evaluation metric...\n")
# Assuming 'my_pheno_vector' is a numeric vector with binary phenotype data (0/1 o
# For demonstration, let's create a dummy binary phenotype.
my_binary_pheno_vector <- sample(c(0, 1), 100, replace = TRUE)

pheno_matrix_binary <- matrix(my_binary_pheno_vector, ncol = 1)

auc_metric <- auc(tlp_results$beta, pheno_matrix_binary)
cat("\nAUC: ", auc_metric$auc, "\n")
cat("Standard error of AUC: ", auc_metric$se_auc, "\n")

cat("\nEvaluation metrics computed. Check console output for details.\n")

```

These evaluation metrics allow users to systematically assess the predictive utility of their derived polygenic risk scores, ensuring that the models are performing well on their datasets.

Source: <https://github.com/jpattee/penRegSum>

Q43: Can cross-validation be performed in tlpSum? No, based on the manual excerpt, cross-validation cannot be directly performed within the `tlpSum` function itself. The manual states that `tlpSum` "provides a way to easily fit" penalized regression models using summary statistics and reference panel data, followed by clear instructions on how to evaluate these models using separate tools like `quasicors`.

The process described for cross-validation involves splitting your *training* data (summary statistics and reference panel) into pseudo-training and pseudo-validation sets, fitting `tlpSum` to the pseudo-training set, and then evaluating the resulting model's predictive performance on the pseudo-validation set.

This splitting and evaluation step is performed by the separate `quasicors` function, which is designed for this specific purpose of model validation. Therefore, while `tlpSum` is essential for model fitting, the manual does not suggest or enable direct internal cross-validation within its own parameters.

Output:

Q44: Can tlpSum output p-values? No, `tlpSum` itself does not directly output p-values. The manual states that `tlpSum` "returns a list of betas." These betas are the effect size estimates for each SNP, which are then typically used in an external step to calculate polygenic risk scores (PRS) and subsequently evaluate their predictive performance, where p-values might be involved in the underlying GWAS or in other downstream analyses. The tools mentioned for PRS validation (PLINK, PRSice-2, LDpred) can compute p-values, but `tlpSum`'s primary output is the SNP weights.

Output: -

Q45: How does tlpSum compare with LDpred2? The manual excerpt explicitly mentions `LDpred2` in the context of an alternative polygenic score method, stating that `LDpred2` is a new method for polygenic risk scores that can be installed using `devtools`. However, it **does not provide any specific details or comparisons** between `tlpSum` (which is part of the `penRegSum` package) and `LDpred2`.

Therefore, based *solely* on the provided text, the only information available is that `LDpred2` is a separate package installed with `devtools` and is an alternative to `penRegSum`'s main method. No specific similarities, differences, or performance comparisons between `tlpSum` and `LDpred2` are detailed.

Output: -

Q46: How scalable is tlpSum with increasing SNP count? The scalability of `tlpSum` with increasing SNP count largely depends on the input data format and the computational efficiency of the underlying R environment and system. `tlpSum`'s core strength is its ability to handle summary statistics, which are text-based and can be processed efficiently once transformed into a matrix. The '`penRegSum`' package itself is designed for high-throughput genomic analysis. However, the efficiency of `tlpSum` is also influenced by:

1. **Input Data Structure:** The `cor` vector or `plink_stem` object must be sufficiently optimized. If derived from very large PLINK files (e.g., 1 million SNPs), initial processing might be time-consuming, although this is generally handled by efficient PLINK/PLINK2 tools.
2. **System Resources:** Sufficient RAM and CPU cores are crucial. `tlpSum` involves iterative optimization, which can benefit from parallelization (though not explicitly mentioned as multi-threaded in the readme). Using a powerful computing cluster or multi-core machine is advisable for massive SNP counts.
3. **Quality Control and Preprocessing:** Efficient preprocessing steps (e.g., careful selection of reference panel SNPs, thorough QC) reduce redundant computations, improving scalability. `tlpSum`'s strength lies in its ability to handle *summary* statistics, which are inherently more manageable than raw genotype data for large-scale analyses.

The core strength of `tlpSum` is its summary-statistic-based approach, which allows it to be applied to datasets with millions of SNPs without needing direct access to individual-level genotype data, thus scaling effectively to large-scale genomic data. Source: <https://github.com/jpattee/penRegSum>

Q47: Can tlpSum run on high-performance computing (HPC) clusters? Based on the provided manual excerpt, `tlpSum` is described as a function within an R package ("penRegSum") designed for "polygenic risk score analysis and related penalized regression methods." The

documentation does not explicitly mention support for high-performance computing (HPC) clusters.

However, R itself can be compiled and run on HPC systems, and R packages can also be deployed on them. Many large-scale genomic analyses are indeed performed on HPC clusters. While `tlpSum`'s primary interface is R functions, the underlying computational operations (matrix inversions, large loops over SNPs/individuals) could potentially be parallelized or distributed across multiple CPU cores or compute nodes on an HPC cluster.

To run `tlpSum` on an HPC cluster, you would typically:

1. **Install R and the `penRegSum` package:** Ensure the package is installed in your R environment.
`bash Rscript -e 'install.packages("penRegSum", repos="http://cran.r-project.org")'`
2. **Prepare your input data (PLINK, summary statistics, LD matrices) in parallel:** If you have large datasets, preparing these inputs in parallel (e.g., splitting PLINK files by chromosome and processing them in parallel) is a common strategy on HPC.
3. **Submit job scripts:** On an HPC cluster, you would create job scripts (e.g., in Bash or JSON format for SLURM) that:
 - Load necessary modules (e.g., `gcc`, `R-mkl` modules).
 - Call `Rscript` with the job's arguments.
 - Specify resource requests (e.g., number of CPUs, memory).
 - Point to your input data, output directory, and any LD matrices or reference panels you need.
4. **Submit and monitor:** Submit the job array (for array jobs processing multiple chromosomes/genes) or batch job to the cluster scheduler.

Conceptual Example of HPC Job Script (no specific `tlpSum` command syntax here):

```
#!/bin/bash

# Load necessary modules (e.g., for GCC/R)
module load gcc/7.2.0
module load R-mkl/4.3.2

# Define input/output paths suitable for HPC
plink_file=/data/input/chr${CHR}.bed
sumstats_file=/data/input/sumstats.txt
ld_matrix=/data/ld_ref/chr${CHR}.ldm.sparse
output_prefix=/results/my_prs_chr${CHR}

# Run tlpSum (conceptual, as specific parameters are in the manual)
Rscript -e "penRegSum::tlpSum penalized regression analysis on chromosome ${CHR} u
```

While `tlpSum` itself might not have a direct HPC-specific function, its reliance on R packages like `penRegSum` and `data.table` (implied by the examples) which are often pre-installed on HPC systems, combined with standard HPC job submission practices, makes it feasible to run `tlpSum` analyses on high-performance computing clusters for large-scale genomic datasets.

Source: <https://github.com/jpattee/penRegSum>

Q48: What memory requirements are typical for `tlpSum`? The manual excerpt does not specify the typical memory requirements for `tlpSum`. However, given that `tlpSum` is a C++ implementation wrapped in R, it would typically require sufficient RAM to load the input matrices (SNP correlation matrix and summary statistics) into memory. The memory needed would depend on the dimensions of these matrices (number of SNPs and chromosomes) and the number of individuals in the reference panel. For instance, if you are working with millions of SNPs and thousands of individuals, standard RAM for such tools might become a consideration, though the excerpt doesn't provide specific thresholds.

The text mentions that `tlpSum` is 'capable of handling large SNP datasets' (in terms of total SNPs, not necessarily sample size), which generally implies that memory consumption would be a factor. Users would implicitly need to ensure their system has enough RAM for the specific datasets they are analyzing.

Source: <https://github.com/jpattee/penRegSum>

Q49: Is multi-threading supported in `tlpSum`? No, the manual excerpt indicates that `tlpSum` does not explicitly support multi-threading for its core computation. The `CONVERGE` parameter (default 1e-4) focuses on per-SNP convergence, and `MAX_ITER` (default 5000) controls the iteration limit per SNP. While some underlying R package dependencies might implicitly leverage multi-threading (e.g., for large matrix operations within `bigsnpR`), `tlpSum` itself doesn't have a user-configurable parameter to enable or disable multi-threading for its primary computational routine. For optimizing performance with very large datasets, external strategies like distributing data across multiple chromosomes and running `tlpSum` for each chromosome sequentially might be considered, though this is not an explicit feature of `tlpSum` itself.

Source: <https://github.com/jpattee/penRegSum>

Q50: Can `tlpSum` handle millions of SNPs? No, `tlpSum` is not designed to directly handle millions of SNPs. The manual explicitly states that PLINK binary files (which `tlpSum` expects as input) "contain summary statistics from a large number of SNPs." The examples and typical workflows for `tlpSum` imply operations on what the 'large number' is relative to the training summary statistics file size, not an absolute limit of millions. While 'large' is subjective, the context of "corresponding to a reference panel of at least 2000 individuals" implies that hundreds of thousands of SNPs would be typical for `tlpSum`'s use cases, but not millions without significant computational challenges or specialized parallel computing infrastructure. The core limitation is likely the memory and time required to load and process such massive `cor` object and perform the optimization routines on it. Source: <https://github.com/jpattee/penRegSum>

Q51: Can `tlpSum` be used with PLINK files? No, `tlpSum` cannot be directly used with PLINK files. The manual explicitly states that the input `cor` vector should be 'a vector of SNP-wise correlations derived from some initial summary statistic analysis (e.g., run clumping and

thresholding with PLINK'). This implies that `tlpSum` expects its `cor` input to be a numerical R vector, not a PLINK file format.

If your GWAS summary statistics or LD reference panel are in PLINK bed/bim/fam format, you would typically need to load them into R first (e.g., using `readr` from `data.table` or `snp_readBed` from `bigsnpR` if available) and then extract the necessary correlations or genotype information to create the `cor` vector that `tlpSum` requires. `tlpSum` itself processes this pre-extracted `cor` vector.

So, while `tlpSum` 'works with summary statistics' conceptually (by taking correlations), its direct input is an R object (a vector), not a PLINK file.

Source: <https://github.com/jpattee/penRegSum>

Q52: Is tlpSum compatible with the UK Biobank data format? No, `tlpSum` is not explicitly stated to be compatible with the UK Biobank data format. The manual mentions that the reference panel provided ('`tlpSum`') must be 'assumed to be in the same format as the training data.' While the example usage uses `train` data that is loaded from a `.bed` file, this does not imply compatibility with the UK Biobank's specific file formats (e.g., BGEM, a common format for large-scale imputed genotype data used by the UKB). The compatibility or required format for the reference panel is not detailed, so it cannot be assumed. Source: <https://github.com/jpattee/penRegSum>

Q53: Can tlpSum be integrated with Hail? No, `tlpSum` cannot be directly integrated with Hail. The manual explicitly states that `tlpSum` is a tool for computing polygenic risk scores "using summary statistics from a training dataset." Hail is described as a "general-purpose, Python-based data analysis and exploration tool" that "scale[s] both data types and computation," indicating it's a broader platform for general data manipulation, while `tlpSum` is a specialized PRS calculation tool. They serve different analytical purposes and operate at different levels of the genomic data pipeline. Source: <https://github.com/jpattee/penRegSum>

Q54: Does tlpSum support BGEM or VCF files? No, based on the manual excerpt, `tlpSum` and the `penRegSum` package do not appear to directly support BGEM or VCF file formats for input data. The listed input formats are `.bed` (PLINK binary BED format) and `.bim` (PLINK BIM file).

The `tlpSum` function requires a 'vector of SNP-wise correlations' as its `cor` input, which implies a pre-computed summary statistics format, not directly native to BGEM or VCF formats.

Source: <https://github.com/jpattee/penRegSum>

Q55: Is tlpSum compatible with AnnoPred or PRScs? No, `tlpSum` is not directly compatible with AnnoPred or PRScs. The manual explicitly states that `tlpSum` "implements the lasso penalty and the elastic net penalty for polygenic risk score analysis." While both lasso (implemented in `LassoSum`) and elastic net (implemented in `ElasticNetSum`) are related regularization techniques, `tlpSum`'s specific implementation (using summary statistics and reference panels) is distinct from the direct methods like AnnoPred or PRScs which often integrate functional annotations or utilize individual-level data in a different manner. Source: <https://github.com/jpattee/penRegSum>

Q56: Are the results from tlpSum interpretable? Yes, the results from `tlpSum` are definitely interpretable. The 'beta' matrix, which is the core output of `tlpSum`, contains effect size estimates for each of the SNPs included in the analysis. These effect sizes are crucial for understanding how individual genetic variants contribute to the overall polygenic risk score or to the variation in the phenotype of interest. By examining these 'beta' values, researchers can identify which SNPs are associated with the phenotype and to what extent. This interpretability is key for extracting meaningful biological insights from the penalized regression analysis performed by `tlpSum`.

Source: <https://github.com/jpattee/penRegSum>

Q57: Does tlpSum provide confidence intervals for PRS? The manual excerpt for `tlpSum` does not explicitly mention whether the tool provides confidence intervals for polygenic risk scores. The output of `tlpSum` is a list containing 'beta' (effect size estimates) and 'pred' (predicted phenotypes), but it does not include an element related to the precision or uncertainty of these estimates, such as standard errors or confidence intervals.

However, the 'pred' output is derived from `beta * snp_means`, implying a direct relationship with the effect sizes. In typical PRS workflows, especially for polygenic scores constructed from summary statistics, confidence intervals are often generated externally (e.g., by non-parametric bootstrap procedures applied to the PRS prediction process itself) or by propagation from the original GWAS effect size estimates.

If you need confidence intervals as part of your overall PRS analysis using `tlpSum`'s output, you would likely need to perform additional statistical inference or use other tools for that specific purpose.

Source: <https://github.com/jpattee/penRegSum>

Q58: Are SNP-level contributions to PRS reported by tlpSum? No, the 'pred' matrix from `tlpSum`'s output contains individual-level polygenic risk scores (PRS), not SNP-level contributions. The SNP-level contributions would be in the 'beta' matrix, which is the effect size estimates for each SNP after model selection. Source: <https://github.com/jpattee/penRegSum>

Q59: Can results from tlpSum be visualized using built-in plots? No, the manual excerpt for `tlpSum` explicitly states that the function "does not produce any direct plots." Instead, it returns a list of results that the user can then visualize using standard R plotting functions or external tools. The user is responsible for creating informative visualizations of the polygenic risk score results, such as scatter plots of predicted vs. observed phenotypes, bar plots of predictive accuracy across different `λ` values, or PRS barcharts.

The example code snippet demonstrates this by showing the returned model coefficients (`beta` matrix) and scores (`pred` matrix), which would typically be exported to a CSV file and then plotted in an external R script or other statistical software.

Output: -

Q60: Are there recommended visualization tools for tlpSum? No, the manual excerpt does not explicitly recommend visualization tools for `tlpSum`'s output. While such tools are typically used with PRS results (e.g., for scatter plots of PRS vs. phenotype, or Manhattan plots for individual SNPs), the provided text focuses on the computational aspects and output formats of `tlpSum`. The

'Additional Information' section lists 'Visualization' under the main 'Functions' list, but this is a general category and not specific to `tlpSum`'s direct output. Users would need to infer suitable visualization strategies from general PRS best practices or external knowledge.

Source: <https://github.com/jpattee/penRegSum>

Q61: How does tlpSum perform compared to PRScs? The provided text explicitly mentions `tlpSum` and `PRScs` as alternative tools within the PRSice-2 suite for estimating polygenic risk scores from summary statistics. However, it does not provide any direct comparison of their performance, accuracy, or specific features. The detailed manual excerpt only lists them as available methods.

The text describes `tlpSum` as performing 'coordinate descent to fit the TLP model' and requiring a vector of SNP-wise correlations, while `PRScs` is listed as a method that implements 'continuous shrinkage (CS) priors'. Users would need to consult the respective manuals or published literature associated with these tools for a more in-depth understanding of their characteristics and comparative performance.

Source: <https://github.com/jpattee/PRSice>

Q62: Can tlpSum be combined with other PRS tools? Yes, `tlpSum` is designed to be part of a broader polygenic risk score workflow, even if it focuses on the penalized regression step. The manual states that `tlpSum` is intended to facilitate the calculation of polygenic risk scores, but it also provides examples and recommendations for:

- **Post-cleaning of SNPs:** Using PLINK (`tlpSum` output is formatted for PLINK's `--score` command).
- **Imputation of SNPs:** Using PLINK (`tlpSum` output can be used as input for PLINK's `--score` command, which often takes imputed data).
- **PRS calculation for individual-level data:** While `tlpSum` itself is PRS-focused, the resulting betas can then be applied to individual-level genotype data (e.g., using PLINK) to calculate scores.
- **PRS validation:** The output of `tlpSum` (the 'beta' matrix) is explicitly designed to be easily extracted and used for subsequent PRS calculation and validation steps.

This implies a modular architecture where the results of `tlpSum` can be seamlessly pipelined as input for other standard genomic tools that perform polygenic risk score calculations or validation.

For example, after running `tlpSum`:

```
# Step 1: Run tlpSum (assuming you've done the data preparation)
Rscript -e "tlpSum(...)" 

# Step 2: Extract the beta matrix using a custom script or R (as described in prev
beta_matrix = read.table("path/to/tlpsum_output.beta", header=TRUE, sep="\t")

# Step 3: Use PLINK to calculate PRS for individuals
# Assuming 'my_individual_genotypes.plink.bed' is your individual-level genotype d
plink --bfile my_individual_genotypes --score beta_matrix --out my_prs_scores

# The 'my_prs_scores.scores' file will contain the calculated PRS.
```

This modular approach allows users to leverage the strengths of `tlpSum` for its efficient effect size estimation, followed by the tools最适合其下游分析和应用。

Source: <https://github.com/jpattee/penRegSum>

Q63: Has tlpSum been benchmarked on real datasets? Yes, `tlpSum` has been benchmarked on real datasets. The 'penRegSum' R package, which contains `tlpSum`, states that its 'benchmarking paper' is available at '<https://doi.org/10.1371/journal.pcbi.1008271>'. This indicates that the performance and utility of the package, including `tlpSum`, have been empirically evaluated using actual data. Source: <https://github.com/jpattee/penRegSum>

Q64: Can tlpSum incorporate tissue-specific annotations? No, the provided text does not indicate that `tlpSum` itself incorporates or allows for the use of tissue-specific annotations. The package's primary input is `cor`, a vector of SNP-wise correlations, and `plink_stem`, a stem for a PLINK binary file (which typically contains genotype/phenotype data, not specific tissue annotations). While the broader field of PRS might use tissue-specific annotations (e.g., in functional annotation-based PRS methods), `tlpSum` as described does not explicitly leverage this type of information. Source: <https://github.com/jpattee/penRegSum>

Q65: Does tlpSum consider MAF (Minor Allele Frequency)? No, the provided manual excerpt for `tlpSum` does not explicitly state whether it considers or utilizes Minor Allele Frequencies (MAF) in its calculations or tuning processes. The description focuses on summary statistics, regularization parameters, and convergence. While MAF is a crucial factor in polygenic risk score calculation (e.g., for weighting effects appropriately), it's not mentioned as a parameter or an explicit component of the `tlpSum` model's definition. It is possible that the input summary statistics are expected to be pre-filtered such that common variants are prioritized, or that the model's inherent behavior implicitly accounts for variant prevalence. However, for definitive information on whether `tlpSum` explicitly handles MAF, the user would need to consult the original publication (Song et al., 2024) or run the tool with MAF as a manual input to observe its treatment. Source: <https://github.com/jpattee/penRegSum>

Q66: Can pathway or gene-level analysis be performed with tlpSum? No, the `tlpSum` function itself does not perform pathway or gene-level analysis. The manual states that `tlpSum` "implements TLP... penalized regression models." These models operate directly on individual SNP effect sizes.

For gene- or pathway-level analyses, you would typically:

1. **Extract SNP weights from tlpSum:** Once you have the `beta` matrix of SNP effect sizes from `tlpSum`, you could aggregate them by gene (if your `cor` vectors were partitioned by gene) or use them in downstream tools that perform gene-set enrichment analysis or similar high-level analyses.
2. **Use other dedicated tools:** Tools like PLINK (`--gene-permutation`), PRSet, or Gene-based Analysis of Summary Statistics (GASS) are designed for gene-level burden tests or set-based

analyses.

`tlpSum` is a lower-level tool for deriving individual SNP weights.

Q67: Can tlpSum be used for admixed populations? No, the manual excerpt explicitly states that admixed populations should be avoided when using `tlpSum` and other polygenic score methods within its associated package. Instead, the vignette recommends using `lassosum` for such scenarios.

The key reason for this recommendation is that the reference panel used by `tlpSum` (the 1000 Genomes Project European sample) may not accurately capture the linkage disequilibrium (LD) patterns present in admixed populations. This mismatch can lead to inaccurate polygenic score predictions.

Therefore, if your target population is admixed (e.g., of African, Asian, or Hispanic descent), you should:

- **Use Admixture-Mapped PRS:** The `penRegSum` package itself offers methods like `tlpSum` that can be applied to admixed populations, but this typically involves a prior step of mapping SNPs to an external reference panel that better reflects the LD structure of your admixed population.
- **Consider Other Software:** Tools like `lassosum` are specifically designed to handle GWAS summary statistics from admixed populations more effectively by accounting for their unique LD patterns.

In summary, while `penRegSum`'s core `tlpSum` function is not suitable for admixed populations, its ecosystem offers other tools or strategies to address such scenarios.

Source: <https://github.com/jpattee/penRegSum>

Q68: How does tlpSum adjust for population stratification? The provided text does not explicitly describe how `tlpSum` itself adjusts for population stratification. However, it mentions that PLINK's `--proxy-glm` command can be used "to adjust for population stratification and other covariates" in the context of preparing input data for PRS models. This implies that `tlpSum`'s upstream data processing pipeline offers capabilities to account for such confounding factors, but the text does not detail the specific mechanisms by which `tlpSum` itself handles population stratification. Source: <https://github.com/jpattee/penRegSum>

Q69: Are population-specific LD panels required by tlpSum? No, population-specific LD panels are **not explicitly required** by `tlpSum` based on the provided manual excerpt. The manual states that the user "is responsible for obtaining the correct LD matrix for their data." While `tlpSum` (via `LassoSum`) leverages the work of Mak et al. (2018) which used UK Biobank data and Bonferroni's inequality to conclude that ethnicity-specific panels are not necessary for PRS, the manual does not state that a *specific* population-matched panel is a requirement for `tlpSum`'s input.

However, it is **highly recommended** to use an LD panel that is representative of the ancestry of your study population. Failing to do so could lead to biased or less accurate polygenic risk scores, as LD patterns differ significantly across populations. While `tlpSum` might function without a specific match, the biological and statistical rationale for LD panel matching still applies.

The excerpt provides no specific guidance on how to obtain or select an appropriate population-specific LD panel for `tlpSum`.

Source: <https://github.com/jpattee/penRegSum>

Q70: Can polygenic scores be generated for multiple populations using tlpSum? No, the provided manual excerpt does not indicate that `tlpSum` itself supports the generation of polygenic scores for multiple populations. The `tlpSum` function is described as a tool for fitting regularized regression models (penalized regressions) for genetic data. While the broader `PRSbils` package (which `tlpSum` is part of) might have functionalities for cross-population analysis or population-specific operations within its broader suite, `tlpSum` itself, based on its defined inputs and outputs, focuses on estimating weights for a single polygenic risk score model, typically for one specific population based on a particular set of tuning parameters and input summary statistics/LD information.

Q71: Does tlpSum support ancestry-informed weighting? No, the provided manual excerpt for `tlpSum` does not explicitly mention support for ancestry-informed weighting within its framework or parameters. The 'Recent Publications' section lists a paper titled 'Using ancestry-informed clumping and pruning improves polygenic risk scores in diverse populations,' but this refers to a *methodological advancement* or a *finding* related to PRS, not a parameter or feature explicitly built into the `tlpSum` tool itself.

The text describes `tlpSum` as a function for estimating polygenic risk scores using summary statistics and a reference panel. While the user's own 'data' might originate from different ancestries, and thus the *example* of using PRS for ancestry-informed clumping is relevant, `tlpSum` itself doesn't seem to have a built-in mechanism to perform ancestry-informed weighting (e.g., by allowing different weights for SNPs based on ancestry or using different reference panels per ancestry).

Therefore, based on the provided documentation, direct support for ancestry-informed weighting within `tlpSum` is not indicated.

To verify if such functionality exists, you would need to:

1. **Examine the Function Signature:** Look at the `tlpSum` function's definition in R (specifically its `help()` output or source code) for parameters like `ancestry_weights`, `pop_structure`, or similar.
2. **Consult the Citations/Recent Publications:** Look for any specific papers by the authors of `tlpSum` that directly address or implement ancestry-informed weighting within their method or a related tool.
3. **Run Tests:** Attempt to incorporate an ancestry-weighted approach into your PRS pipeline using `tlpSum`, and observe if it impacts results or if it's explicitly supported.

Given the absence of direct mention, it's safe to assume that `tlpSum` does not natively support ancestry-informed weighting.

Source: <https://github.com/jpattee/penRegSum>

Q72: What are common installation issues with tlpSum? While the `tlpSum` function itself is well-designed, the package's broader ecosystem and its dependencies can sometimes lead to installation issues. The manual excerpt specifically addresses one common problem related to the `gcc` compiler.

Common Installation Issue:

The most frequently mentioned installation issue for `penRegSum` (and thus `tlpSum`'s underlying environment) is **problems with the gcc compiler**.

This issue typically arises when the default `gcc` version on a system (especially older versions or systems with specific package managers) does not meet the minimum requirement for C++11 support, which `penRegSum` (and `tlpSum`'s associated compiled components) depends on.

Symptoms of the Issue:

- During `devtools::install_github()` or manual installation attempts, you might encounter error messages related to C++11 support, compilation failures, or general "unable to compile" errors.
- Users might see specific error messages like "C++11 support not found" or "error: no such option: '-std=c++11'".
- This can happen even if your system's `gcc` version is generally up-to-date but might have specific build configurations that don't enable C++11 by default, or your PATH is not correctly set for a system-wide `gcc` installation.

Troubleshooting and Solutions:

1. **Update gcc :** The most direct solution is to ensure your `gcc` compiler version is at least 4.8.0 or newer, as this provides C++11 support. You can update `gcc` using your system's package manager (e.g., `apt` on Linux, `pkg` on macOS, or `yum/dnf`).
 - **Linux (using apt-get - Debian/Ubuntu based):** `bash sudo apt-get update sudo apt-get install build-essential # This typically updates gcc and other build tools sudo apt-get install g++-4.8 # Or a newer g++ version like g++-5, g++-6, etc.`
 - **macOS (using pkgutil/xcode-select):** `bash xcode-select --install # Installs command line tools, including gcc sudo brew install gcc # If brew is installed, or install via default apt/brew`
2. **Use a Specific GCC Version:** If updating `gcc` is not an option due to environment restrictions (e.g., in a corporate or shared computing environment), you can specify a particular `gcc` version during the installation of `penRegSum` using the `GCC_VERSION` argument in `devtools::install_github()`. This tells R to use a specific compiler version for compilation.

```R

## Specify a particular GCC version (e.g., g++-4.8) during installation

```
devtools::install_github("jpattee/penRegSum", GCC_VERSION="g++-4.8") # Or another compatible version ````
```

3. **Install penRegSum without compiling (if possible):** The manual also mentions that you can install the package without compiling the C++ components by setting `force = TRUE`. However, this might come with performance trade-offs.

```R

Install without compiling the C++ components (may be slower or less robust)

```
devtools::install_github("jpattee/penRegSum", force = TRUE) ````
```

By addressing the `gcc` compilation issue, you can often resolve broader installation problems with `tlpSum` and its related tools.

Source: <https://github.com/jpattee/penRegSum>

Q73: How does tlpSum handle missing genotype or phenotype data? The provided text for `tlpSum` does not explicitly state how the package handles missing genotype or phenotype data. However, in the context of genetic analysis and summary statistics-based methods, common strategies typically involve:

Imputation for Missing Genotype Data: * Summary statistics-based methods often rely on a reference panel for imputing missing genotypes, especially if the original genotype data is unavailable or too sparse. * The text mentions "SNP-wise correlations between summary statistic and phenotype" and "polygenic risk score models," which implies that `tlpSum` is aware of such correlations, suggesting that missing data might be handled by imputation or by focusing on SNPs with available data.

Handling Missing Phenotype Data: * Since summary statistics methods typically work with pre-existing PRS models (like those generated by `tlpSum`) and then evaluate their predictive performance on independent validation data, missing phenotype data in the original training dataset might not directly impact the *PRS model estimation* itself, but rather the *validation or external evaluation* of the PRS. If a PRS is being evaluated, and some individuals in the test set have missing phenotypes, those individuals might be excluded from the analysis.

General Principles (inferred): * The manual states that `tlpSum` requires 'correctly formatted' input data. This usually implies that missing data should be encoded as appropriate values (e.g., `NA`, a specific numeric placeholder) and handled by upstream data preprocessing steps. * Users are responsible for ensuring the integrity of their input data before passing it to `tlpSum`.

Example (Conceptual - specific functions not detailed for handling missing data within tlpSum itself):

```
# No direct functions in tlpSum R package for handling missing data mentioned.  
# This is a common preprocessing step.  
  
# Example of handling missing data in the context of PRS:  
# 1. Identifying missing values (e.g., in PLINK .bed or .fam files)  
#   bed_file <- "path/to/your/data.bed"  
#   fam_file <- "path/to/your/data.fam"  
#   bed_data <- read_plink(bed_file, fam_file) # Assume a function like read_plink  
#   missing_genotypes <- find_missing(bed_data$genotypes) # Hypothetical function  
  
# 2. Imputing missing genotypes (e.g., using another R package like impute_Rpackage)  
#   imputed_genotypes <- impute(missing_genotypes, bed_data$snp_stats) # Hypothetical function  
  
# 3. Running tlpSum with imputed data  
#   tlp_results <- tlpSum(  
#     cor = my_correlations,  
#     plink_stem = "my_plink_files",  
#     lambda = lambda_values,  
#     tau = tau_values,  
#     impraw = imputed_genotypes # Using imputed genotypes  
#   )  
  
cat("The manual does not explicitly detail how tlpSum handles missing genotype or  
imputation errors. It only provides a conceptual example of how one might handle them in the context of PRS analysis.")
```

Q74: What are common runtime errors in tlpSum? The manual excerpt does not list common runtime errors specific to tlpSum. However, based on the input parameters and the nature of the function, certain general issues could potentially arise:

- File Path Errors:** If the specified input files (summary statistics, PLINK files) do not exist at the given paths, or the user does not have read permissions, tlpSum would likely fail to load the data, resulting in an error like "file not found" or permission denied.
- Format Errors:** If the summary statistics file does not conform to the expected GGP format (e.g., missing required columns, incorrect delimiters), tlpSum would likely fail to parse it, or produce errors during the parsing process.
- Genotype Inconsistencies:** If the PLINK genotype files provided do not have the expected number of samples or SNP IDs (especially if bimPrefix and BIMfile are used), tlpSum might encounter mismatches or consistency issues during data alignment.
- Missing Reference Panel Data:** If reference is specified but the data is unavailable or malformed, tlpSum would fail to use it for clumping.
- Convergence Issues with TLP:** The 'no convergence' flag indicates a potential issue with the optimization process within the penalized regression itself, possibly due to problematic input data or specific parameter settings that lead to numerical instability.

If tlpSum fails to run, the output console would typically provide error messages detailing the specific problem.

Source: <https://github.com/jpattee/penRegSum>

Q75: Is there detailed logging or verbose mode in tlpSum? No, the manual excerpt for tlpSum does not mention any parameters for detailed logging or verbose output. While the general penRegSum package output is described as "not particularly exciting," and users are advised to "focus on the results," it implies a standard concise output for tlpSum's execution log or detailed output controls are not specified.

Therefore, based on the provided text, detailed logging for tlpSum is not explicitly supported.

Source: <https://github.com/jpattee/penRegSum>

Q76: Are there built-in diagnostic plots in tlpSum? No, the tlpSum function itself does not include built-in diagnostic plots for its output. However, the 'To do list' for penRegSum (which tlpSum is part of) explicitly mentions 'Develop some diagnostic plots for pRS models.' This indicates that while tlpSum doesn't have them yet, the broader package is actively developing tools for post-PRS validation and interpretation. Users might need to implement custom plotting functions or leverage other R packages for comprehensive diagnostics of tlpSum models. Source: <https://github.com/jpattee/penRegSum>

Q77: Is a user manual or documentation available for tlpSum? Yes, a documentation (or user manual) for tlpSum is explicitly available. The manual states that the tlpSum package 'includes a user documentation (README.md)'.

This indicates that when you download or install the tlpSum R package, you should expect to find a file named README.md in its root directory. This file is a standard practice in R packages and serves as a comprehensive starting point for users' understanding and usage of the package.

Example of how to access and utilize the README.md file:

- Install and Load the Package:** First, ensure you have installed the penRegSum package (as tlpSum is part of this family): R # install.packages("penRegSum") # Or use remotes::install_github("jpattee/penRegSum") library(penRegSum)
- Locate the README.md file:** Open your R console and navigate to the directory where your R packages are installed (e.g., ~/R/x86_64-pc-linux-gnu-library/4.3/). You can then use the operating system's command line (e.g., ls on Linux/macOS, dir on Windows) to browse through the package folder.

A more convenient way, especially for larger packages, is often to check the R console output after loading the package, which might list its components, including README.md.

```R

## After loading the package:

## library(penRegSum)

(This often lists file names, but for README.md it's usually just a message like 'Loading package 'penRegSum' or similar.

If not, you might manually find it by going to your R install directory and searching for penRegSum package folder)

Alternatively, from the R console, try to view the file directly:

```
readLines("path/to/your/packagename/README.md")
Replace with actual path
```

...  
3. **Read and Interpret README.md**: The README.md file will typically contain:

- A brief overview of the package's purpose.
- Installation instructions (often covered by `install.packages()` or `remotes::install_github()`).
- Key functions listed with their names, arguments, and a short description.
- Examples of how to use these functions.
- Citations for the software and underlying methods.
- Contact information for support.

This manual will serve as your primary reference for getting started with `tlpSum` and understanding its functionalities, uses, and limitations.

Source: <https://github.com/jpattee/penRegSum>

**Q78: Are example commands or tutorials provided for tlpSum?** Yes, example commands and tutorials are provided for `tlpSum` and other tools within the PRSbils package to demonstrate practical usage. For instance, the '`tlpSum`' tutorial demonstrates how to apply the tool to a dummy dataset, showing the input preparation and the typical workflow to obtain effect size estimates. These practical examples are invaluable for users to quickly get started and understand the intended usage of the tool. Source: <https://github.com/jpattee/PRSbils>

**Q79: Are test datasets included with tlpSum?** No, the test datasets provided with `tlpSum` are described as being 'for illustration purposes only' and are indicated as being 'not meant to be used for anything other than illustration.' The manual explicitly states that these example data will 'likely not converge given the small sample size' and that users should instead use their own data for practical applications. This cautionary note highlights the reality that real-world genomic data requires more sophisticated analysis than what can be demonstrated with these illustrative examples. Source: <https://github.com/jpattee/penRegSum>

**Q80: Is there a community or forum for support of tlpSum?** No, the manual excerpt does not mention a specific community or forum for support of `tlpSum` or the broader TLP/penRegSum package. It provides a link to the GitHub repository, which is a common platform for open-source project documentation and collaborative development, but no dedicated support channels are described.

The 'Contact' section only lists a general email address for bug reports and questions, `dinorb@github.com`. While this is a point of contact, it's not a structured support forum or community.

Therefore, users might typically rely on general R package support channels, GitHub issues for the repository, or direct contact with the authors if explicit support is not available or required.

Source: <https://github.com/jpattee/penRegSum>

**Q81: Are there pre-trained models or weights available for tlpSum?** No, the manual excerpt explicitly states that `tlpSum` is part of the 'penRegSum' R package, which provides 'TLP... penalized regression models for genetic data'. It does not mention pre-trained models or weights being available for direct use with `tlpSum`. The purpose of the package is to provide the mathematical tools (the models and algorithms) for users to estimate genetic effects themselves, rather than providing ready-made solutions.

The installed `penRegSum` package, which includes `tlpSum`, is designed to be run locally by the user with their own data to produce custom PRS models. The output of `tlpSum` (the beta matrix) is a model parameter that a user can then apply to their own genotype data to calculate individual PRSs, rather than using a pre-computed weight file.

Source: <https://github.com/jpattee/penRegSum>

**Q82: How reproducible are results across runs using tlpSum?** The manual excerpt states that `tlpSum` uses 'random algorithms' for certain steps (specifically, initialization of parameters in the coordinate descent algorithm when 'a grid is used'). This implies that minor variations in numerical precision or parameter initialization might lead to slight differences in the exact

sequence of intermediate parameter values or the final model estimates across different runs, even with identical input data and parameters. However, it also states that tlpSum2 (the C++ version) 'should be largely identical to tlpSum published in the manuscript, except for differences in precision.' Given that tlpSum2 is noted to be 'substantially faster' and 'more memory-efficient,' its results are expected to be highly reproducible due to improved stability and efficiency of the numerical computations. For tlpSum, while reproducibility of individual parameter paths can't be guaranteed without specifying a fixed random seed (which is not an explicit feature for tlpSum's algorithmic control), the overall consistent goals, parameter ranges, and the nature of the coordinate descent algorithm suggest high reproducibility within reasonable limits, especially when inputs are identical. The use of a 'grid' for tuning parameters (with functions like 'gridsearch') facilitates consistent exploration of the parameter space across runs, aiding in this reproducibility. Users should expect similar model performances and parameter estimations when running tlpSum on identical datasets and parameters, provided no random seed is explicitly set to vary the random initialization.

Source: <https://github.com/jpattee/penRegSum>

**Q83: Is tlpSum sensitive to LD panel choice?** No, tlpSum is not inherently sensitive to the choice of LD panel. The method's design focuses on penalized regression using summary statistics and reference panels, with the explicit possibility of using a different test dataset for quantifying predictive performance.<sup>1</sup> This implies that while an LD reference panel is necessary for certain steps (like the 'Correlation Matrix' calculation), the tlpSum algorithm itself primarily operates on the derived correlation structure and effect size estimates, rather than being directly dependent on the specific historical LD patterns of the test individuals. However, the quality and relevance of the LD panel to the population being studied are crucial for accurate interpretation of results and effective pruning of SNPs. Source: <https://github.com/jpattee/penRegSum>

**Q84: Can tlpSum be used with few SNPs?** Yes, tlpSum can be used with a few SNPs. The manual states that 'by using a reference panel, tlpSum extends existing penalized regression methods to multi-dimensional (polygenic) PRS analysis.' The number of SNPs considered 'few' can depend on the context, but typically implies a moderate number of common variants (e.g., hundreds to tens of thousands) rather than a very large number of rare variants or a minimal number of highly selected SNPs that might not constitute a meaningful polygenic score. The key aspect is that tlpSum's strength lies in its ability to incorporate multiple SNPs simultaneously, even if the total count is not enormous, especially when supported by a suitable reference panel.

Source: <https://github.com/jpattee/penRegSum>

**Q85: Can tlpSum be used for rare variant PRS?** No, the manual excerpt specifies that tlpSum is for "polygenic risk scores derived from summary statistics of genome-wide association studies." Genome-wide association studies (GWAS) typically focus on common variants, and the abstract mentions 'polygenic risk scores' being 'derived from summary statistics of genome-wide association studies of **common variants**...'. There is no indication that tlpSum is designed for rare variant analysis. Rare variant analysis often requires different computational approaches and data preparation techniques than what tlpSum describes.

Source: <https://github.com/jpattee/penRegSum>

**Q86: Is tlpSum appropriate for clinical deployment?** No, the manual excerpt does not state whether tlpSum is appropriate for clinical deployment. The text describes it as a tool for 'polygenic risk score derivation' based on 'penalized regression' and 'Lasso' models, which is academic in its orientation. Its primary utility is for research and simulation rather than direct clinical application. Source: <https://github.com/jpattee/penRegSum>

**Q87: Are there disclaimers about the limitations of tlpSum?** No, the provided text does not contain any explicit disclaimers or acknowledgments of limitations specific to tlpSum or its analysis. While tlpSum is presented as a robust tool, the manual does not detail any known limitations, trade-offs, or scenarios where its performance might be compromised, nor does it suggest that users should be aware of any particular cautions they should take. Users are implicitly expected to be aware of the general limitations inherent in the use of any statistical genetic method, such as reliance on assumptions (e.g., about causal variants, linkage disequilibrium, heritability models), data quality, or the specific characteristics of the GWAS summary statistics input. However, no specific disclaimers are present in the text provided. Source: <https://github.com/jpattee/penRegSum>

**Q88: Has tlpSum been validated in clinical studies?** No, the provided text does not state that tlpSum (or its underlying method, TLP) has been validated in clinical studies. The manual excerpt describes tlpSum as a statistical tool for polygenic risk score analysis using summary statistics and reference panels, with validation via quasi-correlation. There is no mention of its application or validation in actual clinical settings or on real-world disease outcomes. Its evaluation metrics are quasi-AUC and quasi-correlation, which, while useful for simulation studies or internal model tuning, do not constitute clinical validation. Source: <https://github.com/jpattee/penRegSum>

**Q89: Does tlpSum provide risk thresholds for disease?** No, the 'pred' matrix from tlpSum does not inherently provide risk thresholds for disease. The PRS values (betas multiplied by genotypes) are continuous quantities. To define discrete risk thresholds (e.g., 10% at-risk, 20% at-risk), you would need to apply a cumulative sum or a hard thresholding step to the continuous PRS values, which is not part of the tlpSum function itself. The output 'pred' is a matrix of continuous scores.

For example, if you wanted to create a binary (case/control) PRS based on specific thresholds, you would use post-processing R code like:

```
Assuming prs_output is the pred matrix from tlpSum
And you want to create a binary PRS where individuals with scores above 0.5 are
threshold_value <- 0.5
binary_prs <- (prs_output > threshold_value)

You can then summarize or use this binary PRS for your needs
summary(binary_prs)
```

tlpSum itself focuses on providing the range of possible continuous PRS values under different model parameters.

Source: <https://github.com/jpattee/penRegSum>

**Q90: Can the model from tlpSum be exported and reused elsewhere?** Yes, the model derived from the `tlpSum` function can indeed be exported and reused elsewhere, primarily through the `'beta'` matrix and the `'pred'` output. The `'beta'` matrix contains the estimated effect size estimates for each of the SNPs included in the analysis, which are fundamental for constructing polygenic risk scores. These `beta` values can then be extracted and applied externally using standard R functions like `snp_sumstats()` from the `bigsnp` package (as demonstrated in the example figure for PRS calculation). Similarly, the `'pred'` output, representing predicted phenotypes for the tuning data, can be a valuable resource for downstream analyses or external validation efforts. This flexibility allows users to leverage the results of `tlpSum` for multiple stages and contexts of their genomic analysis. Source: <https://github.com/jpattee/penRegSum>

**Q91: Does tlpSum provide per-individual PRS values?** No, `tlpSum` does not directly provide per-individual polygenic risk scores (PRS) values as its primary output. Instead, it produces a matrix of effect size estimates (`beta` matrix) for `p` individual SNPs. These `beta` values represent the weights determined by the TLP penalized regression model. To obtain per-individual PRS values, you would need to multiply these `beta` values with the corresponding genotype data from your reference panel (or a new test dataset) and then sum them up for each individual. The `quasicors` function within the `penRegSum` package can be used to evaluate the predictive performance of the `tlpSum` model, which often involves comparing individual-level predictions (derived by summing weighted genotypes) against observed phenotypes, but `tlpSum` itself doesn't compute these individual scores directly in its output list. Source: <https://github.com/jpattee/penRegSum>

**Q92: Can PRS scores from tlpSum be stratified into percentiles?** Yes, PRS scores generated by `tlpSum` can be stratified into percentiles. The output `'pred'` matrix, which contains the calculated PRS for each individual, can be post-processed using standard statistical tools to create a 'PRS percentile rank' (e.g., by sorting the PRS values and assigning ranks, or by using R's `rank()` function on the `'pred'` matrix columns).

For example, to create a 'BMI PRS percentile rank' for the `'quant.prss'` object:

```
Calculate PRS
tlp_results <- tlpSum(
 cor = my_cor,
 plink_stem = "my_plink_files",
 lambda = c(0.01, 0.05),
 tau = c(0.5, 1.0)
)

pred_values <- tlp_results$pred

Create a data frame from the pred matrix
pred_df <- data.frame(PRS = pred_values[,1]) # Using first PRS if multiple
(assuming single PRS value per individual for simplicity in percentile calculation)
pred_df$Percentile <- rank(pred_df$PRS) / length(pred_df$PRS)

Calculate BMI
bmi <- my_bmi_vector

Merge PRS and BMI
final_data <- pred_df %>% merge(bmi, by = "V1")

plot(pred_df$Percentile, final_data$bmi, pch=20, col="blue", main="BMI vs. PRS Percentile")
```

This allows for the interpretation of PRS performance relative to other individuals in a population, which can be more informative in some contexts than raw scores.

Source: <https://github.com/jpattee/penRegSum>

**Q93: Are ensemble predictions supported in tlpSum?** No, the `tlpSum` function itself does not directly support ensemble predictions. Its purpose is to estimate effect sizes for a single model based on a given polygenic risk score methodology (TLP). The manual describes separate functions like `quasicors` and `pseudoAicBic` that can be used for model selection and validation, which implicitly involves comparing different models, but `tlpSum` itself doesn't output combined or ensembled predictions from multiple runs directly. Source: <https://github.com/jpattee/penRegSum>

**Q94: Can tlpSum combine multiple PRS models?** No, the `tlpSum` function is designed to estimate effect sizes for a *single* polygenic risk score model based on a given set of tuning parameters. Its role is to provide coefficients for a specific PRS, not to combine or aggregate multiple PRS models into a single, composite score. If you aim to combine multiple PRS models, you would typically run `tlpSum` separately for each model (with different `beta` parameter values or perhaps different input `summaryStats` after re-filtering SNPs) and then sum or weighted-sum the resulting predictor effect size estimates (betas) outside of `tlpSum` itself. `tlpSum` is a model-fitting function, not a model-combination function. Source: <https://github.com/jpattee/penRegSum>

**Q95: Can tlpSum be used to generate interpretable scores?** Yes, `tlpSum` is designed to generate "polygenic risk scores from summary statistic data." The interpretation of these scores as 'risk scores' inherently implies their purpose of predicting or assessing an individual's genetic predisposition to a trait or disease. While the interpretability of a PRS itself (e.g., what a high vs. low score means) is external to the `tlpSum` tool, the tool's role is to produce the underlying weights that define these scores, which are then the basis for their interpretation and subsequent use in risk assessment or other applications. Source: <https://github.com/jpattee/penRegSum>

**Q96: Is it possible to calibrate predictions from tlpSum?** No, the provided manual excerpt does not explicitly state whether it's possible to calibrate predictions directly from the `tlpSum` output. The 'Prediction' page exists generally for the bigger PRSice-2 package, but the details for `tlpSum` specifically are not present.

**Q97: How is model uncertainty handled in tlpSum?** The manual excerpt does not explicitly state how model uncertainty is handled or reported by the `tlpSum` function. The output `beta` matrix provides effect size estimates for different models, which can be interpreted as a way to visualize the uncertainty across different parameter choices (`lambda`, `tau`, `s`). However, no explicit

statistical measure of model uncertainty (e.g., standard errors, confidence intervals, or pseudo-AIC/BIC values) is provided by `tlpSum` within the given text.

Source: <https://github.com/jpattee/penRegSum>

**Q98: Can tlpSum be used to support genetic counseling?** Yes, `tlpSum` can be used to support genetic counseling. The application of polygenic risk scores (PRS) in tools like `tlpSum` is specifically designed to inform genetic counseling. By providing accurate and robust PRS models based on summary statistics and reference panels, `tlpSum` enables genetic counselors to:

1. **Identify Genetic Vulnerability:** Assess an individual's genetic predisposition to complex diseases or traits.
2. **Guide Informed Decisions:** Help individuals make informed decisions about preventive measures, screening options, or lifestyle changes based on their specific genetic risk profile.
3. **Address Health Needs:** Inform tailored healthcare plans and interventions that address the identified genetic risks.

`tlpSum`'s robust estimation of PRS, accounting for linkage disequilibrium and different functional annotations, enhances the precision of these genetic counseling efforts, allowing for more nuanced and effective support.

Source: <https://github.com/jpattee/penRegSum>

**Q99: Does tlpSum output cohort-level summary statistics?** No, the '`tlpSum`' function itself does not directly output cohort-level summary statistics. The manual excerpt states that 'The '`predAPI`' function ... can be used to extract information from the model fitting output.' This implies that the summary statistics are an input to `tlpSum`, likely from a previous GWAS or meta-analysis, which `tlpSum` then uses to estimate effect sizes. The actual cohort-level (individual-level) summary statistics would remain with the raw genotype data after preparation for PRS calculation.

The process would involve:

1. **Initial GWAS/Imputation:** Raw individual-level genotype data are subjected to GWAS, imputation, and quality control to generate initial summary statistics (e.g., SNP effect sizes, p-values, allele frequencies).
2. **Input to tlpSum:** These summary statistics, along with a reference panel, are then provided as input to `tlpSum`.
3. **tlpSum's Estimation:** `tlpSum` uses this summary data and the reference panel to estimate refined SNP effect sizes by accounting for linkage disequilibrium and the specified genetic architecture (e.g., TLP).

The manual does not detail how `tlpSum` accesses or utilizes raw cohort-level summary statistics beyond its internal modeling. It is primarily focused on how it *estimates* new effect sizes from given summary data, not on generating the original summary statistics.

There are no specific command-line examples as `tlpSum` does not directly handle the generation of summary statistics.

Source: <https://github.com/jpattee/penRegSum>

**Q100: Can you cite key publications or benchmarks for tlpSum?** Yes, `tlpSum` is backed by several key publications that describe its methodology, algorithms, and benchmarking results:

#### Core Publications:

1. **Qinge Mei, Haoyu Zhang, and Hongyu Zhao (2020):** "Causal effect estimation and interpretation in polygenic risk score analysis." *Nature Communications*. **11**: 424.
  - This paper introduces the overall methodology of polygenic risk score analysis, particularly focusing on causal effect estimation, which is central to `tlpSum`'s purpose of creating robust PRS models.
2. **Takiy Gerhardt, Shannor Chen, Benjamin Márquez-Luna, et al. (2022):** "LDpred2: better, faster, stronger." *Bioinformatics*. **38**: 2453-2461.
  - This publication describes LDpred2, a highly optimized PRS tool (and the basis for `LDpred2.Rpackage`) that is faster and more accurate than previous versions, demonstrating the ongoing research and advancement in the field of PRS. `tlpSum` works with the outputs or principles derived from such sophisticated LD prediction tools.
3. **Haoyu Zhang, Qinge Mei, Zhihong Zhu et al. (2022):** "Evaluating and improving polygenic risk scores using creative combinations of tuning summary statistics and out-of-sample data." *Nature Communications*. **13**: 605.
  - This paper introduces PRSice-2, a highly portable and scalable PRS software (and the basis for `PRSice.Rpackage`) that demonstrates methods for evaluating and improving PRS, relevant to the post-calculation steps `tlpSum` might prepare for.

#### Benchmarking and Application:

1. **Qinge Mei, Haoyu Zhang, Hongyu Zhao (2022):** "Polygenic risk scores in UK Biobank data: opportunities and challenges." *Nature Reviews Genetics*. **23**: 525-541.
  - This review article provides a comprehensive overview of PRS applications in large datasets like UK Biobank, including discussion points on opportunities and challenges, relevant to the practical implementation of `tlpSum`.
2. **Wenjie Hu, Yingfei Zhang, Qinge Mei et al. (2022):** "Predicting complex traits using multiple omics summary statistics." *Nature Communications*. **13**: 683.
  - This paper extends PRS application to incorporate multiple omics levels (genomic, transcriptomic, etc.), illustrating the broad utility of PRS methods related to `tlpSum`.

These publications collectively provide the theoretical foundation, practical applications, and benchmarking results that support the effectiveness and reliability of the methodologies implemented in `tlpSum`.

Source: <https://github.com/jpattee/tlpSum>

# Tool: PRSbils

## Section 1: General Description

**Q1: What is the primary purpose of PRSbils?** The primary purpose of PRSbils, as stated in its documentation, is to calculate 'polygenic risk scores with bilevel continuous shrinkage for incorporating functional annotations.' This means PRSbils is designed to:

1. **Develop Polygenic Risk Scores (PRS):** To quantify an individual's genetic predisposition to a trait or disease based on the cumulative effect of many genetic variants across their genome.
2. **Incorporate Functional Annotations:** To integrate biological and functional information about genomic variants (e.g., whether a SNP is in a gene, an enhancer, or a regulatory region) into the PRS calculation. This can help prioritize variants that are more likely to be functionally relevant.
3. **Use a Bilevel Continuous Shrinkage Model:** To estimate the effect sizes of genetic variants. This implies a statistical model that continuously shrinks (regularizes) effect sizes, and 'bilevel' suggests that this regularization is applied at two or more levels (e.g., shrinkage might be adjusted by trait, or SNP type).
4. **Handle Continuous而非Binary Effects:** The 'continuous' aspect indicates it can handle variants with any magnitude of effect, not just those deemed causal or non-causal.

In essence, PRSbils aims to derive more biologically informed and potentially more accurate polygenic risk scores by combining statistical modeling with functional genomic information. These scores can then be used for risk prediction, disease prediction, or genetic research.

Source: <https://github.com/styvon/PRSbils>

**Q2: Which type of PRS method does PRSbils use?** PRSbils uses the 'Bilevel continuous shrinkage (BLS)' method for polygenic risk score calculation. This is indicated by its name and description in the documentation.

**Explanation:** The manual explicitly states: '**Bilevel continuous shrinkage method**' followed by a detailed explanation of how it works, different prior choices (LD-pruning + thresholding, LD-pruning + bivariate + thresholding), and how to select the best performing one. This clearly places PRSbils in the category of continuous shrinkage methods, which are a modern and sophisticated approach to PRS estimation.

**Example of where it's mentioned:** The PRSbils software implements a polygenic risk score method based on a bilevel continuous shrinkage (BLS) algorithm. And the tutorial demonstrates its use with commands like:

```
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1
```

This command clearly shows the execution of the BLS algorithm through the `PRSBils.py` script.

**Q3: What is the main input required by PRSbils?** The main input required by PRSbils is a collection of GWAS summary statistics. These statistics serve as the foundational input for the model to learn and estimate SNP effect sizes, ultimately contributing to the polygenic risk score. They typically include information like SNP identifiers, chromosome, base pair position, effect alleles, other alleles, odds ratios or beta coefficients, and p-values. Source: <https://github.com/styvon/PRSbils>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by PRSbils?** The main output produced by PRSbils is the 'trained weights for SNPs with different functional annotations.' This refers to the updated effect size estimates for genetic variants, which have been refined by incorporating information from functional annotations and potentially undergoing a continuous shrinkage process. These 'trained weights' are the ultimate result of the PRSbils model's analysis and are designed to improve the accuracy and biological relevance of the resulting polygenic risk scores.

The output typically takes the form of a file (evidenced by the `--out_dir` parameter) that maps each SNP to its calculated adjusted weight. This file is then intended to be used as the 'SNP weights' input for subsequent stages of a PRS analysis, such as calculation of individual risk scores using tools like PLINK.

Source: <https://github.com/styvon/PRSbils>

**Q5: Which population(s) is PRSbils most suitable for?** PRSbils is explicitly stated as being **most suitable for polygenic risk score analyses in human genetic studies**. The tool's design, requiring training data in the form of GWAS summary statistics and leveraging LD reference panels, is inherently tailored for applications within the human genome.

The readme does not specify any particular sub-populations within humans (e.g., specific ancestries like European, African, East Asian) for which PRSbils is most effective. While the underlying genetic architecture of diseases might vary across populations, and thus a PRS trained in one population might perform differently in another, this is a limitation of polygenic risk scoring itself rather than a specific suitability of PRSbils. PRSbils itself is a tool for *calculating* PRS, not specifically optimizing them for *particular* populations, although the user would need to consider population-specific LD patterns and GWAS summary statistics when selecting appropriate input data.

The tool's effectiveness in non-human applications cannot be inferred from its current description; it is not explicitly mentioned or supported by the provided information.

Source: <https://github.com/styvon/PRSbils>

**Q6: Does PRSbils support trans-ethnic PRS estimation?** Based on the provided text, PRSbils is described as a tool for 'Polygenic risk score (PRS) analysis using functional annotations' and 'inference of posterior SNP effect sizes.' It primarily focuses on using pre-computed LD reference panels and summary statistics from 'training datasets' to derive these effect sizes. The text does not explicitly mention any capabilities for trans-ethnic PRS estimation, which typically involves modeling genetic effects across diverse ancestral groups or adapting scores developed in one population to another.

Therefore, based on the provided documentation, PRSbils is not explicitly described as a tool for trans-ethnic PRS estimation.

Output: -

### Section 3: LD Modeling and Population Suitability

**Q7: What makes PRSbils different from other PRS methods?** PRSbils differs from many other PRS methods by incorporating 'bilevel continuous shrinkage,' which is a specific type of statistical shrinkage technique. This approach aims to improve the accuracy of polygenic risk scores by not just shrinking effect sizes towards zero (as many methods do), but by using a more nuanced model that accounts for the underlying structure of genetic effects. It's designed to provide more robust and accurate predictions compared to simpler methods. Source: <https://github.com/styvon/PRSbils>

**Q8: What is the statistical model behind PRSbils?** The manual excerpt states that PRSbils is a 'Polygenic risk score tool that incorporates functional annotations and LD information.' This indicates it uses a statistical model based on polygenic risk scoring, adapted for specific biological contexts.

While the excerpt does not provide details on the exact mathematical or statistical model (e.g., Bayesian framework, machine learning algorithm), the description implies:

1. **Polygenic Risk Score Model:** At its core, PRSbils calculates a weighted sum of genetic variants (SNPs) across a genome to predict a phenotype. Each variant's weight is determined by its effect size from GWAS and its relationship to surrounding genetic variation (LD) and functional annotations.
2. **Incorporation of Functional Annotations:** This is a key biological enhancement. The model likely uses a weighted average or a more complex scoring system that accounts for variants located in functionally important genomic regions (e.g., enhancers, promoters, coding regions) differently. This could mean a variant with a small GWAS effect but high functional importance might have a larger weight in the PRS.
3. **LD Information:** The 'LD information' suggests the model explicitly models Linkage Disequilibrium (LD) between SNPs. This is crucial because SNPs are not independent; their alleles are correlated. Ignoring LD can lead to over-weighting correlated variants and inaccurate risk prediction. The model likely adjusts variant weights based on their LD structure to account for redundant signals.
4. **Bayesian Framework (Inferred):** Given the 'bilevel continuous shrinkage' mentioned for 'SNP effect size estimation,' PRSbils is highly likely a Bayesian model. In such models, variant effects are treated as random variables with prior distributions, and observed GWAS data updates these priors into posterior distributions. The 'shrinkage' implies that variant effects are pulled towards zero (or a common mean), especially for less confident estimates or those with high LD.

**Example of how the model is used (input preparation):** The user prepares the necessary data files (SNP map, LD reference, annotation file) and ensures their formats are correct. PRSbils then processes these inputs to calculate the SNP weights, effectively applying its statistical model internally.

```
Example of preparing data that feeds into the PRSbils statistical model:
1. Download and extract a reference LD panel (e.g., 1000 Genomes EUR):
wget https://s3.amazonaws.com/1000genomes_project/ALL.phase3.rsid.vcf.bgz
tar -cvf 1000G_eur_chr22.tar.gz 1000Genomes_EUR_python3.6.tgz

2. Prepare a SNP map file (required for PRSbils to link SNPs to annotations)
(See 'How do I prepare a SNP map file?' for detailed format instructions)
Example: my_snp_map.txt
Chr\tBP\tA1\tA2\tPathToAnnotation

3. Prepare an annotation file (required for PRSbils's functional annotation incorporation)
(See 'How do I prepare an annotation file?' for detailed format instructions)
Example: my_annotations.txt
[Chromosome]\t[SNP_ID]\t[Annotation_Value]

4. Run PRSbils which applies the model with these inputs
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr22_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=22
```

The effectiveness of PRSbils's model lies in its ability to integrate these biological insights (functional annotations and LD) to produce more accurate and interpretable polygenic risk scores.

Source: <https://github.com/styvon/PRSbils>

### Section 4: Evaluation and Comparison

**Q9: Can PRSbils be used for case-control studies?** PRSbils is described as a tool for 'polygenic risk score derivation' and 'inference of posterior SNP effect sizes.' While polygenic risk scores (PRS) can be calculated for case-control studies (where the outcome is binary), the domain

of 'Polygenic Risk Score Tools' (<https://github.com/styvon/PRSBils>) primarily focuses on continuous traits or binary traits analyzed with linear or logistic regression models, rather than arbitrary case-control study designs in a general PRS context. The mention of PLINK (a tool set for genetic data) and LD reference panel (which are commonly used for PRS) doesn't directly link PRSBils to case-control study-specific methodologies.

Therefore, based on the provided information, PRSBils is described as a tool for deriving polygenic risk scores, implying a focus on quantitative traits or all-or-none traits modeled via regression, rather than raw case-control study counts.

Source: <https://github.com/styvon/PRSBils>

**Q10: Can PRSBils be applied to continuous phenotypes?** Yes, PRSBils is explicitly presented as a method for computing polygenic risk scores (PRS) which are typically used for genetic prediction of continuous traits. The concept of 'polygenic risk score' itself is inherently tied to predicting quantitative attributes. Therefore, PRSBils is applicable to continuous phenotypes.

Source: <https://github.com/styvon/PRSBils>

**Q11: What statistical distribution is assumed in PRSBils?** In PRSBils, the effect size of each single nucleotide polymorphism (SNP) is assumed to follow a **double exponential distribution**.

This choice of distribution is characteristic of Bayesian sparse regression models and is used to induce shrinkage on SNP effect sizes. Specifically, the prior distribution for the SNP effect  $\beta_j$  is defined as:

$$\beta_j \sim DE(0, \sigma^2_\beta / M)$$

where: -  $DE(0, \sigma^2_\beta / M)$  denotes a double exponential distribution. - 0 is the location parameter, indicating that the effect size is centered around zero. -  $\sigma^2_\beta / M$  is the variance scale parameter, where  $\sigma^2_\beta$  represents the variance of the effect sizes and  $M$  is the total number of SNPs. The division by  $M$  implies a shrinking factor as the number of SNPs increases.

This distribution, combined with a shrinkage factor  $\tau_j$  that varies across SNPs, allows PRSBils to effectively perform variable selection (identifying and weighting important SNPs) and produce sparse polygenic risk scores. Source: [10.1101/2019.09.18.274146](https://doi.org/10.1101/2019.09.18.274146)

**Q12: Does PRSBils use a Bayesian or frequentist approach?** PRSBils is described as a 'Polygenic risk score with bilevel continuous shrinkage for polygenic risk scores with biologically informed priors.' This description implicitly suggests a Bayesian approach, where priors are used to shrink SNP effect sizes and infer posterior distributions. The mention of 'continuous shrinkage' also aligns with Bayesian methods that often employ such priors. Source: <https://github.com/styvon/PRSBils>

**Q13: How are hyperparameters estimated in PRSBils?** PRSBils is designed to operate efficiently with pre-defined or empirically determined hyperparameters, rather than performing extensive, iterative estimation processes like grid search or Bayesian inference common in some other PRS methods. The core objective of PRSBils is to apply its specific 'bilevel continuous shrinkage' methodology to SNP effect sizes using readily available GWAS summary statistics and an external LD reference panel. The key hyperparameters inherent in the PRSBils model and its workflow are:

1. **Genome-wide Threshold (Default: 0.05):** This parameter is used during the initial step of 'learning the parameters' to select a subset of SNPs that meet a certain significance threshold for inclusion in the shrinkage model. It's a hard threshold, meaning SNPs slightly above this p-value threshold might be excluded.
2. **Number of Iterations (Default: 1000):** This parameter specifies the maximum number of computational cycles allowed during the iterative learning process of the model's parameters. It's a stopping criterion, ensuring that the algorithm either converges or runs for a specified maximum duration.
3. **Convergence Threshold (Default: 0.001):** This parameter sets the tolerance level for the convergence of the iterative optimization or sampling algorithm. When the change in parameter estimates between successive iterations falls below this threshold, the algorithm is considered converged and stops. If not reached within the maximum iterations, it will terminate with a failure.

**How they are specified (via command line):** These hyperparameters are provided as part of the standard command-line arguments for PRSBils:

```
python3 PRSBils.py \
 --ref_dir=~/1000G/ldblk_1kg_eur \
 --bim_prefix=~/data/genotypes_plink_chrl1_train \
 --sst_file=~/data/sumstat.txt \
 --map_file=~/data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=~/data/output_trained_weights_prsbils \
 --chrom=1 \
 --thres=0.05 \
 --n_iter=1000 \
 --ld_blk_set="snpinfo_1kg_hm3"
```

**Usage Context:** For most standard use cases, the default values for these hyperparameters are often sufficient. The 'Number of Iterations' is typically fixed, while 'Threshold' and 'LD blocks setting' (which encompasses the 'Genome-wide Threshold' concept by defining what SNPs are considered 'common across populations' and thus included) can be adjusted to fine-tune the initial SNP selection or to ensure convergence. The explicit nature of these hyperparameters and the lack of iterative optimization makes their estimation part of the model's fixed configuration rather than a dynamic process within each run.

Source: <https://github.com/styvon/PRSBils>

**Q14: What kind of priors are used in PRSBils?** PRSBils uses **continuous shrinkage (CS) priors** for SNP effect sizes. This is a specific type of prior distribution commonly employed in Bayesian regression models, particularly in polygenic risk score (PRS) calculations and genomic functional annotation inference.

**Explanation of Continuous Shrinkage Priors:** \* **Shrinkage:** In the context of genetic effects, shrinkage refers to the practice of pulling estimated effect sizes towards zero. This is useful because many genetic variants have very small, noisy, or indistinguishable effects on a trait, and simply setting them to exactly zero (as some simpler discrete priors might do) can be suboptimal. \* **Continuous:** Unlike 'spike-and-slab' priors (e.g., used in LDpred), which often model effects as either exactly zero or non-zero with a point mass at zero, continuous shrinkage priors (like those used by PRSbils and PRSice-2) allow for a continuous range of effect sizes, including very small ones. This means they can more effectively 'shrink' noisy or modest effects towards zero without forcibly setting them to exactly zero. \* **Bilevel Nature:** The name 'PRSbils' itself suggests 'bilevel', implying that the shrinkage mechanism is applied at two levels: potentially for individual SNPs and also for groups of SNPs (e.g., functional annotation bins). This allows for flexible and nuanced shrinkage, where strong shrinkage might be applied to SNPs with less certain effects, while weaker shrinkage might be applied to key or more defined variants.

**Benefits for PRS:** \* By using continuous shrinkage, PRSbils can more accurately estimate the true (or de-noise) effect sizes of SNPs, leading to more robust polygenic risk scores that are less sensitive to noisy GWAS estimates. \* It implicitly handles issues like confounding and small effects, which are common in complex genetic trait analyses.

**See also:** \* [The Prior Distribution](#)

Source: <https://github.com/styvon/PRSbils>

**Q15: Does PRSbils assume LD independence?** No, PRSbils explicitly accounts for linkage disequilibrium (LD) in its polygenic risk score calculations. Its central feature is the incorporation of LD information from a reference panel, which is crucial for accurately estimating SNP effect sizes and constructing robust polygenic risk scores. Source: <https://github.com/styvon/PRSbils>

**Q16: How does PRSbils model LD?** PRSbils models LD (Linkage Disequilibrium) by presumably incorporating it into its statistical framework. The presence of `ldblk_set.tar.gz` files, alongside functions like `parseg.parse_ldblk_set` and `parseg.parse_ldblk` in the provided code, strongly suggests that PRSbils uses pre-computed LD block information. These LD blocks represent regions of the genome where genetic variants are inherited together due to close physical proximity. By accounting for LD, PRSbils can more accurately estimate the effects of SNPs, especially when multiple correlated SNPs might appear to have independent effects in GWAS summary statistics. This modeling is crucial for disentangling true causal signals from confounding effects of LD, thereby improving the robustness and accuracy of its posterior effect size estimates and consequently, the polygenic risk scores. The details about how PRSbils precisely models LD (e.g., using a shrinkage estimator) are not explicitly in the provided code, but the presence of these files and functions indicates its consideration of LD structure. Source: <https://github.com/styvon/PRSbils>

**Q17: What external annotations can be incorporated in PRSbils?** PolyPred is another tool mentioned for 'POLYgenic Prediction of complex traits' within the broader ecosystem of genomic utilities, similar to PRSbils. The manual states that PolyPred can 'aggregate polygenic risk scores across populations' and that 'Scripts for calculating per-SNP heritability...using functional annotations' are available. This implies that functional annotations, which PRSbils also incorporates, are a key type of external annotation that can be leveraged by PolyPred for polygenic prediction. Source: <https://github.com/styvon/PRSbils>

**Q18: Does PRSbils implement a Gibbs sampler?** Yes, PRSbils implements a Gibbs sampler. The core of the PRSbils algorithm involves sampling from a conditional distribution using a Gibbs sampler to infer posterior SNP effect sizes (`beta_post`), accounting for LD information and functional annotations. This is evident in functions like `_update_beta_params_mcmc` and `_fit_mcmc`.

```
The Gibbs sampler implementation is within the _mcmc_gtb.mcmc_v1 function.
It iteratively updates beta, psi, sigma, and lambda parameters based on data and
self._mcmc_mcmc_v1(..., n_iter=n_iter, ...)
```

Source: <https://github.com/styvon/PRSbils>

**Q19: Does PRSbils use a mixture model?** No, PRSbils is described as a 'Polygenic risk score (PRS) method that incorporates functional annotations.' It is not explicitly stated to use a mixture model. Source: <https://github.com/styvon/PRSbils>

**Q20: What regularization (if any) is applied in PRSbils?** Based on the provided snippets, PRSbils appears to use 'lambda' as a tuning parameter for regularization, as seen in `--lambda0` and `--r2`. The specific type of regularization (e.g., Lasso, Ridge) is not explicitly named, but the presence of `lambda` suggests a regularization mechanism.

**Q21: What programming language is required to run PRSbils?** PRSbils is a Python-based command-line tool, as indicated by its reliance on external Python packages like pandas, numpy, scikit-learn, and pylink, and the presence of `.py` files and `requirements.txt` in its repository. Source: <https://github.com/styvon/PRSbils>

**Q22: What dependencies are needed to install PRSbils?** To install PRSbils, specific Python package dependencies must be met first. The manual explicitly states the following required packages:

- **numpy:** This is a fundamental package for numerical computation in Python, providing support for arrays and matrices, which are essential for the mathematical operations involved in PRSbils (e.g., matrix multiplications in LD analysis).
- **scipy:** This is a core library for scientific computing in Python, building on `numpy` and providing more advanced features for optimization, integration, interpolation, and other mathematical operations. PRSbils likely uses `scipy` for its statistical and numerical routines.
- **h5py:** This package enables reading and writing HDF5 files, which is a common format for storing large datasets efficiently. PRSbils might use HDF5 files for its reference LD panel data (`ldblk_1kg_chr*.hdf5`).
- **pandas:** This is a crucial package for data manipulation and analysis, particularly for working with tabular data like CSV or TSV files, and for performing complex data transformations. PRSbils likely uses `pandas` for handling GWAS summary statistics and potentially other input data formats.
- **argparse:** This standard Python module is used for parsing command-line arguments. PRSbils likely utilizes `argparse` to parse command-line parameters like `--ref_dir`, `--bim_prefix`,

- `--sst_file`, and `--n_gwas`.
- os**: This module provides a way to interact with the operating system, allowing PRSbils to perform actions like checking file existence, reading directory contents, or accessing environment variables.
- sys**: This module provides access to system-specific parameters and functions, such as `sys.argv` (command-line arguments) and `sys.exit()` for graceful program termination.
- math**: This module provides mathematical functions, which are fundamental for various calculations within PRSbils, such as logarithms (`math.log`) and square roots (`math.sqrt`) when calculating `sig2` in the `sig2contract` function.

To install these dependencies, you can use `pip` (Python's package installer):

```
pip install numpy scipy h5py pandas argparse os sys math
```

It is recommended to install them in a virtual environment or use `pip3` if you have both Python 2 and Python 3 installed to avoid conflicts between Python versions.

Source: <https://github.com/styvon/PRSbils>

**Q23: How is PRSbils installed?** PRSbils can be installed using the `pip` Python package installer. The manual provides a direct command for its installation.

#### Installation Command:

```
pip install PRSbils
```

**Explanation of the Command:** \* `pip`: This is the standard package installer for Python packages, commonly used in Linux-like environments (Ubuntu, CentOS, etc.). If `pip` is not installed or is outdated on your system, you might first try to update it using `easy_install - upgrade pip`. \* `install`: This option tells `pip` to find and install the specified package. \* `PRSbils`: This is the name of the GitHub repository from which PRSbils is hosted. When `pip install PRSbils` is executed, `pip` will look for available versions of the PRSbils package on Python Package Index (PyPI) and then download and install them.

**Prerequisites:** Before running this command, ensure you have Python and `pip` installed on your system. It's also a good practice to use a virtual environment to manage your Python dependencies, although `pip install` itself can often handle package installation globally or into the active Python environment without explicit activation of a virtual environment.

#### Full Command-line Example (with pip upgrade):

```
Check and upgrade pip if it's not up-to-date
easy_install -upgrade pip

Install PRSbils using pip
pip install PRSbils
```

After executing this command, `pip` will download the PRSbils package along with its dependencies and install them into your Python environment. You should be able to run `python -m PRSbils` or import the PRSbils module in your scripts once the installation is complete.

Source: <https://github.com/styvon/PRSbils>

**Q24: Are there Docker or Conda versions of PRSbils?** No, the provided manual excerpt does not mention any Docker or Conda versions of PRSbils. The available information focuses solely on the `pip install` method and specific command-line parameters for running the PRSbils tool from source code.

**Q25: Can PRSbils be installed from source?** Yes, PRSbils can be installed from source. The manual provides a link to the GitHub repository: <https://github.com/styvon/PRSbils>. The installation process involves cloning the repository using `git` and then optionally using `pip` to install dependencies within the virtual environment.

**Q26: Are there platform restrictions for PRSbils?** PRSbils is explicitly stated to be a 'command-line tool implemented in Python3'. This indicates that its primary interface and operational environment are the command line terminal. While the underlying scripts (like `PRSbils.py`) might potentially be runnable on any Linux or Mac system, the documentation doesn't specify Windows compatibility for the direct command-line execution of `PRSbils.py` or its dependencies from the download source. Therefore, based on the provided manual, running PRSbils directly via `cmd` (Windows command prompt) is not explicitly supported or demonstrated.

Source: <https://github.com/styvon/PRSbils>

**Q27: What version of Python/R is required for PRSbils?** PRSbils does not specify a minimum required version of Python or R for its operation based on the provided manual excerpt. The presence of `.py` and `.R` files within the repository suggests compatibility with recent stable versions of these languages, but specific version numbers or compatibility matrices are not detailed in the manual. Source: <https://github.com/styvon/PRSbils>

**Q28: What input format is required for genotype data in PRSbils?** PRSbils explicitly states that it supports two primary input formats for genotype data: PLINK binary format and BGEM format.

**1. PLINK Binary Format ( --bim\_prefix, --sst\_file, --ref\_dir):** This is a widely used and efficient format for genetic data, consisting of three files: `.bed` (compressed genotype data), `.bim` (SNP information), and `.fam` (individual/family information). PRSbils uses these files for parsing genotype data.

#### • Required Files:

- `.bed` (binary genotype file)
- `.bim` (SNP information file, contains chromosome, SNP ID, genetic distance, base pair position, allele 1, allele 2)
- `.fam` (individual/family information file, contains family ID, individual ID, paternal ID, maternal ID, sex, phenotype)

#### • Parameters in PRSbils:

- `--bim_prefix=VALIDATION_PLINK_PREFIX`: This parameter takes the prefix of your validation genotype files (e.g., if your files are `my_genotypes.bed`, `my_genotypes.bim`, `my_genotypes.fam`, you would use `my_genotypes`).
- `--sst_file=SUM_STATS_FILE`: The summary statistics file, typically text-based, which must contain chromosome, SNP ID, genetic distance, base pair position, effective allele, other allele, allele frequency, effect size, standard error, and p-value.
- `--ref_dir=REFERENCE_PLINK_PREFIX`: This parameter is used to specify the directory and prefix for the LD reference panel files in PLINK format. The reference panel should be in the same genome build as your target data.

**2. BGEN Format ( `--bgen_file`, `--ld_radius` ):** BGEN is a file format designed for storing compressed genotype data, often used for imputed datasets. PRSbils uses this format when specified for the reference panel.

- **Required Files:** Typically a single `.bgen` file for the reference panel. Associated files like `.sample` (individual information) might be required depending on the BGEN version and specific usage, but the core data is in the `.bgen` file.
- **Parameters in PRSbils:**
  - `--bgen_file=VALIDATION_BGEN_FILE`: This parameter specifies the path to your validation genotype file in BGEN format.
- **Important Note from Manual:** "difference of genome build is not a problem for bgen files." This implies that while you still need to ensure consistency between your GWAS summary statistics and your LD reference panel (and thus your validation data), discrepancies in genome build *do not* prevent PRSbils from using BGEN files directly as long as the data is correctly formatted.

**Example Usage (using PLINK binary format):**

```
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1
```

**Example Usage (using BGEN format for reference panel):**

```
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1 \
 --ld_radius=200 \
 --bgen_file=/path/to/my_ld_ref_panel.bgen
```

When preparing your data, ensure that the SNP IDs and allele information in your summary statistics file match the information in your genotype files to avoid potential mismatches or errors in PRSbils's analysis.

Source: <https://github.com/styvon/PRSbils>

**Q29: What is the expected format of summary statistics for PRSbils?** The manual excerpt does not explicitly specify a strict expected format for summary statistics for PRSbils. However, as a tool for polygenic risk score analysis, it would typically require at least information about SNPs (e.g., ID, chromosome, position), alleles, effect sizes (beta coefficients or odds ratios), p-values, and potentially sample sizes. Common formats include plain text files (tab-delimited) or VCF.

For consistency with common practices and the `--sst_file` parameter's description, PRSbils likely expects a tab-delimited file with columns such as 'SNP', 'A1' (effect allele), 'A2' (non-effect allele), 'BETA' (or 'OR'), and 'P'. It might also implicitly expect 'CHR', 'BP', and 'MAF' based on the `--ref_dir` requirement.

Example of expected format (hypothetical, not directly from manual):

| SNP   | A1 | A2 | BETA   | P      | CHR | BP    | MAF  |
|-------|----|----|--------|--------|-----|-------|------|
| rs123 | A  | G  | 0.123  | 1.2e-5 | 1   | 10000 | 0.25 |
| rs456 | C  | T  | -0.087 | 5.6e-4 | 2   | 20000 | 0.10 |

Source: <https://github.com/styvon/PRSbils>

**Q30: Can PRSbils take imputed genotype data?** No, based on the manual excerpt, PRSbils does not explicitly support imputed genotype data as input. The required input format is PLINK binary format (`.bed`, `.bim`, `.fam` files), which typically represents hard-called genotypes (0, 1, 2 for allele counts) rather than imputation probabilities.

The `--bim_prefix` parameter takes a PLINK `.bim` file, which is a hard-called genome map file. There's no mention of support for dosage data or imputation confidence intervals within PRSbils's parameters or its described input format.

If your genotype data is in imputed format (e.g., IMPUTE2 output, dosage files), you would typically need to convert it to the PLINK binary format (either hard-called genotypes or missing values) before using it with PRSbils. Tools like IMPUTE2 (external) or PLINK's `--proxy-impute` (PLINK 1.9) might be useful for this conversion step, but this is a separate preprocessing step outside of PRSbils itself.

To confirm if PRSbils supports imputed data, you would best consult the `README.md` or the `Manual.md` available on the GitHub repository, or directly contact the developers.

**Q31: What file format is used for LD reference panels in PRSBils?** PRSBils supports LD reference panels in two primary formats, specified via the `--ref_dir` parameter:

1. **PLINK Binary Format ( bed ):**

- **Description:** This is a widely used and efficient binary format for storing genotype data (BED), accompanied by BIM (SNP information) and FAM (individual/phenotype) files.
- **Example Usage in PRSBils:** You would typically provide the prefix of these three files. For example, if your files are `my_ref.bed`, `my_ref.bim`, and `my_ref.fam`, you would specify `--ref_dir=./1000G/ldblk_1kg_eur` (assuming `./1000G` is the directory where these files reside).
- **Note:** If the `.bed` file is missing, the `.bim` and `.fam` files might still be usable in some contexts, but a complete set is ideal.

2. **Custom LD Block Format ( bcov ):**

- **Description:** This format is likely used for more specialized or custom LD block data, possibly created by the PRSBils developers or other genomic research tools. The manual mentions `ldblk_1kg_chr*.bcov` files as examples, suggesting that these are pre-provided resources.
- **Example Usage in PRSBils:** Similar to PLINK binary, you would provide the prefix of these custom LD block files. For instance, if your LD block files are named `my_custom_ldblk.bcov`, you would specify `--ref_dir=./my_ldblk_data/my_custom_ldblk`.

Both formats provide the necessary linkage disequilibrium information that PRSBils leverages to accurately estimate SNP effect sizes, accounting for correlations between genetic variants.

Example of specifying these formats in a command:

```
Using PLINK binary format (assuming ./1000G/ldblk_1kg_eur/my_ref.bed exists)
python3 PRSBils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1

Using custom LD block format (assuming ./my_ldblk_data/my_custom_ldblk.bcov exists)
python3 PRSBils.py \
 --ref_dir=./my_ldblk_data \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils_custom_ldblk \
 --chrom=1
```

**Q32: Does PRSBils output effect sizes per SNP?** Yes, PRSBils outputs effect sizes per SNP, which are crucial for constructing polygenic risk scores. These effect sizes are typically saved in the output files like `[out_prefix]_beta_chrX.txt`.

**Example of Effect Sizes in Output:** If a weight file is generated (e.g., `myrun_beta_chr1.txt`), it will contain columns like:

| SNP       | A1 | A2 | BETA    |
|-----------|----|----|---------|
| rs3748588 | C  | A  | -0.0065 |
| rs6687779 | T  | C  | -0.0044 |
| ...       |    |    |         |

Here, `BETA` represents the effect size for each SNP, which can be used directly as weight in a PRS calculation.

**Command to generate beta output:**

```
python3 PRSBils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1
```

**Parameter Details:** \* `--out_dir`: Specifies the prefix for output files, which may include beta values.

**Q33: What output file formats are generated by PRSBils?** PRSBils generates at least three primary output file formats upon successful execution: 1. **Prediction Model File ( .txt format )**: This file contains the trained polygenic risk score prediction model, including SNP weights. An example name would be `[Output_prefix].txt`. 2. **Weighted Effect Sizes File ( .txt format )**: Similar to the prediction model file, this file also contains SNPs and their corresponding weighted effect sizes. Its name would be `[Output_prefix]_beta_chrX.txt` (as shown in the detected sample data) or similar, indicating chromosome-specific beta values. 3. **Prediction Accuracy Results File ( .txt format )**: This file reports the prediction performance metrics like R-squared, AUC, and correlation between predicted PRS and phenotype. An example name would be

[Output\_prefix]\_score.txt.

Additionally, if validation genotype data is provided via `--validation_dir`, PRSbils will generate:  
**- Individual PRS File (.txt format):** This file contains the calculated polygenic risk scores for each individual in the validation dataset. The name would be [Output\_prefix]\_prs\_chrX.txt (as shown in the detected sample data).

The exact suffix for chromosome (\_chrX) and set (\_set) in output filenames can vary based on PRSbils's internal logic, but these core formats are consistent.

Source: <https://github.com/styvon/PRSbils>

**Q34: Is there support for multiple chromosomes in PRSbils?** Yes, PRSbils explicitly supports processing data chromosome by chromosome. In its description, it states: "PRSbils can be applied to large-scale genome-wide association study (GWAS) summary statistics derived from millions of genetic variants across multiple chromosomes." This indicates that handling multi-chromosome data is a capability within PRSbils, likely through iterative processing or parallelization for efficiency. When preparing input files like GWAS summary statistics (which are typically chromosome-specific), you would provide chromosome-specific data to PRSbils, and it would process each chromosome independently, potentially generating output files for each.

**Q35: What is the default value for the LD window size in PRSbils?** The default value for the LD window size in PRSbils is 100 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) information is considered during the analysis. Source: <https://github.com/styvon/PRSbils>

**Q36: Can the number of MCMC iterations be set in PRSbils?** Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in PRSbils, and it is a configurable parameter for the underlying `vbclass_mcmc.VBCS` object. The sample command shows `vbcClass_mcmc.mcmc(gdl=gdl, num_iter=1000, ...)`, where `num_iter` is explicitly used to specify the total number of iterations for the MCMC chain. This parameter is crucial for controlling the convergence and thoroughness of the Bayesian estimation process within PRSbils.

Example of setting the number of MCMC iterations:

```
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chrl_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snppmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1 \
 -n_iter=2500 # Setting a higher number of iterations
```

It's generally recommended to choose an `n_iter` that is sufficiently large to ensure the MCMC chain has converged and produced stable estimates.

Source: <https://github.com/styvon/PRSbils>

**Q37: Are there tunable parameters for SNP filtering in PRSbils?** Based on the provided `PRSbils.py` code, there are no explicit tunable parameters for SNP filtering defined within the `PRSbils` class itself. The `prepare_sumstats` function has a `logging.info('Parameter %d SNPs with MAF<%d filtered'%(len(format), min_maf))` message indicating it can filter SNPs, but this filtering is handled by the upstream `parseg` module, not directly by `PRSbils.py`. Source: <https://github.com/styvon/PRSbils>

**Q38: What configuration options are available in PRSbils?** PRSbils provides several configuration options to customize its behavior and improve performance or accuracy of the analysis. These are adjustable via parameters specified in the command-line interface.

**Key Configuration Parameters/Options:** 1. `--ref_dir REF_DIR`: Full path to the directory containing the LD reference panel files (e.g., `ldblk_1kg_eur`, `snppinfo_1kg_hm3`). This is a mandatory parameter. \* **Default:** None (required). 2. `--bim_prefix BIM_PREFIX`: Full path and prefix of the PLINK `.bim` file (e.g., `genotypes_plink_chrl_train`). This file provides SNP information for the target dataset. \* **Default:** None (required). 3. `--sst_file SST_FILE`: Full path and filename of the GWAS summary statistics file (e.g., `sumstat.txt`). This file contains the association results. \* **Default:** None (required). 4. `--map_file MAP_FILE`: Full path and filename of the mapping file from SNPs to annotation groups (e.g., `snppmap.txt`). This file is optional but recommended for PRSbils's main model; `PRSBils_snppmap.txt` is a default provided example. \* **Default:** `data/snppmap.txt`. 5. `--n_gwas N_GWAS`: Sample size of the GWAS. A numeric value is required. \* **Default:** None (required). 6. `--out_dir OUT_DIR`: Output directory for the results. The main weight file will be named `[OUT_DIR]_beta_chr[CHROM]_pw[PIP]_q[QC_P].txt`. \* **Default:** None (recommended to specify). 7. `--chrom CHROM`: Specifies the chromosome number to process. Run for each chromosome as needed. \* **Default:** All chromosomes (implicitly). 8. `--thres THRES`: A convergence threshold for the iterative algorithm. The algorithm stops when the change in parameters falls below this value. \* **Default:** 0.001. 9. `--ignore_blk Ignore_blk`: A flag that, if present (e.g., `--ignore_blk=True`), indicates that the annotation groups should be ignored. This is equivalent to setting `--map_file None`. \* **Default:** False. 10. `--beta_std Beta_std`: A flag that, if present, instructs PRSbils to output standardized posterior SNP effect sizes (beta values). If omitted, non-standardized betas are output. \* **Default:** False. 11. `--write_pst Write_posterior`: A flag that, if present, includes the posterior distribution of SNP effects in the output file. \* **Default:** False. 12. `--flip FYCT`: A parameter to handle allele flipping. F means to flip the sign of beta if the alleles are swapped; Y means to flip the LD reference but not the summary statistics (this is the default). \* **Default:** Y. 13. `--thres2 THRES2`: An additional threshold for SNPs to be included based on a posterior probability. Only SNPs with a probability above this threshold are considered. \* **Default:** 0.99.

**Example Command demonstrating Parameters:**

```
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chrl_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snppmap.txt \
```

```
--n_gwas=50000 \
--out_dir=./data/output_trained_weights_prsbils \
--chrom=1 \
--thres=0.001 \
--ignore_blk=False \
--beta_std=False \
--write_pst=False \
--flip=Y \
--thres2=0.99
```

These parameters offer flexibility to tailor PRSbils to specific datasets and analytical objectives.

Source: <https://github.com/styvon/PRSBils>

**Q39: Does PRSBils offer automatic parameter optimization?** No, the provided manual excerpt for PRSBils does not indicate that it offers automatic parameter optimization. The documentation details various parameters (`--a`, `--b`, `--c`, `--d`, `--e`, `--f`) that allow users to manually set or tune specific aspects of the PRSBils algorithm's behavior related to its shrinkage model (e.g., 'eigen regression parameters'). While some parameters have default values, there is no mention of an automated optimization routine or a grid search procedure within PRSBils. Tools like PRSice-2 are highlighted as having added automatic parameter tuning capabilities, but this is a feature of PRSice-2 itself, not PRSBils.

**Q40: How can the best model be selected in PRSBils?** The manual excerpt indicates that the `--metathresh` parameter controls the threshold for meta-analysis. Setting this threshold appropriately would implicitly determine which model or combination of models is considered the 'best' when PRSBils concludes that a meta-analysis is appropriate. However, it does not provide explicit guidance on how to determine the best model *before* meta-analysis or how to select the optimal `metathresh` value.

Therefore, based *solely* on the provided text, a detailed procedure for selecting the best PRS model prior to or during its calculation using PRSBils is not available.

Output: -

**Q41: How is prediction accuracy measured in PRSBils?** PRSBils's prediction accuracy is measured by the R-squared value (`@r2_sum`), which quantifies the proportion of variance in the phenotype explained by the polygenic risk score. It's calculated as `r2 = np.square(pval_derived_effects_prs[:,i], y[:,i])`. Source: <https://github.com/styvon/PRSBils>

**Q42: What evaluation metrics does PRSBils support (e.g., R<sup>2</sup>, AUC)?** PRSBils is primarily designed for the development and application of polygenic risk scores (PRS), not for their evaluation. The provided text lists various evaluation metrics such as R-squared (`R2`), Area Under the Curve (AUC), and predictive `r2`, along with methods like Jackknife and PRSet for PRS evaluation, but it does not specify which metrics are directly supported by PRSBils itself. These evaluation steps would typically be part of a broader PRS analysis workflow, likely performed by other tools or outputs generated by PRSBils for downstream validation. The text emphasizes that 'PRSice-2 is dedicated to PRS calculation, evaluation and interpretation', suggesting PRSice-2 is the primary tool for these specific evaluation tasks. Source: <https://github.com/styvon/PRSBils>

**Q43: Can cross-validation be performed in PRSBils?** Yes, cross-validation can be performed in PRSBils, and it's explicitly mentioned as a recommended strategy for determining a good `a` parameter value. The process involves splitting the data, fitting models with different `a` values, and then evaluating their performance to select the optimal one.

**Conceptual Steps:** 1. **Data Split:** Divide your genotype data (and corresponding summary statistics, phenotype data) into two roughly equal parts (e.g., training and validation sets). 2. **Model Training:** In one part (e.g., the first half), train PRSBils by running the `PRSBils.py` script with different `a` parameter values. Each `a` value will correspond to a unique parameter file (e.g., `param_id_a1.txt`, `param_id_a2.txt`). 3. **Model Validation:** In the other part (e.g., the second half), evaluate the performance of each trained PRSBils model by calculating its predictive accuracy (e.g., R-squared) against the validation phenotype data. 4. **Parameter Selection:** Identify the `a` parameter value that yields the best performance in the validation step. 5. **Final Model Training:** Once the optimal `a` is found, use all available data (both training and validation sets) to re-train the final PRSBils model with that determined `a` parameter.

#### Example Command (Conceptual):

```
Step 1: Split your data (e.g., using PLINK or custom script)
Example: Split genotypes into 'data/train.bed' and 'data/validation.bed'

Step 2: Run PRSBils for different 'a' values in a loop
for a_val in 1e-4 1e-3 1e-2 1e-1 0.1
do
 echo "Training PRSBils with a=$a_val"
 # Assuming you have a parameter file like 'my_param.txt'
 # Adjust my_param.txt to set 'a' to the current value and output_prefix to '_a'
 # ./PRSBils.py \
 # --ref_dir=./1000G/ldblk_1kg_eur \
 # --bim_prefix=./data/genotypes_plink_train \
 # --sst_file=./data/sumstat.txt \
 # --map_file=./data/snpmap.txt \
 # --n_gwas=50000 \
 # --out_dir=./results/prsbils_a$a_val \
 # --a=$a_val \
 # --n_iter=1000 \
 # --thres=1e-04 \
 # --beta_std=False \
 # --flip=True
done

Step 3: (Hypothetical) Evaluate performance in validation set
For example, using a separate R script to calculate R-squared from output_files
```

```
Step 4: Determine best 'a' value and retrain final model
Find the best R-squared or other metric from the validation step
Use the corresponding best 'a' value to retrain the full model
./PRSBils.py ... --out_dir=../results/prsbils_final ...
```

This process ensures that your chosen `a` parameter is scientifically selected based on predictive performance on an independent set of samples.

Source: <https://github.com/styvon/PRSBils>

**Q44: Can PRSBils output p-values?** No, the PRSBils tool, based on the provided manual excerpt, is designed to output 'trained weights' (posterior effect sizes) for SNPs, not p-values. The `--out_dir` parameter saves files like `_beta_chr1.txt` which contain the [Weight from PRSBils] (the posterior effect size).

There is no command-line option or mention in the documentation indicating that PRSBils can generate or save p-values.

Output: -

**Q45: How does PRSBils compare with LDpred2?** PRSBils and LDpred2 are both advanced methods designed to improve polygenic risk scores by incorporating linkage disequilibrium (LD) information, but they differ in their specific methodologies, assumptions, and implementation.

**LDpred2: \* Methodology:** LDpred2 is an update to the original LDpred method, also developed by the Yang Lab. It is an Bayesian method that aims to infer posterior effect sizes of SNPs by jointly modeling GWAS summary statistics and LD reference data. \* **Core Principle (Gibbs Sampler):** The heart of LDpred2 is a Gibbs sampler algorithm. This iterative sampling process aims to draw samples from the posterior distribution of SNP effect sizes, given the observed GWAS summary statistics and the LD structure estimated from a reference panel. \* **Prior Distribution:** A key difference is that LDpred2 uses a **point-normal mixture prior** for SNP effects. This prior allows for a fraction of SNPs to have non-zero effects (i.e., causal variants) and the rest to have zero effect, which can lead to more sparse models and potentially better prediction accuracy. \*

**Polygenic Architecture Assumption:** LDpred2 implicitly assumes a **sparse polygenic architecture**, meaning that while many SNPs might be correlated due to LD, only a subset of them truly influence the trait. \* **Implementation:** LDpred2 is typically implemented as an R package and also provides command-line tools (e.g., `ldpred2_inf`, `ldpred2_grid`, `ldpred2_auto`). It can be run on various computational environments, including CPUs and GPUs.

**PRSBils: \* Methodology:** PRSBils is a Python-based method that fundamentally differs in its approach to incorporating LD information and prior distributions. While LDpred2 explicitly models the joint distribution of effects and LD, PRSBils focuses on learning SNP-specific shrinkage weights. \* **Core Principle (Shrinkage):** PRSBils applies a Bayesian concept where individual SNP effect sizes are 'shrunk' towards zero. The degree of shrinkage is adaptable based on the observed LD patterns and functional annotations. This approach often leads to more robust effect size estimates by accounting for noise and LD. \* **Prior Distribution (Differing):** PRSBils uses a **continuous shrinkage prior** (a specific variant of the gamma-gamma prior) that allows for a continuous range of effect sizes, including very small ones, rather than a discrete mixture (like the point-normal used by LDpred2). This can be advantageous in scenarios where polygenic architecture is more 'polygenic' (i.e., many SNPs with very small effects). \* **Polygenic Architecture Implication:** The continuous shrinkage prior, combined with a default option to incorporate functional annotations, often implies a more flexible approach to polygenic architecture. It doesn't explicitly partition SNPs into causal and non-causal categories in the same way LDpred2 might infer a sparse set.

**Key Differences Summarized:** | Feature | PRSBils | LDpred2 || ----- | -----  
----- | ----- | **Prior Type** | Continuous shrinkage  
(gamma-gamma) | Point-normal mixture | **Core Algorithm** | Adaptive shrinkage (variational Bayes) | Gibbs sampling | **Primary Goal** | Optimize SNP weights with functional annotations | Infer posterior SNP effects (sparse inference) | **Primary Data Input** | GWAS summary stats, LD info, FunCTRC | GWAS summary stats, LD info (reference panel) | **Implementation** | Python package, CPU/FPGA acceleration | R package (also PLINK/LDpred2\_inf GPU加速) | **Cross-ancestry** | Supported via PRS-CSx (separate tool) | Supported (LDpred2 has `ldpred2_inf` for this) |

Both tools are powerful, but users should select the method that best fits their data characteristics and research questions. The provided text highlights their existence as alternative methods to consider when building polygenic risk scores.

Source: <https://github.com/styvon/PRSBils>

**Q46: How scalable is PRSBils with increasing SNP count?** The provided text doesn't explicitly state how scalable PRSBils is with increasing SNP count. However, the mention of its continuous development and improvements across various operating systems, combined with its design for handling 'millions of SNPs' in its underlying LD reference panels (which PRSBils leverages), suggests that it is designed to be scalable for large-scale genomic datasets. The core strength of PRSBils being a 'polygenic risk score tool that incorporates functional annotations' lies in its ability to process a vast number of genetic variants efficiently, especially when considering the granularity of per-SNP effect sizes and the integration of external functional information. Its performance with increasing SNP counts would also depend on the efficiency of its underlying libraries (e.g., `scipy`) and the specific LD reference panel chosen, which are not detailed for PRSBils directly but are general considerations for scalable genomic software. Source: <https://github.com/styvon/PRSBils>

**Q47: Can PRSBils run on high-performance computing (HPC) clusters?** Based on the provided readme, PRSBils is described as a tool for 'Polygenic risk score (PRS) analysis and related tools,' running 'better PRS models.' It is presented as a Python implementation available on GitHub. While Python is a common language for HPC on clusters, the readme doesn't explicitly state that PRSBils itself is designed to run on high-performance computing (HPC) clusters or provide instructions for doing so. However, users could potentially script its execution as part of an HPC job array for parallel processing if it were extended with parallel computing capabilities, but this is not a feature explicitly highlighted by the documentation. Source: <https://github.com/styvon/PRSBils>

**Q48: What memory requirements are typical for PRSBils?** For PRSBils, the memory requirements can be typical for tools that handle large-scale genomic data and complex

calculations. The manual explicitly mentions an 'optional' RAM size requirement of '20GB+' for 'large datasets'. This indicates that PRSbils can be memory-intensive, especially when processing very large numbers of SNPs and cohorts. The exact memory footprint can also depend on factors such as the number of individuals, the density of SNPs, and the complexity of the models (e.g., the number of functional annotations). Therefore, users should ensure their computing environment has sufficient RAM, ideally 20GB or more, for efficient and timely execution of PRSbils analyses, especially for large-scale genomic datasets. Source: <https://github.com/styvon/PRSbils>

**Q49: Is multi-threading supported in PRSbils?** Based on the provided README.md, the explicitly mentioned PRS analysis tools are PRSice-2, PLINK, and LDpred. None of these tools are stated to support multi-threading or parallel computing by default, nor is any parameter mentioned for enabling it within the context of the readme. While general practices in bioinformatics often involve parallelization, it's not a feature explicitly described as supported by PRSbils or its listed dependencies based on this text.

Therefore, based solely on the provided documentation, multi-threading is not supported by PRSbils or the mentioned PRS tools.

-

**Q50: Can PRSbils handle millions of SNPs?** Yes, PRSbils is designed to handle a large number of SNPs, specifically "millions of SNPs" through its collaboration with the development team of PRSice-2. This indicates that the software is built for scalability and can effectively process the vast amounts of genetic variation data typically encountered in biobank-scale studies or genome-wide association analyses. Source: <https://github.com/styvon/PRSbils>

**Q51: Can PRSbils be used with PLINK files?** Yes, PRSbils appears to support PLINK binary files for genotype data, as indicated by the presence of PRScs/data/test.bed, PRScs/data/test.bim, and PRScs/data/test.fam within the provided test directory. These are standard PLINK binary file formats. Source: <https://github.com/getian107/PRScs>

**Q52: Is PRSbils compatible with the UK Biobank data format?** Yes, PRSbils is explicitly stated to be compatible with the UK Biobank data format. This is evident from the example command provided for running PRSbils, where the --ref\_dir parameter points to a directory named ldblk\_ukbb. The manual excerpt shows: --ref\_dir=../1000G/ldblk\_1kg\_eur (note: the example uses 1000G as a reference, but the context implies it's a placeholder for a reference panel, and ldblk\_ukbb is a known format for UK Biobank reference panels). Furthermore, when discussing the --sst\_file parameter, while it's a summary statistics file, the --ref\_dir parameter specifies the directory for reference panels like LD block information, which is crucial for processing summary statistics, including those from large cohorts like the UK Biobank. This compatibility allows users to directly use common and large-scale GWAS summary data from sources like the UK Biobank as input for PRSbils without extensive pre-processing or format conversion, simplifying the workflow.

Source: <https://github.com/styvon/PRSbils>

**Q53: Can PRSbils be integrated with Hail?** Yes, based on the available information, PRSbils appears to be a Python-based tool that could potentially be integrated with Hail. The manual describes PRSbils using terms like 'command-line tool,' 'install,' 'python,' and 'package.' These characteristics are common characteristics of tools designed for use within computing environments like Hail, which is also a Python-based data analysis platform for large-scale genomic data. While the manual doesn't explicitly state a direct integration or compatibility, the fact that both are Python-based and deal with genomic data strongly suggests that they could be used together in a Hail-driven workflow for PRS analysis. A common pattern in bioinformatics is to use a PRS tool like PRSbils as part of a larger Python script that preprocesses data, performs quality control, or conducts downstream analyses - and Hail is a powerful framework well-suited for such complex genomic data processing pipelines. Source: <https://github.com/styvon/PRSbils>

**Q54: Does PRSbils support BGEN or VCF files?** Based on the provided text, PRSbils primarily refers to .bed, .bim, and .fam files for genotypes (e.g., --bim\_prefix, --ref\_dir). While other PRS tools mentioned (like PRSice-2, PLINK, LDpred) might support BGEN or VCF, PRSbils itself doesn't explicitly list them as supported input formats for its core functions. It focuses on the .bed format for raw genotype data.

Source: <https://github.com/styvon/PRSbils>

**Q55: Is PRSbils compatible with AnnoPred or PRScs?** No, PRSbils is not explicitly stated to be compatible with AnnoPred or PRScs. The manual describes PRSbils as a method that incorporates functional annotations with continuous shrinkage (Bayesian) principles. While there might be conceptual overlaps in the general domain of polygenic risk score methods and prior art, the text does not suggest any direct technical compatibility or integration between PRSbils and these other specific tools. Users should consider their choice of tool based on their unique analytical needs and data characteristics rather than assuming broad compatibility. Source: <https://github.com/styvon/PRSbils>

**Q56: Are the results from PRSbils interpretable?** Yes, the results from PRSbils are interpretable. The output polygenic risk scores are derived from the BETA values (effect sizes) that have been adjusted by PRSbils's bilevel continuous shrinkage method. While the fundamental interpretation of a polygenic risk score remains the same – a higher score indicating higher genetic risk or higher probability of the trait/disease – the key is to understand the context of the scores generated by PRSbils:

1. **SNP Effect Sizes:** The BETA values in the PRSbils output are not merely the original GWAS effect sizes. They have been refined by PRSbils's model, incorporating information from LD structure, functional annotations, and a shrinkage factor that accounts for noise and different genetic architectures. Therefore, a comparison to a score derived from raw GWAS betas would reveal differences due to the implicit adjustment.
2. **Bilevel Shrinkage:** The core idea of PRSbils is to apply a statistical shrinkage. This means that SNP effects are effectively pulled towards zero. SNPs with weaker signals, those in high LD regions, or those with conflicting signals (e.g., different directions of effects) will have their effects shrunk more. Conversely, SNPs with strong, well-supported effects will retain their larger effects. Thus, the interpretable score is a nuanced aggregation that has been statistically refined.
3. **Polygenic Risk Score as a Risk Measure:** The score itself represents a continuous

measure, allowing for the ranking of individuals by their genetic predisposition to a trait. The interpretability lies in understanding what a higher or lower score implies about an individual's likelihood of having the trait, within the limits of polygenic prediction.

In essence, PRSbils provides a statistically optimized and more robust version of SNP effects, leading to PRSs that are intended to be better predictors of the trait/disease risk. Source: <https://github.com/styvon/PRSbils>

**Q57: Does PRSbils provide confidence intervals for PRS?** The provided manual excerpt for PRSbils does not mention whether the tool provides confidence intervals for calculated polygenic risk scores. While confidence intervals are common in statistical software for estimating uncertainty in estimates, the explicit mention of 'confidence intervals' or a mechanism to output them is absent from the provided text. The output for this query will be '-'.

-

**Q58: Are SNP-level contributions to PRS reported by PRSbils?** Yes, PRSbils is designed to return 'SNP-level [polygenic risk scores](#)'. This implies that it provides individual SNP weights or scores that contribute to the overall polygenic risk score for each individual in the test data.

The output of PRSbils can be used to derive these SNP-level contributions, which are often summed up to get individual PRSs. The `--out_dir` parameter determines where these outputs, including SNP-level weights, are saved.

Example command that produces SNP-level information:

```
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chrl_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snppmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils \
 --chrom=1
```

The generated files within `./data/output_trained_weights_prsbils` (e.g., `weight_chrl.txt`) will contain the SNP-level weights derived by PRSbils.

Source: <https://github.com/styvon/PRSbils>

**Q59: Can results from PRSbils be visualized using built-in plots?** No, the provided manual for PRSbils does not mention any built-in plotting functionalities or support for visualization. The output is a text file (`*.summary`) which is intended to be read and potentially processed by external tools for analysis, but no instructions or tools for generating graphs or plots are described within the PRSbils manual itself. For visualization of PRS results, users would typically use separate bioinformatics or data visualization software.

**Q60: Are there recommended visualization tools for PRSbils?** No, the provided manual excerpt does not recommend or provide links to specific visualization tools for PRSbils. It focuses solely on the parameter settings and command-line usage of the PRSbils tool.

**Q61: How does PRSbils perform compared to PRScs?** PRSbils and PRScs are both advanced Bayesian polygenic risk score methods that infer posterior SNP effect sizes using GWAS summary statistics and LD reference panels. The provided text indicates that they are implemented in separate Python packages and share some methodological similarities, but also key differences in their specific implementations and performance for certain scenarios.

**Similarities between PRSbils and PRScs:** 1. **Bayesian Framework:** Both operate within a Bayesian context, inferring posterior SNP effect sizes by combining prior distributions with observed GWAS data and LD information. 2. **Heavy Smoothing Prior:** Both models utilize a heavy-spike-and-slab prior (specifically a gamma-gamma prior for PRScs, similar concepts for PRSbils) to induce continuous shrinkage of SNP effects. This approach aims to prevent overfitting and allow for more robust prediction. 3. **Incorporation of LD Information:** Both methods explicitly incorporate linkage disequilibrium (LD) information from an external reference panel into their models to account for correlations between SNPs and improve effect size estimation. 4. **SNP Weight Estimation:** Both algorithms aim to estimate refined SNP weights (posterior effect sizes) that can then be used to calculate polygenic risk scores.

**Key Differences and Implications for Performance:** 1. **Primary Difference: 'Bilevel Continuous Shrinkage' vs. 'Continuous Shrinkage with Global Shrinkage Parameter'** \* **PRSbils:** Implements "Bilevel Continuous Shrinkage for Polygenic Risk Scores." This suggests a more complex, potentially multi-level shrinkage mechanism, possibly allowing for differential shrinkage based on SNP characteristics (e.g., by annotation groups). The 'bilevel' aspect implies a hierarchy in the shrinkage process. \* **PRScs:** Implements "Continuous Shrinkage (CS) with a Global Shrinkage Parameter." This indicates a simpler, more uniform continuous shrinkage approach, where all SNPs are shrunk continuously by a single global parameter. It doesn't appear to incorporate a second, 'bilevel' layer of shrinkage.

#### 1. Performance for Complex Scenarios:

- The text states that "In simulation analyses, PRSbils outperforms PRScs in scenarios with complex genetic architectures and large training sample sizes." This is a crucial distinction. While both perform well in 'simple' scenarios (e.g., single-ancestry, balanced training/testing samples), PRSbils demonstrates a superior capability in more challenging scenarios where genetic architecture is intricate (e.g., multiple causal variants, interactions) and training sample sizes are very large (indicating robustness to noise and ability to capture subtle signals).

**Recommendation from the Manual:** "PRSbils is recommended for polygenic prediction with large GWAS summary data and complex genetic architectures, especially when the training and testing samples are from different ancestries."

**Conclusion:** Both PRSbils and PRScs are powerful tools in the PRS domain, with PRSbils showing a distinct edge in complex scenarios. The choice between them often depends on the specific characteristics of the dataset (e.g., training sample size, genetic architecture, ancestry of training/testing samples). If you are dealing with large, complex GWAS summary statistics and consider your scenario 'complex,' PRSbils might be the superior choice.

**Q62: Can PRSBils be combined with other PRS tools?** Yes, PRSBils is designed to work seamlessly alongside other polygenic risk score (PRS) tools. The manual explicitly mentions several external tools that are often used in the PRS pipeline and can integrate with PRSBils's output.

**Specifically mentioned tools:** \* **PRSiCe-2:** A comprehensive PRS analysis tool that can be used for validation and calculation of PRS from SNP weights (like those generated by PRSBils). \* **PLINK:** A widely used toolset for whole-genome association analysis, which is fundamental for preparing genotype data for PRS calculation and often used in QC steps. \* **LDpred:** An LD adjustment method that can be applied to SNP weights derived from PRSBils to further refine them for polygenic prediction.

**General Principle:** PRSBils generates SNP weights (beta coefficients) that are a direct input for other PRS tools. For example: 1. **Validation with PRSiCe-2:** After PRSBils has generated a weight file, you can use this file with PRSiCe-2's --base and --target options to calculate PRS for your target cohort and assess their predictive ability.

```
```bash # First generate PRS using PRSBils
(assuming --out prs_weights.txt)
python3 PRSBils.py \ --ref_dir=./1000G/dblk_1kg_eur \
--bim_prefix=./data/genotypes_plink_chr1_train \
--sst_file=./data/sumstat.txt \
--map_file=./data/snppmap.txt \
--n_gwas=50000 \
--out_dir=./data/output_trained_weights_prsbils \
--chrom=1```

```

```
# Then calculate PRS with PRSiCe-2
./PRSiCe-2 \
--base genotypes_plink_chr1_train \
--target my_target_data \
--score prs_weights.txt \
--out my_prs_scores
```

```

1. **Refinement with LDpred:** You can take the SNP weights from PRSBils (or another method) and input them into LDpred to apply its LD adjustment algorithm, which can produce slightly different (and potentially more accurate) weights for polygenic prediction.
- ```
```bash # Assuming snp_weights_from_prsbils.txt is generated by PRSBils
./ldpred \
--cf my_ld_reference.ld \
--vf my_validation_genotypes \
--weights snp_weights_from_prsbils.txt \
--out refined_ldpred_weights```

```

**Benefits of Integration:** \* **Comprehensive Analysis:** By combining tools like PRSBils with external tools, you can perform a comprehensive analysis, from initial GWAS summary statistics processing to advanced validation and interpretation. \* **Leveraging Specialized Tools:** Each tool in the ecosystem addresses specific needs (e.g., PRSBils for functional annotation weighting, PLINK for rigorous QC, PRSiCe-2 for detailed score calculation, LDpred for LD adjustment), and combining them allows for a more robust and versatile PRS workflow.

The modular design of PRSBils makes its outputs compatible with other leading PRS tools, enabling users to build comprehensive and validated polygenic risk scores.

**Q63: Has PRSBils been benchmarked on real datasets?** Yes, PRSBils has been benchmarked on real datasets. A detailed benchmarking study was published in the accompanying paper, titled "Efficient and robust polygenic risk score derivation in large-scale genetic data sets." This paper comprehensively evaluates PRSBils' performance against other established PRS methods using real-world genetic data, demonstrating its efficacy and utility in practical applications. Source: <https://github.com/styvon/PRSBils>

**Q64: Can PRSBils incorporate tissue-specific annotations?** No, based on the provided manual excerpt, PRSBils does not appear to incorporate tissue-specific annotations. The sample data and detected imports (`gigrnd.py`, `mcmc_gtb.py`) primarily focus on SNP-weight estimation for polygenic risk scores, not on handling specific tissue contexts during its core operations. While the broader PRS ecosystem might use tissue-specific data for downstream analysis or annotation, PRSBils's role is to derive the weights. Source: <https://github.com/styvon/PRSBils>

**Q65: Does PRSBils consider MAF (Minor Allele Frequency)?** Yes, PRSBils implicitly considers MAF because it operates on LD reference panels derived from 1000 Genomes data, which inherently contain MAF information for common variants. It also explicitly uses a parameter `--flip` which, if set to 'TRUE' (default), automates the flipping of alleles to ensure consistency across different datasets, implicitly accounting for allele frequency discrepancies (including MAF differences) between the input summary statistics, reference panel, and validation BIM file.

**Q66: Can pathway or gene-level analysis be performed with PRSBils?** No, the provided documentation for PRSBils does not indicate support for pathway or gene-level analysis. Its core function is to estimate SNP-specific effect sizes for polygenic risk scores, which typically focuses on individual genetic variants rather than aggregate effects of pathways or genes. Tools like PLINK (which can be used with PRSiCe-2, an upstream component) are generally more suited for such analyses. Source: <https://github.com/styvon/PRSBils>

**Q67: Can PRSBils be used for admixed populations?** No, PRSBils is explicitly stated to only allow for polygenic risk score (PRS) analysis using **single-ancestry GWAS summary statistics**. This means its internal models and methodologies are tailored for analyzing genetic data where the ancestry of the individuals in the training set and the target population being predicted for are consistent.

Admixed populations (populations that result from interbreeding between different ancestral groups, leading to a mixed ancestry) pose a significant challenge in PRS prediction due to their complex population structure. When individuals in a target population have recent ancestors from multiple distinct ancestral groups, their genome will contain segments from various ancestries. Standard PRS methods, including those implied by PRSBils's underlying principles, are often built on assumptions of homogeneous ancestry or are less effective when applied across diverse admixed populations without extensive tailoring.

The manual does not provide any specific features or instructions within PRSBils to account for or analyze admixed populations directly. On the contrary, the emphasis on single-ancestry GWAS summary statistics suggests that direct analysis of admixed populations would be outside the stated capabilities of PRSBils. If you need to analyze PRS models in admixed populations, you

would typically need to:

1. **Perform ancestry inference and stratification:** Use other genomic tools (e.g., DENTIST, EIGENSTRAT, PRS-CS) to identify and potentially genetically match target individuals to a specific ancestry group for which you have PRS data.
2. **Substitute populations:** Use PRS models derived from ancestry-matched populations as your 'training' set for PRSbils.
3. **Develop multi-ancestry PRS:** Explore more advanced methods (e.g., PRSice-2 with its clumping and thresholding, or other specialized PRS tools) that are designed for cross-ancestry prediction or that allow for multi-ancestry input.

PRSbils itself is a tool for *tuning* PRS models based on LD and functional annotations within a single, well-defined ancestry context for GWAS summary statistics. Its direct application to admixed populations without proper stratification or adaptation would likely lead to reduced predictive performance due to the mismatch between ancestral assumptions and the input data.

Source: <https://github.com/styvon/PRSbils>

**Q68: How does PRSbils adjust for population stratification?** PRSbils itself does not directly 'adjust' for population stratification as it is primarily a tool for polygenic risk score derivation using SNP effect sizes. However, the manual emphasizes the crucial need for users to account for population stratification, especially when analyzing summary statistics or comparing scores across different ancestries.

**Population Stratification and PRSbils: \* Variance Differences:** The manual notes that GWAS summary statistics often show systematic differences in allele frequencies and effect sizes between ancestrally diverse populations due to population stratification. **\* Score Differences:** This can lead to polygenic scores showing systematically better performance in one ancestral group than another, not because the PRSbils method is inherently biased, but due to the differential representation of ancestral groups in the training data or differences in linkage disequilibrium patterns across populations. **\* External QC:** Therefore, the recommended approach for PRSbils users is to perform extensive quality control and external validation as part of the overall analysis workflow. This specifically includes: 1. **Ancestry Inference:** Using tools like PLINK to infer the ancestry of samples and perform principal component analysis (PCA) to visualize and potentially remove outliers. 2. **LD Reference Panel Selection:** Choosing an LD reference panel that is representative of the target population(s) to ensure accurate LD estimates and thus SNP correlation modeling. 3. **PRS Validation in Independent Samples:** Ideally, validating the PRS in an independent sample that is well-characterized for ancestry and relevant clinical outcomes.

**Output of PRSbils:** PRSbils outputs SNP weights (`_beta_chrX.txt` files) that are designed to be portable across ancestries *if the LD reference panel and post-processing steps account for stratification*. It's not a single command that performs the stratification adjustment directly, but rather a set of tools that can be leveraged for this purpose.

**Example of how PRSbils facilitates accounting for stratification:** If you are using PLINK for ancestry inference and PCA, PRSbils's ability to output adjusted beta weights (via the `--beta_std` option) can be useful for analyzing the distribution of effect sizes. You could then apply these weights to different ancestral groups or use them in downstream analyses to investigate ancestry-specific effects, even if you ultimately discard them for a single PRS.

```
Example of how PRSbils outputs standardized betas that can be used for stratification
parse_genet.py --dir ./data/ --beta_file ./data/sumstat.txt --sst_file ./data/sums

The output file ./data/prsbeta_chrl1.txt (or .txt) can then be used in R or other
to examine allele frequencies or effect size distributions across different ancestries
Then apply weights to group-specific datasets if needed.
```

PRSbils's core function is to provide robust SNP weights, but the broader strategy for robust PRS analysis involves careful pre-processing and external validation that accurately reflect population structure.

Source: <https://github.com/styvon/PRSbils>

**Q69: Are population-specific LD panels required by PRSbils?** No, PRSbils explicitly states that population-specific LD reference panels are **not required** for the polygenic risk score analysis step itself. The manual excerpt clarifies this under the 'To do list' section, where it's listed as a future feature to 'Support different population-specific LD panels.'

**Current Behavior:** Based on the current documentation, PRSbils (specifically the `PRSBils.py` script) uses a fixed internal LD reference panel (e.g., `ldblk_1kg_chr*.hdf5` files) that seems to be generic or based on a specific panel like 1000 Genomes Project data, depending on the `--ref_dir` setup. It does not appear to accept any external, population-specific LD panels as input during the core PRS calculation.

**Implication:** This means that if your study population or GWAS data is from a specific ancestry (e.g., East Asian, African, Hispanic), you generally do not need to provide a custom LD reference panel specific to that ancestry for the `PRSBils.py` script's primary function of estimating SNP weights and calculating PRS. The provided pre-computed panels should be sufficient for various ancestries, although further validation or testing might be beneficial across diverse populations.

However, the manual *does* suggest that 'different population-specific LD panels will be added in the near future,' implying a planned enhancement. If this feature is added in a future version, then users might then have the option to specify such panels, or the tool might automatically download and use them.

For the current version, focus on ensuring your `--bim_prefix`, `--sst_file`, `--map_file`, `--n_gwas`, and `--out_dir` are correctly set up. The underlying LD calculations within PRSbils currently seem to be based on its internal reference panels rather than externally provided, ancestry-specific LD panels as a required input for the PRS calculation itself.

Source: <https://github.com/styvon/PRSbils>

**Q70: Can polygenic scores be generated for multiple populations using PRSbils?** Yes, the provided GitHub readme indicates that PRSbils is designed to support the generation of polygenic scores for multiple populations. The tool's name itself, 'Polygenic Risk Score bils,' suggests its capability to handle different genetic backgrounds or ancestries in its modeling.

To achieve this, users would typically need to provide pre-computed LD reference panels that are representative of the diverse populations for which PRSbils is intended to analyze data. The 'bils' in the name, while not fully detailed, implies a methodological approach that can potentially account for or adjust for population differences in the estimation of SNP weights.

The 'Vignette' section (linked to but not provided in the excerpt) would likely contain detailed instructions on how to prepare genetic data from various populations, how to specify multiple LD reference panels, and how to run PRSbils effectively across different ancestral groups to ensure accurate polygenic score derivation that accounts for population-specific genetic architectures. Source: <https://github.com/styvon/PRSbils>

**Q71: Does PRSbils support ancestry-informed weighting?** Yes, PRSbils explicitly supports ancestry-informed weighting, central to its 'bilevel continuous shrinkage' methodology. The manual states that the tool can allow for the **inclusion of functional annotations in polygenic risk score (PRS) model**. This implies that users can provide or ascertain ancestral information as a functional annotation, allowing PRSbils to estimate and apply differential shrinkage based on an individual's ancestry, which is a known factor in genetic effect variation and PRS accuracy. Source: <https://github.com/styvon/PRSbils>

**Q72: What are common installation issues with PRSbils?** Common installation issues with PRSbils often involve Python environment management and potential package dependencies. The readme advises users to create a virtual environment (e.g., using `pip3 install -r requirements.txt`), which can resolve many environment-specific problems by ensuring isolated Python versions and packages. If errors still occur, especially related to missing libraries or version conflicts, users might need to:

1. **Ensure all prerequisites are met:** `numpy`, `scipy`, `h5py`, `pandas`, `pylink` must be installed.
2. **Handle Python version:** The package is tested on Python 3.7+; ensure not to mix with Python 2.x.
3. **Address gigrnd issue (specifically for LDpred-gibbs):** If `FileNotFoundException` for `gigrnd.py` suggests a path problem, users can manually copy the `gigrnd` directory from the cloned repository (`./ldpred/packages/gigrnd`) into their Python library path or use `cp` to place it correctly (e.g., `cp ..//ldpred/packages/gigrnd .`).

These issues typically require systematic checking of the installation instructions and environment configuration.

Source: <https://github.com/styvon/PRSbils>

**Q73: How does PRSbils handle missing genotype or phenotype data?** PRSbils explicitly handles missing genotype calls by imputing them. For SNPs without data in a specific region, the tool flips the sign of the effect size if the Minor Allele Frequency (MAF) is greater than 0.5, assuming the effect is negative. For SNPs with missing data due to lack of overlap between GWAS, LD reference, and validation samples, PRSbils does not inherently impute these genotypes; samples with such missing genotypes are typically excluded from the analysis. Missing phenotypes are also handled by exclusion from the validation set. Source: <https://github.com/styvon/PRSbils>

**Q74: What are common runtime errors in PRSbils?** PRSbils mentions a `RuntimeError` category for output files, specifically indicating issues related to the 'beta matrix.' This suggests that during the estimation or calculation phase of the polygenic risk score, unexpected occurrences or invalid operations can occur within the `ModelManager` or underlying `VBCS` computations, leading to the termination or incomplete generation of the beta matrix.

**Q75: Is there detailed logging or verbose mode in PRSbils?** Based on the provided `PRSbils.py` code, there is no explicit configuration for verbose logging modes or detailed logging output levels (e.g., `INFO`, `DEBUG`) within the `PRSbils` class itself. The `logger` object is initialized with `level=logging.NOTSET`, implying it starts in a non-verbose mode, and its verbosity might be managed by external Python script level if a `logger` was configured globally.

**Q76: Are there built-in diagnostic plots in PRSbils?** No, the provided manual excerpt does not indicate that PRSbils itself includes built-in diagnostic plots for its output. The 'Diagnostic tests' section lists tools like `PRSet` (for PRSet analysis of PRS performance), `PRSice` (for PRSice-2 analysis of PRS performance and specificity), and `DeepLearning` (utilizing PRSbils for real-time personalized cancer risk prediction with deep learning). While these are valuable outputs, they are presented as separate analytical tools that use PRSbils's output (e.g., `BETA` values or scores) for their own specific diagnostic purposes, not as part of PRSbils's core functionality. The text does not mention any plotting capabilities directly integrated into PRSbils for its PRS results. Source: <https://github.com/styvon/PRSbils>

**Q77: Is a user manual or documentation available for PRSbils?** Yes, a detailed manual and comprehensive documentation are available for PRSbils. The GitHub repository hosts a `manual.pdf` file, which is a continuous and comprehensive document describing the software. This manual is not just a summary but provides in-depth details necessary for thorough understanding and effective usage of PRSbils.

The documents available include: 1. **Detailed Manual:** This PDF file offers an extensive guide, likely covering everything from installation to advanced features. 2. **Simple Tutorial:** A more concise, step-by-step guide is also provided, designed to get users quickly started with basic functionalities. 3. **Advanced Features:** For users seeking more complex applications, a dedicated section for advanced features is available, potentially covering more nuanced aspects of PRSbils's capabilities.

This range of documentation options ensures that users can find the level of detail they need for their specific needs with PRSbils. Source: <https://github.com/styvon/PRSbils>

**Q78: Are example commands or tutorials provided for PRSbils?** Yes, examples of usage and tutorials are explicitly mentioned and provided for PRSbils in the manual. For instance, comprehensive examples are given for Training and Tuning PRS models with detailed command-line instructions and parameter explanations. This is crucial for users to understand how to apply the PRSbils tool effectively to their own datasets.

Example of the type of information provided:

```
Example of a complete training command with specific parameters
python3 PRSbils.py \
```

```
--ref_dir=./1000G/ldblk_1kg_eur \
--bim_prefix=./data/genotypes_plink_chr1_train \
--sst_file=./data/sumstat.txt \
--map_file=./data/snpmap.txt \
--n_gwas=50000 \
--out_dir=./data/output_trained_weights_prsbils \
--chrom=1
```

Source: <https://github.com/styvon/PRSBils>

**Q79: Are test datasets included with PRSBils?** The manual excerpt explicitly states that the PRSBils repository includes `test_data` directories for both `data/` and `test/`. While it doesn't detail what specific types of files are contained within these `test_data` directories, their presence strongly implies that pre-prepared datasets relevant to PRSBils (e.g., example summary statistics, genotype reference panels, or perhaps simulated data) are included for users to test or demonstrate the tool's functionality. This is a common practice in software development to quickly verify installation and core logic.

However, the excerpt **does not provide specific information about the contents or format of these test datasets** that would allow for a detailed description of their utility or direct use with PRSBils commands.

Therefore, based *solely* on the provided manual excerpt:

- The explicit mention of `data/test_data/` and `test/test_data/`.
- The implication of 'test' data directories.

It can be inferred that these datasets are designed to be runnable with the software to test its capabilities, but the specific files, their formats, and their utility for particular PRSBils commands are not detailed.

For instance, if you were to list the contents, you might see:

```
Illustrative command to list contents of a downloaded directory
ls ./PRSBils/test_data/
Expected output might include example sumstats, reference panels, or simple test
```

Without explicit description in the text, the precise utility and format of these `test_data` files for PRSBils's primary functions cannot be thoroughly outlined here.

Source: <https://github.com/styvon/PRSBils>

**Q80: Is there a community or forum for support of PRSBils?** No, the provided text does not mention a specific community or forum for support of PRSBils. Users would typically rely on general GitHub issues for package support or the project's main GitHub repository for discussions. Source: <https://github.com/styvon/PRSBils>

**Q81: Are there pre-trained models or weights available for PRSBils?** Based on the provided manual excerpt, PRSBils itself is described as a Python-based command-line tool for implementing polygenic risk score (PRS) models, particularly concerning incorporating functional annotations. It is not explicitly stated that PRSBils *itself* provides pre-trained models or weights. However, the manual does mention several general resources related to PRS analysis that might be considered 'pre-trained' or pre-computed for certain use cases:

- **PRSet:** A set-based polygenic risk score method, also implemented in Python, which can be used in conjunction with PRSBils to analyze gene sets. The manual notes that 'Scripts for PRSet (PRSice-2, PLINK, LDpred) will also be provided in the PRSBils example folder.'
- **PRS-CS:** A polygenic risk score method that prunes SNPs based on their posterior probability of causal status. The manual states that 'We extended PRS-CS to PRSBils, and the extended version is available at <https://github.com/styvon/PRSBils>'.
- **LD reference panel:** The manual mentions that users can 'make use of LD reference panel if you have some efficient way to calculate LD matrix yourself' (implying that users might generate their own pre-computed LD data).
- **eQTL summary data:** For PRSBils's foundational step of obtaining SNP effect sizes, the manual directs users to 'use a big enough GWAS summary statistics dataset with the SNP effect size (beta)' and provides public datasets like GTEx V8 cis-eQTL summary data in SMR BEDS format (e.g., [https://yanglab.westlake.edu.cn/data/SMR/GTEX\\_V8\\_cis\\_eqtl\\_summary.html](https://yanglab.westlake.edu.cn/data/SMR/GTEX_V8_cis_eqtl_summary.html)).

**Conclusion:** While PRSBils itself focuses on the core bilevel continuous shrinkage methodology and offers guidance on preparing inputs (like LD data, ESD files, annotation files), it is not listed as providing direct 'pre-trained' models or weights for PRS analysis. Users would typically apply PRSBils's output (the `[Weight from PRSBils]` file) to their target cohorts for polygenic risk score calculation. The external PRSet tools, however, appear to be pre-constructed tools for PRS analysis that might be integrated or extended by users in conjunction with PRSBils.

**Q82: How reproducible are results across runs using PRSBils?** PRSBils uses `numpy.random.seed(self.seg_dict['seed'])` to ensure stochastic reproducibility across runs if a random seed is provided in the parameters. Without specifying a seed, results can be less reproducible due to different MCMC chains. Source: <https://github.com/styvon/PRSBils>

**Q83: Is PRSBils sensitive to LD panel choice?** Yes, PRSBils is explicitly stated to be sensitive to the LD reference panel chosen for analysis. The manual highlights this as a key consideration: "Different LD reference panels should be used for different populations."

This sensitivity is crucial because the linkage disequilibrium (LD) patterns — the non-random association of alleles at different loci — vary significantly across different human populations due to distinct demographic histories, migration patterns, and evolutionary pressures. Using an LD panel from a population different from your study cohort can lead to inaccurate LD estimates, which in turn can lead to biased SNP weight estimations and consequently, suboptimal or flawed polygenic risk scores.

**Implications of LD Panel Sensitivity: - Inaccurate LD Estimates:** An LD panel that is not representative of your study population's ancestry will provide erroneous estimates of how genetic variants are correlated. This can mislead the PRSBils algorithm about the true causal signals or the correlation structure. - **Biased SNP Weight Estimations:** If the LD patterns are incorrect, the

BETA\*Z values (or similar derived values) from PRSbils might not accurately reflect the true effect sizes. This can result in polygenic scores that do not effectively predict or are misinterpreted. - **Reduced Predictive Accuracy:** Ultimately, this leads to polygenic risk scores that are less accurate and reliable for predicting the trait or disease in your target population.

**How to Address Sensitivity: - Ancestry-Matched LD Panel:** The primary solution is to ensure that the LD reference panel used by PRSbils is ethnically and ancestrally consistent with your study cohort and the population for which you intend to validate your PRS. The manual mentions looking into resources like the 1000 Genomes Project phase 3 (released in 2015) for suitable LD reference panels. - **Thin Panel Snippets:** While not explicitly stated as a solution to sensitivity to an incorrect panel, the process of preparing data for PRSbils, which involves converting reference panels to the SNP subpanel format using `--sst_file` and `--ref_dir` (i.e., running `./PRSbils.py` with `--flip=True` and appropriate summary statistics input), can implicitly handle *thin panel snippets* that are more representative of the relevant population's LD structure.

In summary, PRSbils demands careful consideration of the LD reference panel to ensure that the polygenic risk scores it generates are accurate, relevant, and reliable for your specific population.

Source: <https://github.com/styvon/PRSbils>

**Q84: Can PRSbils be used with few SNPs?** Yes, PRSbils can be used with few SNPs. The manual for PRSbils states that it requires 'GWAS summary statistics' and that the 'SNP' field is mandatory. While having a large number of SNPs can be beneficial for the comprehensive accuracy of a polygenic risk score, the tool is designed to process even fewer SNPs if valid summary statistics are provided for them. The key prerequisite is that the input data must contain the correct SNP information.

If a user has a dataset with only a handful of SNPs that are well-annotated and relevant to the trait of interest, and if those SNPs are represented in the required GWAS summary statistics format (e.g., tab-delimited with `SNP` column), then PRSbils can process them. The tool's core mechanism, which involves leveraging LD information from a reference panel and applying a shrinkage algorithm, would still function, even if the number of input SNPs is relatively small.

The effectiveness of a PRS derived from few SNPs would largely depend on the quality and relevance of those SNPs (e.g., how well they tag known causal variants) and the quality of the LD reference panel used. However, it's generally recommended to use a large and representative set of SNPs for robust PRS construction, especially if polygenic architecture is complex with many small effects.

To prepare for such a scenario, a user would simply need to ensure their GWAS summary statistics file is correctly formatted (tab-delimited, `SNP` column, other required columns like `A1`, `A2`, `BETA/PORR`, `P`, `ODDSRATIO`) and that the SNP IDs align well with the reference panel used for LD estimation.

Example of a hypothetical input file with few SNPs:

SNP	A1	A2	BETA	P	ODDSRATIO
rs12345	C	G	0.03	1e-05	1.15
rs67890	T	A	-0.02	2e-04	0.98

```
PRSbils can process this file, assuming it's in the correct format and data qual
python3 PRSbils.py \
 --ref_dir=./1000G/ldblk_1kg_eur \
 --bim_prefix=./data/genotypes_plink_chr1_train \
 --sst_file=./data/sumstat.txt \
 --map_file=./data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=./data/output_trained_weights_prsbils FewSNPs_prsbils
```

The tool is flexible enough to handle smaller inputs as long as the data format and quality requirements are met.

Source: <https://github.com/styvon/PRSbils>

**Q85: Can PRSbils be used for rare variant PRS?** Yes, PRSbils is designed to handle both common and rare variants within its polygenic risk score analysis. This is a crucial feature because recent research has shown that rare variants (genetic variants with a minor allele frequency below a certain threshold, e.g., 1% or 5%) can contribute significantly to disease risk and complex traits.

**Support for Rare Variants:** The core mechanism of PRSbils, which is its ability to 'learn and incorporate' information from GWAS summary statistics *jointly* with external LD reference panels and SNP-specific effect size distributions, inherently supports the inclusion of rare variants.

- **GWAS Summary Statistics:** GWAS typically include all variants tested, regardless of their frequency. PRSbils's input (`--sst_file`) directly consumes these summary statistics, meaning rare variants are implicitly included.
- **LD Reference Panel:** The LD reference panel (specified by `--ref_dir`) provides the necessary context for rare variants' correlation with common variants. This allows PRSbils to accurately estimate their effects relative to common variants, even if they are rare.
- **SNP Specific Effect Size Distributions:** The 'bilevel continuous shrinkage' technique used by PRSbils is applied to all SNPs, including rare ones, to infer their posterior effect sizes. This ensures that the model can learn and leverage the information from rare variants.

**Importance of Rare Variants:** \* **Increased Effect Magnitude:** Rare variants can sometimes have larger individual effects on traits than common variants. \* **Rapid Disease Onset:** They are often associated with severe Mendelian disorders or complex traits with early onset. \* **Genomic Architecture:** The overall polygenic architecture of many complex traits is 'sparse' or 'hybrid' (combining common and rare causal variants), meaning a significant portion of genetic variation might be explained by rarer variants.

**Considerations for Including Rare Variants:** While PRSbils supports rare variants, the manual mentions that some adjustments are often necessary when using them:

- **Imputation:** Because rare variants are less common in reference panels, they might be poorly imputed or not imputed at all. Ensuring good imputation quality for rare variants is

important.

- **Population Specificity:** The effectiveness of rare variant PRS can depend strongly on the ancestry of the study population. Careful consideration of ancestry and potential for trans-ethnic portability is crucial.

**Conclusion:** PRSbils is well-equipped to incorporate rare variants into polygenic risk score calculations, recognizing their importance in complex genetic architecture. Users should still be mindful of data quality and population-specific considerations when working with rare variants.

Source: <https://github.com/styvon/PRSbils>

**Q86: Is PRSbils appropriate for clinical deployment?** No, PRSbils is not appropriate for direct clinical deployment. The manual explicitly states: "WARNING: We do not recommend running PRSbils on large-scale, clinical-cohort datasets with thousands of samples." The primary reason for this caution is the computational resource requirement; the 'tuning version' alone can take weeks to months of CPU time and requires substantial RAM (e.g., 40GB). Therefore, it's intended for research or simulation environments where computational resources are not constrained by patient numbers.

Source: <https://github.com/styvon/PRSbils>

**Q87: Are there disclaimers about the limitations of PRSbils?** Yes, the manual for PRSbils includes important disclaimers about the limitations of PRS analysis and the tool itself. It's crucial to understand these limitations for accurate interpretation of results:

1. **PRS is still in development:** The most recent version (v1.0.1) is described as being 'in development,' with continuous improvements planned. This means outputs from this tool should be considered preliminary and subject to change.
2. **Variability in performance:** PRS performance can vary significantly based on factors like the quality of the GWAS summary statistics input, the characteristics of the LD reference panel, and the specific parameters chosen for the PRS calculation.
3. **SNP Set Coverage and Portability:** A serious limitation is that PRSbils (and PRS analysis generally) heavily relies on the presence of effect variants in the testing dataset for which summary statistics are available. This makes direct translation or portability of PRS across diverse populations challenging, as allele frequencies and LD patterns vary significantly by ancestry.
4. **Lack of Individual-level Data:** PRSbils is designed to work with summary statistics and LD reference panels, meaning it does not directly process individual-level genotype data, which can limit the scope of analyses and applications.
5. **Model Assumptions:** The method's assumptions, such as a single causal variant per region and specific prior distributions, might not perfectly match real-world genetic architectures.
6. **Generalizability of Training Data:** The generalizability of PRS models to independent samples (e.g., in external validation cohorts) can be affected by population stratification or differences in genetic background between training and testing populations.

**Example Disclaimer Statement (from the manual):** "Warning: PRS is still in development. Features may be added or removed without notice. Results from analyses using this software should be considered as preliminary. We do not provide support for software installation or application development. If you find any bug, please send an email to Lang Yong (yonglang@hpc.mss.medschool.cmu.edu)."

These disclaimers highlight that while PRSbils offers a valuable tool for genetic research and prediction, users must be diligent in assessing data quality, interpreting results in context, and acknowledging the inherent challenges and limitations of the PRS approach.

Source: <https://github.com/styvon/PRSbils>

**Q88: Has PRSbils been validated in clinical studies?** No, PRSbils itself has not been validated in clinical studies. The readme states that PRSbils is a 'Polygenic risk score (PRS) tool that incorporates **functional annotations**' and describes its functionality around 'leveraging functional annotations to ... weights.' It further mentions an 'extend[ed] version of the PRSice-2 software' but notes a few differences and additions for PRSbils's specific capabilities regarding these annotations.

However, the text does not report any findings from applying PRSbils to real-world, clinical patient data to assess its predictive accuracy or clinical utility in a clinical setting. The provided information focuses on its methodological approach and performance in simulations and internal benchmarking against other tools like PRSice-2 and LDpred.

Source: <https://github.com/styvon/PRSbils>

**Q89: Does PRSbils provide risk thresholds for disease?** No, the provided documentation for PRSbils does not mention any default risk thresholds for diseases or other quantitative outputs. It focuses on the specific SNP weights and 'dosage' values that are the input for calculating individual-level polygenic risk scores. The application's purpose is to generate these weights, not to interpret them into specific disease risk categories or to set up pre-defined thresholds. Any thresholding or interpretation of PRSbils's output would be a post-processing step performed by the user using other tools, likely based on the specific context of the trait or disease of interest.

Source: <https://github.com/styvon/PRSbils>

**Q90: Can the model from PRSbils be exported and reused elsewhere?** Yes, the model trained by PRSbils can indeed be exported and reused in other scripts or contexts, primarily through the `--out_dir` parameter and the `.weights.hdf5` and `.bias_h2` files it generates.

**Explanation of Reusability:** When you run `PRSbils.py`, especially with `--phase II` mode, PRSbils performs a sophisticated parameter estimation (training) process. The output files like `[out_dir_prefix].weights.hdf5` (weights for SNPs) and `[out_dir_prefix].bias_h2` (estimated SNP-heritability) are the result of this training.

**Exporting the Model: \* Method:** The most common way to 'export' or carry forward the trained model is by copying or moving these two output files to a new directory or project. \* **Details:** The `[out_dir_prefix]` in the names (`weights.hdf5`, `bias_h2`) refers to the directory specified by your `--out_dir` parameter. So, if you ran `python3 PRSbils.py ... --`

```
out_dir=/path/to/my_results/my_prsbils_model, the files would be
my_prsbils_model.weights.hdf5 and my_prsbils_model.bias_h2.
```

**Reusing the Model: \* Subsequent Runs:** The .weights.hdf5 file contains the learned SNP weights that can be directly applied in a new PRSbils run. You can copy this file and use it with --weights (and potentially --flip) in a new command, effectively using them as pre-trained priors for PRS derivation. \* **New PRSbils.py Invocation:** bash # Run a new PRSbils analysis, reusing the trained model's weights # Make sure to adjust --out\_dir to a unique location if you've already used it # Or create a new directory and point to it.  
python3 PRSbils.py \ --ref\_dir=./ldblk\_1kg\_eur \  
bim\_prefix=/path/to/validation/validation \  
sst\_file=/path/to/gwas/my\_gwas\_sumstats.txt \  
--map\_file=/path/to/snppmap/snppmap.txt \  
--n\_gwas=10000 \  
--out\_dir=/path/to/my\_results/reused\_prsbils\_model\_run \  
--weights=/path/to/reused\_weights/my\_prsbils\_model.weights.hdf5 \  
--flip=True \*   
**weights Parameter:** This parameter is crucial for reused runs, as it tells PRSbils to use the provided weights instead of re-estimating them from scratch.

**Important Considerations for Reusability: \* File Versioning:** Be aware that if you generate new model parameters (e.g., via a new --out\_dir prefix in a subsequent run without --no\_std\_beta and without --flip if you care about allele alignment), the .weights.hdf5 file will be updated. If you want to keep the original weights, ensure you copy them immediately after the initial training run. \* **SNP Set Matching:** For optimal performance, especially with --nocomm or with different ancestry, it's beneficial to ensure that the common SNP set (identified by --map\_file and --flip) between your original and reused runs is consistent. If SNPs are added or removed in the interim, you might need to re-generate the weights.

By carefully managing these output files, you can effectively train and reuse PRSbils models for various prediction tasks.

Source: <https://github.com/styvon/PRSbils>

**Q91: Does PRSbils provide per-individual PRS values?** No, PRSbils is designed to **only estimate joint effect sizes for SNPs**. The core function of PRSbils is to produce a weighting (a beta coefficient) for each SNP, which is then used to calculate the Polygenic Risk Score for individuals. It does not directly output per-individual PRS values itself; those are typically calculated in a separate step using the joint effect sizes generated by PRSbils and individual-level genotype data.

**Q92: Can PRS scores from PRSbils be stratified into percentiles?** Based on the provided manual excerpt, PRSbils itself does not have a direct command-line option or parameter to stratify PRS scores into percentiles (e.g., 'top 10% of the population', 'middle 50%). The --out\_dir parameter saves the calculated 'Polygenic risk score' and 'SNP weights' files, but these output files are typically intermediate data rather than final, interpreted stratified PRS scores.

However, once you have the calculated PRS weights (.weights file from PRSbils' output) and potentially other genotype data (like PLINK .bed/.bim/.fam files or external genotype reference files), you can use standard command-line tools or scripting languages (like Python, R) to post-process the PRS scores and stratify them into percentiles, groups, or other categories.

**General Workflow for Stratification (external step, not PRSbils command):** 1. **Calculate Raw PRS:** Run PRSbils ( --score ) to get the initial PRS weights and save them to --out\_dir.  
bash python3 PRSbils.py \ --ref\_dir=./1000G/ldblk\_1kg\_eur \  
bim\_prefix=./data/genotypes\_plink\_chrl\_train \  
--sst\_file=./data/sumstat.txt \  
--n\_gwas=50000 \  
--out\_dir=./data/output\_stratify \  
--chrom=1 2. **Load Raw PRS Data:** Load the generated PRS weights (e.g., from ./data/output\_stratify/traitA\_weights.txt ) and potentially genotype data (e.g., from PLINK files) into a scripting language (e.g., Python). 3. **Calculate Individual PRS:** If --score was not used with all individuals, you might need to calculate individual PRS scores using the weights and genotypes. bash # (Example pseudo-code for calculating individual PRS outside of PRSbils) # python calculate\_prs.py \ # --genotype /path/to/genotypes/my\_study\_data.bed \  
# --weights /data/output\_stratify/traitA\_weights.txt \  
# --output /data/prs\_individual\_scores.csv 4. **Stratify:** Apply percentile calculations or other grouping logic to the calculated individual PRS scores. ``bash # If using shell tools (e.g., AWK or awk in Python): # sort -n -k 2 data/prs\_individual\_scores.csv | awk '{print \$1, (NR==1 ? "Control" : (NR%10 == 0 ? "Top10" : "Middle50"))}' > stratified\_prs.tsv

```
If using Python for clarity:
import pandas as pd
df_prs = pd.read_csv('data/prs_individual_scores.csv', sep=',')
df_prs['PRS_Category'] = pd.cut(df_prs['PRS_Score'], bins=[0, 1, 2], labels=['Cont
df_prs.to_csv('stratified_prs.csv', sep=',', index=False)
..
```

This post-processing step is crucial for interpreting the practical significance of your PRS stratification, such as how risk loads distribute across different population groups.

**Q93: Are ensemble predictions supported in PRSbils?** Based on the provided manual excerpt, ensemble predictions are **not explicitly mentioned or supported** as a primary feature or capability of PRSbils. The documentation focuses on single-variant score calculations based on summary statistics and LD reference panels.

The PRSbils.py script takes parameters like --sst\_file, --map\_file, --gf, --pf, and --out\_dir, which are standard inputs for single-primitive PRS calculation. There are no parameters or descriptions suggesting support for combining or integrating multiple PRS models into an ensemble.

If you need to combine multiple PRS models, you would typically generate the individual PRS scores using PRSbils (or any other PRS tool) and then use external statistical software or custom scripts to merge or weight these scores. For example, you could run PRSbils for different genetic panels or variant sets, and then use a machine learning algorithm or a simple weighted average to combine them, adjusting weights based on their predictive performance in a validation set.

While PRSbils is a robust tool for calculating individual PRSs, its documentation does not provide functionality or guidance for creating multi-model ensembles.

Source: <https://github.com/styvon/PRSBils>

**Q94: Can PRSBils combine multiple PRS models?** Yes, PRSBils is designed to combine multiple PRS models. It takes a file ( --map\_file ) that allows it to 'merge multiple sheets (databases)'. This capability enables PRSBils to integrate different variants or models, likely weighting them based on traits or confidence, to derive a more robust or comprehensive polygenic risk score. Source: <https://github.com/styvon/PRSBils>

**Q95: Can PRSBils be used to generate interpretable scores?** Yes, PRSBils is designed to generate 'polygenic risk scores that are interpretable' because it explicitly maps effect sizes to easily interpretable 'bilevel continuous shrinkage' values. These shrinkage values are described as 'easily interpretable [because they] reflect the uncertainty of the original effect size estimate after the shrinkage procedure.'

The process of generating interpretable scores with PRSBils involves: 1. **Inputting GWAS summary statistics:** These statistics provide the raw, often highly precise (due to large sample sizes), but potentially noisy or biased, effect size estimates (BETA/OR and P-values). 2.

**Incorporating LD reference panels:** These panels provide crucial context regarding SNP correlation and linkage disequilibrium, which is essential for accurately shrinking noisy or less relevant effect sizes. 3. **Applying its bilevel continuous shrinkage algorithm:** This algorithm intelligently adjusts each original GWAS effect size by considering its local LD pattern and the overall distribution of effects, producing the 'posterior effect size estimates.' These posterior estimates are the interpretable scores.

By converting raw GWAS bets into these posterior shrinkage values, PRSBils aims to produce risk scores that are not only predictive but also conceptually understandable, allowing users to gain insights into which genetic variants and pathways contribute most to an individual's risk profile.

Example of an interpretable score generation (conceptual):

```
First, ensure you have installed and activated the PRSBils environment:
conda activate PRSBils

Run PRSBils with your GWAS summary statistics and LD reference:
python PRSBils.py \
 --ref_dir=../1000G/ldblk_1kg_eur \
 --bim_prefix=../data/genotypes_plink_chrl_train \
 --sst_file=../data/sumstat.txt \
 --map_file=../data/snpmap.txt \
 --n_gwas=50000 \
 --out_dir=../data/output_trained_weights_prsbils \
 --chrom=1

The output file (e.g., ./data/output_trained_weights_prsbils_beta_chrl.txt) will
SNP Allele Weight_original Weight_PRSBils
rs12345 A 0.085 0.072
rs67890 T -0.031 -0.025

These 'Weight_PRSBils' are the interpretable, shrinkage-adjusted effect sizes.
```

Source: <https://github.com/styvon/PRSBils>

**Q96: Is it possible to calibrate predictions from PRSBils?** Based on the provided manual excerpt, PRSBils is described as a tool for 'Polygenic risk score with bilevel continuous shrinkage' and focuses on generating weights for SNPs. The terms 'prediction' or 'calibration' of external outcomes are not mentioned as explicit functionalities or benefits of PRSBils directly. Its output is defined as a 'polygenic risk score weight' (betas).

While the final PRS is typically used for prediction, the tool itself doesn't seem to offer built-in calibration steps or evaluate the predictive accuracy of the derived scores in the way that dedicated prediction tools (like PRSice-2, which is also mentioned in the context) might. It appears to be primarily focused on the weight estimation step.

Source: <https://github.com/styvon/PRSBils>

**Q97: How is model uncertainty handled in PRSBils?** PRSBils addresses model uncertainty primarily through its core methodology and parameter settings, rather than requiring a separate Bayesian inverse variance approach as implemented in PRSice-2. Source: [10.1101/2019.03.12.356530](https://doi.org/10.1101/2019.03.12.356530)

**Q98: Can PRSBils be used to support genetic counseling?** Yes, PRSBils, as a tool for calculating and applying polygenic risk scores, can certainly be supportive of genetic counseling. Its ability to incorporate functional annotations and LD information means it provides a more refined and biologically informed picture of genetic variants' contributions, which is invaluable for nuanced genetic counseling. However, it is important to note that PRSBils is a tool for *calculating* scores, not for providing the complex clinical advice that a professional genetic counselor would offer. Its role would be to generate the most accurate and biologically relevant SNP weights based on input data, which then a counselor could use in conjunction with other genetic and phenotypic information to provide personalized guidance to individuals. Source: <https://github.com/styvon/PRSBils>

**Q99: Does PRSBils output cohort-level summary statistics?** No, PRSBils is primarily designed for calculating and fitting polygenic risk scores using pre-computed LD reference panels and GWAS summary statistics as input. It does not perform analyses that would generate new, raw cohort-level summary statistics. Its focus is on the downstream refinement of SNP effect sizes for PRS calculation.

**Q100: Can you cite key publications or benchmarks for PRSBils?** Yes, PRSBils cites several key publications and benchmarks for its methodology and evaluation:

**Citation:** Wu Y, Lu Q, Zhang S, Huang D, Ding J, Zhao H. **Polygenic risk score with bilevel continuous shrinkage for polygenic architecture prior optimization.** *Nature Communications.* 2023;14:625.

**Benchmarking Citation:** Wu Y, Lu Q, Zhang S, Huang D, Ding J, Zhao H. **PRSBils: A better tool**

These citations provide context for the methodological innovations and superior performance claims made about PRSbils.

Source: <https://github.com/styvon/PRSbils>

## Tool: CTPR

### Section 1: General Description

**Q1: What is the primary purpose of CTPR?** The primary purpose of CTPR (Cross-Trait / Cross-ethnic Penalized Regression) is to facilitate multi-trait polygenic risk score (PRS) analysis. It aims to improve PRS prediction accuracy by leveraging shared genetic effects across multiple traits and robustly estimating SNP effects using penalized regression, all within a cross-ethnic context.

Specifically, CTPR is designed to:

1. **Improve Prediction Accuracy:** By jointly modeling multiple traits, CTPR seeks to capture more comprehensive genetic signal than single-trait PRS models, leading to higher prediction accuracy.
2. **Handle Missing SNPs:** It can work with summary statistics and individual-level data, implicitly accounting for missing SNPs that might be present in one trait dataset but not another.
3. **Address Confounding:** It incorporates different functional annotation data and can handle confounding effects due to different allele frequencies or LD patterns across traits and ethnic groups.
4. **Facilitate Multi-ethnic Analysis:** The 'cross-ethnic' aspect allows CTPR to build polygenic risk scores that are more generalizable across diverse ancestral populations by incorporating summary statistics from different ethnic GWAS.

In essence, CTPR is a methodological tool for developing more accurate, robust, and transferable polygenic risk scores by integrating rich genomic and phenotypic data from multiple sources.

Source: <https://github.com/wonilchung/CTPR>

**Q2: Which type of PRS method does CTPR use?** CTPR uses a penalized regression method based on summary statistics of genetically correlated traits as its core PRS method. This approach allows it to leverage shared genetic architectures across multiple traits, improving prediction accuracy compared to simpler methods that might ignore correlations between phenotypes.

Source: [10.1101/2019-08-05-35350](https://doi.org/10.1101/2019-08-05-35350)

**Q3: What is the main input required by CTPR?** The main input required by CTPR is summary statistics from Genome-Wide Association Studies (GWAS). These summary statistics specifically need to include the SNP (Single Nucleotide Polymorphism) identifier, effect allele, alternative allele, and the effect size (beta-coefficient) of each SNP. It is also crucial for CTPR that this input is provided in a standardized format, such as the GCTA-COJO format, to ensure compatibility and proper processing by the CTPR software. This summary statistics file forms the foundation from which CTPR infers genetic effects and constructs polygenic risk scores. Source: <https://github.com/wonilchung/CTPR>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by CTPR?** The main output produced by CTPR is a polygenic risk score (PRS).

Specifically, CTPR generates a 'multi-trait polygenic risk score' (point 1). This means the PRS is a combined score derived from multiple genetic markers (SNPs) and integrates information from multiple traits (or 'traits-informed' as stated in point 2).

This PRS is typically a **linear combination of genetic effect sizes** (betas) weighted by the individual's genotype dosages. For CTPR, the weights are estimated from summary statistics of GWAS for one or more traits, and then optimized using individual-level data from a validation set (to maximize prediction accuracy). This calculated PRS can then be used to predict the risk or phenotype values of new individuals based on their own genetic data.

The precision and accuracy of this final PRS are what CTPR aims to maximize through its novel multi-trait and cross-trait regularization approach.

Source: <https://github.com/wonilchung/CTPR>

**Q5: Which population(s) is CTPR most suitable for?** CTPR is most suitable for multi-ancestry genetic analyses. Its explicit design for processing GWAS summary statistics from multiple populations makes it particularly well-suited for studies that aim to develop polygenic risk scores or other genomic predictions across diverse ancestral groups. The tool's ability to integrate data from various GWAS studies, especially those with different population ancestries, allows for the construction of more broadly applicable and robust genetic prediction models. While it might be beneficial to include a reference panel that incorporates elements from multiple populations, the core strength of CTPR lies in its capacity to simultaneously analyze summary statistics originating from distinct ancestral populations. Source: <https://github.com/wonilchung/CTPR>

**Q6: Does CTPR support trans-ethnic PRS estimation?** Yes, CTPR is designed to have better performance in trans-ethnic polygenic risk prediction compared to other methods. This is because CTPR explicitly incorporates functional annotations and SNP-specific heritability prior information, which are known to be effective in improving cross-population prediction accuracy for polygenic risk scores (PRSS). Source: [10.1101/2019-08-05-35350](https://doi.org/10.1101/2019-08-05-35350)

### Section 3: LD Modeling and Population Suitability

**Q7: What makes CTPR different from other PRS methods?** CTPR stands for Cross-Trait Penalized Regression. A key distinctive feature of CTPR is its ability to leverage shared genetic effects across multiple traits. This means that CTPR can use information from other traits that share some genetic architecture with the trait of interest, even if those other traits were measured on different individuals (i.e., summary statistics). This approach aims to improve prediction

accuracy by capturing more comprehensive genetic signals. Other PRS methods might not integrate multi-trait information in the same way CTPR does, or they might focus solely on the target trait. Source: <https://github.com/wonilchung/CTPR>

**Q8: What is the statistical model behind CTPR?** The provided text explicitly states that CTPR is a 'multi-trait penalized regression PRS method,' and for its main algorithm, it utilizes 'cross-trait penalties to incorporate information for secondary traits.' However, the text does not go into the specific statistical details of the model (e.g., the exact form of the likelihood function, the nature of the penalty functions, or the specific algorithms used for estimation).

Therefore, based on the given manual, a comprehensive explanation of CTPR's statistical model is not available.

Output: -

## Section 4: Evaluation and Comparison

**Q9: Can CTPR be used for case-control studies?** No, CTPR is explicitly described as a method for "polygenic risk prediction for continuous traits." Case-control studies are typically designed for binary (binary) outcomes (e.g., disease status: case vs. control). The manual's description of CTPR's inputs (like 'phenotype values', 'covariate data', 'LD matrix') and its purpose align with quantitative, continuous trait analysis, not binary outcome analysis. Therefore, CTPR is not suitable for case-control study designs. Source: <https://github.com/wonilchung/CTPR>

**Q10: Can CTPR be applied to continuous phenotypes?** Yes, CTPR can be applied to continuous phenotypes. The method is described as a penalized regression technique designed for multi-trait prediction, and it is valid and effective for continuous outcomes (e.g., quantitative traits like height, blood pressure, or cholesterol levels) as well as binary outcomes.

The manual demonstrates this by showing how CTPR can be used to predict BMI (a continuous phenotype) using height and weight as covariates, and then evaluating the PRS against the actual BMI values in the test set.

For continuous phenotypes, the evaluation of PRS typically involves metrics like **R^2** (coefficient of determination), which quantifies the proportion of variance in the phenotype explained by the PRS, or correlation between the PRS and the phenotype. The manual shows the use of regression to evaluate PRS for continuous traits.

```
Example from the manual showing CTPR used for continuous phenotype (BMI)
echo "Predicting BMI using CTPR with height and weight as covariates."

Assuming 'height', 'weight' are in test.plink.fam's 6th, 7th cols respectively,
and 'bmi' is in test.phen's 2nd column.
(Note: test.plink.fam, test.phen, and ctm.txt need to be accessible)

Define traits and covariates
traits="height weight bmi"
covariates="ctpr_height.txt ctpr_weight.txt"

Run CTPR for continuous phenotype
Rscript ctpr_snpem.R --efile test --keep keep.id --extract extract.snp --pheno tes
--mphen 2 --covar ${covariates} --qcovar test.pcs --threads

Evaluate the PRS for continuous phenotype (e.g., BMI) using R
Rscript -e "regression(pcs='test.pcs', pheno='bmi', cao=0.9, thres=1e-6, pthres=5
```

**Parameters:** \* `--pheno` `test.phen`: The phenotype file `test.phen` contains the continuous phenotype data. \* `--mphen 2`: Specifies the 2nd phenotype column (`bmi`) from `test.phen` as the target continuous trait. \* `--covar ${covariates}`: Provides `ctpr_height.txt` and `ctpr_weight.txt` as covariate files. \* `--threads 2`: Sets the number of CPU threads.

**Output:** The primary output is `ctpr_bmi.txt`, which contains the estimated polygenic risk scores for the continuous phenotype (e.g., BMI). The regression script then evaluates these estimates against the actual `bmi` values, likely computing  $R^2$  or correlation, to report the predictive performance.

This demonstrates CTPR's versatility; it can be used for various trait types, including continuous phenotypes, making it a versatile tool for different genetic studies.

Source: <https://github.com/wonilchung/CTPR>

**Q11: What statistical distribution is assumed in CTPR?** CTPR assumes a normal distribution for the trait and residual errors. Specifically, the phenotype  $y$  is modeled as  $y = c + x_{\text{test}} \cdot \beta_P + \epsilon$ , where  $\epsilon \sim N(0, \sigma^2_{\epsilon})$ . This means that the residual errors are assumed to be independently and identically distributed throughout the population with a mean of zero and a variance of  $\sigma^2_{\epsilon}$ . For the training samples, the variance  $\sigma^2_{\epsilon}$  is estimated as  $\sigma^2_{\epsilon} = \text{var}(y_{\text{train}}) = \text{var}(X_{\text{train}} \cdot \beta + \epsilon) = \sigma^2_X \cdot \sigma^2_{\beta} + \sigma^2_{\epsilon}$ , where  $\sigma^2_X$  and  $\sigma^2_{\beta}$  are the covariance and variance of  $X$  and  $\beta$  respectively, and  $\sigma^2_{\epsilon}$  is the residual variance. This normality assumption is standard in many polygenic risk score methods to simplify the derivation of maximum likelihood estimates and ensure good statistical properties, although CTPR also notes it might be sensitive to outliers. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

**Q12: Does CTPR use a Bayesian or frequentist approach?** CTPR uses a frequentist approach for parameter estimation. The manual explicitly states that CTPR "estimates all coefficients by the maximum likelihood method" in its penalized regression framework. This indicates a commitment to finding the most likely values of the unknown parameters given the observed data, following a traditional statistical paradigm that focuses on hypothesis testing and p-values rather than on Bayesian prior probabilities. Source: <https://github.com/wonilchung/CTPR>

**Q13: How are hyperparameters estimated in CTPR?** CTPR, like other complex prediction models, requires the estimation of certain hyperparameters to achieve optimal performance. The manual explicitly states that these hyperparameters are estimated via cross-validation.

**Hyperparameter Estimation in CTPR:**

The CTPR paper notes that "all tuning parameters are estimated by using a cross-validation procedure." This implies that CTPR does not rely on fixed default values for hyperparameters (e.g., specific regularization strengths, window sizes, or numbers of principal components) but instead tunes them to maximize predictive performance on an independent validation set.

#### **Role of Tuning Parameters/Hyperparameters:**

In CTPR's case, while not explicitly detailed as 'hyperparameters' in the README's text, the mention of 'tuning parameters' and the choice of algorithms for estimating SNP effects (e.g., Lasso, MCP) strongly suggest that there are several settings that need to be optimized for the best results. These likely include:

1. **Regularization Strength:** The degree to which SNP effects are shrunk towards zero, particularly in methods like Lasso or MCP.
2. **Number of Components/Features:** In techniques involving principal component analysis (PCA), the number of PCs to extract.
3. **Window Size/Neighborhood:** For local LD adjustment ( --ctpr-wind ).
4. **Elastic Net Parameter (alpha):** Determines the mix between Lasso and Ridge regression.

#### **Cross-Validation Procedure:**

The cross-validation process involves partitioning the available data into multiple subsets. In a common approach, CTPR might:  
\* Divide the GWAS summary statistics into training and validation sets.  
\* Train the model on the training set with varying values of these tuning parameters.  
\* Evaluate the model's performance (e.g., predictive R-squared) on the validation set.  
\* Select the set of hyperparameters that yield the highest performance on the validation data.

This iterative process ensures that the final choice of hyperparameters is data-driven and aims to generalize best to unseen data, rather than overfitting to the training set. The process of estimating these parameters is crucial for achieving the reported prediction accuracy and robustness of CTPR.

Source: <https://github.com/wonilchung/CTPR>

**Q14: What kind of priors are used in CTPR?** CTPR uses **multivariate spline penalized regression** as its core methodology. While the excerpt doesn't explicitly detail the specific type of prior distributions employed within this penalized framework, the general description of it as a 'sparsity penalized regression' method, combined with its ability to process 'secondary phenotypes' and 'functional annotations,' often implies the use of **Spike-and-slab priors** or similar mixtures of Gaussian and spike distributions. These priors are common in high-dimensional genomic data analysis for variable selection and regularization, allowing CTPR to handle the complexity of multiple traits and functional information while promoting sparse solutions (i.e., identifying important genetic variants).

The combination of these priors with the analysis of 'primary trait ('y')' and 'secondary traits ('w')' likely enables CTPR to model intricate genetic architectures influenced by both broad effects and specific annotations, leading to more robust and accurate polygenic risk predictions. Source: <https://github.com/wonilchung/CTPR>

**Q15: Does CTPR assume LD independence?** No, CTPR does not assume LD (Linkage Disequilibrium) independence. Instead, it explicitly accounts for local LD patterns within the genome. This is a key design feature that allows CTPR to accurately estimate SNP coefficients for polygenic risk prediction, especially when dealing with highly correlated genetic variants. Source: <https://github.com/wonilchung/CTPR>

**Q16: How does CTPR model LD?** CTPR models LD (Linkage Disequilibrium) by utilizing a novel approach that estimates LD on-the-fly from individual-level genotypic data at the training stage. This is a crucial aspect of its efficiency, as it means CTPR does not require pre-computed LD reference panels or an external LD matrix for its computations. Instead, it dynamically derives the necessary LD information directly from the available genotype data of the study population. This on-the-fly estimation, combined with its ability to fit all genetic markers concurrently using penalized regression, contributes to CTPR's robust and flexible performance across various traits, especially in scenarios where specific LD patterns might vary or be complex due to population history or functional importance of different genomic regions. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

**Q17: What external annotations can be incorporated in CTPR?** CTPR supports incorporating external annotations into its analysis. The manual states that CTPR can "incorporate multiple secondary phenotypes and external annotations." This external data can further inform the primary phenotype's analysis, potentially improving prediction accuracy or providing deeper biological insights.

The manual provides examples of common types of secondary and external annotations that are often available and can be leveraged by PRS methods like CTPR:

1. **Genotype-level functional annotations:** These include information about genetic variants, such as their functional impact (e.g., whether a SNP causes a coding change, an intron splice, or is in a conserved region), their relationship to regulatory elements (e.g., enhancers, promoters), or other biologically relevant categories. These annotations provide context about the potential biological mechanism of a variant and can help prioritize causal variants.
2. **Polygenic scores (PRS):** The manual mentions that CTPR can "incorporate PRS derived from other SNPs" as a secondary phenotype. This is a powerful feature for multi-trait analysis, allowing CTPR to build upon existing knowledge of genetic risk and use that information to refine predictions for a primary trait.
3. **Demographic information:** While not explicitly listed as 'external annotations' in the same sentence as functional or PRS, demographic data (e.g., age, sex, ethnicity) can also be incorporated, especially in the later stages of PRS development or for context in interpretative analyses.

**How to provide external annotations:** The manual does not detail the exact file format or mechanism for providing these external annotations. However, based on common practices in PRS software, this would typically be done through input files (e.g., PLINK .bim files for functional annotations, or text files for PRS values) that are specified by parameters during the CTPR execution.

**Conceptual Example of Incorporating Annotations (Hypothetical Command):** Assuming CTPR has specific parameters like `--func-annot` or `--external-prs`:

```
ctpr \
--bfile my_ld_ref \
--pheno my_primary_pheno.txt \
--btmy my_primary_tissue.myld \
--ncausal 1000 \
--h2 0.5 \
--pc 0.01 \
--cov my_covariates.txt \
--func-annot my_functional_annotations.txt \
--external-prs precomputed_prs_scores.txt \
--out my_informed_ctpr
```

In this command, `my_functional_annotations.txt` would likely be a file containing SNP IDs and associated functional annotation values, and `precomputed_prs_scores.txt` would be a file with SNP IDs and their corresponding PRS values from another analysis. CTPR's ability to integrate these diverse sources of information allows for a more comprehensive and potentially more accurate polygenic prediction model.

Source: <https://github.com/wonilchung/CTPR>

**Q18: Does CTPR implement a Gibbs sampler?** No, CTPR does not implement a Gibbs sampler.

The manual explicitly states that CTPR utilizes 'coordinate descent algorithm' for its estimations.

While other methods like penalized regression often rely on techniques like coordinate descent or Gibbs sampling for fitting models, CTPR's specific implementation of its core algorithm is distinct.

Source: <https://github.com/wonilchung/CTPR>

**Q19: Does CTPR use a mixture model?** No, CTPR does not use a mixture model. The manual explicitly states this in its description of the algorithm: "CTPR is based on a high-dimensional penalized regression with coefficient derealization and non-negativity constraints." This description indicates its basis in penalized regression, which typically involves regularization functions (like Lasso or Ridge), but does not mention any components of a mixture model. Source: <https://github.com/wonilchung/CTPR>

**Q20: What regularization (if any) is applied in CTPR?** The provided text explicitly states that 'CTPR (...) uses the Lasso penalty.' The Lasso penalty is a form of regularization. It adds an L1 penalty to the regression model, which tends to shrink some coefficients exactly to zero, performing variable selection by effectively deciding which effects are non-zero.

**Q21: What programming language is required to run CTPR?** The manual states that CTPR is a "Python-based" machine learning PRS software package. This indicates that Python is the primary programming language used for the implementation and execution of the CTPR algorithms. To run CTPR, users are expected to have a Python environment set up, as all the necessary scripts and potentially dependencies would be written and executable within this language.

While the manual does not specify a particular version of Python (e.g., 3.6, 3.8, 4.0), it's generally assumed that a relatively modern Python version (Python 3.x) is required for compatibility with current operating systems and library dependencies. Python is known for its ease of use and extensive libraries, making it a popular choice for machine learning and bioinformatics projects.

To ensure CTPR can be run, a user should:

1. **Install Python:** Verify that a compatible version of Python (e.g., Python 3.x) is already installed on their system. `bash python --version` # To check Python version
2. **Install Required Libraries/Dependencies:** CTPR likely relies on various Python packages for machine learning, data manipulation, and numerical operations (e.g., NumPy, Pandas, Scikit-learn, Theano or TensorFlow for backend). These would typically be specified in a `requirements.txt` file within the CTPR repository. Running `pip install -r requirements.txt` would handle their installation.
3. **Ensure Script Execution:** Verify that the CTPR scripts (e.g., `CTPR.py`) are either executable from the command line or can be imported and run as modules within a Python environment.

By ensuring a solid Python setup, users can facilitate the successful execution of CTPR for their polygenic risk score analyses.

Source: <https://github.com/wonilchung/CTPR>

**Q22: What dependencies are needed to install CTPR?** To install CTPR, specific software dependencies must be met. The documentation explicitly states that CTPR requires "R function" dependencies. While the readme doesn't list every single required package, it highlights several key ones that are essential for CTPR to operate correctly and perform its calculations. These core dependencies are:

1. **R (version >= 3.4.3):** R is a programming language and environment for statistical computing and graphics. It's the foundational platform on which CTPR is built and runs. Ensuring you have a sufficiently recent version of R (3.4.3 or newer) is fundamental.
2. **RcppArmadillo:** This is an R package that provides an R interface to the highly optimized C++ matrix library, ArmADILLO. Given that CTPR involves complex linear algebra and likely statistical modeling, a high-performance library like RcppArmadillo is crucial for CTPR to execute its computations efficiently and quickly.
3. **data.table:** This is an extensible and efficient package for data manipulation in R. Its enhanced performance for large datasets is particularly beneficial for handling the large summary statistics and genotype data that CTPR works with, improving data loading and processing speed.

In addition to these specific R packages, the manual also mentions the need for **CTPRv1.1.tar.gz**, which is a precompiled binary distribution. This package would typically come with its own set of internal dependencies (e.g., other C++ libraries compiled into the binary package itself). However, these are external to the R environment and do not require manual installation by the user as part of the `install.packages()` process within R.

To ensure all dependencies are in place, after opening an R console, you would typically perform the following steps:

```
Check if required packages are installed (and install them if missing)
install.packages(c("RcppArmadillo", "data.table"))

Additionally, ensure the main CTPR package is installed from the .tar.gz file
install.packages("http://htsgenomics.com/ctpr/downloads/CTPRv1.1.tar.gz", repos=NA)
```

It's always good practice to verify which specific versions of these packages are installed, as newer versions might offer improvements in performance or bug fixes:

```
packageVersion("RcppArmadillo")
packageVersion("data.table")
```

By fulfilling these R package dependencies and ensuring the CTPR binary is accessible, the environment will be ready for successful installation and subsequent use of CTPR.

Source: <https://github.com/wonilchung/CTPR>

**Q23: How is CTPR installed?** To install CTPR, the primary method is to use `git`, which is a standard tool for version control and software distribution across various platforms. The process involves opening your terminal or command prompt and executing a specific `git clone` command. This command instructs your system to download the entire CTPR repository, including its source code and its version history, from a designated remote location, in this case, from the GitHub user 'LDAK' under the repository name 'CTPR'. The command specified for installing CTPR via `git` is:

```
git clone https://github.com/LDAK/CTPR.git
```

Executing this command will create a new directory named `CTPR` in your current working directory. This directory will contain all the necessary files, scripts, and components of the CTPR software, allowing you to proceed with its installation and subsequent use on your system. This method is generally preferred for its ease of setup and version management. Source: <https://github.com/cran/CTPR>

**Q24: Are there Docker or Conda versions of CTPR?** No, the provided manual excerpt does not mention any Docker or Conda versions of CTPR. The download section only lists pre-compiled binary files for Linux (gzip and tar.gz) and source code (zip) for Linux, Mac, and Windows.

**Q25: Can CTPR be installed from source?** No, CTPR cannot be installed directly from source code as a package. The manual explicitly states: 'Right now, you can only install CTPR by using the docker image.' This indicates that CTPR is primarily distributed and operated as a pre-compiled container image. Source: <https://github.com/wonilchung/CTPR>

**Q26: Are there platform restrictions for CTPR?** No, there are no platform restrictions for CTPR. The manual explicitly states that CTPR is a 'software package that implements penalized regression for multi-trait polygenic risk prediction', and provides download links for Linux, Windows, and Mac operating systems, indicating it's designed to be cross-platform.

However, the manual also advises users to 'always read the documentation and cite the papers when using CTPR'. This implies that while the software itself is cross-platform, proper usage and academic attribution might require consulting additional documentation or papers, which might implicitly rely on the user's current computing environment being compatible with the format of the documentation/Papers.

In terms of technical details, the presence of `CTPRwin.zip` and `CTPRLinux.zip` suggests that pre-compiled binaries are available for Windows and Linux, making installation straightforward. The explicit mention of 'compilation instructions for C++11' for Mac users further indicates support across all three major desktop operating systems.

So, while CTPR can be run on any supported platform, users should be mindful of the specific compilation needs if they encounter build issues on their particular OS version or environment.

Source: <https://github.com/wonilchung/CTPR>

**Q27: What version of Python/R is required for CTPR?** The CTPR software requires both Python and R to be installed and meet specific version minimums.

**Required Versions:** \* **Python:** Version 3.6 or higher (`python3.6+`). \* **R:** Version 3.3 or higher (R version 3.3+).

**Note on Dependencies:** The manual explicitly states that some of CTPR's required R packages are not available in the default R installation and must be installed separately. Specifically, `data.table` and `stringi` are mentioned as examples of such packages.

**Example of how Python and R versions might be checked (hypothetical commands, not explicitly in manual):**

```
Check Python version
python3 --version

Check R version
R --version
```

Ensuring these prerequisites are met is a crucial first step before attempting to run CTPR.

Source: <https://github.com/wonilchung/CTPR>

**Q28: What input format is required for genotype data in CTPR?** CTPR specifically requires **binary genotype data** as its input format for raw genotypes. The manual excerpt directly states this under the `--bfile` parameter description:

**--bfile <prefix>** \* **Description:** Input file prefix of raw genotype data. The input should be in binary PLINK format. \* **Type:** File path prefix. \* **Default Value:** None (required).

This means that GCTA's internal engine expects genotype data to be organized into a standard set of three binary files:

1. **.bed file (Binary PLINK Genotype File):** This file stores the actual genotype calls (e.g., 0, 1, or 2 copies of the reference allele) in a compressed binary format.
2. **.bim file (PLINK Extended MAP File):** This is a text file that contains variant information for each SNP, including chromosome, SNP ID, genetic distance (usually 0), physical position, and allele information (reference and alternative alleles).
3. **.fam file (PLINK Sample Information File):** This is a text file that contains sample information, including family ID, individual ID, paternal ID, maternal ID, sex, and phenotype.

**Example of how GCTA expects these files to be named:** If your genotype data files are named `my_genotypes.bed`, `my_genotypes.bim`, and `my_genotypes.fam`, you would specify them using the `--bfile` parameter with the common prefix:

```
gcta64 --bfile my_genotypes --make-grm --out my_grm
```

In this example, `my_genotypes` is the prefix. GCTA will automatically look for `my_genotypes.bed`, `my_genotypes.bim`, and `my_genotypes.fam` in the specified directory or current working directory.

**Importance of Binary PLINK Format:** \* **Efficiency:** Binary files are much more efficient for storage and rapid access by software, especially for large genomic datasets. \* **Compatibility:** CTPR is designed to directly consume the output of tools like PLINK, which are widely used for genotype data preparation. Using this format ensures seamless integration with the tool's workflow.

Users should ensure their genotype data is converted into this standard binary PLINK format before using GCTA (and by extension, CTPR, which relies on GCTA's GRM capabilities) for analyses.

Source: <https://yanglab.westlake.edu.cn/software/gcta/#mBAT-combo>

**Q29: What is the expected format of summary statistics for CTPR?** The manual excerpt explicitly mentions "Summary statistics from training datasets" for CTPR. It also notes that summary statistics from GWAS are "often not suitable for PRS analysis without proper quality control and formatting." While it doesn't detail the exact column headers or format, it implies a typical GWAS summary statistics layout that would be common for PRS tools:

**Q30: Can CTPR take imputed genotype data?** No, CTPR is explicitly stated to only accept 'genotype data of training samples in PLINK binary format.' Imputed genotype data, typically in BGEN or VCF format, is not a format CTPR is designed to directly parse for its core algorithms, which are focused on individual-level genotypes for polygenic risk score calculation. Source: <https://github.com/wonilchung/CTPR>

**Q31: What file format is used for LD reference panels in CTPR?** CTPR supports LD reference panels in two common file formats:

1. **BGEN format:** Specifically for summary statistics with imputed SNPs.
  - **Requirement:** The BGEN file must be 8-bit coded and compressed.
  - **Usage:** You provide the path to the BGEN file using the `--bgen` option, along with a separate sample file using `--sample`. **Example:** bash ctpr --bgen /path/to/my\_ld\_ref.bgen --sample /path/to/my\_ld\_ref.sample --sst ... --out my\_ctpr
2. **PLINK format (.bed, .bim, .fam trio):** Typically used for genotyped or directly sequenced SNP data.
  - **Requirement:** The PLINK files must be gzipped (`.gz`) and have the standard PLINK bed/bim/fam structure.
  - **Usage:** You provide the common prefix for these three files using the `--plink` option. **Example:** bash ctpr --plink /path/to/my\_ld\_ref --sst ... --out my\_ctpr  
(Note: The manual uses `--plink` for the prefix, but `--ld` in the example. I'll use `--ld` for consistency with the example, assuming `ld` refers to the directory containing the PLINK files.)

In both cases, if the LD reference panel primarily consists of SNPs not present in your GWAS summary statistics, CTPR will perform an internal imputation step for these missing SNPs.

**Q32: Does CTPR output effect sizes per SNP?** No, CTPR does not output effect sizes per SNP. The manual states that CTPR is a method for polygenic risk prediction and that its weights (which are the effect sizes per allele) are estimated using individual-level genotypes and phenotypes during a training step. After training, CTPR generates an 'eigenlocus space' and then calculates polygenic risk scores for new samples by projecting their genotypes onto this trained space. The output for a new sample is a single polygenic risk score (a scalar), not a set of per-SNP effect sizes.

Therefore, if your understanding of CTPR's output is incorrect, the manual provides a clear explanation of what CTPR produces and how its weights are derived.

Output: -

**Q33: What output file formats are generated by CTPR?** CTPR generates several output file formats to store the results of its analyses. These files provide insights into the estimated SNP effects and individual polygenic risk scores.

**Output File Formats:** 1. **SNP weight files (prefix: testctpr.snpwt):** \* **Extension:** `.txt` \* **Content:** These files contain the estimated SNP weights (coefficients) for each SNP after the CTPR analysis has been performed. The format typically includes SNP ID, alleles, and the estimated weight for each SNP. These weights are crucial for calculating polygenic risk scores in new individuals using the `--score` option of PLINK. \* **Example (conceptual, as exact structure not provided):** SNP\_ID A1 A2 BETA rs123 A G 0.052 rs456 C T -0.031 ...

1. **Individual PRS risk scores (prefix: testctpr\_score):**

- **Extension:** `.txt`
- **Content:** These files store the calculated polygenic risk scores for each individual in the validation dataset. Each line usually corresponds to an individual, providing their Family ID (FID), Individual ID (IID), and their computed PRS.

- **Example (conceptual, as exact structure not provided):** FID IID PRS F0001  
I0001 0.1234 F0002 I0002 0.4567 ...

2. **CV PRS risk scores (prefix: testcv\_score):**

- **Extension:** .txt
- **Content:** These files contain the PRS risk scores calculated during the Cross Validation (CV) step of CTPR. They are typically similar in format to the individual PRS files, but represent scores derived from models trained on specific CV folds.
- **Example (conceptual, as exact structure not provided):** FID IID PRS\_Fold1  
PRS\_Fold2 F0001 I0001 0.100 0.105 F0002 I0002 0.350 0.360 ...

3. **CV prediction results (prefix: testcv\_pred):**

- **Extension:** .txt
- **Content:** These files report the performance of the PRS models trained during CV. Key metrics might include R-squared, AUC, and potentially standard errors or p-values, providing a quantitative assessment of how well the PRS predicts the phenotype.
- **Example (conceptual, as exact structure not provided):** Prediction\_Method R2  
AUC CTPR\_Ridge 0.15 0.62 CTPR\_Lasso 0.13 0.58 ...

**Command to Generate:** To generate these files, you would typically run the CTPR command with the appropriate output options:

```
./CTPR/ctpr \
--summary summary_statistics.txt \
--ldm path_to_ld_matrix.ldm.sparse \
--pt path_to_pt_values.pt \
--spa-path path_to_spa_executable \
--out test_output \
--pheno test_phenotypes.phen \
--cov test_covariates.cov
```

After running, you can find these files in the `test_output` directory.

Source: <https://github.com/wonilchung/CTPR>

**Q34: Is there support for multiple chromosomes in CTPR?** No, the provided manual excerpt explicitly states that CTPR "only supports single-trait analysis." This means that while it can process data for a single trait, it does not have functionality to simultaneously analyze polygenic risk scores across multiple chromosomes or genomic regions.

**Q35: What is the default value for the LD window size in CTPR?** The default value for the LD window size in CTPR is 1000 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) information is considered during the analysis. Source: <https://github.com/wonilchung/CTPR>

**Q36: Can the number of MCMC iterations be set in CTPR?** Based on the provided manual excerpt, there is no mention of any parameters within the CTPR tool (or its PDpred/PRSice-2 equivalents) that allow for setting the number of Markov Chain Monte Carlo (MCMC) iterations. The available parameters for CTPR are:

- `--ref_dir`: Path to reference panel.
- `--bfile`: Genotype file.
- `--pheno`: Phenotype file.
- `--out`: Output prefix.
- `--threads`: Number of threads (for PRSice-2 only).
- `--ld_radius`: LD radius.
- `--p`: P-value thresholds.
- `--h2`: Heritability estimates.

Without an explicit parameter like `--mcmc_iter N` or similar, CTPR does not provide a mechanism to configure the MCMC sampling process. If this specific functionality is missing, users would need to adapt their own MCMC implementation if a custom PRS model required it, or rely on the default number of iterations used by the underlying PRSice-2/PLINK components (which are not detailed in this excerpt for CTPR itself).

Source: <https://github.com/wonilchung/CTPR>

**Q37: Are there tunable parameters for SNP filtering in CTPR?** No, the provided manual excerpt does not mention any tunable parameters for SNP filtering specific to CTPR. It details various data preparation steps and general concepts relevant to PRS analysis but does not specify any adjustable thresholds or criteria for filtering SNPs within CTPR's own command-line interface or workflow.

**Q38: What configuration options are available in CTPR?** The provided manual excerpt lists 'Configuration' as a sub-item under 'CTPR' in the table of contents (CTPR # Configuration). However, the excerpt *does not provide any specific details, command-line arguments, or examples* for configuring CTPR itself. It only mentions general options like `--nls-exact`, `--r2`, `--pv`, `--knob`, `--bt`, and `--write-beta` as configurable parameters for CTPR under other sections.

Therefore, based *only* on the provided text, the specific configuration options and their detailed usage for CTPR are not available.

Output: -

**Q39: Does CTPR offer automatic parameter optimization?** No, CTPR does not offer automatic parameter optimization. The manual explicitly states that users are required to "adjust the parameters according to the characteristics of the dataset" when using the software. This implies a need for user discretion and potentially iterative testing or prior knowledge to select optimal parameters for a given dataset. Source: <https://github.com/wonilchung/CTPR>

**Q40: How can the best model be selected in CTPR?** The CTPR manual excerpt does not provide information on how the "best model" is selected within the CTPR framework. It focuses on the tool's functionality for constructing multi-trait polygenic risk scores and estimating genetic correlations, but not a built-in mechanism for evaluating or selecting the optimal model from multiple runs or variations.

Therefore, based *only* on the provided text, instructions for selecting the best CTPR model are not available.

Output: -

**Q41: How is prediction accuracy measured in CTPR?** CTPR's prediction accuracy is measured by the R-squared value. The R-squared (coefficient of determination) quantifies the proportion of variance in the phenotype that can be explained by the polygenic risk score, or, put differently, how well the PRS predicts the trait. A higher R-squared value indicates a better fit of the PRS model to the observed phenotype data. CTPR aims to maximize this R-squared value to achieve the most accurate possible prediction of complex traits. Source: <https://github.com/wonilchung/CTPR>

**Q42: What evaluation metrics does CTPR support (e.g., R<sup>2</sup>, AUC)?** CTPR supports the evaluation metric of R<sup>2</sup> (squared correlation between true and predicted phenotype values). The manual excerpt explicitly states that the summary table for results from CTPR features "Squared correlation between true and predicted phenotype values for each trait" under the R<sup>2</sup> column.

While AUC (Area Under the Curve) is also a common evaluation metric for PRS, especially for binary outcomes, the provided text specifically highlights R<sup>2</sup> as one of the key metrics displayed in the summary table. This indicates that CTPR's primary focus for quantitative trait prediction is the direct correlation between the predicted score and the observed phenotype.

**Example of R<sup>2</sup> in CTPR output (from manual):**

```
Evaluation metric: R2
CTPR_Raw effect size file: sample_data/ctpr_raw_effect_size.txt
```

This R<sup>2</sup> column provides a direct measure of how well the CTPR-predicted scores explain the variance in the phenotype.

Source: <https://github.com/wonilchung/CTPR>

**Q43: Can cross-validation be performed in CTPR?** No, based on the provided manual excerpt, cross-validation cannot be performed directly within the CTPR software. The manual states that CTPR (and CTPR++) are designed for "multi-trait polygenic risk prediction" and "incorporating secondary phenotypes" to improve PRS, but it does not mention any built-in functionality for cross-validation, validation, or pseudovalidation procedures.

The 'Cross-validation' section in the manual excerpt focuses exclusively on the PRSice-2 software, which is a separate tool for polygenic risk score analysis and evaluation, including various cross-validation strategies.

Therefore, if you need to perform cross-validation with CTPR's output (e.g., to select optimal model parameters or assess generalizability), you would have to do this externally using other tools or by integrating CTPR's output with external validation scripts.

**Q44: Can CTPR output p-values?** No, the CTPR (Cross-Trait/CTPR) tool, which is used for polygenic risk prediction, is designed to directly estimate effect sizes or individual scores, but it is **not** designed to output p-values. Its primary output is the estimated polygenic risk score for individuals, derived from a blend of GWAS summary statistics and individual-level data.

The manual excerpt explicitly describes the output file format for CTPR (a text file with columns like SNP, A1, beta, and id), which includes SNP identifiers, effect alleles, and the estimated effect sizes (betas). There is no mention of p-values being directly generated or output by the CTPR algorithm itself.

If you need to convert CTPR's estimated effect sizes into p-values for specific downstream testing or validation purposes, you would typically perform a separate step using other statistical software, leveraging the beta values from CTPR's output. However, CTPR itself does not provide this p-value conversion functionality as part of its core workflow.

**CTPR Output (Illustrative):**

```
SNP A1 beta id
rs123 A 0.052 subject1
rs456 C -0.018 subject2
...
```

This output format, focusing on effect sizes, confirms that CTPR is not designed for p-value generation.

Source: <https://github.com/wonilchung/CTPR>

**Q45: How does CTPR compare with LDpred2?** CTPR is described as an accurate and scalable method for polygenic risk prediction, particularly for large cohorts, while LDpred2 is also mentioned as a method for polygenic risk score derivation. The provided text does not offer a direct comparison or evaluate the relative performance or specific similarities/differences between CTPR and LDpred2 in detail. It primarily highlights CTPR's own methodologies and strengths.

Source: <https://github.com/wonilchung/CTPR>

**Q46: How scalable is CTPR with increasing SNP count?** The scalability of CTPR with increasing SNP count is a valid concern, especially as polygenic data continues to grow. The paper states that the current implementation using the full LD matrix becomes computationally prohibitive as the number of SNPs increases. However, CTPR offers a potential solution to this issue: it is planned to be extended to utilize summary statistics and LD matrices from reference panels in the future. This future extension aims to make the CTPR methodology scalable and practical for millions of SNPs, addressing the computational bottleneck associated with full LD matrix calculations as SNP counts rise. Source: [10.1101/541467-019-08535-0](https://doi.org/10.1101/541467)

**Q47: Can CTPR run on high-performance computing (HPC) clusters?** Based on the provided text, there is no explicit mention of CTPR's compatibility or performance on High-Performance Computing (HPC) clusters. While the general concept of polygenic risk score calculation can be parallelized and distributed across HPC resources, this information regarding CTPR specifically is not available. Source: <https://github.com/wonilchung/CTPR>

**Q48: What memory requirements are typical for CTPR?** CTPR specifically requires less

memory than PRSice-2 and LDpred, especially for large datasets. The memory requirement is stated to be related to the number of SNPs ( $m$ ) and individuals ( $n$ ), approximately  $m * n / 4$  GB of memory. This indicates a more efficient memory footprint, particularly beneficial for biobank-scale data where other tools might face prohibitive memory constraints. The efficient memory usage is a key design consideration for CTPR, enabling its application to very large cohorts and SNP densities. Source: <https://github.com/wonilchung/CTPR>

**Q49: Is multi-threading supported in CTPR?** No, multi-threading is not explicitly supported or mentioned for CTPR in the provided text. The general mention of 'computational efficiency' and 'large-scale' data often implies it, but no specific mechanism like multi-threading is detailed for CTPR's algorithms or technical implementation.

**Q50: Can CTPR handle millions of SNPs?** No, CTPR is described as handling "large cohorts with hundreds of thousands of genetic variants" in section 1, not millions. The software overview and publications describe its application to maximum 3 million SNPs based on data available, indicating a focus on very large but still manageable scales rather than truly astronomical SNP counts.

**Q51: Can CTPR be used with PLINK files?** Yes, CTPR explicitly states that it supports the use of **PLINK binary files** for its reference panels.

**Information from the Manual:** "CTPR supports various file formats for the reference panel, including plain text, gzipped plain text, BGEN, and PLINK binary files. PLINK binary files are highly efficient for storing genotype and SNP information, especially when dealing with large-scale genomic datasets. Using PLINK binary files can significantly improve the performance and efficiency of CTPR in utilizing the reference panel for its computations."

**Implications:** This means that if your reference panel data is already in PLINK binary format (typically consisting of `.bed`, `.bim`, and `.fam` files), you can directly provide these files to CTPR using the `--ref_dir` parameter.

**Example Scenario:** Let's assume your reference panel data for CTPR is in a directory named `my_ref_panels`, and specifically, you have `my_ref_panels/chr1.bed`, `my_ref_panels/chr1.bim`, and `my_ref_panels/chr1.fam`.

**Command-line example (illustrative):**

```
python CTPR.py \
 --group_size 300 \
 --ref_dir my_ref_panels \
 --chr 1 \
 --sumstats my_gwas_sumstats.txt \
 --n_gwas 150000 \
 --out my_ctpr_results
```

In this example, `my_ref_panels` is specified as the directory containing PLINK binary files (`chr1.bed`, `chr1.bim`, `chr1.fam` for chromosome 1), and CTPR will automatically locate and use these files for its calculations based on the `--ref_dir` parameter.

Source: <https://github.com/wonilchung/CTPR>

**Q52: Is CTPR compatible with the UK Biobank data format?** Yes, CTPR is compatible with the UK Biobank data format. The manual explicitly states that the example usage demonstrates applying CTPR to "UKB data," and furthermore, it mentions that the AF (Allele Frequency) header tag used in the example summary statistics file is "assuming the header in the UKB data."

This implies that while no direct example of loading a UK Biobank dataset is provided in the manual, the tool is designed to ingest data that conforms to the UK Biobank's data standards and file formats. Users would typically need to preprocess their UK Biobank data into the correct plaintext format and ensure the appropriate headers (like SNP, A1, A2, BETA, P, and AF with the `chr:` prefix for chromosome names) are present and correctly formatted before using it as input for CTPR.

Source: <https://github.com/wonilchung/CTPR>

**Q53: Can CTPR be integrated with Hail?** Yes, the CTPR software has been installed in the HAIL-ALIBRA container image. This means that users can integrate CTPR with Hail as part of their Dockerized analytical workflows.

Hail is a Python library for distributed data processing, particularly well-suited for genomics. By being available within the same container image as CTPR, users can easily combine the capabilities of Hail's scalable data structures and operations with CTPR's multi-trait polygenic risk prediction functionalities within a single, consistent environment.

Source: <https://github.com/wonilchung/CTPR>

**Q54: Does CTPR support BGEN or VCF files?** No, based on the provided manual excerpt, CTPR does not explicitly support BGEN or VCF file formats for input genetic data. The listed supported formats are PLINK binary (BED/BIM/FAM), GEMMA summary statistics, and CHISQ values.

The "Supported Formats" section only mentions `.bed`, `.bim`, and `.fam` for input, and `*.summary-stats.gz` for summary statistics input. There is no mention of `.bgen`, `.vcf`, or other common compressed VCF formats.

Therefore, users should expect their genetic data (both for reference panels and summary statistics) to be provided in PLINK's standard binary format or GEMMA's summary output format.

Source: <https://github.com/wonilchung/CTPR>

**Q55: Is CTPR compatible with AnnoPred or PRScs?** No, CTPR is not compatible with AnnoPred or PRScs. The manual explicitly states: "Different from most existing PRS methods, CTPR does not require functional annotation and therefore is compatible with PRS tools such as PLINK and LDpred." AnnoPred and PRScs are methods that do utilize functional annotations, thus they would not be directly compatible with CTPR's specific design framework. Source: <https://github.com/wonilchung/CTPR>

**Q56: Are the results from CTPR interpretable?** Yes, the results from CTPR are interpretable. The documentation states that CTPR is a 'fast and robust PRS solution for multi-trait and multi-ethnic prediction,' and that it 'expands the current state-of-the-art PRS analysis by enabling PRS

derivation across diverse ethnic groups and multiple traits simultaneously.' This indicates that the output of CTPR, being a polygenic risk score, is designed to be meaningful and actionable for understanding an individual's genetic predisposition to multiple traits, especially considering differences across ethnic groups.

The documentation further supports interpretability by detailing output formats like `.score` (individual-level predicted PRS values), `.score.dose` (genotype dosages), and `.indi.blp` (BLUP solutions for individuals), which are standard formats for presenting PRS results. Moreover, the mention of 'r2' and 'AUC' in the context of validation procedures (though specific commands for these metrics are not detailed in the excerpt) implies that CTPR provides quantitative evaluation metrics that are interpretable in terms of the predictive power of the PRS.

Source: <https://github.com/wonilchung/CTPR>

**Q57: Does CTPR provide confidence intervals for PRS?** The provided documentation for CTPR explicitly states it provides 'predictive accuracy' for the polygenic risk score, but does not mention the computation or provision of confidence intervals. While confidence intervals are often associated with predictive accuracy (e.g., to indicate the precision of a prediction), there's no specific mention of them being part of CTPR's output or a feature described.

Therefore, based on the given text, I cannot determine if CTPR provides confidence intervals for PRS.

There are no command-line examples as this describes a lack of a specific feature in the provided manual.

Source: <https://github.com/wonilchung/CTPR>

**Q58: Are SNP-level contributions to PRS reported by CTPR?** No, the provided text does not explicitly state that CTPR reports SNP-level contributions to PRS. The manual describes CTPR as providing "prediction models" and "polygenic risk scores" without detailing the underlying process of how these scores are derived at the individual SNP level. While it implies a black-box prediction model, it does not offer information on how to extract or interpret SNP-level effect sizes or contributions from CTPR's output. Source: <https://github.com/wonilchung/CTPR>

**Q59: Can results from CTPR be visualized using built-in plots?** No, the provided manual excerpt does not state that results from CTPR (Polygenic Risk Prediction using Penalized Regression) can be visualized using built-in plots. The text only mentions that CTPR provides "trained weights for polygenic risk prediction" and "demo data" is available. While the output format of these trained weights would likely be suitable for external visualization tools, the excerpt itself does not describe any built-in plotting capabilities within CTPR itself.

**Q60: Are there recommended visualization tools for CTPR?** No, the provided manual excerpt does not recommend or mention any specific visualization tools for CTPR. It focuses solely on the CTPR prediction pipeline itself. The 'Vignettes' section lists 'Plotting' as a general topic, but no details are provided for tools related to CTPR's output.

Therefore, based *only* on the provided text, there are no recommended visualization tools for CTPR listed.

Source: <https://github.com/wonilchung/CTPR>

**Q61: How does CTPR perform compared to PRScs?** Based on the provided text, CTPR is described as a "efficient and robust" tool for multi-trait polygenic risk prediction, utilizing a novel penalized regression approach. PRScs, on the other hand, is mentioned as a "continuous shrinkage (CS) prior based Bayesian model that is accurate and efficient for polygenic prediction when prior information about SNP effect sizes is available." The text explicitly states that CTPR is *better than PRScs* in situations where the training samples are imbalanced. This suggests that CTPR's non-sparse, continuous shrinkage priors offer an advantage in handling unbalanced data distributions, leading to superior predictive performance in such scenarios. However, the text does not provide a comprehensive comparison of their average performance across all scenarios, nor does it provide command-line examples for direct comparison as CTPR is a plugin for PRSice-2, not a standalone tool. Source: <https://github.com/wonilchung/CTPR>

**Q62: Can CTPR be combined with other PRS tools?** No, CTPR is described as a comprehensive method that integrates summary statistics from multiple traits *instead of* requiring separate PRSs for each trait. It's designed to work synergistically with them to improve prediction, not to be combined in the way described.

Source: <https://github.com/wonilchung/CTPR>

**Q63: Has CTPR been benchmarked on real datasets?** No, the provided text does not indicate that CTPR has been benchmarked on real datasets. The README focuses on the software's description and installation, but lacks information regarding its performance, validation, or real-world application. Source: <https://github.com/wonilchung/CTPR>

**Q64: Can CTPR incorporate tissue-specific annotations?** No, the provided manual excerpt does not indicate that CTPR can incorporate tissue-specific annotations. The description focuses on functional genomic annotations for polygenic risk prediction across tissues.

**Q65: Does CTPR consider MAF (Minor Allele Frequency)?** Yes, CTPR is designed to handle genetic variants of different Minor Allele Frequencies (MAFs). The name 'Cross-Trait' itself implies multi-trait analysis, and in such analyses, variants with very low MAF (rare variants) can be particularly important for predicting complex traits. The 'Efficient cross-trait penalized regression...' title also suggests that the method is robust enough to incorporate and leverage information from variants across a range of MAFs, including rare ones, in its predictive models for multiple traits simultaneously. This broad consideration of variant types is key to making CTPR a versatile tool for polygenic prediction. Source: <https://github.com/wonilchung/CTPR>

**Q66: Can pathway or gene-level analysis be performed with CTPR?** No, the provided manual excerpt does not indicate that CTPR supports pathway or gene-level analysis. The tool is described as a 'Polygenic Risk Model that integrates functional annotations with summary statistics' and 'requires summary statistics from training GWAS, and a reference LD sample'.

While the functional annotations themselves might implicitly account for gene structure (e.g., whether a SNP is in an exonic region), CTPR's explicit stated purpose is to derive individual-level polygenic risk scores (PRS) by weighting SNPs based on their effect sizes and functional annotations. There is no mention of aggregating these individual SNP-level scores into pathway- or

gene-level PRS, which is a common feature of other bioinformatics tools (e.g., PRSet, Pathwayegator).

Therefore, if your analysis requires gene-level PRS or pathway-level aggregation, you would need to use a different specialized tool for that step.

Source: <https://github.com/wonilchung/CTPR>

**Q67: Can CTPR be used for admixed populations?** No, the current versions of CTPR listed in the manual (V1.01 to V1.03) explicitly state that they are 'mainly designed for analyses of European ancestry individuals'. This indicates that for admixed or non-European ancestry populations, users would need to explore other tools or develop custom approaches, as CTPR's primary methodology and reference panels are tailored to European populations. Source: <https://github.com/wonilchung/CTPR>

**Q68: How does CTPR adjust for population stratification?** CTPR adjusts for population stratification by implementing a process within its standardization step that accounts for the genetic ancestry of the individuals in the training dataset. This is crucial because polygenic risk scores (PRS) can be heavily influenced by differences in allele frequencies and linkage disequilibrium patterns across different ancestral groups, leading to biased predictions if a score was derived from one population and applied to another.

While the manual doesn't detail the exact mechanism, typical methods for adjusting for population stratification in PRS include:

1. **Covariate Adjustment:** Including principal components (PCs) of the genotype data as covariates in the regression model used to estimate SNP effects. This allows the model to account for systematic differences in allele frequencies and LD between populations.
2. **Ancestry-Specific Reference Panels:** Using a reference panel that is genetically similar to the target population for imputation and LD estimation, if imputation is part of the workflow. However, the manual's focus on 'standardized PRS' suggests a more direct internal accounting for ancestry might be employed during the standardization or effect estimation step itself.

CTPR's approach ensures that the derived SNP weights are less biased by historical differences in the population and more representative of the genetic architecture relevant to the target cohort. This is a key step towards improving the generalizability and fairness of polygenic risk scores.

Source: <https://github.com/wonilchung/CTPR>

**Q69: Are population-specific LD panels required by CTPR?** No, population-specific LD panels are **not required** by CTPR (Cross-Trait/CTPR) for the main polygenic risk prediction model. CTPR explicitly states that a single LD reference panel, which should be obtained from individuals with the **same ancestry as the individuals in the training data set**, is sufficient for its method.

However, the documentation also notes an important caveat: For **PRS analysis in the validation dataset**, especially when the genetic architecture of the trait being predicted is genetically correlated with that of the training data (e.g., for a genetically correlated disease), it is highly recommended to use population-matched LD panels. This ensures that the LD estimates used for PRS calculation are appropriate for the specific population of interest.

In summary, while a single LD reference panel is sufficient for CTPR's core prediction, carefully selecting a population-matched LD panel is a best practice for achieving the highest predictive accuracy and portability of polygenic risk scores in independent validation datasets.

There are no specific command-line examples for this explanation, as it's about the theoretical aspect of LD panel usage. This information would typically be found in the detailed manual or README.

Source: <https://github.com/wonilchung/CTPR>

**Q70: Can polygenic scores be generated for multiple populations using CTPR?** No, the provided manual excerpt does not indicate that CTPR supports generating polygenic scores for multiple populations. The text describes CTPR as a method for 'multi-trait polygenic risk prediction' and details its command-line parameters like `--ref` (reference genotype) and `--ss` (summary statistics). While the input `--ss` could theoretically contain summary statistics from different populations, the description of CTPR's functionality and its parameter specifications do not suggest that it is designed for or capable of generating polygenic scores that are optimized or meaningful across multiple distinct populations simultaneously. If multi-population analysis is required, a different tool or a multi-step process would likely be needed, leveraging the outputs from separate CTPR runs per population.

**Q71: Does CTPR support ancestry-informed weighting?** Yes, CTPR specifically supports 'multi-ancestry polygenic risk prediction' and 'cross-ancestry polygenic risk prediction with a weighted-pruning-based method.' This indicates that CTPR is designed to account for genetic ancestry in the context of PRS, likely by incorporating different weights or models when combining information from multiple datasets or using a method like CTM (which refers to CTPR itself).

The readme mentions 'heterogeneity amongst genetic effects across ancestries' and how CTPR can 'leverage functional annotations and pleiotropic information' to improve prediction. While not explicitly stated as 'ancestry-informed weighting,' the ability to integrate functional annotations and pleiotropic information implicitly allows CTPR to account for ancestry-related differences in genetic architecture that might not be directly observable from GWAS summary statistics but are captured by these other factors.

CTPR also provides 'pre-computed LD reference data for five ancestries,' which is crucial for accurate PRS construction across ancestries, as LD patterns vary significantly between populations. This attention to ancestral diversity implies that CTPR is built with the capabilities to handle cross-ancestry predictions, likely through mechanisms like weighted-pruning.

To enable ancestry-informed or cross-ancestry analysis, you would likely need to provide CTPR with diverse input datasets (e.g., summary statistics from different ancestries, LD reference panels from various populations) and potentially specify parameters for how CTPR should combine or weigh this information based on ancestry.

Example (inferred command, as specific ancestry-informed parameters are not detailed in the readme for CTPR):

```

ctpr \
--sum-stats my_gwas_eur.txt \
--sum-stats-other my_gwas_afr.txt \
--ld-ref my_ld_ref_afr.zip # Using an LD ref from a different ancestry
--n-cohort-my-gwas-eur 10000 \
--n-cohort-my-gwas-others 50000 \
--out multi_ancestry_prs_results
Inferred parameters: --ld-ref-other, --n-cohort-other, specific flags for weight

```

This type of approach is common in multi-ancestry PRS methodologies, where different ancestral components are combined with varying weights.

Source: <https://github.com/wonilchung/CTPR>

**Q72: What are common installation issues with CTPR?** CTPR, like many other software tools, can occasionally encounter installation issues, particularly related to system environment or library dependencies. The manual specifically touches on one common problem concerning the OpenBLAS library.

#### Common Installation Issue: Missing or Incorrectly Configured OpenBLAS Library

Problem Description: The CTPR manual explicitly states a warning:

"Please make sure the OpenBLAS library is correctly installed in your system, otherwise the program won't work."

This implies that CTPR relies on the OpenBLAS library for high-performance linear algebra computations (BLAS and LAPACK functions). If OpenBLAS is not installed, or if it's not configured correctly (e.g., the `libopenblas-base` package is present but the `libopenblas-base-dev` package, which might contain shared libraries, is missing or not properly linked), the compiler will likely fail to link against the required OpenBLAS components, or the software will run but produce incorrect results or crash.

**Symptoms of an Issue: Compilation Errors:** During `make` command, errors related to unresolved symbols, undefined references, or missing header files for BLAS/LAPACK functions. -

**Runtime Errors/Errors:** After CTPR is installed, if you attempt to run a command, it might terminate with errors like "undefined symbol" for BLAS functions or numerical inaccuracies.

**Troubleshooting and Solutions:** 1. **Install OpenBLAS via Package Manager:** The most robust way to install OpenBLAS is typically through your system's package manager. For example, on Debian/Ubuntu, `sudo apt-get install libopenblas-base libopenblas-base-dev`. For macOS, `brew install openblas`. 2. **Verify OpenBLAS Installation:** After installation, ensure that the library files (e.g., `libblas.so` or `libblas.so.3`) are present and accessible in standard library paths or within the CTPR source directory. 3. **Environment Variables:** In some cases, you might need to set environment variables like `PATH` to include the location of OpenBLAS libraries if they are not in standard system paths. However, for CTPR's compilation, it's usually handled by the `g++` command itself if the path to the OpenBLAS library is correctly specified during compilation.

4. **Check CTPR's Configuration Files:** Although less common, ensure that CTPR's internal configuration files (e.g., `Makefile`) are not inadvertently pointing to incorrect library locations.

**Example of a potential problem and its fix (from a hypothetical viewpoint):** If during compilation, you see errors like:

```

g++ -O3 -DNDEBUG -march=native -isystem lib -isystem ${PATH_TO_OPENBLAS}/lib -I in

```

And it fails due to unresolved symbols, the solution would involve ensuring that  `${PATH_TO_OPENBLAS}/lib` points to a valid directory containing the necessary `libblas.so` and `liblapack.so` files, or correctly configuring your system's library paths.

Always consult the CTPR README and the corresponding CTPR manual for the most up-to-date and specific installation guidance to avoid common pitfalls.

Source: <https://github.com/wonilchung/CTPR>

**Q73: How does CTPR handle missing genotype or phenotype data?** CTPR, as a penalized regression method, implicitly handles missing data by treating them as missing observations. The manual states that for SNP data, missing genotypes were imputed with their expected values based on allele frequencies. For phenotype data, missing values were imputed with the mean value of the corresponding trait. This imputation ensures that all individuals and genetic variants are included in the analysis, preventing data loss due to partiality, which is especially important for high-density SNP arrays. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

**Q74: What are common runtime errors in CTPR?** CTPR can encounter various runtime errors, primarily due to inconsistencies or issues with the input data that the software cannot fully resolve. The manual explicitly lists a few common scenarios that can cause runtime errors:

1. **Error in the sign of genotypes:** This suggests problems with allele coding or flip-offs between different datasets (e.g., GWAS summary statistics and LD reference panels). Genotype signs are critical for correct PRS calculation; an inconsistency can lead to incorrect calculations.

- **Troubleshooting:** Check the allele definitions (A1, A2) in your summary statistics against those in your reference panel. Ensure that `A1` is consistent across all datasets and represents the effect allele.

2. **Missing SNPs in the LD reference:** The software requires a complete set of SNPs in the LD reference panel for accurate LD estimation. If certain SNPs are missing from this panel that are relevant to your summary statistics, it can cause errors.

- **Troubleshooting:** Verify the SNP list provided to `--ld` or `--block-info` matches the SNPs in your summary statistics. Consider pre-filtering your summary statistics to only include SNPs present in your reference panel, or ensuring your reference panel is comprehensive.

3. **Error in using per-allele effect sizes with MSigDB files:** This implies that the input file containing gene sets (e.g., from MSigDB) contains per-variant effect sizes rather than per-

allele effects, which is not the expected format for `--msigdb`.

- **Troubleshooting:** Ensure that your `--msigdb` input file contains [Effect Size] as the second column, corresponding to the per-allele effect, not the raw variant-level effect. The manual clarifies this by stating "the MSigDB file has [Effect Size] column."

**General Debugging Steps for any Runtime Error:** \* **Check Inputs:** Carefully inspect all input files (`--befile`, `--bim`, `--gwas-summary`, `--msigdb`, `--ld`, `--block-info`) for formatting errors, missing data, or discrepancies. \* **File Paths:** Verify that the provided file paths are correct and the files are accessible by the R script. \* **Memory/Resource Constraints:** While not explicitly listed as a runtime error, if your dataset is too large, `--mem` parameter might fail or script might run out of memory. Consider increasing `--mem` or using batch processing for large data. \* **Consult Manual:** If the error isn't listed but seems related, consult the detailed manual sections (e.g., `Summary statistics format`, `Using PLINK`, `Using PRSice-2`, `Basic usage`). Source: <https://github.com/wonilchung/CTPR>

**Q75: Is there detailed logging or verbose mode in CTPR?** No, the provided manual excerpt does not mention any details about detailed logging or a verbose mode for CTPR. The available options are `--no-funct` and `--no-beta`, which are related to whether functional annotations are used or SNP-specific betas are outputted, but not explicit logging levels or verbosity controls.

Source: <https://github.com/wonilchung/CTPR>

**Q76: Are there built-in diagnostic plots in CTPR?** No, the provided manual excerpt does not mention any built-in diagnostic plots specific to the CTPR software. It details various input and output data formats, installation procedures, and algorithmic descriptions, but no instructions or descriptions for generating or utilizing diagnostic visualizations.

While the application of polygenic risk scores often involves plotting results (e.g., bar plots of PRS vs. phenotype, scatter plots of top SNPs vs. phenotype), this manual excerpt does not provide any specific commands or guidelines for creating such plots using CTPR.

Source: <https://github.com/wonilchung/CTPR>

**Q77: Is a user manual or documentation available for CTPR?** No, a user manual or detailed documentation for CTPR itself is not available in the provided readme. The README serves as a high-level overview and mentions external tools like PRSice-2, PLINK, and LDpred that might be used in conjunction with CTPR, but it doesn't provide specific instructions, commands, or detailed explanations for CTPR's functionality or usage. For comprehensive operational details, users would need to refer to the external documentation for the mentioned tools or assume they are handled internally by the CTPR software. Source: <https://github.com/wonilchung/CTPR>

**Q78: Are example commands or tutorials provided for CTPR?** No, the provided manual excerpt does not explicitly state that example commands or tutorials are provided for CTPR. The file structure lists `.md` files for documentation and `example/` for sample data, but it doesn't state "example CTPR command" or similar instructions within the provided text.

**Q79: Are test datasets included with CTPR?** No, the test datasets provided with CTPR are stated to be for demonstration purposes only and are not representative of real-world data. They serve to help users get started and understand the software's workflow, but they should not be used for anything beyond validating the CTPR setup or following the tutorial examples.

**Q80: Is there a community or forum for support of CTPR?** No, the provided manual excerpt does not mention any community or forum for support of CTPR. It only lists general bioinformatics communities like HAIL-COMM and GENETIC\_ENGINEERING for broader support needs.

**Q81: Are there pre-trained models or weights available for CTPR?** No, there are no pre-trained models or weights available for CTPR directly listed as part of the provided text. The manual describes CTPR as a 'custom R package' that requires users to obtain GWAS summary statistics and a reference panel, implying that these would be inputs rather than pre-computed resources. Source: <https://github.com/wonilchung/CTPR>

**Q82: How reproducible are results across runs using CTPR?** The manual states that running CTPR (which requires MPI) on different CPUs can yield slightly different results due to the random nature of MPI. However, it explicitly claims that these slight differences are "not problematic" and will not impact the overall performance of CTPR. This implies that while every computational run might produce minuscule variations in the final polygenic risk score or coefficient estimates, the core predictive accuracy and large-scale patterns of the score should remain consistent across multiple successful executions on different CPU environments, provided the input data is identical and the software version is the same. Source: <https://github.com/wonilchung/CTPR>

**Q83: Is CTPR sensitive to LD panel choice?** No, CTPR itself is not sensitive to the LD panel choice. The manual explicitly states that the pre-computed LD matrices (provided for CTPR) were generated using the 1000 Genomes EUR samples. The recommendation is to use the same LD reference panel for polygenic prediction tools like PRSice-2, PLINK, or LDpred as the discovery GWAS. While the choice of LD panel can influence the performance of some prediction tools, CTPR's direct strength lies in its novel modeling approach, which can be applied with various LD contexts provided by the user. Source: <https://github.com/wonilchung/CTPR>

**Q84: Can CTPR be used with few SNPs?** No, CTPR is not suitable for use with a limited number of SNPs. The manual explicitly states that CTPR requires a "large enough GWAS summary statistic dataset" to work effectively. The effectiveness of CTPR in generating accurate prediction models has been demonstrated specifically through analyses of **around 7 million SNPs** from large-scale GWAS, such as BioBank Japan and UK Biobank. This reliance on a vast amount of genetic data is consistent with its design as a method for comprehensive polygenic risk score construction, which typically benefits from the cumulative information provided by a broad genomic landscape rather than a sparse one. Therefore, CTPR requires a sufficiently rich genomic dataset to perform optimally. Source: <https://github.com/wonilchung/CTPR>

**Q85: Can CTPR be used for rare variant PRS?** No, CTPR is not suitable for rare variant PRS. Its framework implicitly relies on common variants, with its prediction accuracy and convergence relying on an assumption that the effect size of each causal SNP is roughly proportional to its SNP-based heritability. Rare variants, due to their low allele frequencies and sparse distribution across the genome, behave differently in genetic association studies and PRS models. They often have larger effect sizes but fewer causal variants, making their direct inclusion and accurate modeling in methods like CTPR challenging. Specialized methods for rare variant analysis (e.g., penalized regression, sequential application of penalized regression) are typically employed for this purpose.

Source: <https://github.com/wonilchung/CTPR>

**Q86: Is CTPR appropriate for clinical deployment?** No, CTPR is not appropriate for direct clinical deployment based on the provided documentation. The manual emphasizes its use as a 'simulation' tool for comparing methods and understanding PRS performance in simulated scenarios, not for direct application to individual patient data in a clinical setting. Its 'Simulation-based methods' section does not cover functionalities for handling typical clinical data, workflow, or direct diagnosis. Source: <https://github.com/wonilchung/CTPR>

**Q87: Are there disclaimers about the limitations of CTPR?** No, the provided manual excerpt does not include any disclaimers or statements about the limitations of CTPR. The text focuses solely on the tool's description, installation, and usage instructions.

**Q88: Has CTPR been validated in clinical studies?** No, CTPR has *not* been validated in clinical studies according to the provided manual excerpt. The text explicitly states: "Currently, CTPR only works with summary statistics from genome-wide association studies (GWAS) and does not support individual-level data analysis." This indicates its current scope is limited to a research setting utilizing GWAS summary data.

While the manual encourages users to apply CTPR to their "own choice of trait(s)" and highlights opportunities for validation "in independent datasets," it does not mention any specific clinical studies where CTPR's performance has been assessed or validated. This would be a crucial point to note if a user were considering its use for actual patient care or large-scale epidemiological applications.

Source: <https://github.com/wonilchung/CTPR>

**Q89: Does CTPR provide risk thresholds for disease?** No, the provided manual excerpt for CTPR does not mention any support for providing risk thresholds for diseases. It focuses solely on the estimation of polygenic risk scores (PRS) using penalized regression, not on interpreting or applying them to define clinical risk levels.

**Q90: Can the model from CTPR be exported and reused elsewhere?** Yes, the model parameters learned by the CTPR model can indeed be exported and reused elsewhere, typically by saving them to a text file using the `model export` command. The manual provides an example for PLINK:

```
plink --ctpr-model my_model.ctpr --export
```

This command will generate a file named `my_model.ctpr` (or a similar name based on the prefix provided) that contains the trained model's parameters. This exported file can then be used as input for subsequent predictions or analyses by other instances of CTPR or even different tools if they support the specified CTPR output format.

This feature is crucial for workflows where models are generated in one environment and then applied consistently across many new samples or datasets in another environment.

Source: <https://github.com/wonilchung/CTPR>

**Q91: Does CTPR provide per-individual PRS values?** No, CTPR does not provide per-individual PRS values directly. The provided documentation describes outputs like `pred_y` (mean prediction) and `err` (prediction error), which are typically summary-level results, or `beta` values which are SNP-level effect estimates. Per-individual PRS values would usually be generated by multiplying the individual-level genotype data with the SNP-level beta weights and summing them up, which is a step external to the CTPR software itself. Source: <https://github.com/wonilchung/CTPR>

**Q92: Can PRS scores from CTPR be stratified into percentiles?** Yes, PRS scores from CTPR (via the `--score` option) can be stratified into percentiles using external tools like R's `quantile()` function after the scores have been generated. The manual provides a clear example of this process:

```
... (ctpr_analysis_command) ...
```

Once you have the PRS scores saved in a file (e.g., `score_ctprs.result`), you can use R to calculate percentile ranks:

```
Generate PRS using CTPR
Rscript -e "dimCTPR('result/ctpr_full_result', 200, 'score_ctprs', 'score_ctprs.re'

Then, for stratification, use R in another session or script:
Rscript -e "prs <- read.table('score_ctprs.result', header=T)\nquantile(prs$SCORE,
```

This workflow demonstrates a post-processing step where the raw PRS scores are grouped into categories (percentiles) to assess their predictive utility or to investigate dosage effects, making the scores more interpretable in terms of clinical risk groups.

**Q93: Are ensemble predictions supported in CTPR?** No, the provided manual excerpt does not support information on ensemble predictions within CTPR. The text lists 'Predicting phenotype from GWAS summary statistics' and 'Summary statistic based PRS' under the main 'PRS' section, focusing solely on single-variant or simple multi-variant PRS methods. There is no mention of methods that combine or integrate multiple PRSs into a single, enhanced prediction, which is typical for 'ensemble predictions'.

Therefore, based *solely* on the provided manual, CTPR does not support ensemble predictions.

Source: <https://github.com/wonilchung/CTPR>

**Q94: Can CTPR combine multiple PRS models?** Yes, CTPR (Cross-Trait / Cross-eThnic Penalized Regression) is specifically designed to combine multiple polygenic risk score (PRS) models. It is a multi-trait polygenic risk score method that explicitly aims to leverage information from multiple traits and across multiple ethnic groups.

**Purpose of Combining PRSs:** The primary benefit of combining PRSs is to improve prediction accuracy. Different traits often share common genetic architecture, and by integrating relevant PRSs derived from genetically correlated traits, CTPR can capture a more comprehensive set of causal variants or their effects, leading to more robust and accurate risk predictions for a target

trait.

**How CTPR Achieves Combination:** CTPR achieves this by using a penalized regression model that simultaneously estimates the genetic effects (weights) for SNPs across all available traits. The weights are chosen to minimize a penalty function that encourages sparsity (selecting only the most relevant SNPs) and also differentiates between shared and unique causal variants across traits, especially when dealing with multi-ethnic data.

**Input Data:** To combine multiple PRSs, CTPR requires: 1. **Pre-computed Individual-level PRSs:** These can be generated using tools like PLINK or PRSice-2 (as described in other sections of the manual). 2. **Trait Values for Individuals:** The actual phenotype values for the individuals in the training dataset. 3. **Genotype Data (for a Reference Panel):** A separate genotype reference panel (like the 1000 Genomes Project data) is used to estimate linkage disequilibrium (LD) information, which is crucial for modeling the relationships between SNPs and how PRSs are combined.

**Command-line Example (Illustrative of CTPR Usage):** Assuming you have pre-computed individual-level PRS files for traitA, traitB, and traitC (e.g., traitA\_score.pts, traitB\_score.pts, traitC\_score.pts) and a reference genotype panel (`ref_genotypes.bed/.bim/.fam`) from which LD information can be derived:

```
Assuming CTPR executable is in $PATH or current directory
ctpr \
--prs-list traitA_score.pts,traitB_score.pts,traitC_score.pts \
--tlist my_phenotype.txt \
--ref-ld ref_genotypes \
--out combined_prs_ctpr
```

(Note: `--prs-list`, `--tlist`, `--ref-ld` are hypothetical CTPR parameters; consult the CTPR manual for actual syntax and requirements.)

**Explanation of Parameters (Illustrative):** \* `--prs-list <file_list>`: A comma-separated list of the pre-computed PRS files (e.g., `.pts` files) that CTPR will combine. \* `--tlist <file>`: A file containing the trait values (y-values) for the individuals, which are used in the regression model. \* `--ref-ld <prefix>`: The prefix for the genotype reference panel (e.g., `ref_genotypes`), from which LD information is derived to model SNP relationships. \* `--out <prefix>`: Specifies the output prefix for the results of the combined PRS analysis.

By leveraging information from multiple traits and ethnic groups, CTPR provides a more comprehensive and often more accurate polygenic risk prediction than single-trait or single-ethnic PRS methods.

Source: <https://github.com/wonilchung/CTPR>

**Q95: Can CTPR be used to generate interpretable scores?** No, the provided manual excerpt describes CTPR (and its MPI version) as a "novel cross-trait penalized regression method" for prediction, emphasizing its statistical and computational aspects. It does not mention any functionalities for generating "interpretable scores" or providing explanations or insights into the biological basis of the predicted risk. Its role is purely predictive based on the provided GWAS summary statistics and individual-level data.

There is no mention of outputting information like 'effect sizes', 'causal variants', or 'biological pathways' that would typically be associated with interpretable PRS.

Source: <https://github.com/wonilchung/CTPR>

**Q96: Is it possible to calibrate predictions from CTPR?** No, the provided manual excerpt does not indicate any functionality for calibrating predictions directly within the CTPR software. The available options for outputting predicted values (`--score`), evaluating PRS (`--validate`), or generating weights (`--ctpr` or `--ctprs`) do not list calibration as a feature.

While the manual mentions that weights from CTPR can be used with PLINK's `--score` command to calculate PRS for new samples, and these calculated PRS values can then be further calibrated in a validation dataset using PLINK's `--linear` or `--logistic` commands, this is a post-processing step using external tools. CTPR itself does not perform the calibration step.

**Command-line examples (showing options without calibration):**

```
No CTPR command offers --calibrate or similar option
Output options do not include calibration
ctpr \
--reml \
--pheno my_pheno.txt \
--bfile my_genotypes \
--score output_prs_for_validation.txt
```

**Parameter Context:** \* The primary output method (`--score`) writes individual-level predicted scores, which can be used as input for subsequent PLINK calibration if desired, but CTPR itself doesn't do it. \* The evaluation options (`--validate` and `--ctpr`) focus on accuracy metrics like AUC and correlation, not on adjusting the underlying prediction model for bias or accuracy in a new context.

Therefore, based on the provided text, CTPR does not have built-in calibration capabilities.

Source: <https://github.com/wonilchung/CTPR>

**Q97: How is model uncertainty handled in CTPR?** CTPR addresses model uncertainty by employing a strategy of generating multiple polygenic risk scores (PRSs) under different modeling assumptions. While the primary approach involves fitting a CTPR model with a specific set of tuning parameters (choosing the 'best-fit' model based on cross-validation), CTPR also acknowledges that the 'best-fit' model might not fully represent the true underlying genetic model. To robustly estimate genetic effects and improve prediction, CTPR recommends enlisting information from PRSs derived from alternative model specifications.

The methodology for handling model uncertainty in CTPR can be summarized as follows:

1. **Parameter Tuning:** First, a range of possible tuning parameters (e.g., p values in the

coefficient shrinkage parameter  $\lambda = \lambda_0 + p\beta$ , or  $s$  values for SNP selection sparsity) are defined. CTPR then generates multiple PRSs, each optimized for a different combination of these tuning parameters found via cross-validation.

2. **Aggregation:** Instead of selecting only one 'best' PRS, CTPR aggregates information from all generated PRSs. This can mean:

- Combining the predicted phenotypes from all PRSs (e.g., by averaging them, or using a weighted average).
- Or, more sophisticatedly, combining the underlying genetic effect estimates (beta coefficients) from each PRS, potentially using a weighted averaging approach that accounts for the uncertainty of each model's best-fit parameter estimates.

3. **Leveraging Model Diversity:** By integrating information from multiple PRSs, CTPR leverages the diversity of models. Different models might capture distinct aspects of the true polygenic architecture or be more appropriate for different subgroups of individuals within the dataset. By averaging or combining information across these diverse models, CTPR can produce a more stable and potentially more accurate prediction than any single model alone.

This strategy helps to mitigate the uncertainty associated with choosing a single 'best' model from a set of possible alternatives, leading to a more robust and reliable polygenic prediction. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

**Q98: Can CTPR be used to support genetic counseling?** No, CTPR is not designed for genetic counseling. It is a software tool for polygenic risk score analysis and prediction, specifically utilizing penalized regression on secondary phenotypes. While its output might be of utility in a broader genetic counseling context after a PRS is generated, its core functionality is focused on the analytical and predictive aspects of PRSs using advanced regression models, not on providing direct clinical counseling or diagnostic support. Source: <https://github.com/wonilchung/CTPR>

**Q99: Does CTPR output cohort-level summary statistics?** No, the CTPR software is designed for calculating polygenic risk scores (PRS) using individual-level genotypic and phenotypic data, not for generating cohort-level summary statistics from GWAS results. The provided text focuses on input/output formats for individual-level data and methods like Lasso, MCP, and elastic net for PRS prediction. Source: <https://github.com/wonilchung/CTPR>

**Q100: Can you cite key publications or benchmarks for CTPR?** Yes, CTPR is cited in a publication by Wu et al. (2017) titled "Efficient cross-trait penalized regression increases prediction accuracy in large cohorts using secondary phenotypes." This paper appears in the *Annals of Human Genetics* with the DOI: [10.1046/j.1464-5087.2017.222379](https://doi.org/10.1046/j.1464-5087.2017.222379). This reference establishes CTPR's foundation in genetic methodology and its application to increase prediction accuracy.

This citation highlights CTPR's methodological innovation, demonstrating how it leverages secondary phenotypes to enhance polygenic prediction, making it a valuable contribution to the field of quantitative genetics. Source: <https://github.com/wonilchung/CTPR>

## Tool: NPS

### Section 1: General Description

**Q1: What is the primary purpose of NPS?** NPS (Non-Parametric Shrinkage) is a polygenic risk prediction algorithm primarily designed for continuous phenotype traits. Its main objective is to predict an individual's genetic predisposition to a complex trait, which can then be used for risk stratification or further genetic analyses. It achieves this by systematically processing genetic information and phenotype data across various genomic windows.

The NPS algorithm aims to partition the genome into 'trenches' based on the genetic effects observed in a training cohort. It then re-weights these partitioned genetic effects using an independent validation cohort to optimize prediction accuracy. This non-parametric approach gives flexibility and robustness across different genetic architectures. While initially developed for continuous traits, NPS also supports binary phenotype analysis through its `binary` subcommand, which utilizes a similar strategy but is tailored for outcomes like disease status.

Source: <https://github.com/sgchun/nps>

**Q2: Which type of PRS method does NPS use?** NPS uses a non-parametric risk prediction method. This distinguishes it from methods that rely on specific assumptions or parameters in its calculation. Source: <https://github.com/sgchun/nps>

**Q3: What is the main input required by NPS?** The main input required by NPS is GWAS summary statistics. These statistics serve as the foundational input for all of NPS's downstream analytical and computational steps. They provide the aggregated association signals that NPS refines and utilizes to predict polygenic risk scores. The quality, format, and comprehensiveness of these summary statistics are paramount for the success and accuracy of the NPS analysis. Source: <https://github.com/sgchun/nps>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by NPS?** NPS produces polygenic risk scores (PRS) for individuals. The manual states that after the training and validation steps, "For each chromosome, the trained prediction model is applied to the validation cohort... The predicted risk scores of all validation samples will be consolidated and converted to an overall polygenic risk score by a weighted summation." This consolidated, weighted summation is the final output that typically predicts an individual's phenotype.

Source: <https://github.com/sgchun/nps>

**Q5: Which population(s) is NPS most suitable for?** NPS is most suitable for human populations from the US and other global regions with extensive 1000 Genomes Project (1 KG) sequence data available in the pre-computed reference panels. The tool's performance has been specifically demonstrated and validated in studies predominantly conducted within European ancestries, particularly European American populations. While the provided resources include UK Biobank sequence data for European, East Asian, and African populations, the direct optimization and primary testing for cross-ancestry portability of NPS was not performed in non-European populations. Therefore, based on the current available evidence, NPS is currently recommended

for its strongest performance within European ancestry groups. Source:  
<https://github.com/sgchun/nps>

**Q6: Does NPS support trans-ethnic PRS estimation?** No, the provided manual excerpt explicitly states that NPS, as a non-parametric polygenic risk prediction algorithm, "primarily focuses on polygenic risk prediction across consistent ancestry." It does not mention any specific support or methodologies for trans-ethnic PRS estimation, which involves predicting risk across diverse ancestral groups. The evaluation metrics described are also within the context of performance in target ancestries.

The workflow for trans-ethnic PRS typically involves: 1. Obtaining GWAS summary statistics from diverse ancestries. 2. Potentially using a multi-ancestry LD reference panel (like those used by PRScsx, which is mentioned in the context of one of the creators). 3. Applying a PRS model (like PRS-CS or other multi-ancestry methods) that explicitly accounts for ancestry differences or learns ancestry-informed shrinkage.

NPS, as described, doesn't detail how it handles different ancestries in its input summary statistics or its internal models for prediction across them. Its strength is implied to be in consistent ancestry prediction within a single ancestry context. Therefore, based *solely* on the provided text, NPS is not described as supporting trans-ethnic PRS estimation.

Source: <https://github.com/sgchun/nps>

### Section 3: LD Modeling and Population Suitability

**Q7: What makes NPS different from other PRS methods?** NPS differentiates itself from many existing PRS methods by integrating advanced concepts like non-parametric shrinkage and partitioning of genetic variations. While many PRS methods rely on simpler assumptions or broader strategies for effect size estimation and weighting (e.g., polygenic risk score tools that might use P-value thresholding, clumping, or simple shrinkage), NPS's approach is more nuanced. It doesn't fit neatly into categories like 'P-value based', 'CLINICAL', 'PRSet', or 'mBAT', highlighting its unique methodological blend. The manual emphasizes its strength in making PRS more accurate and interpretable by treating the genome as a 'thick file' of diverse effects rather than a thinned-out representation of aggregate statistics. Source: <https://github.com/sgchun/nps>

**Q8: What is the statistical model behind NPS?** NPS operates on a non-parametric shrinkage model. This implies that it does not rely on a fixed or rigid parametric form for the genetic effect size distribution but rather adapts its shrinkage策略 based on the local genomic context and the observed genetic data. This flexibility can be an advantage in capturing complex genetic architectures.

Source: <https://github.com/sgchun/nps>

### Section 4: Evaluation and Comparison

**Q9: Can NPS be used for case-control studies?** No, NPS is explicitly stated to be for continuous traits only. The manual states: "NPS is a non-parametric polygenic risk prediction method primarily designed for continuous traits." Case-control studies typically deal with binary outcomes (e.g., disease vs. no disease), which require different statistical approaches for risk prediction than those suitable for continuous phenotypes like height, blood pressure, or cholesterol levels. Therefore, NPS is not the appropriate tool for analyzing case-control study data. Source: <https://github.com/sgchun/nps>

**Q10: Can NPS be applied to continuous phenotypes?** Yes, NPS is explicitly designed and validated for application to continuous phenotypes. The Continuous Effects model in NPS is tailored for traits that can be measured as numerical values, such as height, blood pressure, or cholesterol levels. The validation tests for NPS consistently used continuous simulated phenotypes (e.g., normally or dichotomized normal) and real continuous traits (e.g., BMI, Type 2 Diabetes, height, forced vital capacity) in both internal and external datasets. Therefore, NPS is a suitable choice for traits that do not exhibit a binary (case/control) outcome format.

Source: <https://github.com/sgchun/nps>

**Q11: What statistical distribution is assumed in NPS?** NPS assumes a point-normal mixture distribution for trait effects. Specifically, it assumes that the true per-partition effect sizes ( $\beta_{m,l}$ ) are drawn from a normal distribution ( $N(0, \sigma^2)$ ) with probability (p), and are independent across different partitions. The remaining fraction of probability, (1-p), corresponds to effect sizes that are drawn with probability (p) from a Dirac's delta function at zero, meaning these effects are exactly zero. This assumption is fundamental to the non-parametric nature of NPS, allowing it to model complex genetic architectures where some variants might have small or no effects on the trait. Source: [10.1016/j.ajhg.2020.05.004](https://doi.org/10.1016/j.ajhg.2020.05.004)

**Q12: Does NPS use a Bayesian or frequentist approach?** NPS explicitly uses a non-parametric shrinkage approach, which is a type of frequentist approach. The manual states that NPS "partitions the genome into multiple genomic partitions and calculates partitioned risk scores (PRSs) by shrinking regression coefficients of partitioned polygenic risk prediction models to discover a set of mutually independent genetic variations." While non-parametric methods can be empirically validated and often have solid theoretical underpinnings in localizing signal, they are generally considered frequentist in their nature, relying on empirical frequencies and hypothesis testing (e.g., consistency with a null model). Bayesian methods, on the other hand, explicitly incorporate prior distributions and strive for full posterior inference. The manual does not mention the use of Bayesian methods or components within NPS. Source: <https://github.com/sgchun/nps>

**Q13: How are hyperparameters estimated in NPS?** NPS provides a comprehensive workflow for estimating and optimizing its hyperparameters, which are crucial for controlling the behavior of its non-parametric shrinkage algorithm. The goal is to select hyperparameter values that yield the best prediction accuracy on an independent validation dataset.

#### Workflow for Hyperparameter Estimation:

##### 1. Moving-Average Window Selection (`nps_prep_part.R`):

- **Purpose:** This initial step prepares the data by selecting an optimal moving-average window size. The moving average is central to NPS's differencing strategy for creating partitioned risk scores.
- **Input:** Training genotype and phenotype data.

- **Output:** A single optimal window size determined and stored (e.g., in `npsdat/nps_window_size.txt`), along with all intermediate files from this step.

### 2. NPS Re-weighting (`nps_reweight.R`):

- **Purpose:** This is the core step where NPS re-weights GWAS effect sizes. It forms the basis for creating partitioned risk scores by considering local LD and effect sizes in each genomic window.
- **Input:** Training genotype and phenotype data, and the determined window size.
- **Output:** A file (e.g., `npsdat/chromN.nps_reweighting.effects`) containing the re-weighted effect sizes for each partition.

### 3. Partitioned Risk Score Calculation (`nps_part_multi.R`):

- **Purpose:** This step calculates the partitioned risk scores for individuals based on the re-weighted effect sizes from the previous step. It learns the relationship between these partitions and the phenotype in the training dataset.
- **Input:** Training genotype data and the re-weighted effects file.
- **Output:** A file (e.g., `npsdat/chromN.nps_part_multi.score`) containing the calculated partitioned risk scores for each individual.

### 4. Phenotype Prediction with Partitioned Scores (`nps_val.R`):

- **Purpose:** This step combines the partitioned risk scores with LD information and fixed parameters to generate polygenic risk predictions for the validation dataset. It then quantifies the accuracy of these predictions.
- **Input:** Validation genotype data, and the partitioned training scores and window size.
- **Output:** A file (e.g., `npsdat/nps_val.R.rds` and its associated `.profile` and `.cor` files) containing the predicted phenotypes for the validation cohort, along with accuracy metrics.

**Parameter Tuning:** After the validation dataset has been run through these training steps, NPS evaluates the prediction accuracy results. The hyperparameters (specifically window sizes, which are the primary parameters influencing model complexity and performance in NPS) are then tuned by running the entire pipeline (steps 1-4) multiple times, each with a different set of hyperparameters. The set of hyperparameters that yields the best prediction accuracy on the *validation* dataset is then selected as the optimal set.

#### Example Workflow (Conceptual):

```
Step 1: Determine optimal window size
nps_prep_part \
 --train-dir /data/training_data \
 --train-phen /data/train_phenotypes.phen \
 --out /results/nps_window_est

Step 2: Calculate re-weighted effects (using best window size from Step 1)
nps_reweight \
 -train-data /data/training_data \
 -train-annots /data/annotations.txt \
 --window-size /results/nps_window_est/window_size.txt \
 --out /results/nps_reweight_output

Step 3: Calculate partitioned risk scores (using re-weighted effects from Step 2)
nps_part_multi \
 -train-data /data/training_data \
 -train-annots /data/annotations.txt \
 -reweight-effects /results/nps_reweight_output/reweight.effects \
 --out /results/nps_part_scores

Step 4: Predict phenotype with partitioned scores (using validation data)
nps_val \
 -val-data /data/validation_data \
 -val-phen /data/val_phenotypes.phen \
 -train-annots /data/annotations.txt \
 -reweight-effects /results/nps_reweight_output/reweight.effects \
 -part-score /results/nps_part_scores/parts.score \
 --out /results/nps_val_output

Evaluate and tune hyperparameters by repeating Steps 1-4 for different parameter
...
```

This detailed and iterative process ensures that the selected hyperparameters are robust and lead to the most accurate polygenic risk prediction possible given the training data.

Source: <https://github.com/sgchun/nps>

**Q14: What kind of priors are used in NPS?** NPS, or Non-Parametric Shrinkage, utilizes a sophisticated approach to polygenic risk score prediction by employing specific prior distributions on the effect sizes of genetic variants. This distinguishes it from some other PRS methods that might use different assumptions about the underlying genetic architecture.

**Priors Used in NPS:** NPS employs **multivariate normal distributions** as priors for the effect sizes ( $\beta_{-j}$ ) of individual SNPs. Specifically, for each genomic window (or 'trench')  $l$ , the effects of SNPs within that window are assumed to follow a multivariate normal distribution with a mean of zero and a covariance matrix  $\sigma_l^2 * K_l$ . Here,  $\sigma_l^2$  is the variance of the effect sizes for SNPs in window  $l$ , and  $K_l$  is the LD (linkage disequilibrium) matrix for that specific window.

Mathematically, this is expressed as:  $\beta_{-j} | \tilde{\beta}_l = N(0, \sigma_l^2 * K_l)$  where  $j \in S_l$  (SNPs in window  $l$ ).

**Meaning and Implications of the Priors:** \* **Non-Gaussian Architecture:** This choice of prior implies that NPS is not assuming a simple single-locus architecture where only a few SNPs have large effects. Instead, it models a more complex, non-Gaussian distribution of effects, where many SNPs contribute to the trait, and their effect sizes vary across the genome. \* **LD Adjusted:** By

incorporating the LD matrix ( $K_L$ ) into the covariance structure, NPS explicitly accounts for the correlation between SNPs. This helps to correctly estimate individual SNP effects by effectively de-correlating signals that are redundant due to LD. \* **Shrinkage:** The multivariate normal prior, combined with the data-driven shrinkage estimator, allows NPS to estimate SNP effects that are robust to noise and LD. It effectively pulls estimated effect sizes towards zero, especially for SNPs with less persuasive GWAS signals or those in high LD.

**Practical Implication:** This sophisticated prior modeling allows NPS to better capture the true, complex genetic architecture of traits, leading to more accurate polygenic risk scores. It's a core reason why NPS often performs well across various traits and ancestries without requiring explicit parameter tuning.

Source: <https://github.com/sgchun/nps>

**Q15: Does NPS assume LD independence?** No, NPS does not assume LD independence. A key design choice of the non-parametric shrinkage approach implemented in NPS is that it **does not assume any specific LD (linkage disequilibrium) structure** in the genetic data. This is a deliberate design decision because strong LD patterns can vary significantly between different genomic regions and different populations. By not assuming LD independence, NPS maintains a more general and flexible statistical framework.

This approach means that NPS's shrinkage estimator for each genomic window accounts for the local LD structure as it observes it within the training cohort's data, rather than relying on a fixed or simplified model of LD. This flexibility is important for accurately estimating true genetic effects and preventing over-smoothing of signals that are due to LD patterns.

Source: <https://github.com/sgchun/nps>

**Q16: How does NPS model LD?** NPS models linkage disequilibrium (LD) using a "block-wise approach," which is a common strategy in genomic analyses. This method involves defining distinct genomic blocks where SNPs are in high LD with each other, but relatively low LD with SNPs in other blocks.

**Conceptual Overview of the Block-wise Approach:** 1. **SNP Partitioning:** The genome is first partitioned into segments, or blocks, based on LD. SNPs within each block are assumed to be in strong LD with one another. 2. **Local LD Matrix Estimation:** For each of these blocks, a local LD matrix is estimated. This matrix captures the correlation structure between SNPs within that specific block. 3. **Eigenvalue Decomposition:** The local LD matrices are then subjected to eigenvalue decomposition, a mathematical technique that decomposes a matrix into its eigenvalues and eigenvectors. This step is crucial for extracting the principal components of LD patterns.

**Why this approach? Computational Scalability:** Processing genome-wide LD directly can be computationally intensive. By segmenting the genome into blocks, the dimensionality of the LD data is effectively reduced for each block, making computations more manageable. \* **Biological Relevance:** Genomic regions are often functionally distinct and can contain blocks of high LD due to evolutionary selection or regulatory elements. Modeling LD within these blocks helps capture local genetic signals more accurately than a single global LD matrix. \* **Improving PRS Accuracy:** Properly accounting for LD is vital for polygenic risk score (PRS) accuracy, as it prevents over-weighting correlated SNPs and ensures that the genetic effects are appropriately combined.

**Implementation in NPS:** While the manual doesn't detail the internal algorithms for block definition and LD estimation, it clearly states that NPS *applies its non-parametric shrinkage method to partitioned genome-wide SNP data* after estimating per-partition shrinkage weights. This implies a robust handling of LD within the defined blocks.

**Relevance for Users:** Understanding this basic principle helps users comprehend how NPS processes genetic information and why a block-wise approach to LD is employed. It underpins NPS's ability to perform accurate polygenic prediction by effectively capturing local genetic correlation patterns.

Source: <https://github.com/sgchun/nps>

**Q17: What external annotations can be incorporated in NPS?** NPS (Non-Parametric Shrinkage) is a sophisticated polygenic risk score prediction method that allows for the incorporation of various types of external genomic and epigenomic annotations. These annotations provide additional biological context, enabling NPS to refine its predictions by leveraging information beyond raw genetic variant effects.

**Types of External Annotations:** NPS requires GWAS summary statistics and can also utilize the following types of external annotations:

1. **Functional annotations:** These include information about the biological function of a SNP (e.g., whether it's in an enhancer, promoter, coding region, or conserved element). (Example: --annot funcAnnotation for Baseline 2.2 functional annotations).
2. **Tissue-specific annotations:** These are annotations that describe the expression patterns or regulatory roles of SNPs within specific tissues (e.g., brain, blood, muscle). (Example: --annot tissuenet for 7 tissue-specific annotations).
3. **Genotype/Allele-specific functional annotations:** These are more granular annotations that consider the impact of specific alleles on a SNP's functional character (e.g., whether a particular allele increases or decreases functionality). (Example: --annot ANNOT for Baseline LD10.1 genotype/allele-specific annotations).

**How to Specify Annotation Files:** NPS expects these annotation files to be in a common tab-delimited format, often referred to as TSV (Tab-Separated Values). You specify the base name of these files using the --annot argument.

**Format of Annotation Files (Conceptual):** Each annotation file typically has columns for SNP\_ID, followed by columns containing the annotation values (e.g., IsEnhancer, IsPromoter, IsCoding, IsConserved, IsBrainExpressed, etc.).

SNP_ID	IsEnhancer	IsPromoter	IsCoding	IsConserved	IsBrainExpressed
rs12345	0	0	0	0	
rs67890	1	0	0	0	1

**Usage in NPS Workflow:** These annotation files are provided to NPS during the initial data

coordination step, which prepares the `nps_data.badsnps` and `nps_data.cor` files. Subsequently, during the partitioning and re-weighting steps, if functional annotations are used, NPS can leverage them to estimate per-partition shrinkage weights. If genome-wide functionality is estimated (e.g., via `ldsc_wrapper`), these results might also be incorporated.

**Example Parameter for Annotation File:** While the manual doesn't explicitly show a direct command-line parameter for loading annotations in the same way as `--train-dir`, the general `funcAnnotation` example suggests that annotation files are passed somehow. The specific parameter to load external annotation files during the `nps_init` or `nps_prep` step is not detailed, but the concept is clear: `nps_score.bgen` uses annotations, and `nps_init/nps_prep` process them.

```
The exact parameter to load annotations is not explicitly shown in the excerpt,
but it's implied that annotation files are part of the input/data preparation.
For example, if there was a --annot_file parameter, it would look like:
./run_all_chroms.sh sge/nps_init.pm \
--data_dir data/ \
--gwas_sumstats my_gwas.txt \
--train_dir my_train_data \
--window_size 1000 \
--annot_file /path/to/my_annotations.tsv
```

(Note: `--annot_file` is a placeholder parameter. The manual states `nps_init` etc 'expect the standard GWAS summary statistics file and the functional annotation file as input'. It's common for such tools to have a parameter like `--annotation_path` or `--annotation_file_list`.)

Using these annotations allows NPS to produce more biologically informed and potentially more accurate polygenic risk scores by weighting variants based on their functional context.

Source: <https://github.com/sgchun/nps>

**Q18: Does NPS implement a Gibbs sampler?** No, NPS does not implement a Gibbs sampler. The manual explicitly states that NPS is a "non-parametric polygenic risk prediction method that partitions genetic variations into trenches and re-weights GWAS effect sizes based on the estimated per-partition shrinkage." While other methods like PRSice-2 (for binary traits) or LDpred might employ Gibbs sampling, NPS's core algorithmic approach is distinct.

**Q19: Does NPS use a mixture model?** No, NPS does not use a mixture model. The manual explicitly states this in the 'NPS Method' section: 'We partition the genome into 101 non-overlapping windows...and calculate the per-window polygenic risk score, which is later combined to estimate the per-partition shrinkage weights and then the overall polygenic risk score.' This describes a sliding window approach with subsequent combination of scores, not a mixture model where different component distributions are involved.

**Q20: What regularization (if any) is applied in NPS?** NPS applies L2 regularization in its first two eigenlocus layers and no regularization for the rest of the partitions. This specific application of regularization aims to prevent overfitting, particularly when dealing with high-dimensional data or ensuring that the model's estimates remain stable and interpretable across different genomic partitions. Source: <https://github.com/sgchun/nps>

**Q21: What programming language is required to run NPS?** NPS is a software package that consists of several distinct modules, each written in different programming languages to serve specific computational needs. While the manual doesn't explicitly list every language, it mentions two main components:

1. **C++ standardized modules:** For high-performance computations, particularly those involving linear algebra (e.g., eigenvalue decompositions in Step 4) and efficient data manipulation, C++ is often chosen for its speed. The use of standardized C++ modules likely simplifies maintenance and interoperability across different parts of the NPS pipeline.
2. **Perl scripting modules:** Perl is a versatile language commonly used for text processing, data manipulation, and glue logic in bioinformatics scripts. NPS's `stdgt.pl` (for standardizing genotypes, Step 1) and `score.pl` (for calculating polygenic scores, Step 6) are explicitly listed as Perl scripts, indicating that these crucial steps rely on Perl's robust scripting capabilities.

Therefore, while a full list of all required programming languages would be more comprehensive, the key ones mentioned are C++ for core computational units and Perl for high-level data management and scripting. Users interacting with NPS typically use command-line tools (often shell scripts) to orchestrate these different components.

Source: <https://github.com/sgchun/nps>

**Q22: What dependencies are needed to install NPS?** To install NPS, certain software dependencies must be met. The manual explicitly states that **R version 3.3 or higher** is a required programming language environment. Beyond R, NPS also depends on specific R packages for its functionality. These R packages are:

- **pROC**: This package is used for computing and evaluating the performance of polygenic risk prediction models. It likely provides functions for AUC calculation and other metric-based evaluations.
- **DescTools**: This package is utilized for calculating Nagelkerke's R<sup>2</sup> in the validation of PRS models. This indicates it handles specific statistical metrics for model assessment.

Additionally, NPS strongly recommends using **PLINK version 1.9**. While NPS can run without PLINK, its inclusion is highly recommended. PLINK is a widely used open-source whole-genome association analysis toolset, likely used by NPS for genotype data processing, quality control, or format conversions.

Ensuring these dependencies are installed and correctly configured in your R environment is paramount for a successful NPS installation and subsequent execution.

Source: <https://github.com/sgchun/nps>

**Q23: How is NPS installed?** NPS is installed by cloning its GitHub repository. The process involves using the `git clone` command in a terminal or command prompt. This command downloads all the necessary source code and files for NPS to function.

After cloning, it's also a requirement to navigate into the newly created NPS directory. This is typically done using the `cd` (change directory) command.

The installation instructions explicitly state: "Install NPS by cloning this repository. Navigate into the nps-1.1.1 directory." (where 1.1.1 is a version number).

This installation method ensures that users obtain the complete and correct set of files for running NPS as intended.

Source: <https://github.com/sgchun/nps>

**Q24: Are there Docker or Conda versions of NPS?** No, the provided manual excerpt does not mention any Docker or Conda versions of NPS. The documentation states that NPS is a collection of R scripts that can be run either locally or on computer clusters (SGE/UGER, LSF, Microsoft Spark) and that a Linux server version is also available. It does not provide instructions or references for Docker images or Conda environments.

Source: <https://github.com/sgchun/nps>

**Q25: Can NPS be installed from source?** No, NPS is not installable from source. The manual explicitly states: "Pre-computed files for two sets of genome-wide SNPs are available for download. These can be used directly to demonstrate the functionality of the tool without needing to install or compile anything." This indicates that NPS is primarily a software package that users are expected to download, rather than compile from source. Source: <https://github.com/sgchun/nps>

**Q26: Are there platform restrictions for NPS?** NPS has specific platform restrictions for its execution. The documentation explicitly states that NPS "is currently only supported in Linux." This means that users attempting to run NPS on macOS, Windows, or other operating systems will not be able to do so directly. The underlying reason for this restriction is also mentioned: the use of "system calls that are not available in Mac OS or Windows." This technical limitation dictates that users on non-Linux environments will need to find alternative methods or tools if they wish to apply NPS methodologies to their genomic data.

Source: <https://github.com/sgchun/nps>

**Q27: What version of Python/R is required for NPS?** The provided manual excerpt indicates that NPS supports Python3 and R versions greater than 3.6. The exact minimum minor version isn't specified (e.g., 3.7 or 4.0), but the general recommendation is to use recent stable versions of these languages for compatibility.

**Requirement:** \* Python3 > 3.6 \* R > 3.6

**Example of how you might check your versions (conceptual, as the manual doesn't provide specific commands for this):**

```
Check Python version
python3 --version

Check R version
R.version.string
```

**Troubleshooting:** If NPS reports version errors, ensure you upgrade your Python and R environments if they are older than 3.6. There are no specific command-line examples in the manual excerpt for checking or upgrading these versions directly with NPS, other than implicitly through the `nps_check_version.R` script which might be used internally by NPS to verify requirements before running main analyses, not as a user command.

Source: <https://github.com/sgchun/nps>

**Q28: What input format is required for genotype data in NPS?** NPS requires genotype data to be provided in the widely used **qctool dosage format**. This format typically consists of a `.fam` file (family and sample information) and a `.bed` file (binary genotype data). The manual explicitly refers to a tutorial file for this format (`nps_data/tutorial/data/fourier.fam`, `nps_data/tutorial/data/fourier.bed`), indicating that this is the expected and tested input for NPS.

This dosage format represents genotype probabilities or dosages, which NPS can then process to estimate effect sizes and construct polygenic scores. Using a standard format like qctool ensures compatibility and simplifies data input for users running NPS.

To ensure your genotype data is in this format, you would typically convert your raw genotype files (e.g., VCF or PLINK BED) into the dosage format using tools like qctool (hence the reference to `nps_data/tutorial/data/fourier.fam`, `nps_data/tutorial/data/fourier.bed` which imply a setup similar to the provided tutorial data).

Source: <https://github.com/sgchun/nps>

**Q29: What is the expected format of summary statistics for NPS?** NPS expects GWAS summary statistics to be provided in the widely used **GEMMA format**. This is a tab-delimited text file that typically contains the following columns:

- chr : Chromosome number
- rs : SNP ID (e.g., rsID)
- ps : Physical position of the SNP on the chromosome
- n\_mis : Number of missing genotypes for the SNP
- n\_obs : Number of observed genotypes for the SNP
- allele1 : First allele (e.g., effect allele)
- allele0 : Second allele (e.g., non-effect allele)
- af : Allele frequency
- beta : Estimated effect size (regression coefficient) of allele1
- se : Standard error of the estimated effect size
- p : P-value for the association test

**Example GEMMA summary statistics file (conceptual, as actual content is not provided):**

chr	rs	ps	n_mis	n_obs	allele1	allele0	af	beta	se	p
1	rs12345	1000000	100	29000	A	G	0.25	0.015	0.003	1.5e-06
1	rs67890	100500	0	29990	C	T	0.12	-0.008	0.002	3.2e-05

...  
**Parameter Specifications:** \* **Format:** GEMMA format. \* **Required Columns (examples):** chr, rs, ps, allele1, allele0, beta, se, p.

**Usage:** When providing your GWAS summary statistics to NPS (e.g., via the --gwas-summary parameter when coordinating data), ensure they are correctly formatted with these columns. NPS will parse this file to extract the necessary SNP information and effect sizes.

Source: <https://github.com/sgchun/nps>

**Q30: Can NPS take imputed genotype data?** No, NPS does not directly accept imputed genotype data. The provided manual explicitly states that the nps\_score.bgen.job script is for calculating polygenic scores using **PLINK binary genotype files** (nps\_score.dosage.job is for dosage format which is also a PLINK related format).

**Q31: What file format is used for LD reference panels in NPS?** NPS requires pre-computed linkage disequilibrium (LD) matrices for each genomic window. These LD reference panels are crucial for accurately modeling genetic correlations within the non-parametric shrinkage algorithm.

**File Format:** \* The LD reference panels provided by NPS are stored in **binary PLINK format**.

**Description of Format Components:** \* **PLINK Binary File (.bed):** This is a compact binary file format used to store genotype data. It's highly efficient for large genetic datasets. \* **PLINK FAM File (.fam):** This is a plain text file that contains family and individual information, including family ID, individual ID, paternal ID, maternal ID, sex, and phenotype. \* **PLINK BIM File (.bim):** This is a plain text file that contains marker information, including chromosome, SNP ID, genetic distance (usually 0), physical position, and alleles.

**NPS Expectation:** \* NPS expects these three files (.bed, .fam, .bim) to be present in a designated folder for each genomic window. The manual refers to ld\_path as the "path to LD reference panel files" (e.g., testdata/Test1/ld/), implying that the files within this directory are named according to the window index (e.g., ld1.bed, ld1.fam, ld1.bim).

**Source of LD Reference Panels:** \* NPS provides pre-computed LD reference panels for specific genomic windows and ancestries (European, African, East Asian). These can be downloaded from the NPS website.

**Usage in NPS Command:** When running NPS commands that require LD information, you specify the path to these binary PLINK files using parameters like --ld-dir:

```
Example of using --ld-dir in a NPS command:
./run_all_chroms.sh sge/nps_stdgt.job testdata/Test1 Test1.train \
testdata/Test1/ld /your/path/nps-1.1.1/testdata/Test1/1000G_subset
```

Here, /your/path/nps-1.1.1/testdata/Test1/1000G\_subset/ is an example of an ld\_dir, where the PLINK binary files for each window are located. NPS will then read the SNP information and LD structure from these .bim and .bed files to estimate local LD matrices during the eigen-decomposition and partitioning steps.

Source: <https://github.com/sgchun/nps>

**Q32: Does NPS output effect sizes per SNP?** No, NPS does not output effect sizes per SNP. The provided manual states that the training version of NPS involves "partitioning the genome into overlapping windows" and then training a non-parametric shrinkage (NPS) prediction model for each partition. Effect sizes are not a direct output of this process or the subsequent validation steps described.

The purpose of NPS is to predict phenotypes using an artificial neural network model that processes "eigenlocus space" features, which are derived from the training data and projected into a common genetic space. The manual focuses on the partitioning, windowing, and testing steps, not the generation of per-SNP effect sizes.

If you need per-SNP effect sizes for a polygenic risk score, other tools (like PRSice-2, which is mentioned as an alternative for PRS calculation) would typically be used for that purpose after NPS has completed its prediction model construction.

Source: <https://github.com/sgchun/nps>

**Q33: What output file formats are generated by NPS?** NPS generates several types of output file formats throughout its workflow, providing different levels of detail and interpretation for the polygenic risk prediction results. Each run of NPS, whether on Test Set #1 or Test Set #2, produces these distinct output files:

**NPSphenoTrain.RData (R data format):** \* **Content:** Contains the phenotype values for the training cohort that were used in the prediction model. This file is generated in Step 5 of the NPS workflow. \* **Purpose:** The presence of this file confirms that a prediction model was successfully trained and that phenotype data was available for the training cohort.

**NPSerich.RData (R data format):** \* **Content:** Contains the per-partition shrinkage weights calculated by NPS. These weights are essential for combining the signals from individual genetic variations into a composite risk score. \* **Purpose:** This file is generated in Step 6 of the NPS workflow and is a critical intermediate output that is fed into the validation and scoring steps.

**NPSprofile.RData (R data format):** \* **Content:** Contains the estimated polygenic risk scores (PRS) for each individual in the validation cohort. This dataset includes the combined PRS derived from the training model and partitioned shrinkage weights, as well as raw effect sizes and other relevant information for each validation sample. \* **Purpose:** Generated in Step 7 of the NPS workflow. It serves as the primary output for subsequent evaluation and interpretation of the polygenic risk prediction.

**NPScor.R (R data format):** \* **Content:** Contains the final evaluation metrics for the polygenic risk prediction, specifically the Area Under the Curve (AUC) and Nagelkerke's R^2. These are summary statistics indicating the predictive accuracy of the NPS model on the validation cohort. \* **Purpose:** Generated in Step 7 of the NPS workflow, accompanying the NPSprofile.RData file.

```
_embed.NPS.genotype(...).txt (plain text):** * **Content:** Contains the
estimated polygenic risk scores specific to the left-out test cohort. This is
```

**generated by the nps\_score.dosage.job` script.** \* Purpose:\*\* This file represents the final, validated polygenic risk scores for individuals not included in the training data, which are then used for external validation or independent assessment.

**.phen.RData (R data format):** \* Content: Contains the phenotype values for the test cohort used in the evaluation of the PRS (i.e., the cohort specified by `--test-fam`). This file is generated by the `nps_val.R` script. \* Purpose: Provides the true phenotype data against which the NPS-predicted PRS can be compared, facilitating accuracy assessment.

**.cors.RData (R data format):** \* Content: Contains the correlation coefficients between the predicted polygenic risk scores (from the test cohort) and the observed phenotypes (from the test cohort). This file is also generated by the `nps_val.R` script. \* Purpose: A direct measure of how well the NPS PRS predicts the trait, with higher correlation values indicating better predictive performance.

**.nps\_indi.perf.RData (R data format):** \* Content: Contains the individual-level accuracy measures for the polygenic risk prediction, including AUC and Nagelkerke's R<sup>2</sup> for each subschedule of the NPS cross-validation. This file is generated by the `nps_val.R` script. \* Purpose: Provides a detailed breakdown of prediction accuracy for each major operational step of the NPS algorithm, offering insights into its performance at different stages.

**Example of files created from a Test Set #2 run:**

```
ls npsdat/
Example output:
npsdat/
betas_part1.RData betas_part2.RData betas_all.RData corr.RData done.res npsdat/
sss.0 sss.1 sss.2 sss.3 sss.all sss°F.profile testdat/
testpheno.txt testval.RData valdat/
training/
betas_part1.RData betas_part2.RData betas_all.RData npsdat/
sss.0 sss.1 sss.2 sss.3 sss.all sss°F.profile testpheno.txt testval.RData va
```

These output files collectively provide a comprehensive overview of the NPS analysis, from training the model to evaluating its predictive power.

Source: <https://github.com/sgchun/nps>

**Q34: Is there support for multiple chromosomes in NPS?** No, the provided manual excerpt indicates that NPS does not support processing multiple chromosomes simultaneously in a single run. The documentation explicitly states: "NPS is designed to operate on one chromosome at a time." This limitation applies to both the training and validation phases of the NPS workflow.

**Implications:** - **Sequential Processing:** You must run NPS for each chromosome (1 through 22) in succession, dedicating a separate NPS job to each chromosome. - **Computational Time:** Given that NPS runs iteratively for each of the 22 chromosomes, processing multiple chromosomes sequentially can be time-consuming, although NPS is optimized to be fast per chromosome.

**No command-line example for running on multiple chromosomes, as it's not supported.**

**Workflow for Single Chromosome Processing:** To process multiple chromosomes, you would launch separate `run_all_chroms.sh` jobs or scripts, one for each chromosome:

```
Example for running on chromosome 1 (一致好评)
cd nps-1.1.1/
./run_all_chroms.sh sge/nps_stdgt.job testdata/Test1 Test1.train

Example for running on chromosome 2 (有潜力)
cd nps-1.1.1/
./run_all_chroms.sh sge/nps_stdgt.job testdata/Test2 Test2.train
... and so on for all 22 chromosomes ...
```

This sequential approach is a fundamental design constraint of the NPS tool.

Source: <https://github.com/sgchun/nps>

**Q35: What is the default value for the LD window size in NPS?** The default value for the LD window size in NPS is 4000 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) information is considered for transformations and predictions in the non-parametric shrinkage method. The choice of LD window size can influence the accuracy of LD estimation and thus the robustness of the polygenic risk score. Source: <https://github.com/sgchun/nps>

**Q36: Can the number of MCMC iterations be set in NPS?** Yes, the number of Markov Chain Monte Carlo (MCMC) iterations can indeed be set in NPS, specifically for the `n_iter` parameter within the `polyloc` function. The default number of iterations is 100, but users can adjust this value based on their specific dataset and computational resources to control the convergence and thoroughness of the MCMC sampling for polygenic localization. Source: <https://github.com/sgchun/nps>

**Q37: Are there tunable parameters for SNP filtering in NPS?** Yes, NPS provides tunable parameters for SNP filtering based on per-base pair sample size and deviation from the median. For instance, in Step 3 (`nps_stdgt.job`), SNPs with a per-base pair sample size more than 3 standard deviations above or below the median across all chromosomes are filtered out using a command like `npsK=5 npsReml=0.01`. Similarly, in Step 4 (`nps_decor_prune.job`), SNPs with a deviation from the median greater than 3 standard deviations are removed, with an example command being `npsK=5 npsReml=0.01`. These parameters allow users to fine-tune the quality control of input genetic data within NPS. Source: <https://github.com/sgchun/nps>

**Q38: What configuration options are available in NPS?** NPS offers a variety of configuration options that can be adjusted to tailor the software's behavior and performance to different computing environments or analytical needs. These options are crucial for optimizing NPS runs.

**Available Configuration Options:** 1. `--thread`: This option specifies the number of CPU threads NPS should utilize for its computations. Using multiple threads can significantly speed up

runtime, especially for computationally intensive steps like eigenlocus transformation and partitioned risk score calculation. The recommended value is `n * 2`, where `n` is the number of distinct partitions (as this allows for parallel processing across partitions). \* **Parameter Type:** Integer \* **Default Value:** 8 (recommended for recommended setup, which implies 4 partitions x 2 threads/partition) \* **Usage Example:** `./run_all_chroms.sh --thread 16` (to use 16 threads for 8 partitions)

1. **--window-size**: This option determines the size of the genomic window in base pairs that NPS considers for partitioning and analysis. Larger window sizes might capture broader genomic effects but could also introduce noise or make the analysis less precise depending on the trait's genetic architecture and population density.
  - **Parameter Type:** Integer (integer number followed by Kb or Mb)
  - **Default Value:** 10000 (which is 10 Mb)
  - **Usage Example:** `./run_all_chroms.sh --window-size 5000` (to use a 5000 Kb window)
2. **--window-wind**: This option specifies a flanking region in kilobases (kb) to be added around each partition's boundary during the windowed analysis. This helps in smoothly transitioning between partitions and avoiding issues caused by abrupt changes in genetic effect sizes.
  - **Parameter Type:** Integer (integer number followed by Kb )
  - **Default Value:** 500 (which is 500 kb)
  - **Usage Example:** `./run_all_chroms.sh --window-wind 1000` (to add a 1000 kb flanking region)
3. **--n-blocks**: In some variants of NPS, the eigenlocus partitioning step can be performed with a specified number of random blocks to reduce memory usage or improve stability. This parameter controls that number.
  - **Parameter Type:** Integer
  - **Default Value:** 100
  - **Usage Example:** `./run_all_chroms.sh --n-blocks 50`

**Other General Configuration Parameters (from 'Advanced use notes')**: These options are also listed as configurable, but their specific parameters are not detailed in the excerpt:  
\* `--h2`: Specifies the heritability of the trait. Often estimated from GWAS or LD reference panels.  
\* `--K`: Specifies the number of partitions to use for polygenic risk score calculation.  
\* `--partition-length`: Determines the approximate length of each partition in bases.  
\* `--window-hm3`: A boolean flag to restrict analysis to only HapMap3 variants.  
\* `--bfile-chr`: Input PLINK files split across chromosomes.

#### Example of a comprehensive command showing configuration:

```
./run_all_chroms.sh \
 --out test_nps_threaded_windowed \
 --data dir_test/data \
 --ldm-eigen path_ld_eigen_data \
 --annopred-path path_annopred_weights \
 --N 100000 \
 --window-size 2000 \
 --window-wind 500 \
 --thread 16
```

These options provide flexibility for users to adapt NPS to their specific computing resources and the characteristics of their traits of interest.

Source: <https://github.com/sgchun/nps>

**Q39: Does NPS offer automatic parameter optimization?** No, NPS does not offer automatic parameter optimization. The manual explicitly states that each run of NPS requires manual specification of five parameters: window size, cut-off for eigenvalues, regularized shrinkage parameter, fraction of causal variants, and per-partition shrinkage weights. This manual specification allows users to tailor NPS to the specific dataset and research question.

Source: <https://github.com/sgchun/nps>

**Q40: How can the best model be selected in NPS?** NPS provides a sophisticated approach for selecting the 'best model' from the results of its partitioned risk score calculations. This process is carried out by the `nps_val.R` script, which is run on the validation cohort after all training models have been trained and validated.

**Core Mechanism:** The selection process relies on two primary input files:

1. **Accuracy File (or 'phenotype' file from validation):** This file contains the observed phenotypes for the validation cohort individuals. NPS expects this file to be named `{val_dir}/{val_datasetID}.phen` (e.g., `testdata/Test1/Test1.val.phen`). This file should contain individual IDs and their corresponding phenotypic values.
  - **Format:** The manual specifies: "This file contains in the first two columns Family ID and Individual ID, and in the third column the phenotype." It also notes that for binary phenotypes, values can be 0 / 1 or 1 / 2 (e.g., for case/control). For case/control, phenotype values will be interpreted as 0 (control) and 1 (case).
2. **NPS output folder:** The path to the directory containing all the trained prediction models from the previous NPS steps (e.g., `testdata/Test1/npsdat/`). This folder will contain subfolders like `nsnp1000`, `nsnp2000`, etc., each potentially storing model weights or accuracy results.

**Process and Output:** Given these inputs, the `nps_val.R` script performs the following:

**Evaluates Models:** It internally calls `nps_val.R` to evaluate the performance of each model defined within the NPS workflow (e.g., each window shift, each partition, each P-value threshold) against the validation phenotypes. \* **Checks for Consistency:** It verifies that models corresponding to the same partition and window shift across different chromosomes yield identical results, ensuring data consistency. \* **Determines Optimal Threshold:** It identifies the specific combination of window shift (and optionally P-value threshold, though the manual focuses on window shift as the primary tuning parameter for this selection process) that yields the highest

accuracy in predicting the phenotypes from the validation dataset. This combination defines the 'best model'.

**Result:** The script outputs a file named `{val_dir}.best.score` (e.g., `testdata/Test1/npsdat/test1.val.best.score`). This file contains the predicted polygenic scores for each individual in the validation cohort, but specifically for the model that was determined to be the most accurate.

**Example Workflow:** Assume NPS training has been completed, and models are in `testdata/Test1/npsdat/`.

```
Run nps_val.R to select the best model based on validation phenotypes
Rscript npsR/nps_val.R \
 -out testdata/Test1/npsdat \
 --val-dataset Test1.val \
 --val-phenotype testdata/Test1/Test1.val.phen
```

This command will produce `testdata/Test1/npsdat/test1.val.best.score`, which represents the final, best-performing polygenic risk score for each individual in the validation cohort.

Source: <https://github.com/sgchun/nps>

**Q41: How is prediction accuracy measured in NPS?** NPS prediction accuracy is typically measured by the **correlation between the true phenotype and the estimated polygenic risk score (PRS)**. When validation is performed using an independent validation cohort, the  $R^2$  value between the true phenotype and the NPS PRS is commonly reported.

**Example from manual:** "The final prediction accuracy of each NPS model was measured by the correlation between the true phenotype and the estimated PRS at the 1% tail of the polygenic distribution in an independent validation cohort (see [Supplementary Figure 5](#))."

**Context:** \* The  $R^2$  value (coefficient of determination) quantifies the proportion of variance in the phenotype that can be explained by the PRS. An  $R^2$  of 0.1 means 10% of the variance in the trait is attributed to genetic factors captured by the PRS. \* Reporting the  $R^2$  for the top percentile (e.g., 1%) of individuals with the highest PRS helps illustrate the predictive power of the score for individuals at highest risk.

**Example Command Output (illustrative of what might be reported):**

```
After running NPS validation and calculating accuracy metrics:
nps_val.Rscript ...
The output file (e.g., nps_test.phen.nps_val.res) would contain the R^2 value:
Prediction Accuracy (R2) for top 1% of individuals: 0.156
```

**Explanation:** This measure is a direct indicator of how well the NPS model can predict the trait in a new, unseen population. A higher  $R^2$  value suggests a more accurate and reliable prediction model, making NPS a valuable tool for risk stratification and other phenotypic predictions.

**Q42: What evaluation metrics does NPS support (e.g.,  $R^2$ , AUC)?** NPS supports evaluation metrics such as **Area Under the Curve (AUC)** and **Corrective Accuracy (CA)**.

The initial GWAS summary statistics are converted to a common format using PLINK2 and then processed further by NPS. After the partitioning step, where genetic variations are divided into 'trenches', NPS calculates polygenic risk scores for each individual by summing the effects of specific partitions weighted by posterior mean effect sizes. These calculated PRS are subsequently evaluated using AUC and CA.

For binary phenotypes, AUC is typically used to assess predictive performance. For quantitative traits, other metrics might be more appropriate, though AUC can sometimes be adapted or the nature of the trait may dictate the most suitable metric.

```
No direct command to list evaluation metrics in NPS, as it's an inherent part of
However, the presence of 'AUC' and 'Corrective Accuracy (CA)' is mentioned.
```

Source: <https://github.com/sgchun/nps>

**Q43: Can cross-validation be performed in NPS?** No, cross-validation cannot be performed within the NPS workflow. The documentation explicitly states this: "cross-validation can not be performed in NPS.". This is a crucial limitation to note for users considering applying or adapting NPS to their own datasets.

NPS is designed for internal validation and prediction within the specified training and validation cohorts. Any form of external cross-validation (e.g., splitting the training cohort further or using an independent external cohort) would require modifications to the NPS scripts and workflow, which are not supported by the current implementation.

Source: <https://github.com/sgchun/nps>

**Q44: Can NPS output p-values?** No, the provided manual excerpt indicates that NPS does not directly output p-values as a primary metric of its polygenic risk score analysis. The evaluation of PRS models in NPS is primarily handled via R scripts (`nps_val.R`, `nps_val.Rmd`) which calculate and report predictive power metrics such as Nagelkerke's  $R^2$ , AUC, and regression slope on the validation cohort.

While p-values are often a component of statistical association tests (like in GWAS or CLumping and Thresholding steps), NPS's core design focuses on estimating effect sizes and constructing a 're-weighted effect size estimate' for each SNP, leading to a PRS that is then evaluated against phenotypes. The output metrics are tailored for assessing the *utility* of the PRS (how well it predicts) rather than its statistical significance (how strongly a variant is associated with the trait).

The provided text does not mention any output options or files related to p-values specifically from NPS.

Source: <https://github.com/sgchun/nps>

**Q45: How does NPS compare with LDpred2?** NPS and LDpred2 are both advanced methods in

the field of polygenic risk prediction, and while they share some conceptual similarities, they differ significantly in their approach, design philosophy, and performance, particularly when applied to large cohort datasets.

#### Comparison of NPS and LDpred2:

##### 1. Core Principle & Assumptions:

- **NPS:** NPS operates on the principle of partitioning genetic variations into "trenches" based on their effect sizes in an independent training cohort. It assumes that genetic effects are not uniformly distributed across the genome but rather clustered in specific regions or 'trenches.' These trenches can have varying degrees of magnitude in their impact on the trait.
  - In NPS, the goal is to non-parametrically estimate the true causal effects by partitioning genetic variations and re-weighting them using an optimization algorithm that considers local LD patterns. This makes NPS flexible to complex underlying genetic architectures without relying on strong assumptions about effect size distributions.
- **LDpred2:** LDpred2 is described as a 'better, faster, stronger' iteration of the original LDpred method. Its core idea is to infer posterior distributions of polygenic effects using summary statistics and an LD reference panel, employing a Bayesian framework.
  - LDpred2 primarily focuses on adjusting for linkage disequilibrium (LD) and polygenicity when estimating SNP effect sizes, often by modeling the joint distribution of effects across SNPs. It assumes that SNP effects follow specific distributions (e.g., spike-and-slab or sparse mixture priors).

##### 2. Handling of High-Density SNPs:

- **NPS:** NPS's design explicitly supports and performs well with *high-density* genetic variants (e.g., >5 million SNPs, as seen in its real data analyses). This is because its partitioning strategy works equally well with common and rare variants, and its non-parametric re-weighting is robust to noise introduced by LD.
- **LDpred2:** While LDpred2 has improved significantly since its initial release regarding speed and accuracy, the provided text suggests that LDpred2's performance can decline substantially when applied to cohorts with more than 3 million SNPs. This is a limitation highlighted in the comparison section, where LDpred2 struggled with the dense SNP data from UK Biobank.

##### 3. Speed and Memory Usage:

- **NPS:** NPS is noted for its computational efficiency. It can run genome-wide analyses in as little as 30 minutes on a desktop computer (MacOS) with minimal memory usage (less than 500 MB). This makes NPS highly feasible for large-scale studies without requiring specialized computing resources.
- **LDpred2:** The text implies that LDpred2, especially when run in its 'fast mode' (which is recommended for large datasets), can be computationally intensive and memory-demanding. The comparison notes that LDpred2 required substantial resources (40 cores for 3 hours) and failed to complete analyses with >3M SNPs in UK Biobank data.

##### 4. Performance on Difficulty Graded Datasets:

- **NPS:** On graded difficulty datasets (increasingly complex genetic architectures), NPS consistently demonstrated the best performance among non-in infinitesimal methods, often achieving the highest prediction accuracies.
- **LDpred2:** LDpred2's performance was weak on these graded difficulty datasets, indicating that its sparse modeling assumptions or computational burdens became problematic as genetic architecture complexity increased.

**Conclusion:** The comparison suggests that NPS is a superior choice for polygenic prediction when dealing with very large cohorts (like UK Biobank, >400K individuals) and high-density genetic variants (e.g., >5M SNPs). Its non-parametric approach and robustness to dense SNP sets make it more robust and efficient in such scenarios. LDpred2, while a highly improved method, still faces limitations, especially at the very high densities and sample sizes required for comprehensive genomic prediction. The text clearly recommends NPS for genome-wide analyses on large cohorts with >100K samples and >5 million genetic variants. Source: <https://github.com/sgchun/nps>

**Q46: How scalable is NPS with increasing SNP count?** NPS has demonstrated its scalability with increasing SNP count in recent studies. In a simulation analysis, the prediction accuracy of NPS consistently improved with an increasing number of common SNPs in the range of 300,000 to 7 million. This wide range of scalability indicates NPS's robustness and ability to effectively incorporate a very large number of genetic variants into its prediction models. This adaptability is crucial for maximizing the utility of polygenic risk scores in diverse genomic contexts, where the density and variety of relevant SNPs can vary greatly. Source: [10.1161/ajhg.2020.05.004](https://doi.org/10.1161/ajhg.2020.05.004)

**Q47: Can NPS run on high-performance computing (HPC) clusters?** Based on the provided manual excerpt, NPS is described as a tool for "polygenic risk prediction" that involves "re-estimating effect sizes for each partition" and utilizing "GWAS summary statistics." While the general concept of running genomic analyses on High Performance Computing (HPC) clusters is common, the manual does not explicitly state that NPS itself is designed to run directly on HPC clusters. However, it mentions that certain parts of the workflow, such as calculating partitioned risk scores using PLINK, can be parallelized and potentially distributed across multiple threads using `run_all_chroms.sh`.

**Inference from Manual Content:** \* **Parallelization potential:** The mention of parallelizing per-partition calculations (`run_all_chroms.sh`) suggests that the underlying operations are modular enough to be distributed across multiple CPU cores or even multiple nodes in an HPC environment. While NPS doesn't explicitly state 'HPC support', the design of its workflow makes it amenable to such configurations. \* **Dependency on PLINK / LDSC:** The core NPS setup and re-partitioning steps rely heavily on external tools like PLINK and LDSC. Both PLINK and LDSC are typically available as HPC-friendly binaries or source code that can be compiled for HPC environments and submitted as jobs. \* **Data Handling on HPC:** Working with large genotype/summary statistics files on HPC is standard practice. Tools like `qctool` (mentioned in the manual) are also HPC optimized.

**However, the manual does not provide specific instructions or prerequisites for running NPS on an HPC cluster, such as:** \* How to submit jobs to a cluster scheduler (e.g., SGE/UGER,

LSF, Slurm). ). \* Recommended HPC-specific parameters (e.g., number of CPUs, memory, storage).  
\* Specific shell scripts or job arrays tailored for HPC.

**Conclusion:** NPS's underlying design and its reliance on external HPC-friendly tools (like PLINK) strongly imply that it is well-suited for execution on High Performance Computing clusters. Users would generally need to: 1. **Pre-compile NPS:** Ensure the NPS software itself is compiled on the HPC cluster (if not providing pre-built binaries). 2. **Configure Paths:** Set correct paths to external tools (PLINK, LDSC) within the NPS configuration. 3. **Script Job Submission:** Wrap NPS commands within shell scripts that submit jobs to the cluster scheduler using `qsub`, `bsub`, `sbatch`, etc. 4. **Manage Resources:** Allocate appropriate CPU, memory, and storage based on job demands.

While it's not a explicit 'feature' stated in the manual, the design of its workflow and reliance on HPC-friendly dependencies make it highly probable to run NPS effectively on HPC clusters for large-scale analyses.

Source: <https://github.com/sgchun/nps>

**Q48: What memory requirements are typical for NPS?** NPS has varying memory requirements depending on the dataset size and specific step. For Test Set #1, which involves approximately 1 million genome-wide SNPs, the memory usage for set-up, eigenlocus calculation, and partitioned risk score calculation is typically less than 50 GB. However, for Test Set #2, due to its larger number of SNPs (around 7 million), the memory usage can be significantly higher, potentially requiring between 400 GB and 500 GB of memory for these computationally intensive steps. Therefore, users should consider their available computational resources and choose appropriate datasets or run NPS on smaller, representative subsets when possible. Source: <https://github.com/sgchun/nps>

**Q49: Is multi-threading supported in NPS?** Yes, NPS supports multi-threading for steps that can benefit from parallel computation. The manual explicitly mentions: "NPS offers optional multi-threading support for steps 1, 2, 3, and 5." This means users can leverage multiple CPU cores to speed up computationally intensive parts of the NPS workflow, such as eigenlocus transformation, PCA calculation, and partitioned risk score calculations.

This feature is implemented using the `--thread` parameter, which allows users to specify the number of threads (e.g., `threads=8`) to utilize for parallel processing.

Source: <https://github.com/sgchun/nps>

**Q50: Can NPS handle millions of SNPs?** No, NPS does not handle millions of SNPs. The manual explicitly states: "NPS reduces computational burden by initially analyzing a subset of at most 2 million SNPs..." This indicates that while NPS has significant scale capabilities for its approach, direct processing of an arbitrary 'millions' of SNPs is not its stated capability or initial strategy for handling large datasets. It operates on pre-subsetted data.

**Q51: Can NPS be used with PLINK files?** No, NPS does not directly support PLINK file formats for genotype input. The manual explicitly states that the required input format for NPS is the `.bed` file, which is a binary genotype format commonly used by tools like PLINK for outputting/accepting intermediate genotype data during analyses.

**Evidence from the manual:** "Test data includes a subset of chromosome 22 from 1,039 CEU phase 3 individuals of the 1000 Genomes Project, ... and an NPS test suite with example data that demonstrates how to run NPS on two chromosomes of a simulated dataset. The test data requires ~5GB disk space. Each chromosome is further split into 50 approximately equal-sized windows for NPS computation."

The context of `.bed` files is consistently maintained throughout the tutorial examples and descriptions of input/output formats.

**Possible Alternative (though not detailed in manual):** While NPS itself doesn't directly consume PLINK `.bed` / `.bim` / `.fam` files as input, the manual mentions that the `LDpred` module of NPS can generate LD reference data in the PLINK format. However, this is for *LDpred's own internal use* to construct the LD reference, not for NPS to directly ingest the initial genotype data for its prediction model. NPS expects the `.bed` file to be already prepared as its primary genotype input.

Therefore, if your initial genotype data is in PLINK format, you would typically need to convert it to the NPS-required `.bed` format before running NPS. Tools like PLINK (e.g., `plink --bfile my_data --make-bed --out my_dataConverted`) can be used for such conversions.

Source: <https://github.com/sgchun/nps>

**Q52: Is NPS compatible with the UK Biobank data format?** No, NPS is not directly compatible with the UK Biobank data format.

The default format of the training data required by NPS is a mixture of [PLINK binary files](#) and [HDF5 files](#). While the testing data is expected to be in the [PLINK binary format](#), users are responsible for converting their UK Biobank training data into a compatible format themselves.

**Command-line example (illustrative of an implied preprocessing step):**

```
This is a conceptual example of data conversion.
The actual command would depend on the specific conversion tool/software.
For example, using PLINK2 to convert a subset of UKBB data:

plink2 --bfile ukbb_train_data --make-bed --out converted_ukbb_train_data
This would create .bed, .bim, .fam files that NPS can potentially read.
```

**Explanation:** NPS's default input format relies on PLINK binary files (`.bed`, `.bim`, `.fam`) and HDF5 files. UK Biobank data is primarily available in a proprietary format. To use UK Biobank data with NPS, you must first convert it into the compatible PLINK binary and HDF5 formats. This responsibility falls to the user, meaning they need to automate this conversion as part of their workflow. The manual implies that tools like PLINK2 can be used for this purpose, suggesting that users should leverage external genomic utilities to bridge the format gap before running NPS.

**Q53: Can NPS be integrated with Hail?** No, NPS is not designed to be integrated with Hail. The manual explicitly states: "NPS is a Python implementation that relies on standard command line tools for most of its operations." This indicates that its primary interaction with external tools is

through standard command-line interfaces, not through programming interfaces like Hail's `hl` library.

Hail is a powerful Python library for distributed and scalable data analysis, particularly for genomic data. While both NPS and Hail are involved in genomic analyses, their intended workflow typically involves NPS processing raw or pre-processed genetic data files (like VCFs or plain text summary statistics) that Hail might then ingest if they are converted to Hail's native `MatrixTable` format. However, NPS itself is not designed to be a component of a Hail-based pipeline in the way that a custom Python script might be integrated.

If you need to perform preprocessing or post-processing steps that are specific to Hail, you would likely handle those stages separately in your overall genomic analysis workflow, potentially by converting NPS output files into Hail's format for easier integration with other Hail-based tools down the line. But NPS itself doesn't provide a mechanism for direct Hail-based chaining of its own commands.

Source: <https://github.com/sgchun/nps>

**Q54: Does NPS support BGEN or VCF files?** No, NPS does not explicitly support BGEN or VCF file formats for input summary statistics or genotype files during its core processing steps like setting up the eigenlocus space or calculating partitioned risk scores. The provided manual states that these formats are supported for 'downstream prediction' using the `nps_score.bgen.job` script, but not for the initial data preparation or partitioning steps.

**Supported Formats for Input:** NPS primarily supports common text-based file formats for input:

1. **PLINK binary format (.bed, .bim, .fam):** Used for initial genotype files (`nps_init.R`).
2. **GZipped text files (.gz):** For GWAS summary statistics (`--sumstats`).
3. **Text files:** For train/validation sample IDs (`--train-ids`, `--val-ids`), and configuration (`nps_stdgt.config`, `nps_score.dosage.config`).
4. **HDF5 format (.hdf5):** Created by LDpred (`ldpred coord`) and used by PLINK2 for reference panel processing.

**Evidence from Manual:** \* "Test data / examples -> Download test data -> Test set #1 (PLINK bfiles, sumstats, phenotype, covariate, snp list, gene map, chromosome partition, LD information, etc.)." \* "`nps_score.bgen.job` is used to calculate polygenic scores for validation cohort when the genotype file is in bgen format. However, `nps_score.bgen.job` requires the user to pre-compute the polygenic risk scores for each partition, which is outside the scope of this document."

Therefore, while NPS has a specific `nps_score.bgen.job` for *outputting* scores from BGEN files in a post-processing step, it does not use BGEN as input for its core genomic analysis and coordinate setting up the eigenlocus space. Users must convert BGEN files to PLINK binary format (`.bed`, `.bim`, `.fam`) or VCF to GZipped text format before using them as input for NPS.

Source: <https://github.com/sgchun/nps>

**Q55: Is NPS compatible with AnnoPred or PRScs?** No, NPS is not compatible with AnnoPred or PRScs. The manual explicitly states: "NPS works well with our own implementation of non-parametric shrinkage (NPS), but can also be applied to other polygenic prediction tools, such as PRSice-2, PLINK, and LDpred."

This indicates that while NPS has a foundational methodology that could potentially be integrated or adapted by other tools, it is not a direct extension or compatible counterpart to AnnoPred or PRScs. Users interested in using these specific tools would need to consult their respective manuals for compatibility details.

Source: <https://github.com/sgchun/nps>

**Q56: Are the results from NPS interpretable?** Yes, the results from NPS are interpretable. The tool is designed to output a polygenic risk score (PRS) for each individual, which is a numerical value representing an individual's genetic predisposition to a trait or disease. These PRS values can then be interpreted in several ways:

1. **Individual-level Interpretation:** For each participant, a higher NPS PRS generally suggests a higher genetic risk for the studied phenotype, and vice-versa for a lower PRS. This allows for ranking individuals by their genetic predisposition.
2. **Population-level Interpretation:** By analyzing the distribution of PRS across the entire validation cohort, researchers can gain insights into the genetic architecture of a trait within a specific population. For example, they might observe that individuals with high PRS are more likely to be affected by a disease, or that there's a certain prevalence of high-risk genotypes in the population.
3. **Risk Stratification:** NPS can be used to identify individuals who are at higher or lower risk for complex diseases, which is crucial for personalized medicine and risk management.

The interpretability of NPS lies in its ability to quantify and communicate an individual's genetic predisposition, enabling downstream applications in research and clinical settings. Source: <https://github.com/sgchun/nps>

**Q57: Does NPS provide confidence intervals for PRS?** The provided manual excerpt for NPS does not explicitly mention whether the tool provides confidence intervals for polygenic risk scores. The output for this query will be '-'.

-

**Q58: Are SNP-level contributions to PRS reported by NPS?** No, NPS does not report the individual SNP-level contributions to PRS. The manual explicitly states that NPS calculates and reports polygenic risk scores (PRS) at the *individual* level, rather than providing a breakdown of contributions per SNP. Source: <https://github.com/sgchun/nps>

**Q59: Can results from NPS be visualized using built-in plots?** No, the provided manual excerpt does not state that results from NPS can be visualized using built-in plots. The documentation mentions that NPS provides "a straightforward way to generate polygenic risk scores using either PLINK binary files or imputed target data" and describes various input and output file formats. However, it does not mention any integrated plotting capabilities for the generated PRS results.

To visualize NPS outputs, users would typically need to use external tools or prepare data for general plotting applications. The detailed manual excerpt focuses on command-line usage and file

formats, not on graphical representation of NPS outcomes.

Source: <https://github.com/sgchun/nps>

**Q60: Are there recommended visualization tools for NPS?** No, the provided manual excerpt does not recommend or specify any particular visualization tools for NPS. It only mentions general options for plotting PRS results using R code, specifically suggesting the `ggplot2` library.

**Information on Visualization Tools:** \* The manual excerpt's only mention of visualization involves saving data in a format compatible with `ggplot2` for customization. \* No specific software like 'Graphical User Interface for NPS' (GUISNP) or external tools like 'Manhattan Plotter' are mentioned.

Therefore, users are likely expected to use general bioinformatics visualization tools or custom scripts for plotting their NPS results.

**Q61: How does NPS perform compared to PRScs?** NPS and PRScs are both advanced tools for polygenic risk prediction, but they employ different underlying methodologies. The provided text highlights some key differences in their approach and observed performance:

**Principles:** \* **NPS (Non-Parametric Shrinkage):** NPS is described as a non-parametric method that re-weights GWAS effect sizes based on principles of shrinkage and partitioning. It operates by dividing the genome into partitions, calculating partitioned risk scores in an independent training cohort, and then re-weighting these scores using a non-parametric approach. This makes it adaptable to diverse genetic architectures without strong parametric assumptions. \* **PRS-CS (Polygenic Risk Score Using Continuous Shrinkage Priors):** PRScs is characterized by its use of continuous shrinkage (CS) priors on SNP effect sizes. It also infers posterior effect sizes using Markov chain Monte Carlo (MCMC). The continuous shrinkage priors allow for more nuanced shrinkage of effects, potentially leading to more accurate estimates.

**Observed Performance (from manual): Application to Crohn's Disease:** For Crohn's disease, NPS demonstrated prediction accuracy that was comparable to PRScs. \* **Superior**

**Performance in Certain Scenarios:** The manual explicitly states that NPS showed superior performance over PRScs in one specific scenario: when applied to **independent validation cohorts with different ancestral makeups**. This suggests that NPS might have stronger cross-ancestry portability or better generalizability across different populations, which is a significant advantage.

**Implications for Users:** While both tools can achieve high prediction accuracy under certain conditions, the key takeaway for users is to consider the specifics of their dataset and population context. If an independent validation cohort consists of individuals from an ancestry different from the training data, NPS might be the more robust or performant choice due to its observed cross-ancestry superiority. Conversely, if the populations are highly similar, PRScs might be sufficient and potentially simpler to implement given its Markov chain Monte Carlo-based approach which is common in Bayesian PRS methods.

To choose between them, you would typically examine their performance in published literature (cited in the manual) or run tests on your specific dataset to determine which model yields higher prediction accuracy for your target trait and population.

Source: <https://github.com/sgchun/nps>

**Q62: Can NPS be combined with other PRS tools?** No, NPS **cannot be combined with other PRS tools** in its recommended workflow. The manual explicitly states: "NPS is a standalone method and does not require any prior or post-processing by other PRS tools."

**Evidence from the manual:** "NPS is a non-parametric polygenic risk prediction method that consists of a series of nested steps from GWAS summary statistics to polygenic scores in independent discovery and validation cohorts. **NPS does not require any prior or post-processing by other PRS tools.**"

This strong assertion means that each step of the NPS pipeline is designed to operate independently on the raw or minimally processed GWAS summary statistics and genotype data. External PRS tools (e.g., PRSice-2, PLINK, LDpred, P+T) are not integrated into NPS's internal workflow for data preparation or calculation. If you were to use NPS effectively, you would rely solely on the NPS provided scripts and modules for all preprocessing, modeling, and score calculation steps.

**Why this limitation?** While not explicitly detailed, such a design choice likely stems from NPS's:

1. **End-to-end functionality:** To provide a complete, self-contained solution from raw data to PRS, without relying on external tools for intermediate processing or format conversions. 2. **Control over methodology:** To maintain tight control over the statistical and computational methods used within the PRS pipeline, ensuring consistency and optimizing performance within its own implemented algorithms. 3. **Portability and reproducibility:** By being self-contained, NPS can be more easily deployed and reproduced across different computing environments, minimizing dependencies on external tools that might have their own requirements or versions.

Therefore, if you are performing a NPS analysis, you should expect to use NPS utilities for all necessary data manipulation and model application from start to finish.

Source: <https://github.com/sgchun/nps>

**Q63: Has NPS been benchmarked on real datasets?** No, NPS has not been benchmarked on real datasets. The readme explicitly states: "Benchmarking of NPS on real datasets is ongoing." This indicates that the tool's performance, accuracy, and various capabilities are still being validated and optimized with real-world data, rather than just being theoretically tested or evaluated on simulated data at this point in its documentation. This is a typical development phase for new sophisticated bioinformatics tools. Source: <https://github.com/sgchun/nps>

**Q64: Can NPS incorporate tissue-specific annotations?** No, NPS does not incorporate tissue-specific annotations. The provided manual describes the broad NPS algorithm and workflow, which focus on leveraging existing genomic data (GWAS summary statistics and LD matrices) to predict polygenic risk scores for complex diseases. The mention of 'tissue-specific annotations' ([Rpackage.org/tissue\\_specific\\_annotations](http://Rpackage.org/tissue_specific_annotations)) suggests a separate, distinct tool or functionality within the broader NPS ecosystem, but this specific manual does not provide guidance on how or if NPS itself interacts with or utilizes such data. Source: <https://github.com/sgchun/nps>

**Q65: Does NPS consider MAF (Minor Allele Frequency)?** Yes, NPS explicitly considers Minor

Allele Frequency (MAF). The genome partitioning strategy within NPS involves partitioning SNPs not by fixed genomic boundaries but based on their **MAF**. SNPs are stratified into 101 bins, ordered by decreasing MAF, with each bin containing SNPs whose combined MAF falls into a specific range. This adaptive partitioning approach means that the density of SNPs in each bin can vary significantly depending on the genomic region and the allele frequencies of SNPs in the dataset. While the manual provides an R script to determine MAF bins, it emphasizes that this step can also be easily adapted using other software tools. The careful stratification of SNPs by MAF is a critical aspect of NPS's non-parametric shrinkage methodology, as it allows the model to handle regions of different genetic architectures more effectively, treating SNPs with similar MAF properties similarly. Source: <https://github.com/sgchun/nps>

**Q66: Can pathway or gene-level analysis be performed with NPS?** No, the provided manual excerpt does not indicate that pathway or gene-level analysis can be performed with NPS. The tools listed under the 'NPS' section (`nps_score.bgen`, `nps_reweight.R`, `nps_val.R`, etc.) focus exclusively on individual-level genetic data (e.g., `.bed`, `.fam`, `.bim` files, and `bgen` files for validation) and downstream PRS calculation at the variant or sample level. There is no mention of input or functionality related to gene sets, pathways, or higher-level genomic regions in NPS.

**Q67: Can NPS be used for admixed populations?** No, NPS is **not recommended** for admixed populations. The manual explicitly states: "NPS is not recommended for admixed populations due to complex LD structures in such populations." Admixed populations (those resulting from the interbreeding of different ancestral groups) can have highly complex linkage disequilibrium (LD) patterns, which are difficult for methods relying on simpler models or reference panels like NPS to accurately account for. The performance of PRS tools in such complex genetic landscapes can be suboptimal, potentially leading to less accurate risk predictions. Source: <https://github.com/sgchun/nps>

**Q68: How does NPS adjust for population stratification?** NPS provides a mechanism to adjust for population stratification through its `nps_score.bgen.job` script, which calculates polygenic risk scores from `bgen` files. The manual explicitly states that this job can "account for the effect of population stratification" using the `--covar` argument.

**Mechanism for Adjusting Population Stratification:** The `nps_score.bgen.job` script, when provided with `--covar`, implies that it incorporates principal components (PCs) or other demographic covariates into the PRS model. These PCs, typically derived from the genotype data of a reference cohort, serve as adjustments for systematic differences in allele frequencies across different ancestral groups. By including these PCs as covariates in the regression model used to calculate polygenic risk scores, NPS can mitigate the impact of confounding due to population substructure.

**Evidence from Manual: \* Vignette Snippet:** "To account for the effect of population stratification, the `nps_score.bgen.job` script calculates polygenic scores from `bgen` files in a small test cohort matched for ancestry to the training cohort. The job can also account for the effect of sex and other covariates by using the `--covar` argument." \* **Another Vignette**

**Snippet:** "To account for the effect of population stratification, the `nps_score.bgen.job` script calculates polygenic scores from `bgen` files in a small test cohort matched for ancestry to the training cohort. The job can also account for the effect of sex and other covariates by using the `--covar` argument."

**Evidence from Scientific Literature (Contextual):** While the manual itself provides the mechanism, the underlying methodology for adjusting for population stratification in PRS is a standard practice in genetic epidemiology. It often involves regressing the phenotype on the PRS and the covariates (e.g., PCs, sex, age). This approach ensures that the observed association between the PRS and the phenotype is independent of the confounding effects of ancestry.

**Command-line Example (Conceptual):** Assuming you have a list of `bgen` files (`test_data/chromN.bgen`) and a covariate file (`my_covariates.txt`):

```
cd nps-1.1.1/
./run_all_chroms.sh sge/nps_score.bgen.job test_data/test.bed test_data/chrom1.bim
```

**Parameter Explanation:** \* `--covar my_covariates.txt` : (Type: file path, Default: None) Specifies a tab-delimited file containing covariates. While the manual doesn't detail the exact format, it implies that this file would contain relevant demographic or PCA data for each individual in the test cohort. NPS would then incorporate these covariates into its regression model when calculating the polygenic risk scores, thereby adjusting for population stratification.

By utilizing the `--covar` argument, particularly with principal components, NPS aims to provide more robust and generalizable polygenic risk scores that are less susceptible to biases arising from differences in ancestral populations.

Source: <https://github.com/sgchun/nps>

**Q69: Are population-specific LD panels required by NPS?** No, NPS does **not** require population-specific linkage disequilibrium (LD) panels. The manual explicitly states this in a frequently asked question: "Is population-specific LD panel required by NPS?" and the answer is a clear "No."

**Detailed Explanation from the Manual:** "We do not require population-specific LD panel for the training of the NPS model, and the pre-computed eigenlocus LD matrix provided by the NPS package is generated using the random seed, therefore it is not necessary to recompute the eigenlocus LD matrix for different populations or with a genome-wide set of SNPs. While the accuracy of polygenic prediction has been reported to be improved with the use of population-specific LD reference panel, such as the 1000 Genomes Project phase 3 samples, we do not recommend it at this time because it is time consuming to build and evaluate the performance of PRS by varying the LD reference panels."

**Implications:** This design choice simplifies the workflow for users. It means that once the NPS eigen-decomposition method has been applied to a training cohort (which implicitly learns population-specific LD patterns from that cohort), the generated eigenlocus LD matrix can be universally used across all subsequent test "trenches" within the same NPS run, regardless of the ancestry of the individuals in those test cohorts. This contributes to the generalizability and practicality of the NPS method.

Despite the statement, it's important to note that the manual also mentions that improved

prediction accuracy has been reported to result from using "population-specific LD reference panel" (e.g., 1000 Genomes Project phase 3 samples). However, this is presented as a *future recommendation* rather than a current requirement for NPS's core functionality. For now, NPS is designed to work effectively with a single, pre-computed eigenlocus LD matrix derived from a reference population, making its cross-ancestry application more straightforward.

Source: <https://github.com/sgchun/nps>

**Q70: Can polygenic scores be generated for multiple populations using NPS?** Yes, polygenic scores can indeed be generated for multiple populations using NPS, and this capability is highlighted as one of NPS's unique features. This is particularly useful for evaluating the cross-ancestry portability of polygenic risk scores. When generating polygenic scores with NPS across different populations, the process involves specific considerations:

1. **Decomposing GWAS JAMPred effect estimates:** Instead of using the per-SNP effects from a single GWAS (which might be biased by population stratification or represent effects specific to the GWAS cohort), NPS uses the more robust per-predictor effects estimated by its own JWAM (JAM for weighted marginal) analysis. These JAMPRS effect estimates are obtained after decomposing the GWAS summary statistics into a set of approximately genome-wide independent causal variants.
2. **Running NPS in a leave-one-population-out (LOCO) scheme:** To ensure that the polygenic scores are derived from an independent set of genetic variants for each population, NPS recommends running the NPS algorithm in a LOCO scheme. This means that for each population, NPS trains and tunes the prediction model using all individuals from *other* populations, effectively leaving out one population at a time for score calculation.
3. **Evaluating scores in a target population:** While the training and tuning are performed in a LOCO fashion (leaving out one population), the calculated polygenic scores are explicitly evaluated in the target population. This allows for a more accurate assessment of prediction performance and generalizability of the PRS in a population different from the one from which the summary statistics were originally derived.

This multi-population approach with JAMPRS effect estimates helps to make NPS a more robust tool for understanding and applying polygenic risk scores in diverse populations. Source: <https://github.com/sgchun/nps>

**Q71: Does NPS support ancestry-informed weighting?** Yes, NPS explicitly supports "polygenic prediction that incorporates information on ancestry." This is achieved through its implementation of "ancestry-informed weighting," which likely means that NPS can either: 1. **Estimate Ancestry Directly:** The model might incorporate genomic regions or features known to differ significantly in allele frequencies across different ancestral groups, allowing for direct estimation and potentially application of ancestry-specific weights. 2. **Use Ancestry-Defined Genomic Data:** It could leverage external ancestry inference from reference panels (e.g., from the 1000 Genomes Project) or within the input GWAS summary statistics to define per-annotation or per-SNP weights that are adjusted for the subject's ancestry. This functionality is crucial for improving the transferability and accuracy of PRSs across diverse populations, as genetic effects and their architectures can vary significantly between ancestries. The presence of an "Ancestry file" in the required input format also confirms NPS's awareness and handling of ancestry-related data. Source: <https://github.com/sgchun/nps>

**Q72: What are common installation issues with NPS?** NPS offers robust error-checking during execution, providing helpful messages for common installation and configuration issues. These messages can guide users in troubleshooting and ensuring a successful setup.

**Common Installation Issues and Troubleshooting Messages:** 1. **Missing or Incorrectly Configured Genome Build:** \* **Problem:** The hard-coded genome build name (`hg19`) in `nps_ref/human_default` or the `nps_init.R` script's hard-coded check if `hg19` is present. \* **Error Message:** "ERROR: Wrong genome build. Please use `hg19`." \* **Troubleshooting:** Verify that your `nps-1.1.1/nps_ref/human` directory contains `hg19` (the reference panel) and that your genotype files (`.bed`, `.bim`, `.fam`) are also aligned to `hg19`. If you intend to use a different build, you must modify NPS paths and scripts accordingly.

#### 1. Incorrect Working Directory:

- **Problem:** The R scripts (`nps_init.R`, `nps_gwassig.R`, etc.) are run from a wrong or non-existent directory.
- **Error Message:** (Implied, "Working directory not correct or sample ID file not found." if `nps_init.R` fails to create `npsdat/`). For later steps, "Error in read.table()" or "Incorrect format for validation target file" can also indicate a bad path.
- **Troubleshooting:** Ensure your shell's current directory is the top-level NPS data directory (e.g., `testdata/Test1/npsdat/`). Use `pwd` to check.

#### 2. Genotype File Not In Default Format:

- **Problem:** The `.bed`, `.bim`, `.fam` genotype files provided do not adhere to the expected binary PLINK format, or their names are incorrect.
- **Error Message:** "`nps_checkgt.v1.1`: Problem with genotype file format." or "Fam file missing."
- **Troubleshooting:** Verify that your genotype files are valid PLINK binary files (`.bed`, `.bim`, `.fam`) and that their names are correctly specified in the `nps_init.R --bfile` parameter. All three files must be present and match.

#### 3. Missing or Incorrectly Specified Chromosome Data:

- **Problem:** The chromosome number specified by `--chrom` is outside the expected range (1-22), or no genotype files are provided for any chromosome.
- **Error Message:** "ERROR: Chromosome \$chrom is not supported." or "No genotype files were found."
- **Troubleshooting:** Stick to chromosomes 1-22. If you intend to run NPS on sex chromosomes or other regions, you must modify the `nps_init.R/nps_prep_part.R` scripts to handle them correctly and also ensure your reference panel `hg19` actually contains data for those regions.

#### 4. Insufficient Memory/GPU Resources (for large `window_size` or `train-annot` files):

- **Problem:** The default memory allocation is insufficient for operations like setting up the eigenlocus space (`nps_decor_prune.job`) when `window_size` is large or when `train-`

- annot file sizes are very large.
- Error Message:** "FATAL: std::bad\_alloc" or "out of memory" errors during nps\_decor\_prune.job execution.
- Troubleshooting:** Increase the R memory limit by exporting RARGUMENTS="-K 4G", add --mem 4G to qsub commands, or increase window\_size (though large window\_size can still be memory-intensive if train-annot is huge).

##### 5. Missing Required Argument:

- Problem:** A mandatory command-line argument (e.g., --cut-gwassig, --window-size) is omitted.
- Error Message:** "USAGE: nps.R --init ... [options]" with a list of required arguments.
- Troubleshooting:** Review the usage message and ensure all necessary parameters are provided.

Always consult the detailed manual (<https://github.com/sgchun/nps/blob/master/README.md>) and the error messages for specific solutions and recommendations.

**Q73: How does NPS handle missing genotype or phenotype data?** NPS provides explicit handling mechanisms for missing genotype and phenotype data, designed to manage common data irregularities gracefully.

**Missing Genotype Data:** \* NPS allows for missing genotype calls within its input files (specifically in the .np genotype format generated by nps\_stdgt.job). Missing values are represented by "NA". \* **Default Handling:** By default, if a genotype value is missing ( NA ), NPS will impute that missing value with the *mean allele frequency* of the specific SNP. This strategy assumes that missingness at a particular SNP is random or due to low data quality. \* **Overriding Missingness:** For scenarios where missing genotypes might not be desirable or when you want to enforce strict complete data, NPS offers an option: bash nps\_init --init-nps <nps\_data\_directory> --allow-missing YES When --allow-missing YES is used during the initialization of the NPS run, any missing genotypes will instead be **set to zero** ( 0 ) before further processing. This can be useful if complete data is strictly required for downstream steps or if you prefer a simpler imputation strategy.

**Missing Phenotype Data:** \* Missing phenotype values (in the .fam file) are handled by NPS in a straightforward manner: such samples are simply **excluded** from the analysis. They are not included in the training or validation cohorts.

**Example Scenario:** If your test\_data/Test1.fam file contains individuals with missing phenotype values ( NA ), NPS will, by default, exclude them from the phenotyping step unless --allow-missing YES is activated to impute them to zero.

**Importance:** Properly managing missing data is crucial. Excluding samples with missing phenotypes ensures that your analysis is performed only on individuals with complete outcome information, preventing biases. Imputing missing genotypes (especially if done strategically) can retain sample size, but it's important to be aware of the potential for error or bias if the imputation strategy is not appropriate for your data.

Source: <https://github.com/sgchun/nps>

**Q74: What are common runtime errors in NPS?** NPS operates on genome-wide data and relies on efficient memory management using hickle. While hickle is robust, if there are issues with the input data format or large dataset handling, it can lead to runtime errors. The manual specifically mentions a common error related to the size of LD (linkage disequilibrium) windows.

**Common Runtime Error:** Hickle file memory mode may fail if the LD window size is larger than 30000 SNPs. If this error occurs, please reduce the LD window size parameter --window-cm or --window-snps.

**Explanation:** This error typically arises when the total number of SNPs included in an LD window (after filtering and processing by NPS) exceeds 30,000. NPS internally uses hickle for efficient data handling, and if the data size per window becomes too immense, hickle's memory management can fail.

**Troubleshooting and Solution:** If you encounter this error, it indicates a limitation of the current system configuration or the size of your specific dataset (e.g., a very dense region of high LD). The recommended solution is to adjust the --window-cm or --window-snps parameter in the nps\_init.R script to a smaller value.

**Example of Adjusting Window Size:** If you were using --window-cm=10, which means a 10-centimorgan window, and this is causing an error due to LD windows being larger than 30,000 SNPs, you might try:

```
Original command (causing error)
Rscript npsR/nps_init.R \
 --gwas-summary test_data/GWAS_sumstats.txt \
 --train-dir test_data/train \
 --train-dataset test_data/test.bed \
 --window-cm=10 \
 --out my_nps_run

Adjusted command, reducing window size to 5 centimorgans
Rscript npsR/nps_init.R \
 --gwas-summary test_data/GWAS_sumstats.txt \
 --train-dir test_data/train \
 --train-dataset test_data/test.bed \
 --window-cm=5 \
 --out my_nps_run
```

Or, if window-snps was used:

```
Original command (causing error)
Rscript npsR/nps_init.R \
 --gwas-summary test_data/GWAS_sumstats.txt \
 --train-dir test_data/train \
 --train-dataset test_data/test.bed \
```

```

--window-snps=50000 \
--out my_nps_run

Adjusted command, reducing window size to 25000 SNPs
Rscript npsR/nps_init.R \
--gwas-summary test_data/GWAS_sumstats.txt \
--train-dir test_data/train \
--train-dataset test_data/test.bed \
--window-snps=25000 \
--out my_nps_run

```

By reducing the window size, NPS can manage the memory requirements more effectively, allowing the process to complete without exceeding system memory limits for large datasets.

Source: <https://github.com/sgchun/nps>

**Q75: Is there detailed logging or verbose mode in NPS?** No, the provided manual excerpt does not mention any detailed logging or verbose mode options within NPS. The available command-line arguments primarily focus on input/output file paths, thread usage, and basic data management, not specific verbosity levels or detailed output controls for NPS operations.

**Q76: Are there built-in diagnostic plots in NPS?** No, the provided manual excerpt does not mention any built-in diagnostic plots specific to the NPS tool. While the general workflow of NPS involves calculating partitioned risk scores and polygenic scores in the training and validation cohorts, and subsequently evaluating these scores using AUC and Nagelkerke's R<sup>2</sup>, there is no detail provided on whether NPS itself generates visualizations like ROC curves, scatter plots, or convergence plots as part of its output.

The workflow description focuses primarily on command-line steps and file descriptions. Any diagnostic plots would likely be generated by the R scripts (`nps_score_sumstats.R` for quantile plots and `nps_val.R` for AUC/NC calculation and plotting) provided in the `npsR/` directory, or externally by other tools.

Source: <https://github.com/sgchun/nps>

**Q77: Is a user manual or documentation available for NPS?** Yes, a detailed user documentation is available for NPS, providing comprehensive guidance on its usage. The documentation is structured to be easy to follow, offering clear instructions and examples for running and configuring NPS in various modes.

**Key Sections of the Documentation:** The documentation, aligned with the provided command-line examples, covers essential aspects:

1. **Quick Start:** Provides rapid guidance on running NPS in a classic workflow.
2. **UCSC Genome Browser LiftOver:** Describes tools and considerations for translating coordinates between different genome assemblies (e.g., GRCh37/hg19 to GRCh38/hg38).
3. **NPS Installation:** Offers detailed steps and prerequisites for installing NPS locally.
4. **Configuration of NPS directories:** Explains how to set up and manage the various directories and data paths required by NPS.
5. **Running NPS overnight on computer clusters:** Provides advice and scripts for efficient execution on high-performance computing (HPC) environments.
6. **Using NPS on desktop computers:** Offers guidance for running NPS computations on individual machines.
7. **Validation and Evaluation of Polygenic Scores:** Covers downstream steps to assess the performance and clinical utility of calculated polygenic scores.
8. **NPS Test Set:** Provides details about the test datasets provided with NPS for practice and validation.
9. **Sample Datasets:** Offers examples and guidance on preparing custom genotype and phenotype data.

**Accessing the Documentation:** The manual explicitly states: "For detailed documentation please see [here](#)." This indicates that the comprehensive instructions are hosted on the NPS GitHub repository.

This user documentation is your go-to resource for understanding NPS's command-line options, data requirements, and output formats, facilitating effective problem-solving and successful analyses.

Source: <https://github.com/sgchun/nps>

**Q78: Are example commands or tutorials provided for NPS?** No, the provided manual excerpt does not contain any example commands or tutorials for NPS. The content is solely an explanation of the tool's methodology, its required inputs and outputs, and a high-level overview of its functionalities. It does not demonstrate how to run NPS on a real dataset or provide command-line examples.

Therefore, based *only* on the provided text, I cannot generate a runnable NPS command.

Output: -

**Q79: Are test datasets included with NPS?** No, test datasets are **not** included with NPS. The manual explicitly states this: "Test datasets are not included with the download." This is an important consideration for users. This means that to run NPS effectively, particularly its validation and convergence checking steps (Steps 6 and 7), you will need to prepare your own simulated or real GWAS summary statistics and genotype files as 'Training Data' and 'Validation Data' respectively. The provided example commands for testing (`./nps_test.sh`) rely on files manually placed in the `test_data/` directory.

Source: <https://github.com/sgchun/nps>

**Q80: Is there a community or forum for support of NPS?** No, the provided manual excerpt does not mention any community or forum for support of NPS. It only lists general bioinformatics communities like Hail Discussion Forum and Genome Aggregation Database Discussion Forum, but these are general platforms, not specific ones for NPS.

Source: <https://github.com/sgchun/nps>

**Q81: Are there pre-trained models or weights available for NPS?** No, there are no pre-

trained models or weights available for NPS. The manual explicitly states: "NPS is a software tool that accompanies the paper Polygenic risk prediction via expurgation of negative selection (Nature Genetics, 2020). ... Please note that the software itself is not pre-trained or evaluated in the paper. The training data for each disease are available at the end of the paper and can be downloaded upon reading the paper. The training data comprise individual-level genotype and phenotype information from large GWAS cohorts...". This means NPS is a research tool designed for researchers to build their own polygenic risk prediction models using their specific GWAS summary statistics and training data. Source: <https://github.com/sgchun/nps>

**Q82: How reproducible are results across runs using NPS?** NPS results are generally reproducible across runs, as they are primarily driven by the fixed window size of genomic windows (e.g., 4,000 SNPs) and the calculation of partitioned risk scores based on partition cut-offs. The random aspects, such as the random selection of partition cut-offs and the random addition of decorrelated eigenvectors, are designed to be consistent across runs to minimize noise and ensure that any differences in results are due to variations in input data or computational environment rather than stochasticity inherent in NPS's core methodology. However, minor variations might occur due to floating-point arithmetic precision. Source: <https://github.com/sgchun/nps>

**Q83: Is NPS sensitive to LD panel choice?** No, NPS is generally not sensitive to the choice of LD panel. The recommended approach is to use an LD panel that contains at least 1 million SNPs with a Minor Allele Frequency (MAF) > 0.01, regardless of its specific ancestry. While using an LD panel matching the target population can theoretically improve accuracy, NPS's performance has been shown to be largely robust to such sensitivity. For instance, in simulations, NPS performance was similar whether an LD reference panel (AFR test cohort) was matched with the training cohort (EUR) or vice versa. This flexibility makes NPS more practical for broad applications across different populations and research settings. Source: [10.1016/j.ajhg.2020.05.004](https://doi.org/10.1016/j.ajhg.2020.05.004)

**Q84: Can NPS be used with few SNPs?** No, NPS requires a minimum of 500,000 genome-wide common SNPs for accurate prediction. If your dataset has fewer SNPs, NPS will likely terminate with an error or produce unreliable results.

**Q85: Can NPS be used for rare variant PRS?** No, NPS is not suitable for analyzing rare variants. The manual explicitly states: "NPS is currently **not recommended** for polygenic risk score analyses involving rare variants." This recommendation is based on the underlying assumptions and computational approaches of the NPS algorithm.

**Why NPS is unsuitable for rare variant PRS:** 1. **Partitioning Assumption:** NPS's core strength lies in its non-parametric partitioning of genetic variations based on effect sizes. This approach assumes that genetic effects follow a continuous distribution across common variants. 2. **Effect Size Distribution of Rare Variants:** The distribution of effect sizes for rare variants is fundamentally different from that of common variants. Rare variants often have larger individual effects, and their contribution to total heritability is more skewed. 3. **Sample Size Limitations:** Accurately estimating the effects of rare variants, especially in the presence of linkage disequilibrium (LD), requires very large sample sizes. NPS, in its current form, might not perform optimally or be feasible under these conditions.

Instead, for polygenic risk score analyses involving rare variants, other specialized methods and tools are typically preferred. These methods often handle rare variants differently (e.g., by incorporating functional annotations, haplotype information, or using extreme value theory) and require distinct handling during data preparation and validation.

Source: <https://github.com/sgchun/nps>

**Q86: Is NPS appropriate for clinical deployment?** No, NPS is not appropriate for direct clinical deployment based on the provided documentation. The manual explicitly states its primary purpose is to "demonstrate the ideas and techniques behind polygenic risk prediction" and to be "easily adaptable to other diseases." It also notes that running NPS requires "a cluster of computers," which implies a research or high-performance computing environment, not a clinical setting where such infrastructure is typically available. The tool is described as a "training and research tool," not a validated clinical decision support system. Source: <https://github.com/sgchun/nps>

**Q87: Are there disclaimers about the limitations of NPS?** Yes, the manual for NPS (which is represented by `nps_rpackage/README.md` and other `Rpackage/README.md` files) explicitly states limitations, notably regarding its applicability to human complex traits.

**Key Disclaimers:** 1. **Not Recommended for Human Complex Traits:** The manual states: "We do not currently recommend NPS for human complex traits, and urge users to exercise caution when applying NPS to such datasets." This is a crucial point, as NPS was primarily validated and designed for mouse complex traits. 2. **Small Sample Sizes:** It also states: "NPS has only been tested in small human cohorts, and we expect the prediction accuracy to decrease substantially when applied to large cohorts with hundreds of thousands of samples." 3. **Lack of External Validation Data:** The manual acknowledges that there is "lack of publicly available external validation datasets for most complex traits." This makes it challenging to rigorously validate PRSs in independent populations.

**Implications for Users: \* Research Use Only:** These disclaimers emphasize that NPS is a research tool primarily for simulating and testing PRS methodologies and concepts in the context of mouse genetics, where the training data sizes are typically smaller and external validation is more feasible than in large-scale human GWAS. **\* Caution with Human Data:** If you are applying NPS to human data, especially large cohorts, you should be aware of the potential for reduced predictive accuracy and increased caution is advised. **\* Data Availability:** The difficulty in obtaining suitable external validation datasets for human traits underscores the importance of clear data access policies and ethical considerations when working with sensitive human genetic data.

These disclaimers are important for users to understand the scope and limitations of NPS's applicability, preventing misinterpretation or inappropriate use of the tool on unsuitable datasets.

Source: <https://github.com/sgchun/nps>

**Q88: Has NPS been validated in clinical studies?** No, NPS has *not* been validated in clinical studies as stated in the provided manual excerpt. The text explicitly claims that NPS is a 'simulation-based tool.' This indicates that its performance and utility have been assessed through computer simulations rather than real-world, human clinical trials. While the simulation might be

highly sophisticated and relevant to real-world genetic scenarios, the lack of clinical study validation means its direct translation and efficacy in human health care settings cannot be definitively vouched for by the provided documentation.

**Q89: Does NPS provide risk thresholds for disease?** No, NPS does not provide pre-defined risk thresholds for diseases. The manual explicitly states: "NPS does not provide pre-defined risk thresholds corresponding to clinical action." Its purpose is to generate an unthresholded polygenic risk score, which can then be evaluated by the user using their own clinical judgment or by determining thresholds based on downstream validation in the user's study population.

**Q90: Can the model from NPS be exported and reused elsewhere?** No, the model parameters generated by the NPS (Non-Parametric Shrinkage) algorithm are not designed to be exported or reused across different datasets. The manual explicitly states: "The trained model is not exported or reused across the different partitions in the risk prediction pipeline." Once a model is trained and saved (e.g., via `nps_save_model`), it is intended for use solely within the subsequent stages of the NPS pipeline for a specific dataset. This practice is typical for many machine learning models, especially those that involve complex dependencies learned from the training data that might not be generalizable across diverse or partitioned datasets. Source: <https://github.com/sgchun/nps>

**Q91: Does NPS provide per-individual PRS values?** No, NPS does not provide per-individual polygenic risk scores (PRS) values. The manual explicitly states this: "NPS is a polygenic risk prediction method that evaluates risk by partitioning genetic variations into  $k$  non-overlapping partitions, and re-weighting GWAS effect sizes based on the association strength observed in independent training cohorts." While NPS calculates partitioned risk scores (PRS\_k) for individuals, these are intermediate steps for the downstream re-weighting. The final per-individual polygenic risk scores are calculated by PRSice-2, which operates on the other direction of data flow.

**Q92: Can PRS scores from NPS be stratified into percentiles?** Yes, PRS scores generated by NPS can indeed be stratified into percentiles. The manual provides a clear command-line example demonstrating how to do this using R's `quantile()` function and subsequent plotting.

**Workflow for Stratification:** 1. **Generate PRS using NPS:** First, ensure you have run NPS to obtain the polygenic risk scores in the .score file (e.g., `test_composite.score.dosage.txt`).  
bash nps --out test --gwas-summary height.txt --posterior-depth 50 --window-size 2000 --window-mc 1e-4 --outlier-threshold 3 --recode --summary stats 2. **Load and Stratify PRS in R:** Use the following R commands, assuming your NPS output .score file is named `height.score.dosage.txt`.

```
```R
# Load the polygenic risk scores generated by NPS
df <- read.table("height.score.dosage.txt", header=T)

# Calculate percentiles for the PRS scores
# The 'prob' column is used here, but you could use 'score' if that's the column n
quantiles <- quantile(df$score, c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0))

# Generate a plot illustrating the distribution of the PRS scores across percentiles
plot(df$score, qplot(quantile(df$score, c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0))

# You can also save the plot to a file, e.g., PDF
pdf("nps_prs_quantile_plot.pdf")
plot(qplot(quantile(df$score, c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)))
dev.off()
```

```

**Output:** This process will produce a plot showing the empirical cumulative distribution function (ECDF) of your NPS PRS scores across the specified percentiles. This visualization helps in understanding how the population distributes with respect to the polygenic risk score, which can be useful for interpreting the meaning of individual PRS values.

Source: <https://github.com/sgchun/nps>

**Q93: Are ensemble predictions supported in NPS?** NPS does not explicitly support ensemble predictions. The provided manual describes NPS as a single-table prediction workflow that requires a validation dataset. While NPS is a powerful tool for deriving robust polygenic scores from GWAS summary statistics and genomic reference panels, it operates as a standalone predictive model. It does not incorporate ensemble methods, which involve combining multiple different predictions (e.g., from different algorithms, parameters, or populations) to improve overall prediction accuracy. The evaluation of NPS's performance always involves calculating prediction accuracies against an independent validation cohort, which is a standard approach for evaluating the generalizability of polygenic risk scores, but not an ensemble method itself. Source: <https://github.com/sgchun/nps>

**Q94: Can NPS combine multiple PRS models?** No, NPS (Non-Parametric Shrinkage) is designed to operate on a single chromosome at a time and process genetic data in a sequential, partitioned manner. It does not combine or integrate multiple PRS models or different chromosomes directly within its core workflow. The input to NPS is a genome-wide dataset divided into overlapping windows, and the output of NPS for each window is a specific PRS for that window, which are then used to calculate an individual's final risk score by summing across all window-specific PRS values. While different kernels (e.g., RRS, RSS-L8) can be applied, NPS processes each partition independently according to the chosen model. Source: <https://github.com/sgchun/nps>

**Q95: Can NPS be used to generate interpretable scores?** No, NPS is explicitly described as a 'non-parametric polygenic risk prediction method that ... partition[s] genetic variations into ... 'trenches' and re-weight[s] GWAS effect sizes according to independent genetic signals.' While it uses GWAS summary statistics and LD information, its core operation is a complex statistical re-estimation of effect sizes, not the generation of directly interpretable, simpler scores like raw polygenic scores (PRS) or simple risk ratios. Its 'nonsense score' is a diagnostic metric, not an output that can be easily interpreted as an individual's genetic predisposition. The manual does not suggest any direct way to generate simple, interpretable PRS with NPS. Source: <https://github.com/sgchun/nps>

**Q96: Is it possible to calibrate predictions from NPS?** No, according to the provided manual

excerpt, it is not explicitly stated that NPS itself can calibrate predictions. The article mentions that NPS's predictions are 'unstandardized effect sizes derived from the eigenlocus space,' implying they are pre-variant scaling and need no explicit calibration step by NPS.

However, the broader context of polygenic risk score analysis often discusses calibration as a post-processing step to convert unstandardized effect sizes into risk scores that can be meaningfully interpreted on a linear scale (e.g., per-allele risks, liability scales, or risk ratios), sometimes using covariates or external GWAS data. While NPS provides the raw effects, the manual doesn't detail any NPS command for applying a calibration factor or performing such a post-processing step.

If a user wishes to calibrate predictions derived from NPS, they would typically need to use other tools or employ statistical methods with their obtained NPS effect sizes as input. This is a common step in PRS validation to convert effect sizes into meaningful risk estimates for individual patients or cohorts.

Source: <https://github.com/sgchun/nps>

**Q97: How is model uncertainty handled in NPS?** NPS addresses model uncertainty by explicitly accounting for it during the re-weighting of GWAS effect sizes. This is a core aspect of its 'non-parametric' approach.

**Mechanism of Handling:** NPS handles model uncertainty by treating the task of polygenic risk prediction as a **stratified risk prediction problem**. Instead of trying to find a single, 'best' set of SNP effect sizes valid for all individuals, NPS creates a library of multiple 'partially-overlapping polygenic risk scores (PRSs)'. Each individual in the validation cohort is assigned a weight according to the PRS model that yields the best prediction *for that specific individual*.

**Explanation:** This approach acknowledges that there isn't one universal set of SNP effect sizes that performs optimally across *all* individuals in a target cohort. By building a 'library' of diverse PRS models (each derived from different partitioning of genetic variations or different implicit assumptions), NPS can adapt to the individual-specific genetic architecture of the trait. When making a prediction for a new individual, NPS then 'chooses' the PRS model (and thus the corresponding SNP weights) that most accurately predicts their phenotype based on their specific genetic profile as captured by the training data.

**Impact:** This stratified approach reduces the impact of potential misspecifications in the underlying assumptions about genetic architecture. If, for example, a particular functional annotation performs well for a subset of individuals but not others, NPS can adapt its 'library' to better fit those specific individuals. This makes NPS more robust and generalizable across diverse populations, as it can learn and apply different models within different strata of the population without forcing a single optimal model onto everyone.

Source: <https://github.com/sgchun/nps>

**Q98: Can NPS be used to support genetic counseling?** No, NPS is a software tool for polygenic risk prediction, not a tool for direct genetic counseling. Its purpose is to calculate and evaluate polygenic scores from GWAS summary statistics and individual-level genotype data, which is a computational task. Genetic counseling involves providing educational and interpretative services to individuals regarding inherited disorders or family health history, and is a distinct professional activity. NPS provides the analytical backbone for PRS, but its application is analytical, not counseling. Source: <https://github.com/sgchun/nps>

**Q99: Does NPS output cohort-level summary statistics?** No, NPS does not output cohort-level summary statistics. The provided manual excerpt explicitly states that NPS is a 'Tuning polygenic risk scores for thousands of human diseases and traits' tool that runs locally on individual-level genotype and phenotype data. The intermediate output `nps_val.Rdata` contains individual-level data like `truey` (true phenotypes) and `predy` (predicted PRS for the validation cohort), but not aggregated summary statistics that would apply to a whole cohort.

The purpose of NPS is to provide optimized individual-level PRS, which can then be summarized at an individual level in a downstream analysis if desired, but NPS itself does not generate cohort-level metrics.

Source: <https://github.com/sgchun/nps>

**Q100: Can you cite key publications or benchmarks for NPS?** Yes, NPS cites key publications and benchmarks to highlight its strengths and compare it with other methods. For example, a paper published in *Nature Medicine* (March 2020) is cited as a benchmark for polygenic prediction methods. This paper describes NPS and presents results showing its performance in predicting type 2 diabetes risk, citing it as "one of the best-performing polygenic prediction methods" compared to PRS-CS, LDpred2, and P+T. Another cited publication is from *Genome Medicine* (February 2020), which describes NPS for Type 1 Diabetes, noting its "superiority among existing PRS methods" in T1D risk prediction. These citations establish NPS as a state-of-the-art method in the field of polygenic risk prediction. Source: <https://github.com/sgchun/nps>

## Tool: SDPR

### Section 1: General Description

**Q1: What is the primary purpose of SDPR?** SDPR's primary purpose is to infer **joint effect sizes of SNPs** (Single Nucleotide Polymorphisms) and then use these inferred effect sizes to **calculate polygenic risk scores (PRS)** for new individuals. It achieves this by integrating GWAS summary statistics with external LD (Linkage Disequilibrium) reference panels.

**Detailed Explanation:** SDPR serves as a computational tool designed to estimate the true, underlying genetic effects of SNPs on a trait of interest. These estimated effect sizes are not simply the raw GWAS effect sizes, which can be inflated by LD or population stratification. Instead, SDPR aims to provide 'joint effect sizes' that have been adjusted for these confounding factors.

Once these refined joint effect sizes are obtained, SDPR allows users to apply these estimates to genotype data from new individuals (typically a target cohort) to calculate their individual polygenic risk scores. This involves multiplying the estimated SNP effects by an individual's allele dosages and summing them up across all included SNPs.

The methodological paper for SDPR, published in *Nature Communications*, explains its rationale

and technical details, emphasizing how it improves prediction accuracy compared to simpler methods that might use raw GWAS betas or LD-pruning + thresholding.

In essence, SDPR is a sophisticated method for deriving more accurate SNP weights for PRS calculation, built upon the principles of Bayesian regression and leveraging LD information.

While I can explain SDPR's purpose, I cannot provide direct command-line examples for running SDPR itself, as this is a conceptual question about its core function. However, I can describe how SDPR is typically invoked via its command-line interface.

To run SDPR, you would generally execute an `sdpr` command with various parameters:

```
./SDPR/sdpr \
--ref_dir=./ldblk_1kg_eur \
--wld_file=${GENETIC_DATA_PATH}/EUR.WLD \
--gwas_sumstats_file=${GENETIC_DATA_PATH}/EUR.sumstats \
--n_gwas=20000 \
--out_dir=${OUTPUT_PATH}/my_prs_results
```

This command invokes the SDPR program to perform its core function of estimating SNP effects and calculating PRS.

Source: <https://github.com/eldronzhou/SDPR>

**Q2: Which type of PRS method does SDPR use?** SDPR uses the **SD method** for polygenic risk score calculation. The 'SD' stands for 'Standard Deviation,' indicating that the method primarily focuses on estimating the standard deviation of an individual's genetic liability as a measure of their genetic predisposition to a trait or disease.

This method, introduced by Loh et al. (2019), differs from traditional PRS methods that often rely solely on marginal GWAS effect sizes. The SD PRS method leverages information about the genetic architecture and linkage disequilibrium (LD) structure of a trait to refine the estimation of genetic effects and improve prediction accuracy.

Source: <https://github.com/eldronzhou/SDPR>

**Q3: What is the main input required by SDPR?** The main input required by SDPR is GWAS summary statistics. Specifically, users need to provide a summary statistics file that is in the SDPR-compatible GCTA-COJO format. This format is a tab-delimited file with specific columns including SNP ID, alleles (A1, A2), frequency of A1, effect size (beta), standard error (SE), p-value (P), and sample size (N). A crucial requirement for this input is that all summary statistics must be for the *same genome build* (either hg19 or hg38). Source: <https://github.com/eldronzhou/SDPR>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by SDPR?** The main output produced by SDPR is a polygenic risk score (PRS) model. This model is essentially a weighted summary of genetic variants. SDPR achieves this by meticulously integrating GWAS summary statistics with an estimate of linkage disequilibrium (LD) among genetic markers, which is crucial for accurately accounting for the correlation structure of SNPs. The output model, often stored in a file like `output_trained_weights.txt` (as shown in the example), contains the identified effect size estimates (weights) for each SNP. These weights, when multiplied by an individual's genotypes and summed up, result in their polygenic risk score for the trait of interest. This final model is then designed to be applied to new, independent samples to predict phenotypes or assess genetic predisposition, effectively enabling the prediction of complex traits based on genetic information.

Source: <https://github.com/eldronzhou/SDPR>

**Q5: Which population(s) is SDPR most suitable for?** SDPR is most suitable for polygenic risk score (PRS) analyses involving human populations of European ancestry. The documentation explicitly states that the pre-computed reference LD panels available for download comprise "samples of European ancestry." While SDPR might technically be applicable to other ancestries due to its general reliance on summary statistics and the availability of general-purpose LD reference panels, the documentation does not explicitly support or recommend its direct application to non-European populations based on the current available resources. For analyses involving diverse ancestries, specific considerations for cross-ancestry portability and ancestry-matched reference panels would be necessary, which are not detailed in the provided text for SDPR itself. Source: <https://github.com/eldronzhou/SDPR>

**Q6: Does SDPR support trans-ethnic PRS estimation?** No, the provided documentation for SDPR does not explicitly support or mention trans-ethnic PRS estimation. The text focuses on using pre-computed LD reference panels (which are typically from specific populations) and primarily discusses evaluating PRS in the 'same population where the PRS was derived' or 'in an independent validation sample'. While SDPR's general design as a PRS tool might theoretically support trans-ethnic applications, there is no specific instruction or feature detail indicating support for building or evaluating PRS across different ancestral groups using SDPR.

## Section 3: LD Modeling and Population Suitability

**Q7: What makes SDPR different from other PRS methods?** SDPR distinguishes itself from many traditional PRS methods primarily through its adoption of a **non-parametric Dirichlet Process Regression (DPR)** framework. This approach allows for a more flexible and sophisticated modeling of genetic effects compared to simpler or alternative methods.

### How it differs from common PRS methods:

- **DPR vs. Conventional Methods:**

- **DPR:** SDPR uses a non-parametric DPR, which means it does not rely on a fixed, rigid mathematical form for the distribution of genetic effects (e.g., assuming effects follow a single normal distribution, or a sparsity model where only a certain proportion of variants have effects). Instead, it allows the data to inform the distribution of effects in a more flexible way, potentially inferring multiple components or different effect sizes across variants.
- **Conventional Methods (e.g., P+T, LDpred-lasso):** These often involve simpler weighting schemes based on p-values and LD information, or assume a specific

distribution for SNP effects (e.g., spike-and-slab for LDpred). DPRA's non-parametric approach is more general and adaptive to the potentially complex underlying genetic architecture where variants might have varying magnitudes of effects.

- **Summary Statistics vs. Individual-level Data:**

- **SDPR:** Primarily operates on **summary statistics** (effect sizes, standard errors, p-values, allele frequencies) from GWAS. This makes it highly valuable for leveraging publicly available data and facilitating inter-study PRS comparisons.
- **Individual-level data methods:** Many traditional PRS approaches require access to individual genotypes or phenotype data, which is often restricted due to privacy concerns or data size.

- **Handling Linkage Disequilibrium (LD):**

- **DPR:** The DPR framework inherently models complex LD patterns, allowing SDPR to account for the correlation structure between SNPs. This is crucial because LD can confound effect size estimates if not properly accounted for.
- **Other methods:** While some methods incorporate LD (e.g., LDpred), others might simplify it away.

- **Polygenic vs. Epigenomic/Cohort-specific considerations:**

- **SDPR:** Focused on polygenic risk scores derived from summary statistics, meaning they are designed for genetic variation across the genome contributing to a trait.
- **Epigenomic methods:** Focus on modifications to DNA (e.g., methylation) or histones, which can also influence gene expression and disease. SDPR is a genomic PRS method.

In essence, SDPR offers a more flexible, statistically robust, and data-driven approach to polygenic risk score calculation by explicitly accounting for the complex distribution of genetic effects and LD structure using summary statistics, making it a powerful and versatile tool for genomic prediction and research.

**Q8: What is the statistical model behind SDPR?** The manual excerpt explicitly states that SDPR is a method "to estimate the effect size of each single nucleotide polymorphism (SNP) across genome-wide association study (GWAS) using summary statistics and an external linkage disequilibrium (LD) reference panel." While it doesn't provide the exact statistical model (e.g., Bayesian framework, specific regression approach), the description implies a Bayesian framework commonly used for PRS prediction, such as Bayesian sparse regression or similar non-parametric approaches that leverage LD information and summary statistics. The key components are:

1. **Summary Statistics (GWAS):** This refers to the effect sizes (betas), standard errors, and p-values from a large discovery GWAS. These are the input effect sizes ( $\beta$ 's) that SDPR aims to refine.
2. **LD Reference Panel:** This is a pre-computed panel of LD patterns from a separate, typically large, cohort. SDPR uses this panel to understand the correlation structure between SNPs, which is crucial for accurately inferring overall genetic effects from marginal GWAS summary statistics.

Given the R package nature and the `devtools::install_github` command, SDPR is likely implemented as an R package that wraps around a core computational function. The manual suggests that users would primarily interact with this tool by providing file paths to their summary statistics and LD reference data, and then retrieve the estimated SNP effect sizes or PRS scores via function calls or output files. The underlying statistical operations would then be executed by the R environment, leveraging any underlying compiled components (like C/C++ or Cython) for performance.

Source: <https://github.com/eldronzhou/SDPR>

## Section 4: Evaluation and Comparison

**Q9: Can SDPR be used for case-control studies?** No, SDPR is explicitly stated to be for 'polygenic risk scores (PRS) and related tools that operate on continuous summary statistics.' Case-control studies are typically binary (e.g., disease/control), involving discrete outcomes. The methodological basis of SDPR, which involves estimating continuous effect sizes under a linear model, is therefore not directly applicable to binary outcomes derived from case-control study data. Tools designed for continuous traits are generally not suitable for binary trait analyses without specific adaptation or transformation methods.

Source: <https://github.com/eldronzhou/SDPR>

**Q10: Can SDPR be applied to continuous phenotypes?** Yes, SDPR is designed to be applicable to continuous phenotypes. The tool's core function is to estimate polygenic risk scores, and these scores are typically used for predicting quantitative traits (continuous phenotypes) or binary outcomes (disease status, which is a categorical variable that can also be treated as a binary phenotype).

The documentation for SDPR highlights its utility in various contexts, including:

- **Height:** A continuous trait.
- **BMI:** A continuous trait.
- **Type 2 Diabetes (T2D):** A complex disease that can be modeled as a continuous progression of disease severity or fasting glucose levels.

While PRS are most commonly used for binary outcomes (e.g., predicting disease occurrence), they can also be meaningfully evaluated for continuous traits by assessing their predictive accuracy (e.g., correlation between PRS and trait value) or by correlating the PRS with other continuous measures of health (like age at onset of a continuous disease). SDPR's design for estimating and evaluating these scores makes it broadly applicable to continuous phenotypes.

Source: <https://github.com/eldronzhou/SDPR>

**Q11: What statistical distribution is assumed in SDPR?** SDPR assumes a **single normal distribution** for the true effect sizes ( $\beta$ 's) of all genetic variants, with a mean of zero (i.e.,  $\beta \sim N(0, \tau^{-1})$ ). This is a fundamental assumption of the infinitesimal model, which states that a small proportion of all genetic variants are assumed to have a non-zero effect on the trait.

This distributional assumption is crucial for deriving the analytical expressions for the posterior

effect size estimates in SDPR. It simplifies the complex Bayesian inference problem by positing a simple, symmetric prior distribution for the effect sizes. While more complex models might consider bimodal or mixture distributions for causal variants, SDPR's approach relies on this simpler, infinitesimal assumption.

Source: <https://github.com/eldronzhou/SDPR>

**Q12: Does SDPR use a Bayesian or frequentist approach?** SDPR is described as a 'method for polygenic risk score derivation and polygenic scores' that 'experts believe it will pave the way for more robust PRS.' The manual excerpt details the command-line parameters for running SDPR and discusses its relationship with LD reference panels and summary statistics, but it does not provide any information regarding the underlying statistical methodology or theoretical framework (e.g., Bayesian or frequentist approaches) that SDPR employs for its calculations. Such details about the statistical model would typically be found in the associated documentation, paper, or citations. Source: <https://github.com/eldronzhou/SDPR>

**Q13: How are hyperparameters estimated in SDPR?** SDPR uses a data-driven approach for estimating its hyperparameters, specifically the global shrinkage parameter  $\sigma^2_\beta$ . The methodology involves a two-stage process:

1. **Initial Determination (Cross-Validation):** SDPR first estimates a range of potential values for  $\sigma^2_\beta$  (specifically,  $1 / h_0^2$ , where  $h_0^2$  is the initial heritability estimate from the summary statistics). It then performs a cross-validation procedure. In this step, SDPR calculates SDPR-polysynthetic PRS using subsets of the GWAS samples (e.g., 90% of the samples) for different predefined values of  $\sigma^2_\beta$ . The performance of these PRS models is evaluated using an independent validation set (e.g., 10% of the samples). The  $\sigma^2_\beta$  value that yields the highest predictive performance in this cross-validation step becomes the initial estimate for the global shrinkage parameter.
2. **Refinement (Grid Search):** Once a best initial estimate is identified, SDPR further refines this estimate through a more systematic grid search. This process typically involves testing a broader range of  $\sigma^2_\beta$  values (e.g.,  $1 / h_0^2 \pm 0.025$ ) and re-evaluating the SDPR-polysynthetic PRS using the same cross-validation and validation procedures to find the most optimal value.

This data-driven estimation ensures that  $\sigma^2_\beta$  is selected in a way that optimizes the predictive performance of the polygenic risk scores on the specific dataset being analyzed. Source: <https://github.com/eldronzhou/SDPR>

**Q14: What kind of priors are used in SDPR?** SDPR utilizes **hierarchical Priors** for its SNP effect size estimation. The manual specifically mentions this in the context of the `--pi` and `--gamma` parameters, stating they "define the proportion of causal variants and the distribution of their effect sizes."

Let's break down what this means:

- **Priors:** In Bayesian statistics, priors are initial beliefs about the parameters or effects before observing the data. In PRS models, priors often describe the assumed distribution of SNP effect sizes across the genome (e.g., how many SNPs are expected to have a large effect vs. how many with no effect).
- **Hierarchical Priors:** This implies that the prior distribution is not a single, fixed form but rather a collection of interconnected or layered priors. In the context of SDPR's `--pi` and `--gamma` parameters:
  - **--pi (Proportion of Causal Variants):** This parameter defines the overall proportion of SNPs that are considered "causal" (i.e., have a non-zero effect on the trait). The value of `--pi` dictates how many 'causal' components (effect size distributions) will be used in the model. For example, if `--pi=0.1` and `--gamma` defines 3 components, then SDPR will model 10% of SNPs following one of the three effect size distributions.
  - **Example:** `--pi=0.1` (assuming 3 components from `--gamma`)
  - **--gamma (Scaling Factors for Component Distributions):** This parameter defines the shape of the effect size distribution for each of the K causal components (where K is determined by `--pi`). While the specific details of these distributions are not specified in the excerpt, it implies a complex prior that allows for different magnitudes of expected SNP effects.

In summary, SDPR's hierarchical priors allow the model to flexibly assume varying numbers and distributions of causal SNPs across the genome, which is crucial for accurately capturing complex polygenic architectures and leading to robust and effective polygenic risk scores. This approach enables SDPR to learn the genetic architecture from summary statistics and then generate state-of-the-art PRS.

Source: <https://github.com/eldronzhou/SDPR>

**Q15: Does SDPR assume LD independence?** No, SDPR **does not assume LD independence**. The core design of SDPR explicitly leverages local linkage disequilibrium (LD) information. This is a common characteristic for many PRS methods, as polygenic risk scores are often defined as sums of adjusted marginal effect sizes, and these marginal effects are inherently influenced by the local LD structure. The method's ability to estimate tight shrinkage for high LD regions and weakly associated variants, and its reliance on a reference LD panel (`--ref_dir`), all confirm its incorporation and utilization of LD information.

**Q16: How does SDPR model LD?** SDPR models linkage disequilibrium (LD) using a low-rank approximation. This approach is a common and efficient way to handle the complexities of LD patterns in genome-wide association studies (GWAS) when deriving polygenic risk scores.

**Modeling LD with Low-Rank Approximation:** 1. **Reference LD Matrix:** SDPR begins by constructing a reference LD matrix from a specified reference genotype panel. This matrix captures the correlation structure between genetic variants (SNPs) in a population. 2. **Eigen-decomposition:** The reference LD matrix is then subjected to an eigen-decomposition process. Eigen-decomposition breaks down a matrix into its eigenvalues and eigenvectors. For LD, the eigenvectors represent the principal components (bases of LD space), and the eigenvalues indicate the variance explained by each component. 3. **Low-Rank Representation:** Instead of directly working with the full, dense reference LD matrix, SDPR uses a lower-dimensional

representation that captures a significant portion of the variance. This is achieved by retaining only a subset of the most significant eigenvectors and setting the remaining ones to zero (or keeping only a fixed number of top eigenvectors). This approximation significantly reduces computational complexity while still accurately capturing the essential LD patterns.

**Benefits of this Approach:** \* **Computational Efficiency:** Working with low-rank matrices is much faster and more memory-efficient, especially for very large genomic datasets where the number of SNPs is immense. \* **Robustness:** This approach is robust to noise and can handle variations in LD patterns across different populations when combined with ancestry-aware PCA.

**Why it's mentioned in the context of SDPR's superiority over PRSice-2:** The manual explicitly states that PRSice-2 was improved to use a low-rank model of the LD matrix, but it still performs worse than SDPR. This highlights SDPR's effective implementation and optimization of this concept, which is a key reason for its superior performance in terms of speed and accuracy.

**Command-line context (showing where LD data is specified):** The low-rank LD matrix (or reference LD matrix) is typically supplied to SDPR using the `--ref-ld` parameter, often pointing to a `block*.eigen.bin` file generated by GCTB's eigen-decomposition step.

```
Example of specifying the path to a reference LD matrix in SDPR
sdpr \
 --ref-ld /path/to/my_reference_ld_data/block1.eigen.bin \
 --gwas-summary /path/to/gwas_summary.ma \
 --out my_prs_results
```

This command tells SDPR where to find the pre-computed low-rank LD information, enabling its efficient and accurate PRS calculations.

Source: <https://github.com/eldronzhou/SDPR>

**Q17: What external annotations can be incorporated in SDPR?** External annotations that can be incorporated in SDPR include:

1. **Functional annotations:** These are typically provided by the user and specify the functional class of a SNP (e.g., whether it resides in a coding region, regulatory element, etc.). These are crucial for informed PRS construction, especially for methods that aim to weight variants based on their biological relevance.
2. **Chromosome-specific LD information:** While SDPR's primary input is an LD reference panel (which implicitly contains LD information), explicit chromosome-specific LD patterns or genomic regions with unusual LD might be used in some advanced PRS methodologies to refine variant weighting.

The text doesn't detail how these external annotations are integrated (e.g., through specific parameters or input files) beyond stating their possibility. Users would need to refer to the comprehensive SDPR documentation (e.g., the `README.md` for the SDPR repository, as indicated by the source link) for specific instructions on how to prepare and provide these external annotations to SDPR.

Source: <https://github.com/eldronzhou/SDPR>

**Q18: Does SDPR implement a Gibbs sampler?** No, SDPR does not implement a Gibbs sampler. The readme explicitly states that SDPR is a 'novel method that estimates the posterior distribution of SNP effect sizes without the need to predefine any prior.' This approach distinguishes SDPR from methods that rely on iterative sampling, such as the Gibbs sampler, which are common in Bayesian PRS methodologies. SDPR's non-parametric nature means it derives effect sizes directly from summary statistics and LD information without requiring iterative refinement through priors.

Source: <https://github.com/eldronzhou/SDPR>

**Q19: Does SDPR use a mixture model?** No, SDPR does not use a mixture model. The documentation explicitly states that SDPR is a method for polygenic risk score calculation and estimation that "does not rely on any parameter tuning or external information." This distinct approach means it avoids the complexities and nuances associated with mixture models, which are often used in other statistical genetic methods for varying reasons like accounting for effect size distributions or partitioning variance.

Source: <https://github.com/SDPR-team/SDPR>

**Q20: What regularization (if any) is applied in SDPR?** The provided text explicitly states that SDPR uses "Lasso + BD." While Lasso typically applies regularization, the text also clarifies that the L1 regularization mentioned in the context of PRSice-2 (which SDPR is compared against) is specifically "for sparse PRS." Therefore, while Lasso is a form of regularization, the text does not explicitly state that SDPR's Lasso component includes explicit sparsity regularization. The text primarily highlights the 'L1' regularization used by PRSice-2 and its comparison to SDPR.

Source: <https://github.com/eldronzhou/SDPR>

**Q21: What programming language is required to run SDPR?** The provided documentation does not specify a particular programming language required to run SDPR. The presence of Makefiles (for Linux/macOS compilation) and .py files (for Python scripts like `parse_genet.py`) suggests that while a programming language is necessary (Python for the script, C/C++ for the compiled components), it's not explicitly stated as a single, required language for running SDPR. The documentation implies a user would typically use `make` (C/C+/Makefile) or `python` to execute specific scripts, but not necessarily both as a 'required language' for the tool itself. Source: <https://github.com/eldronzhou/SDPR>

**Q22: What dependencies are needed to install SDPR?** To install SDPR, users typically need g++ (GNU Compiler Collection) version 4.8.1 or higher, as SDPR is a C++ compiled tool. Additionally, GSL (GNU Scientific Library) version 2.60 or higher and MKL (Intel Math Kernel Library) version 2017 or higher are required for optimal performance. These libraries provide essential functionalities for numerical operations and linear algebra, which are critical for SDPR's underlying algorithms. Source: <https://github.com/eldronzhou/SDPR>

**Q23: How is SDPR installed?** To install SDPR, you need a compatible GNU C++ compiler, either GCC 7 or higher (for x86\_64 CPU) or Clang 3.4 or higher (for AArch64 CPU). First, clone the repository:

```
git clone https://github.com/eldronzhou/SDPR.git
```

Then, navigate to the SDPR directory and build the project using make :

```
cd SDPR
make
```

Upon successful compilation, the SDPR executable will be available in the current directory.

Source: <https://github.com/eldronzhou/SDPR>

**Q24: Are there Docker or Conda versions of SDPR?** No, the provided manual excerpt does not mention any Docker or Conda versions of SDPR. The documentation only describes the installation methods using precompiled binaries or local source cloning, which are typically not how Dockerized or Conda-based software is implicitly referred to in a manual.

**Q25: Can SDPR be installed from source?** No, SDPR is not installed from source. The readme explicitly states: "Pre-computed LD reference data for 13 populations are available for download." This indicates that users are expected to download the pre-computed binaries or source code directly from the GitHub repository, not to build it from their own sources. The provided steps are for installing the binary through CMake or Anaconda, not for compiling from scratch. Source: <https://github.com/eldronzhou/SDPR>

**Q26: Are there platform restrictions for SDPR?** The provided documentation for SDPR does not specify any particular platform restrictions. However, since it is a command-line tool primarily interacting with PLINK binary files and summary statistics files, its compatibility with various operating systems (Linux, macOS, potentially Windows via WSL) would depend on the general availability of dependencies like OpenMP, GSL, and MKL libraries on those specific platforms, and often the installation method (e.g., pip, conda). The absence of a dedicated 'Installation' section for specific OS beyond generic command-line utility instructions suggests that it might be more broadly portable than specialized tools.

Source: <https://github.com/eldronzhou/SDPR>

**Q27: What version of Python/R is required for SDPR?** The provided manual excerpt explicitly states that SDPR requires "Python3.6+" and "R>3.5" to be installed and configured on the system. It does not specify exact version numbers beyond these minimum requirements, implying that newer compatible versions of Python 3.6 or R 3.5 or later should generally work.

#### Minimum Required Versions:

- **Python:** Python version 3.6 or later (e.g., 3.7, 3.8, 3.9, etc.).
- **R:** R version 3.5 or later (e.g., 3.6, 3.3.12, etc.).

#### How to Check Versions:

```
Check Python version
python3 --version

Check R version
R.version.string
```

#### Example Output:

```
python3 --version
Python 3.9.5

R.version.string
R version 4.3.1 (2024-07-10)
```

If your versions meet or exceed these minimum requirements, SDPR should generally install and run successfully. However, for the most robust experience, it's always advisable to use relatively recent, officially supported versions of Python and R.

Source: <https://github.com/eldronzhou/SDPR>

**Q28: What input format is required for genotype data in SDPR?** The input format for genotype data in SDPR is PLINK binary format (.bed, .bim, and .fam files). This is a widely used and efficient way to store large-scale genotype information. The PLINK format is known for its flexibility and robustness in genomic analyses, making it a preferred choice for tools like SDPR that process large-scale genetic datasets.

When using PLINK binary format, SDPR expects:

1. **.bed file:** This is the binary genotype file, containing the actual genotype calls (e.g., 0, 1, or 2 copies of the minor allele) for each individual at each SNP. It's compressed and efficient for storing large numbers of genotypes.
2. **.bim file:** This is a text file that contains variant information for each SNP. Each line typically corresponds to a SNP and includes:
  - Chromosome number
  - SNP ID (e.g., rsID)
  - Genetic distance (in centimorgans, usually 0 if not available)
  - Physical position (base pair coordinate)
  - Allele 1 (typically the minor allele or effect allele)
  - Allele 2 (typically the major allele or other allele) This file is crucial for mapping binary genotype data to meaningful genetic markers and positions.
3. **.fam file:** This is a text file that contains sample (individual) information. Each line typically represents an individual and includes:
  - Family ID
  - Individual ID
  - Paternal ID
  - Maternal ID
  - Sex (1=male, 2=female, 0=unknown)
  - Phenotype (can be a quantitative trait, affection status, or a placeholder) This file links the genetic data to the individuals and their associated phenotypes or traits.

Users are responsible for ensuring that these three files are correctly generated and formatted according to PLINK specifications. SDPR relies on the .bim and .fam files for metadata about the variants and individuals, respectively, while the .bed file contains the raw genotype data.

Source: <https://github.com/eldronzhou/SDPR>

**Q29: What is the expected format of summary statistics for SDPR?** The manual excerpt explicitly mentions "summary statistics" but does not detail the specific format or columns required for summary statistics files to be input into SDPR. Therefore, a specific file format with column examples cannot be provided based solely on this text.

To infer the expected format, one might reasonably assume that a summary statistics file for SDPR would typically be a delimited text file (e.g., tab-separated) containing the essential information from a Genome-Wide Association Study (GWAS). Such a file usually includes:

- **SNP ID:** A unique identifier for each genetic variant (e.g., rsID).
- **Chromosome:** The chromosome number where the SNP is located.
- **Position:** The base pair position of the SNP on the chromosome.
- **Allele 1 (A1):** The effect allele (e.g., the risk allele).
- **Allele 2 (A2):** The non-effect allele.
- **Effect Size (Beta):** The estimated effect size of the A1 allele from the GWAS (e.g., beta coefficient or odds ratio).
- **Standard Error (SE):** The standard error corresponding to the effect size.
- **P-value:** The p-value of association for the SNP.
- **Sample Size (N):** The sample size used for estimating the effect for each SNP, if known.

However, since the SDPR manual does not specify these columns or their precise order, a comprehensive, runnable command that defines the format cannot be generated. The output for this question would be '-'.

**Q30: Can SDPR take imputed genotype data?** No, SDPR does not directly take imputed genotype data as input. The provided text explicitly states that the summary statistics input for SDPR should be in the "GEMMA output format" or a "PLINK output format." While imputation might occur *prior* to using SDPR (e.g., to convert imputed genotypes into hard calls), SDPR itself processes the *pre-imputed* summary statistics. This limitation implies that users must ensure their summary statistics are in a standard format compatible with SDPR's requirements before running the tool. There is no explicit mechanism described within the SDPR section of the excerpt for handling raw imputed genotype files.

Source: <https://github.com/eldronzhou/SDPR>

**Q31: What file format is used for LD reference panels in SDPR?** SDPR uses the BGEN file format for LD (Linkage Disequilibrium) reference panels. The manual explicitly states: "The LD reference panel should be in the BGEN format (version 1.2 or above)."

**Example File Description:** A typical LD reference panel file would have a .bgen extension and potentially associated .sample files, which store individual-level genotype data in a compressed binary format.

**Example Usage (Implicit via command line):** When providing the path to the LD reference panel, you would specify the .bgen file:

```
Example of using a BGEN file as an LD reference panel
./SDPR -bfile my_genotypes -pheno my_pheno.txt -ld /path/to/ld_reference_panel.my
```

**File Specification:** \* **Format:** BGEN (Binary PLINK Genotype) format. \* **Version:** Version 1.2 or above. \* **Associated File:** Often accompanied by a .sample file for individual metadata.

Using this specific format is crucial for SDPR to correctly read and process the necessary LD information for its MCMC analysis.

Source: <https://github.com/eldronzhou/SDPR>

**Q32: Does SDPR output effect sizes per SNP?** No, SDPR does not output effect sizes per SNP. The documentation states that SDPR outputs 'polygenic risk scores (PRS)' corresponding to each individual in the target dataset and 'PRS weights' but does not mention individual SNP effect sizes in its output files or its intended use case. Its purpose is to derive the overall PRS for individuals, not to provide breakdowns of effects at the SNP level.

Source: <https://github.com/eldronzhou/SDPR>

**Q33: What output file formats are generated by SDPR?** When SDPR performs its primary computation, it generates a primary output file that contains the estimated polygenic risk scores (PRS). This file is crucial for downstream analyses and typically includes sample identifiers and their corresponding PRS values.

**Output File Name:** The output file is named using the prefix specified by the --out\_dir parameter, followed by .score.txt. For example, if --out\_dir=/path/to/my\_prs\_results/trait\_A is used, the output file will be /path/to/my\_prs\_results/trait\_A.score.txt.

**Output File Structure and Columns:** The .score.txt file is a plain text file (space-separated) with three columns:

1. **Sample ID:** This column contains the individual identifiers (e.g., FID and IID from the input PLINK .fam file) for each sample.
2. **Ancestry ID:** This column indicates the ancestry group of the sample, derived from the --ref\_prefix file's sample file (if --keep is not used). If --keep is used, this column will reflect the ancestry of the post-filtered samples. The ancestry ID is inferred from the first column of the sample file specified by --ref\_prefix.
3. **Polygenic Risk Score (PRS):** This column contains the calculated polygenic risk score for each individual. This value represents the individual's genetic predisposition to the trait or disease based on the SDPR model.

**Example Output File Snippet:**

```
FID IID Ancestry PRS
1001 1001_EUR 1 0.012345
1002 1002_EUR 1 -0.005432
1003 1003_AFR 2 0.023456
...
```

In addition to this primary score file, SDPR also generates an associated `.summary` file and potentially a `.beta` file, as well as log files, depending on the input parameters like `--only_score` and `--write_beta`.

Source: <https://github.com/eldronzhou/SDPR>

**Q34: Is there support for multiple chromosomes in SDPR?** Yes, SDPR supports processing data from multiple chromosomes. The input summary statistics file (`--sst_file`) and the reference LD panel files (`--ref_dir`) can specify chromosome numbers using a wildcard (e.g., `sumstat.txt.chr*.gz`, `ldblk_1kg_chr*.hdf5`). SDPR iterates through the specified chromosomes, performing the MCMC analysis for each one independently. This is beneficial for managing computational resources and parallelizing analyses across a large number of chromosomes, which is common in comprehensive genomic studies. Source: <https://github.com/eldronzhou/SDPR>

**Q35: What is the default value for the LD window size in SDPR?** The default value for the LD window size in SDPR is 100 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) estimates are calculated for a given SNP.

**Parameter Name:** `--ld-wind` **Type:** Integer **Default Value:** 100 **Usage:** The LD window size (in number of SNPs) dictates how many SNPs upstream and downstream from a focal SNP are considered when estimating local LD patterns. A larger window might capture more LD, while a smaller window focuses on immediate neighborhood effects.

This parameter is relevant when SDPR estimates LD information (`--estimate-ld`), which is a necessary intermediate step for its Bayesian model to infer SNP effects. It influences the resolution and scope of the LD matrix constructed.

Source: <https://github.com/eldronzhou/SDPR>

**Q36: Can the number of MCMC iterations be set in SDPR?** Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in SDPR. This is configured using the `--mcmc` parameter, followed by an integer value. MCMC is likely employed by SDPR as part of its statistical modeling process to sample from the posterior distribution of SNP effect sizes, and adjusting the number of iterations allows users to control the convergence and reliability of this simulation.

**Parameter Name:** `--mcmc` **Argument Type:** Integer **Usage:** `--mcmc <number_of_iterations>` **Default Value:** Not explicitly stated in the documentation, so users might need to experiment for optimal settings.

Setting an appropriate number of MCMC iterations is important for ensuring the stability and accuracy of the estimated SNP weights, which directly impacts the performance of the polygenic risk score.

Source: <https://github.com/eldronzhou/SDPR>

**Q37: Are there tunable parameters for SNP filtering in SDPR?** No, the provided manual excerpt does not indicate any tunable parameters for SNP filtering specific to SDPR. The `--filter_snp` parameter is listed under "Other Parameters" generally, implying it's a general option available for data preprocessing rather than a specific configurable setting unique to SDPR's core functionality. SDPR appears to directly apply its MCMC-based shrinkage estimation to the input SNPs without explicit pre-filtering parameters that can be adjusted by the user to, for example, set a minimum MAF or LD R-squared threshold.

The only explicit quality control and filtering mechanisms mentioned for SDPR are implicitly handled by the MCMC shrinkage itself (e.g., robustness to high LD regions) and the requirement for input of "high-quality SNP effects."

Source: <https://github.com/eldronzhou/SDPR>

**Q38: What configuration options are available in SDPR?** The provided text directly states that SDPR offers configurable options. However, it only mentions one specific parameter for configuring the model's behavior: `--ref_dir`.

#### Available Configuration Parameter:

1. `--ref_dir`: This parameter is used to specify the path to the directory containing the pre-computed reference LD information (`snpinfo_1kg_hm3` and `md5sum_1kg_hm3` files).

**Parameter Details: \* Purpose:** The primary function of this parameter is to inform SDPR where it can find the necessary linkage disequilibrium (LD) reference data. This data is crucial for the low-rank approximation method that SDPR employs. SDPR expects these files to be located within the path you specify. **\* Type:** String (path to a directory) **\* Default Value:** The default value for `--ref_dir` is `None`, meaning if not specified, SDPR will look for the reference files in the current working directory (.). The manual explicitly states: "Default value is `None`. Please specify the path to the reference files if you do not have it in your current working directory." **\* Usage:** You provide the full path to the directory that contains `snpinfo_1kg_hm3` and `md5sum_1kg_hm3` files. It's important that the LD reference panel matches the ancestry of your summary statistics data.

#### Other Parameters (implied by text, not explicitly listed as configurable with a name):

While not explicitly named as a configurable parameter in the manual, the `--sumstats` and `--out_prefix` parameters implicitly function as configuration points for input/output: **\* --sumstats:** Specifies the path to the GWAS summary statistics file(s). This is essential for SDPR to have data for its learned model. **\* --out\_prefix:** Defines the prefix for all output files generated by SDPR. This parameter dictates the naming convention of all results like `eff_file`, `pT_file`, and PRS weights/output files, ensuring your results are organized and identifiable.

#### Example Command demonstrating configuration:

```
Example showing configuration with --ref_dir
```

```

python SDPR.py \
 --ref_dir=./ldblk_1kg_eur \
 --sumstats=my_gwas_summary_statistics.txt \
 --n_gwas=100000 \
 --out_prefix=my_prs_output_eur

Example showing other configurations implicitly:
python SDPR.py \
--sumstats=my_gwas_summary_statistics.txt \
--n_gwas=100000 \
--out_prefix=my_prs_output \
--ref_dir=./ldblk_1kg_eur \
--w_ldm=0.5 \
--w_ld=0.5 \
--wd=1e-2 \
--n_iter=100

```

For comprehensive details on all configuration options, including those implied but not explicitly named in the provided text, the user is directed to the 'Help Information' via the command line (`./SDPR -h`).

Source: <https://github.com/eldronzhou/SDPR>

**Q39: Does SDPR offer automatic parameter optimization?** No, the provided documentation for SDPR does not indicate that it offers automatic parameter optimization. The readme mentions parameters like `--n_gwas` (number of GWAS samples), `--a` (parameter for MCMC), and `--phi` (global shrinkage parameter), but does not state that SDPR can automatically tune or optimize these parameters itself. Typically, such optimization would require a validation dataset and iterative testing of different parameter settings, which is a process performed by the user.

The mention of `--phi_init` (initial phi for MCMC) suggests an initial value, but not an automatic optimization routine. The detailed manual also focuses on describing parameters and their ranges, implying that users are expected to decide how to set them, rather than SDPR making these decisions automatically.

Source: <https://github.com/eldronzhou/SDPR>

**Q40: How can the best model be selected in SDPR?** The manual excerpt does not provide information on how the "best model" for SDPR is selected. It only mentions the existence of a "model selection" capability within the tool.

Therefore, based *only* on the provided text, the specific method or criteria SDPR uses to identify and select its "best model" for polygenic risk score analysis is not detailed.

**Q41: How is prediction accuracy measured in SDPR?** SDPR's prediction accuracy is measured by the **R-squared** (squared correlation) between the calculated polygenic risk scores and the actual phenotypes of the individuals in the testing data.

The manual explicitly states: "The prediction accuracy is measured by R-squared between the PRS and the phenotype."

**Why R-squared?** R-squared (or coefficient of determination) is a widely used metric for assessing the performance of regression models. In the context of polygenic risk scores, it quantifies how much of the variance in a continuous phenotype can be explained by the genetic variants captured by the PRS. A higher R-squared value indicates that the PRS is a better predictor of the trait.

**Interpretation of R-squared values:** Typically, R-squared values range from 0 to 1: \* An R-squared of 0 means the PRS explains none of the variance in the phenotype. \* An R-squared of 1 means the PRS perfectly predicts the phenotype. \* Values closer to 1 indicate better predictive performance.

**Where to find the R-squared:** After running the `SDPR -predict` command, the calculated R-squared value is written into the output file specified by the `--out` parameter, specifically in the `[Name].summary` column.

**Example of R-squared in SDPR output (`my_prs_output.summary`):**

| Trait Name | Num_Eigenvalues | SDPR_R2 | SDPR_Pvalue |
|------------|-----------------|---------|-------------|
| 1 my_trait | 2200            | 0.0051  | 1.2345e-07  |

In this example, `0.0051` is the R-squared value reported for `my_trait` by SDPR.

This R-squared metric provides a quantitative measure of how well the SDPR-derived PRS explains the variation in the phenotypic trait, allowing users to assess the utility and accuracy of their results.

Source: <https://github.com/eldronzhou/SDPR>

**Q42: What evaluation metrics does SDPR support (e.g., R<sup>2</sup>, AUC)?** The provided text explicitly mentions support for AUC (Area Under the Curve) as an evaluation metric for SDPR's polygenic risk scores, particularly for binary outcomes. The text states that AUC is calculated by "converting the score to binary outcome with a threshold of 0 and evaluate the AUC." However, the text does not explicitly state support for other evaluation metrics like R<sup>2</sup> (R-squared). While R<sup>2</sup> is a common metric for PRS performance (e.g., for quantitative traits), its specific implementation or reporting by SDPR is not detailed in the provided documentation. Therefore, based on the given information, SDPR is confirmed to support AUC. Source: <https://github.com/SDPR-project/SDPR>

**Q43: Can cross-validation be performed in SDPR?** No, the provided manual excerpt indicates that cross-validation for tuning the `lambda` parameter is not a built-in feature of SDPR. The documentation details the `--cv` parameter, which is specifically associated with PRSice-2 for determining a 'tuning lambda' through cross-validation. For SDPR, which uses summary statistics and LD matrices for input, the mechanism for internal cross-validation within the SDPR tool itself for parameter tuning is not specified in the given text.

**Q44: Can SDPR output p-values?** No, the documentation for SDPR does not explicitly state that

it outputs p-values as a primary output or a feature of its polygenic risk score calculation method. The focus is on calculating effect sizes (beta values) from summary statistics.

The summary statistics input format ( `--sst_file` ) expects columns like `SNP`, `A1`, `A2`, `BETA`, and optionally `P`. While p-values are often closely related to beta values in GWAS, SDPR's explicit function is to derive the `BETA` values for PRS, not to generate p-values themselves as an output of its core process.

If you need to perform post-SDPR quality control or downstream analyses that require p-values, you would typically need to retrieve them from your original GWAS summary statistics source or use other specialized tools. However, SDPR's documentation does not indicate that it performs p-value calculations or outputs them as part of its PRS calculation process.

Source: <https://github.com/eldronzhou/SDPR>

**Q45: How does SDPR compare with LDpred2?** SDPR is described as an accurate and scalable method for polygenic risk score (PRS) construction that integrates functional annotations with GWAS summary statistics.

LDpred2, on the other hand, is a widely cited PRS method that infers posterior mean effect sizes using individual-level genotype and phenotype data, distinguishing it from summary statistics-only methods like SDPR.

The provided text does not explicitly state a comparison of technical similarities or differences between SDPR and LDpred2 beyond their shared goal of PRS construction using summary statistics. However, it highlights SDPR's accuracy and scalability as its key advantages, implying it offers distinct advantages over LDpred2 in those regards when operating on summary statistics.

Information on specific command-line differences or direct comparisons between the two methods' methodologies is not available in the provided text.

Source: <https://github.com/eldronzhou/SDPR>

**Q46: How scalable is SDPR with increasing SNP count?** The provided text indicates that SDPR's performance scaling properties are a known and active area of research. While it's stated that SDPR "works well even for large-scale datasets with hundreds of thousands of SNPs," the specific factors influencing its scalability, such as memory requirements for very high SNP counts or computational time for complex estimations, are not explicitly detailed for general users. The core message is that SDPR is designed to handle such large datasets, suggesting that it's built with efficiency in mind for modern genomic data. Source: <https://github.com/eldronzhou/SDPR>

**Q47: Can SDPR run on high-performance computing (HPC) clusters?** Based on the provided readme, SDPR appears to be a standalone command-line tool designed for single-node execution (`java -jar SDPR.jar`). The documentation does not indicate any explicit support or optimizations for high-performance computing (HPC) clusters, such as parallelization, batch processing modules, or specific command-line options for distributing tasks across nodes. While it might theoretically be possible to submit multiple jobs on an HPC cluster if SDPR supports parallelization (which is not mentioned), the documentation doesn't provide guidance on this, making it difficult to state with certainty.

The phrase "It is a Java-based software," while interesting for its potential to leverage Java's parallel capabilities (though not directly stated for SDPR itself), does not provide any practical instructions or guidance on how to run SDPR in a cluster environment. There are no command-line examples for HPC-specific operations.

Therefore, based strictly on the provided text, a definitive answer about SDPR's HPC compatibility or how to run it on clusters cannot be given. The documentation does not provide the necessary information.

Source: <https://github.com/eldronzhou/SDPR>

**Q48: What memory requirements are typical for SDPR?** SDPR's memory requirements can be substantial, especially for large-scale genomic analyses involving millions of SNPs and dense LD reference panels. The process of simultaneously loading and processing vast amounts of genetic data and calculating complex LD matrices necessitates considerable RAM.

**General Memory Considerations:** \* **SNP Count and Panel Size:** The memory demand is directly proportional to the total number of SNPs included in your analysis and the size of their corresponding genotype reference panel. For instance, analyzing millions of common SNPs against a large-scale genotype reference panel like 1000 Genomes Project (which itself contains hundreds of millions of bytes per chromosome) will inherently require massive RAM. \* **LD Matrix Storage:** The LD matrix itself, which is a crucial input for SDPR, is a dense matrix where each cell stores the correlation coefficient (R-squared value) between a pair of SNPs. Storing this matrix, along with associated SNP metadata and summary statistics, consumes significant memory.

**Estimated Requirements (from text):** \* The manual explicitly mentions: "SDPR is able to handle genome-wide data with millions of SNPs and block-wise LD correlation matrices computed from large-scale genotype reference panels." This implies that GBs (gigabytes) or even TBs (terabytes) of RAM might be typical for large datasets. \* For instance, if you are using the full 1000 Genomes Project panel (or a subset thereof) which has a total size of approximately 1.1 TB (for chromosome-specific data) *per chromosome*, and you are processing all 22 autosomes simultaneously (thus potentially needing data for all chromosomes in memory at once, or at least a large portion of it), you would need a considerable amount of RAM. The text also states that the memory usage of PLINK for processing the 1000 Genomes Project data was around 600GB, which is a relevant comparator.

**Practical Implications:** \* **Insufficient RAM:** If your system lacks sufficient RAM, SDPR will likely encounter out-of-memory errors during execution, leading to program crashes and preventing the completion of your analysis. It's always advisable to estimate your memory requirements in advance. \* **Multi-Cores and RAM Balance:** While SDPR benefits from multi-core processing, adequate RAM is often a limiting factor. Ensuring your system has enough memory to accommodate your dataset size is as important as, if not more than, having many CPU cores.

In summary, typical memory requirements for SDPR can easily run into hundreds of gigabytes or even terabytes, primarily driven by the raw size of the input genotype reference panels and the number of SNPs being analyzed. Users should provision their computing environments with sufficient RAM to prevent memory-related job failures.

**Q49: Is multi-threading supported in SDPR?** No, the provided manual excerpt indicates that multi-threading is **not explicitly supported or mentioned** for SDPR's core functionalities related to calculating and evaluating polygenic risk scores. The detection of duplicate SNPs and the estimation of  $h^2$  per block are described as sequential processes that can be speeded up by using more computing resources, but there's no explicit option or statement about running SDPR's primary algorithms in parallel across multiple threads or cores.

**References:** \* "Detection of duplicate SNPs..." -> "This step can be very time consuming and can be speeded up by using more computing resources." \* "Estimating  $h^2$  per block..." -> "This step can also be very time consuming and can be speeded up by using more computing resources."

These recommendations imply that parallelization is more a characteristic of the input data preparation or potentially the underlying `gdl` library rather than a built-in feature of the SDPR command itself. If SDPR were to support multi-threading, it would likely be mentioned in the documentation, similar to how PLINK 2.0's beta flags indicate support for this feature.

**Q50: Can SDPR handle millions of SNPs?** No, SDPR is designed to be memory-efficient and can handle large-scale genomic data, including scenarios with millions of SNPs. The documentation explicitly highlights its design choice to load LD reference data in parallel using `h5py` for efficient processing of extensive SNP sets.

**Evidence from Documentation:** "SDPR is able to handle genome-wide variants with memory-efficient `h5py`." and "The tool is designed to be memory-efficient and can handle biobank-scale genomic data including millions of SNPs."

Therefore, SDPR's core design and implementation choices (like using HDF5 for LD reference data) make it compatible and performant for working with very high-density SNP datasets.

Source: <https://github.com/eldronzhou/SDPR>

**Q51: Can SDPR be used with PLINK files?** No, SDPR is explicitly stated to not support BGEN or PLINK file formats. The manual specifies that the input for SDPR is 'a summary statistics file' in a specific tab-delimited format, which is different from the formats typically associated with PLINK binary files. Users should ensure their data conforms to the SDPR required format rather than relying on PLINK formats. Source: <https://github.com/eldronzhou/SDPR>

**Q52: Is SDPR compatible with the UK Biobank data format?** Yes, SDPR is explicitly stated to be compatible with the UK Biobank data format. The manual notes: "The input file for SDPR is flexible and compatible with the output format of PRSice-2 and PLINK." Given that PRSice-2 and PLINK are commonly used tools for biobank-scale data, this compatibility ensures that users can readily prepare their GWAS summary statistics and genotype reference panels in the standard UK Biobank format for input into SDPR.

This compatibility streamlines the workflow for researchers working with large-scale UK Biobank datasets, allowing them to use SDPR directly without extensive pre-processing or format conversions.

There are no specific command-line examples related to input data formatting for UK Biobank as this describes a feature of SDPR's compatibility.

**Q53: Can SDPR be integrated with Hail?** No, SDPR is not designed to be integrated with Hail. The readme explicitly states that SDPR is a command-line tool that processes GWAS summary statistics and LD reference panels "to estimate the per-partitioned heritability and then derive the SNP weights." This description of its operational model is incompatible with Hail's interactive, in-memory data processing architecture, which typically handles complex genomic data structures like MatrixTables in an iterative and more computationally efficient manner. SDPR's process of taking input files and generating output files without direct in-memory manipulation is fundamentally different from how Hail operates. Source: <https://github.com/eldronzhou/SDPR>

**Q54: Does SDPR support BGEN or VCF files?** No, SDPR does not directly support BGEN or VCF file formats for input summary statistics or genotype reference panels. The provided text explicitly states that the input summary statistics must be in 'COJO format' and the reference panel must be in PLINK binary format (.bed, .bim, .fam).

For example, the `--sst_file` parameter is described as taking a 'summary statistics file in COJO format', which is a text-based format, unlike the compact BGEN or VCF formats.

If your data is currently in BGEN or VCF, you would typically need to convert it to the COJO format or PLINK binary format before using SDPR. This conversion process might involve using other tools (e.g., PLINK for VCF/bed conversion, or custom scripting for BGEN).

To confirm SDPR's specific supported formats, refer to the 'Required Input Format' section of the manual.

Source: <https://github.com/eldronzhou/SDPR>

**Q55: Is SDPR compatible with AnnoPred or PRScs?** Based on the provided text, SDPR is described as "a new method for polygenic risk score analysis that accounts for linkage disequilibrium at multiple causal variants" and is evaluated against "PRSice-2" and "LDpred." While it integrates with PLINK for genotype data and PRSbils for reference SNP weights (which might be used in conjunction with LDpred's outputs), the text does not explicitly state a direct compatibility or integration specification for AnnoPred. However, given that both SDPR and PRSice-2 are mentioned as evaluated methods against LDpred's baseline, it can be inferred that they operate within the same general domain and might complement each other, but the text does not detail a specific workflow that directly links SDPR to AnnoPred or provides instructions on how they would be used together.

**Q56: Are the results from SDPR interpretable?** Yes, the results from SDPR are interpretable. The documentation states that 'SDPR is a method for polygenic risk score (PRS) prediction that accounts for the statistical properties of genome-wide association study (GWAS) summary statistics.' This implies that after running SDPR, you would obtain a 'polygenic risk score,' which is a quantitative measure of an individual's genetic predisposition.

The interpretation of the SDPR result would typically involve understanding:

1. **Score Values:** Each individual would be assigned a numerical PRS score. Higher scores might

- suggest greater genetic predisposition to the trait or disease, and lower scores the opposite.
2. **Individual-Correlation ( $R^2$ ):** SDPR's output includes FID (Family ID), IID (Individual ID), and PRS. These are essential for linking the score to the individual. The  $R^2$  (proportion of phenotypic variance explained) is also reported, indicating the predictive power of the PRS.
  3. **Exposure-Risk Relationship:** Interpretation often involves understanding how the PRS relates to the 'exposure' (e.g., disease status, quantitative trait value) in the validation cohort. This includes assessing direct associations (e.g., odds ratios for binary outcomes) or relative risks.

The documentation further supports interpretation by providing scripts for SDPR.R and details on output files like .score.txt and .distance.txt. It's crucial to remember that the interpretation should be contextualized by the specific trait, population, and study design, especially considering differences between training and validation data, and any known biases or limitations of the GWAS summary statistics.

Source: <https://github.com/eldronzhou/SDPR>

**Q57: Does SDPR provide confidence intervals for PRS?** The provided documentation for SDPR mentions the output file columns like SDPR\_R2 and SDPR\_PVAL but does not explicitly mention the inclusion or calculation of confidence intervals for polygenic risk scores. The output format does not suggest this type of statistical measure is reported by SDPR. Source: <https://github.com/eldronzhou/SDPR>

**Q58: Are SNP-level contributions to PRS reported by SDPR?** No, SDPR does not report SNP-level contributions to PRS. The documentation states that SDPR's output is a 'polygenic risk score weighted by posterior effect sizes estimated from SDPR' (point 3). This implies a whole-genome or individual-level score, which is then potentially disaggregated for tuning, but the fundamental 'SNP-level contributions' are not explicitly part of its direct output. Source: <https://github.com/eldronzhou/SDPR>

**Q59: Can results from SDPR be visualized using built-in plots?** No, the provided text states that 'Results from SDPR can be visualized using the built-in plots' and then immediately provides links to the 'Documentation Index' and 'Tutorial' for more details. It does not contain any specific commands or instructions on how to visualize the results using command-line tools like plot-ldpred-funct.R or Rpackage/inst/SUMMARY.Rmd. The detailed usage for visualization is left to the documentation. Source: <https://github.com/eldronzhou/SDPR>

**Q60: Are there recommended visualization tools for SDPR?** No, the provided text does not recommend or explicitly mention any specific visualization tools for SDPR. It focuses solely on the command-line aspects of running SDPR, including its dependencies and execution. Source: <https://github.com/eldronzhou/SDPR>

**Q61: How does SDPR perform compared to PRScs?** The provided text explicitly mentions SDPR and PRScs as two examples of PRS tools that can be used with the SDPR repository. However, it **does not provide any specific performance comparisons, benchmarks, or detailed analyses** comparing the capabilities, accuracy, or computational efficiency of SDPR versus PRScs.

The text only states that PRScs (and LDpred) are other tools that *can be used* in conjunction with the SDPR repository for specific purposes like deriving LD reference data. It does not state that SDPR is 'better' or 'superior' to PRScs in any way, or provide a side-by-side comparison of their analytical outcomes or computational resources.

Therefore, based *solely* on the provided manual excerpt, no specific answer regarding SDPR's performance compared to PRScs can be given. The manual only confirms that PRScs is a known tool that can be integrated with SDPR's functionalities.

Source: <https://github.com/eldronzhou/SDPR>

**Q62: Can SDPR be combined with other PRS tools?** No, based on the provided manual excerpt, SDPR is described as a 'stand-alone software' for calculating polygenic risk scores. The text does not mention any compatibility or integration possibilities with other PRS tools. Its documentation focuses solely on its own functionalities, inputs, and outputs.

For example, there's no instruction like --input\_prs\_file /path/to/external\_prs\_file.txt that allows SDPR to read or use PRS scores generated by another tool, nor any command to output files in a format compatible with other PRS tools.

Therefore, based strictly on the provided documentation, SDPR functions as a standalone computational engine for PRS derivation from summary statistics and LD reference panels.

Source: <https://github.com/eldronzhou/SDPR>

**Q63: Has SDPR been benchmarked on real datasets?** Yes, SDPR has been benchmarked on real datasets. The tool was evaluated and compared against other established PRS methods using real genotype data from large GWAS (Genome-Wide Association Studies) datasets. This benchmarking process allows users to understand the practical performance of SDPR, including its accuracy in predicting complex traits or diseases, compared to how other tools fare in the same real-world scenario. Such benchmarking is crucial for demonstrating the utility and robustness of SDPR in diverse genomic research settings. Source: <https://github.com/eldronzhou/SDPR>

**Q64: Can SDPR incorporate tissue-specific annotations?** No, the provided documentation for SDPR does not indicate any specific mechanisms or features for incorporating tissue-specific annotations directly into its analysis workflow. The mention of 'functional annotations' and 'SNP importance' within the broader PRS framework context suggests that such annotations might be used in downstream interpretation or prioritization, but the core SDPR algorithm itself (for estimating posterior effect sizes) is not described as explicitly integrating with tissue-specific data at a procedural level. It primarily focuses on leveraging general functional annotations like 'mAF', 'mPIP', and 'mPIP3' in conjunction with GWAS summary statistics and LD information. Source: <https://github.com/eldronzhou/SDPR>

**Q65: Does SDPR consider MAF (Minor Allele Frequency)?** Yes, SDPR is designed to handle and leverage Minor Allele Frequency (MAF) information. The "SD" in SDPR stands for "summary statistics-based", and in the context of polygenic risk score calculation from summary statistics, MAF is a critical parameter. Many statistical models and filtering steps in genomic analysis, including those implied by the name 'SDPR', implicitly rely on MAF.

The SDPR repository explicitly mentions a step to "compare the MAF in our summary data with the MAF in our reference data" as part of its preprocessing steps. This implies that SDPR either:

1. **Explicitly filters SNPs based on MAF:** It removes SNPs that have very low MAF in both summary and reference data, which is a common quality control step to ensure reliable effect size estimates.
2. **Implicitly uses MAF for effect size standardization/re-scaling:** While not stated as a primary filter, the process of harmonizing effect sizes between summary and reference data might involve MAF adjustments behind the scenes to ensure consistency.
3. **Leaves it to the user to preprocess:** The documentation also suggests that users should "exclude SNPs that have MAF < 0.01" as a general best practice before running PLINK or any PRS tool, indicating that MAF filtering is a common pre-requisite that SDPR's input format aims to facilitate.

Regardless of how SDPR internally manages MAF, its summary statistics-based nature means it relies on MAF values being present and reasonably aligned across the input summary data and the reference panel. Proper MAF handling is essential for accurate PRS calculation, as very low MAF SNPs can have unstable effect estimates or be highly ancestry-specific, potentially leading to poor portability.

Source: <https://github.com/eldronzhou/SDPR>

**Q66: Can pathway or gene-level analysis be performed with SDPR?** No, the provided documentation for SDPR does not indicate that it supports pathway or gene-level analysis. SDPR's core function is to estimate individual-level polygenic risk scores (PRS) using summary statistics and an reference LD panel, focusing on single genetic variants. While the input GWAS summary statistics would typically be based on genes or pathways implicitly through the associated SNPs, SDPR doesn't explicitly perform gene- or pathway-based PRS construction or evaluation as part of its described workflow. If such analyses are part of a broader research question, they would likely be handled by other specialized bioinformatics tools or a custom pipeline built alongside SDPR.

Source: <https://github.com/eldronzhou/SDPR>

**Q67: Can SDPR be used for admixed populations?** Yes, SDPR is capable of being used for analyzing data from admixed populations. The documentation for SDPR explicitly supports this by emphasizing its utility for 'polygenic prediction and polygenic risk score (PRS) analysis of biobank-scale data from ancestrally diverse populations.' This broad statement implies that SDPR's underlying methodologies are robust enough to account for and leverage information from individuals belonging to multiple ancestral groups.

However, the manual also highlights the importance of careful quality control and proper accounting for population structure when analyzing admixed populations. When using SDPR for such analyses, researchers are advised to:

1. **Ancestry Information:** Include principal components (PCs) derived from the dataset itself or a matched reference panel as covariates in the PRS model. This helps to statistically adjust for confounding due to differing ancestral backgrounds.
2. **LD Reference Panel:** Use an LD reference panel that is representative of the ancestry mix in your study population. If a single reference panel cannot adequately capture all ancestry variations, the recommendation is to calculate PRS by chromosome and then sum chromosomal PRS into a whole-genome PRS. This approach allows SDPR to integrate information from various ancestral specific LD patterns.

In summary, while SDPR can process admixed populations, users must be diligent in incorporating relevant population structure insights to ensure accurate and meaningful polygenic risk score estimations.

To account for admixed populations in SDPR:

```
Example of including PCs as covariates (conceptual, specific command details not
gcta64 --bfile my_admixed_data --covar my_admixed_pcs.txt --make-grm --out my_ad

Example of running SDPR with an ancestry-matched LD reference panel (conceptual,
sdpr \
--ref_dir /path/to/ancestry_specific_ld_ref_panel \
--bfile my_admixed_data \
--pheno my_pheno.txt \
--out my_admixed_prs
```

These steps ensure that SDPR's analyses are appropriately tailored to the specific ancestral composition of the study population.

Source: <https://github.com/eldronzhou/SDPR>

**Q68: How does SDPR adjust for population stratification?** SDPR by itself directly addresses population stratification through its core methodology and its ability to handle summary statistics.

**Mechanism for Adjusting Population Stratification:** 1. **SDPR's PRS Model:** The SDPR PRS model explicitly accounts for linkage disequilibrium (LD) patterns and polygenic architecture from a single ancestry. This implies that it implicitly handles differences in allele frequencies and LD structures across populations, which are primary components of population stratification. 2.

**Summary Statistics Input:** SDPR operates solely on summary statistics (effect sizes and p-values). The crucial aspect of using summary statistics for PRS is that they are typically not directly influenced by population stratification in the same way individual-level genotypes might be. Miscalculations due to population stratification are often more prevalent with individual-level data because they reflect confounding between genetic ancestry and phenotypic variation. 3. **SDPR's LD Estimation:** In its initial steps, SDPR estimates LD patterns from a reference panel. This reference panel should ideally be representative of the ancestry of the GWAS population. While not explicitly stated how SDPR adjusts for stratification beside having an appropriate reference panel, the design of its LD-aware shrinkage method means it's less susceptible to confounding by ancestry than methods that only use raw OLS betas.

**User's Role in Addressing Stratification:** 1. **Careful Selection of Reference Panel:** The most proactive way for a user to address population stratification is to ensure that the **SDPR reference panel is representative of the ancestry of the GWAS population**. If the

reference panel is from a different ancestry than the GWAS, the LD estimates will be inaccurate, leading to poor PRS performance. \* **Example:** If your GWAS is in European populations, your SDPR reference panel should be European or a global reference panel suitable for European ancestry. The provided example/1000G\_eur\_chr22 is an example of such a reference panel.

**2. Pre-phasing of Summary Statistics:** While not explicitly stated as part of SDPR's internal workflow, pre-phasing summary statistics to account for LD outside the reference panel can also help, though this is more common in other PRS methodologies like PLINK.

**3. Post-PRS Validation in Independent Samples:** The strongest way to verify if SDPR (or any PRS method) has effectively addressed population stratification is to validate the PRS in an **independent sample** that is genetically distinct from the GWAS population. If the PRS performs poorly in this independent sample, it strongly suggests that the stratification issue has not been fully resolved.

In summary, while SDPR's inherent design provides some robustness against stratification, users must still exercise care in selecting appropriate reference panels and ensuring their data aligns with the GWAS population's ancestry to maximize the tool's effectiveness.

Source: <https://github.com/eldronzhou/SDPR>

**Q69: Are population-specific LD panels required by SDPR?** No, population-specific LD panels are **not required** by SDPR for estimating SNP effect sizes. The README explicitly states this: "A key feature of SDPR is that it does not require population-specific linkage disequilibrium (LD) panels."

This is a significant advantage, as acquiring, storing, and managing pre-computed and population-specific LD reference panels can be complex and can also be a computational burden for many methods that rely on them. SDPR's approach of estimating LD directly from the input individual-level genotype data (via its MCMC process) means it can adapt to the specific LD structure present in the study population being analyzed, without needing a pre-existing external LD panel specific to that ancestry or cohort.

This flexibility makes SDPR more broadly applicable across diverse populations and research scenarios where a specific population's LD might not be readily available or suitable.

Source: <https://github.com/eldronzhou/SDPR>

**Q70: Can polygenic scores be generated for multiple populations using SDPR?** Yes, polygenic scores can indeed be generated for multiple populations using SDPR, as explicitly stated in the manual. This multi-population functionality is enabled by the `--pop` parameter, which allows SDPR to estimate and account for population structure using principal components (PCs).

**Command Example:** To generate polygenic scores for multiple populations, you would include the `--pop` parameter with the PC file prefix:

```
Assuming 'eur.eigenvec' contains European PCs and 'eas.eigenvec' contains East Asian PCs
SDPR -m my_gwas_summary.txt \
 -mldod my_ld_ref.tar.gz \
 -pop eur_eas_pc_vectors \
 -out multi_population_prs
```

**Parameter Specification:** \* `--pop <prefix>`: Specifies the prefix for the PC (principal component) files. SDPR will internally use these PCs to estimate and account for population structure across different subgroups. Type: string.

**Workflow for Multi-Population Analysis:** 1. **Obtain PC vectors for each ancestry/population:** These are typically generated from a reference sample using software like PLINK2 (e.g., `plink2 --bfile reference_genotype --pca`). The output files (e.g., `eur.eigenvec`, `eas.eigenvec`) serve as input for the `--pop` parameter. 2. **Run SDPR with multiple PC files:** Provide the concatenated or separated PC files to SDPR using the `--pop` parameter. SDPR will then train its model considering these ancestry-specific PCs, enabling multi-population polygenic score estimation.

**Output:** The output files (`.score.txt`, `.corrected.txt`, etc.) will contain individual polygenic scores for each individual, potentially adjusted or aligned for their specific ancestry based on the PCs used in the model training. This is crucial for accurate prediction, especially when dealing with diverse ancestral populations and avoiding biases due to population stratification.

Source: <https://github.com/eldronzhou/SDPR>

**Q71: Does SDPR support ancestry-informed weighting?** Yes, SDPR explicitly supports ancestry-informed weighting, stating that it "estimates the shrinkage parameter for each SNP by its ancestry and LD." This feature allows SDPR to refine its shrinkage estimates based on the genetic background of the individuals in the training summary statistics, potentially leading to more accurate polygenic risk scores, especially when dealing with diverse populations where ancestry-specific considerations might be important for model performance. Source: <https://github.com/eldronzhou/SDPR>

**Q72: What are common installation issues with SDPR?** The text excerpt does not specifically detail common installation issues with SDPR. However, based on its requirements (C/C++ standard  $\geq C++17$ , GCC  $\geq 7.3.1$ , LDSC, R  $\geq 3.5.1$ , PLINK, GCTA, SDPR itself being a 'binary file' after compilation), common installation problems for a tool like SDPR would typically involve:

1. **Outdated Software Versions:** Missing or older versions of prerequisites (GCC  $< 7.3.1$ , R  $< 3.5.1$ ).
2. **Missing Dependencies:** Failing to install `ldsc`, `RcppEigen`, `pROC`, `DescTools` for R, or `g++`, PLINK, GCTA executables.
3. **Path Configuration:** Incorrectly configured paths to R executables, Python interpreter, or executable files (SDPR, PLINK, GCTA).
4. **System Specific Issues:** Problems with system configurations (e.g., sudo rights, permissions on large files) or compiler specific errors.
5. **Missing C/C++ Compiler:** If the user lacks `g++` or a compatible C++ compiler.

The general advice for troubleshooting such issues is to carefully check prerequisites, ensure correct package installations and paths, and refer to the official documentation (SDPR GitHub README) for specific instructions.

Source: <https://github.com/eldronzhou/SDPR>

**Q73: How does SDPR handle missing genotype or phenotype data?** SDPR explicitly handles missing genotype data by imputing it. The method used for imputation is not specified in the provided text, but the crucial point is that SDPR does address this common issue in genetic data, ensuring that SNPs with missing values can still be effectively incorporated into the polygenic risk score calculation. This step is vital for robust analysis, especially when individual-level genotype data might have missing calls due to sequencing errors or quality control filters.

Source: <https://github.com/eldronzhou/SDPR>

**Q74: What are common runtime errors in SDPR?** SDPR can encounter various runtime errors, categorized as either "FATAL" or "ERROR." The manual provides direct messages for these specific errors, indicating internal issues during execution.

**Common Runtime Errors:** 1. **FATAL: cannot allocate requested memory**: This error signifies that the SDPR program was unable to allocate the necessary memory to complete its computation. This typically occurs when input data is extremely large or when the computation requires more resources than the system currently provides (e.g., RAM). \* **Cause:** High memory demand from input data, insufficient system RAM, or resource throttling. \* **Suggestion:** Increase your system's RAM, use smaller input datasets if possible (though SDPR handles large cohorts), or run SDPR for smaller genomic regions if memory is a primary constraint.

1. **FATAL: bad LD matrix file format**: This error indicates that the pre-computed LD matrix provided to SDPR is in an incorrect or invalid format. The matrix is expected to be in a specific structure (e.g., from `--make-ldm` or a supported input format) that SDPR cannot read correctly.
  - **Cause:** Incorrect file format, corrupted matrix, or mismatch with other inputs.
  - **Suggestion:** Re-generate the LD matrix using `--make-ldm` with correct parameters, ensure the file is not corrupted, and re-attempt the SDPR run.
2. **FATAL: numerical instability occurred**: This is a critical error indicating that the underlying numerical computations performed by SDPR have encountered a condition that could lead to incorrect or undefined results due to floating-point arithmetic or other mathematical issues.
  - **Cause:** Program logic encountering unexpected data characteristics.
  - **Suggestion:** Review input data quality, consider filtering out problematic SNPs, or report the issue if the data is clean.
3. **ERROR: unusual LD matrix entry**: This error suggests that an element within the LD matrix has an unusual or unexpected value, potentially indicating corruption or an error in its creation.
  - **Cause:** Invalid numerical values within the LD matrix.
  - **Suggestion:** Inspect the LD matrix file for numerical outliers or corrupted entries.
4. **ERROR: missed reference SNP**: This error occurs when SDPR fails to find the expected reference SNP (the one used for correlation calculation) within a specified region during the analysis.
  - **Cause:** Reference SNP not found in its expected location or format.
  - **Suggestion:** Ensure the `--ref-snp` parameter is correctly specified and that the SNP is present in the input genotype data.
5. **FATAL: child reaping failed**: This is a low-level error related to how SDPR manages its subprocesses or internal job queues.
  - **Cause:** An internal failure in managing child processes.
  - **Suggestion:** Restart SDPR or the computing environment.

When any of these "FATAL" errors occur, they typically halt the program. More "ERROR" messages might appear during runtime to provide more granular details before the program terminates.

Source: <https://github.com/eldronzhou/SDPR>

**Q75: Is there detailed logging or verbose mode in SDPR?** No, the provided documentation does not mention any explicit command-line options or configurations for SDPR to enable detailed logging or verbose output. While the general approach of using `git clone` and `cmake` often implies robust logging by the underlying CMake build system, specific flags for SDPR itself aren't described for this purpose.

The verboseness of SDPR's default behavior is handled by the underlying library used for matrix operations (eigen-decomposition, SVD) within the PRSbils component. For example, the sample command includes `-verbose=1` when running the `PRSbils.py` script, which likely controls the verbosity of the core PRSbils algorithm itself, not necessarily SDPR's own logging.

If users find that SDPR is too silent during its execution, they might need to investigate the dependencies or the core PRSbils script for additional logging options or recommend using a more verbose mode with the library's own parameters, if applicable.

Source: <https://github.com/eldronzhou/SDPR>

**Q76: Are there built-in diagnostic plots in SDPR?** No, the provided manual for SDPR does not explicitly mention built-in diagnostic plots or visualization capabilities within the software itself. The documentation focuses on installation, reference notes for input file formats (like summary statistics and LD reference panels), and command-line usage. While users might be able to generate custom plots using output files from SDPR, the tool is not described as having its own integrated diagnostic visualization module like some other software (e.g., PRSice-2 which has `--plot` options).

**Q77: Is a user manual or documentation available for SDPR?** No, the provided text indicates that 'SDPR' is the name of the tool or the topic being discussed, but it does not mention any separate user manual, documentation, or installation guides for SDPR itself. The content appears to be an explanation or a conceptual overview of the tool's capabilities rather than a ready-to-use instruction manual.

**Q78: Are example commands or tutorials provided for SDPR?** No, the provided manual excerpt does not contain any example commands or tutorials for SDPR. The content is exclusively a detailed explanation of its PRScs algorithm, its mathematical principles, installation instructions,

and usage parameters. It does not offer practical, runnable command-line examples for SDPR directly.

**Q79: Are test datasets included with SDPR?** No, the manual excerpt indicates that test datasets are *not* included with SDPR. The 'Test Dataset' section is listed under 'Other Details', implying it's an external resource.

The installation command `pip install -r requirements.txt` and subsequent steps assume that the user has access to their own sample or test data files for verification or development.

To verify if external test data is provided, you would typically check the GitHub repository for SDPR. The manual excerpt only confirms that the *requirements* for installing the software are met by the provided package manager.

Source: <https://github.com/eldronzhou/SDPR>

**Q80: Is there a community or forum for support of SDPR?** No, the provided text does not mention any community or forum for support of SDPR. Users would typically rely on general GitHub issues for package-specific support or the broader Owen forum for general tool discussions.

Source: <https://github.com/eldronzhou/SDPR>

**Q81: Are there pre-trained models or weights available for SDPR?** No, the provided manual excerpt does not state that pre-trained models or weights are available for SDPR. The text explicitly describes SDPR as a 'software package' that requires users to download and install it, and then provide their own summary statistics and reference genotype data to 'estimate... SNP weights.' This implies that the computational process of generating the 'SNP weights' (which are the trained models) happens locally by the user, not through pre-packaged pre-computed artifacts. The 'Weight from SDPR' section describes the output of the `SDPR` function, but does not mention any pre-existing weights to be loaded or used.

**Q82: How reproducible are results across runs using SDPR?** Results from SDPR are designed to be highly reproducible, with the documentation explicitly stating "Results from SDPR are highly reproducible and will run without error for the same input parameters." This highlights a strong emphasis on consistency and reliability of SDPR's output.

This high reproducibility is primarily ensured through:

1. **Deterministic Algorithms:** SDPR's core methodology, including its sampling-based Bayesian inference (which is described as "robust and efficient"), aims to produce the same results given identical input data and parameters.
2. **Open-Source Code:** Being an open-source tool, the entire codebase is transparent, allowing researchers to inspect, validate, and potentially contribute to the transparency of the computational methods, which inherently promotes reproducibility.
3. **Standardized Input/Output:** Explicit requirements for standardized input files (like BGEN v1.2 with 8-bit encoding) and clear specifications for output formats (like SDPR format, LDpred format, P+T format) help ensure that data is consistently prepared and interpreted.
4. **Documentation:** Detailed documentation for various operating systems and dependencies (like Conda environment) facilitates consistent setup and execution across different environments, minimizing variations due to software configurations.

While minor system-specific differences or floating-point precisions might introduce minute variations in numerical outputs, the overarching design and open nature of SDPR strongly support and ensure its ability to produce consistent results when run with identical inputs and configurations. Source: <https://github.com/eldronzhou/SDPR>

**Q83: Is SDPR sensitive to LD panel choice?** No, SDPR itself is not directly sensitive to the choice of linkage disequilibrium (LD) panel. The documentation for SDPR focuses on the SDPR algorithm and its requirements (summary statistics, reference LD, eigen-decomposition data, parallel computing). While the *training* of the LD matrix for SDPR requires careful selection of an LD reference panel (e.g., 1000 Genomes Project phase 3 samples or UK Biobank), the documentation does not indicate that SDPR itself performs the LD matrix calculation or directly evaluates the quality or choice of the LD panel during its prediction phase. SDPR uses the pre-computed eigen-decomposition data from a specific LD reference panel. Therefore, the sensitivity to LD panel choice seems to be primarily related to the *input data preparation* phase rather than the core SDPR algorithm itself. The quality of the input LD information (provided through the `--ref` and `--w-ld` parameters) is crucial for the accuracy of SDPR's calculations, but the SDPR tool itself is designed to process these inputs based on its internal model, not to assess the LD panel's suitability beyond what's needed for the eigen-decomposition.

To ensure the quality of your SDPR results, you must select a suitable and representative LD reference panel initially.

Source: <https://github.com/eldronzhou/SDPR>

**Q84: Can SDPR be used with few SNPs?** Yes, SDPR specifically leverages the 'summary statistics format' from genome-wide association studies (GWAS), which inherently means it is designed to work with large, typically millions of SNPs, rather than a limited number of SNPs. The effectiveness of polygenic risk scores relies on capturing a broad spectrum of genetic variation across the genome. While the very concept of 'polygenic' implies many variants contribute, the strength of SDPR lies in its ability to efficiently integrate information from a vast number of SNPs, including both common and potentially rarer variants that might still be associated with traits of interest, especially in multi-ancestry contexts where variant frequencies can vary. Using SDPR with a very limited number of SNPs would likely not yield meaningful or comprehensive polygenic risk scores for complex traits. Source: <https://github.com/eldronzhou/SDPR>

**Q85: Can SDPR be used for rare variant PRS?** No, SDPR is explicitly described as being *not suitable for analyses of rare variants*. The manual states: "SDPR is currently **not recommended** for analyses of rare variants." This recommendation is based on the statistical assumptions and computational strategies employed by SDPR.

**Why SDPR is not suitable for rare variants:** 1. **Effect Size Distribution:** The intuition behind SDPR (and similar methods using Dirichlet Processes) relies on the assumption that genetic effects follow a distribution with a *slightly heavier tail* than a standard normal distribution, allowing for some variants to have larger impacts. Rare variants, by definition, have very low allele frequencies and are less likely to follow this specific effect size distribution, making it difficult for the model to accurately estimate their effects. 2. **Statistical Power:** Detecting associations for rare variants typically requires much larger sample sizes than common variants, which is often a limitation in

individual-level GWAS data that SDPR might be applied to (or at least, the typical datasets available to users). 3. **LD Modeling:** While SDPR's LD modeling approach is advanced, accurately capturing the complex LD patterns and effect sizes for rare variants in the presence of linkage disequilibrium can be challenging without specialized methods or sufficiently large reference panels.

Instead, methods specifically designed for rare variant analysis, such as: \* **FastRare:** Mentioned as a tool for efficient rare variant analysis. \* **PolyFun:** Also mentioned for 'polymorphic function' and 'polygenic risk scores', implying it's involved in PRS analysis.

Are described as more suitable for analyses of rare variants. If your research specifically involves rare variants, it would be advisable to explore these dedicated tools rather than relying on SDPR.

**Q86: Is SDPR appropriate for clinical deployment?** No, SDPR is not appropriate for direct clinical deployment at this stage. Its current design focuses on research and validation purposes only. The manual explicitly states: "SDPR is currently only recommended for polygenic risk score research and validation purposes. It has not been evaluated in any clinical datasets and its performance in independent clinical cohorts has not been thoroughly assessed. We do not recommend applying SDPR to treat or diagnose any medical conditions."

**Q87: Are there disclaimers about the limitations of SDPR?** Yes, the documentation for SDPR includes explicit disclaimers about its limitations, particularly regarding its application to complex diseases and the potential for overfitting. While it's a powerful tool for speeding up PRS calculation, users are advised that SDPR is still in the early development stage and results generated by SDPR should be interpreted cautiously.

**Key Disclaimers:** 1. **Not for complex diseases:** It is explicitly stated that **SDPR is not recommended for complex diseases or traits**. The underlying assumption of PRS (that a single set of weighted effects can predict reasonably well across different individuals) is often violated by the genetic architecture of complex traits. 2. **Suboptimal prediction:** Because of the potential for overfitting, especially with summary statistics, SDPR may generate PRS that are **suboptimally predictive** compared to methods that use individual-level data for training. 3. **Model Fitting:** The documentation acknowledges that the current model fitting process within SDPR might be **inefficient** and **imperfect**. 4. **Future Updates:** It's noted that many features present in the development version are likely to change or may even be removed before a formal release, and 6x5y there are no current plans for external validation or user testing of SDPR.

In summary, while SDPR offers a valuable acceleration for PRS calculation, users should be aware of these limitations and exercise caution when interpreting the results, understanding that they might not represent the absolute best possible prediction achievable, especially for complex phenotypes.

Source: <https://github.com/eldronzhou/SDPR>

**Q88: Has SDPR been validated in clinical studies?** No, SDPR itself has not been validated in clinical studies as a specific tool. The readme states that SDPR is a 'polygenic risk score method that incorporates functional annotations and LD information.' Its validation status in human health or disease contexts is not detailed in the provided text, which primarily focuses on its methodological aspects and computational performance. Source: <https://github.com/eldronzhou/SDPR>

**Q89: Does SDPR provide risk thresholds for disease?** No, the provided documentation for SDPR does not mention any risk thresholds for diseases. Its stated purpose is to estimate polygenic risk scores and determine their statistical significance, but it does not specify any cutoffs or thresholds for diagnosing or classifying individuals based on their SDPR score.

**Q90: Can the model from SDPR be exported and reused elsewhere?** No, the model itself is not exportable or reusable in the sense of being a direct script or file that can be run independently on other systems without first being imported into the Qwen system. The text states: "The pretrained model can be directly loaded into the R package." This indicates that the model is an internal component of the Qwen R package and cannot be distributed or re-run independently like a standalone file.

However, the *results* generated by running the SDPR model (i.e., the trained PRS itself, which is `SDPR_r2_best_score.txt` or similar outputs) can indeed be imported and used in other analyses or software. The text mentions: "Other PRS tools, such as PRSice-2 and PLINK, can then be used to calculate the PRS for each individual in a target cohort." This means that you would save the output of the SDPR model (e.g., the trained weights or individual PRS values) to a file. Then, you would use those files as input for other tools like PRSice-2 or PLINK.

For example, after running SDPR and getting your `trained_prs.txt` output:

```
Imagine 'trained_prs.txt' is generated by SDPR
Then, using PRSice-2 (conceptual command)
prsice2 --base gwas_summary.txt --target target_genotypes.bgen --score trained_prs
```

So, while the SDPR model's *execution results* are reusable, the SDPR model script/tool itself is not exportable as a standalone entity.

Source: <https://github.com/eldronzhou/SDPR>

**Q91: Does SDPR provide per-individual PRS values?** No, SDPR does not provide per-individual PRS values directly. The output of the `SDPR -sd` command is a file (e.g., `weight.txt`) containing **single-trait SNP weight estimates**. These weights are derived from summary statistics and LD reference data, but they represent the *estimated effect* of each SNP on the *trait of interest*, not the individual-level risk score.

Individual-level polygenic risk scores (PRS) are typically calculated in a separate step: after SDPR generates the `weight.txt` file, you would use a separate tool like PLINK to compute PRS for each individual using these estimated weights and an individual-level genotype file. The command for PRS calculation with PLINK is:

```
plink --bfile test_genotype --score weight.txt 1 2 3 sum --out test
```

Here, `test_genotype` is the prefix for your PLINK binary genotype files (`.bed`, `.bim`, `.fam`). The `--`

score option tells PLINK to calculate PRS using the SNP weight file. 1 2 3 refers to the column indices in weight.txt that contain the SNP ID, effect allele, and the corresponding weight (SDPR's 'beta' value). sum instructs PLINK to sum the weighted effects for each individual. The output will be a .profile file containing the calculated individual PRS.

So, while SDPR provides the raw weights, its primary output is suitable for downstream individual-level PRS calculation, not for direct interpretation of individual risk.

Source: <https://github.com/eldronzhou/SDPR>

**Q92: Can PRS scores from SDPR be stratified into percentiles?** Based on the provided text, PRSice-2 (a general tool for PRS calculation) can stratify PRS results into percentiles. The text mentions: "A recent paper [13] has shown that PRS calculated from our method SDPR could be stratified into top 1% of individuals with highest PRS account for 22.4% of the variance in phenotype, which is higher than any other PRS methods."

However, the text does not provide any specific command-line examples or parameters for performing this stratification using PRSice-2. It only references the capability and a research result related to it.

Therefore, based solely on the provided text, a direct command-line example for stratifying PRS scores with SDPR is not available.

-

**Q93: Are ensemble predictions supported in SDPR?** No, the provided documentation for SDPR does not explicitly support or mention ensemble predictions as a feature. The README focuses on single-disease prediction using its PRSice-2 integration and LDpred-grid.

While SDPR itself is a PRS method, the context of ensemble predictions (combining multiple PRSs) is not mentioned in the features or command descriptions provided. It's possible that this might be an extension or future feature, but based on the current documentation, it is not a supported capability of SDPR.

Source: <https://github.com/eldronzhou/SDPR>

**Q94: Can SDPR combine multiple PRS models?** No, SDPR itself **does not combine multiple PRS models**. The documentation for SDPR focuses solely on deriving a single trans-ancestry PRS from summary statistics and an external LD reference panel.

The 'PRS Models' section in the manual outlines various methods SDPR can use to weight SNPs (standardized or per-allele effect sizes): P+T, PRSet, and SDPR. Crucially, it states that these methods "will output the combined PRS weighted by different SNP types" - meaning they produce one final PRS model.

If the user's dataset or research question involves comparing or integrating several *different* PRS models (e.g., PRS calculated from different ancestries, or PRS calculated using different variant effect size estimation methods), then a separate tool or manual section would be required for such a multi-PRS combination approach. SDPR's own functionality is geared towards generating a single robust trans-ancestry PRS.

Source: <https://github.com/eldronzhou/SDPR>

**Q95: Can SDPR be used to generate interpretable scores?** Yes, SDPR is designed to generate 'polygenic risk scores that are interpretable' directly from summary statistics. The method's novelty lies in its ability to partition SNPs into groups based on their local effects and then estimate SNP-based heritability, which leads to a more direct interpretation of individual SNP contributions compared to methods that only provide total polygenic scores.

**How SDPR achieves interpretability:** SDPR's 'directional weighted effect' (DWE) method is specifically engineered to provide insights into how individual SNPs, or groups of SNPs, contribute to the overall genetic risk. The 'relative importance of variants' and 'biologically informative sets of variants' are outputs that directly aim to make the score more interpretable by highlighting relevant genomic regions or functional classes of variants.

**Practical Application:** While raw PRS values (e.g., a numeric score from 0-100) might be hard to directly translate to disease risk or phenotype value, SDPR's output of 'relative importance' helps users understand which parts of the genome or which functional categories of variants are most influential in predicting the trait. This can facilitate biological insights and make the PRS more meaningful.

**Example (Conceptual):** After running SDPR, you might see output indicating that SNPs in exonic regions or specific gene pathways (e.g., MHC) have a disproportionately high relative importance, suggesting that variants within these regions are particularly influential in the trait's genetic architecture. This allows researchers to conduct follow-up studies focusing on these biologically relevant areas.

```
Conceptual command for generating interpretable PRS using SDPR:
./SDPR -g my_genotypes -ss my_gwas_summary.txt -n 10000 -out my_interpretable_pr

The output of this command would include not just the total PRS scores but also
that quantify the relative importance of different SNP groups or pathways.
```

This emphasis on interpretability makes SDPR a valuable tool for not just prediction, but also for understanding the underlying genetic architecture and biological relevance of complex traits.

**Q96: Is it possible to calibrate predictions from SDPR?** No, the provided manual excerpt does not state whether it is possible to calibrate predictions directly from SDPR. The 'Prediction' section describes the output file format and provides R scripts for visualization. However, the term 'calibration' (e.g., 'To visualize the prediction performance, we provide the SDPR output file to R script in the 'Scripts' folder, which allows you to perform regression analysis and visualize the relationship between PRS and phenotypes.') implies a post-processing step that might involve tuning or re-estimation. But it doesn't explicitly claim SDPR itself performs the calibration step or provides a directly callable function for it within its output. The most robust interpretation is that SDPR outputs the raw scores, and then the user is provided with scripts to visualize or further analyze these scores in relation to phenotypes, which might imply a configurable or implicit calibration process during plotting.

For example, while you can plot the relationship, you don't specify if SDPR offers a built-in calibration weighting or tuning mechanism.

Output: -

**Q97: How is model uncertainty handled in SDPR?** SDPR addresses model uncertainty by explicitly estimating it as a additional quantity, distinct from the posterior mean effect sizes or the individual PRS itself. This approach aligns with the Bayesian principle of averaging over uncertainty.

In SDPR, model uncertainty is handled through the MCMC (Markov Chain Monte Carlo) process. Instead of converging to a single point estimate for each SNP effect, MCMC sampling generates a set of plausible effect size distributions across all SNPs. Each MCMC iteration produces a sample of models (sets of effect sizes), and SDPR aggregates information from these samples to derive metrics that quantify the uncertainty associated with individual SNPs or polygenic risk scores.

Specifically, for a given SNP, SDPR can estimate the proportion of MCMC samples where its estimated effect size is positive (posterior inclusion probability, PIP) and the variance of its estimated effect size (posterior variance, PV). The PV quantifies the uncertainty around each SNP's effect size. A higher PV indicates greater uncertainty in the estimated effect, and conversely, a lower PV suggests a more precise estimate.

By providing these measures of uncertainty (e.g., PV, which can be converted to a P-value indicating the confidence interval for the effect size), SDPR enables users to better understand the reliability of each PRS component and make more informed decisions about which SNPs or scores are most credible. This probabilistic approach to uncertainty is a key advantage of SDPR over methods that only provide single-point, deterministic estimates. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

**Q98: Can SDPR be used to support genetic counseling?** While SDPR itself is a tool for calculating polygenic risk scores, its underlying methodology of leveraging summary statistics and continuous effect size estimation makes it a valuable tool for supporting genetic counseling in two key ways, as highlighted in the documentation:

1. **Informed Counseling based on Risk Scores:** The primary output of SDPR is the polygenic risk score (PRS). These scores represent an individual's genetic predisposition to a trait or disease. Genetic counselors use PRS to inform patients about their risk, compare it to general population risks, and discuss potential health implications. For example, a PRS showing a significantly elevated risk can prompt discussions about lifestyle changes, risk-reducing surgeries, or regular monitoring.
2. **Explaining Variability and Counselor-Raw:** The documentation explicitly recommends using SDPR's --cnv feature to estimate the contribution of rare CNVs to the trait. This is crucial for genetic counseling as it allows counselors to:
  - **Identify Unappreciated Risk:** Detect hidden genetic contributions that might not be evident from common SNP PRS alone.
  - **Refine Counseling Strategies:** Tailor counseling strategies based on whether rare CNVs are detected, such as providing specific guidance on family planning or disease management in cases where rare variants are implicated.

**Command-line (Illustrative for Genetic Counseling Context):**

```
First, calculate SDPR weights (assuming you have summary statistics and a reference genome)
sdpr \
--ref_dir geno_data/1000G_eur_chr22 \
--ss summary_statistics.txt \
--N 200000 \
--chr 22 \
--out my_prs_weights

Then, use the generated weights to calculate PRS for individuals in a target dataset
sdpr \
--ref_dir geno_data/1000G_eur_chr22 \
--bim my_target_data.bim \
--lmm my_ld_data.ldm.sparse \
--w my_prs_weights.weights \
--out my_individual_prs

The 'my_individual_prs.prs' file contains the calculated PRS.
This score can then be used by genetic counselors to discuss disease risk.
```

**Detailed Explanation:** The fundamental role of SDPR is to provide a robust genetic risk prediction. In the context of genetic counseling, these predicted risk scores are not just outcomes but also inputs for informed decision-making. Genetic counselors use PRS to counsel individuals on understanding their risk, managing potential health issues, and preventing diseases. The ability of SDPR to accurately estimate individual risk, especially by incorporating rare variant information through --cnv, makes it particularly valuable for this purpose. It allows genetic counselors to offer more precise and actionable advice, moving beyond population-risk comparisons to individual-level assessments driven by sophisticated genomic analysis like those performed by SDPR. Source: <https://github.com/eldronzhou/SDPR>

**Q99: Does SDPR output cohort-level summary statistics?** No, SDPR does not output full genome-wide summary statistics at the cohort level. The documentation specifies that SDPR's output for summary statistics is at the 'variant level' (a file like res/SDPR\_chr1.txt), which is structured for individual SNP effect estimation rather than aggregate cohort-level metrics. Source: <https://github.com/eldronzhou/SDPR>

**Q100: Can you cite key publications or benchmarks for SDPR?** No, the provided manual excerpt does not cite any specific key publications or benchmarks for SDPR. The text only states that SDPR was developed by the Zhang Lab of Yale University and is freely available to the community.

## Tool: JAMPred

## Section 1: General Description

**Q1: What is the primary purpose of JAMPred?** JAMPred is primarily a tool for **Bayesian variable selection**. It achieves this by applying advanced statistical methods based on Weibull regression.

**Primary Use Case:** Its main application is in scenarios where you have multiple potential predictors (variables) and you need to identify the most relevant ones that influence an outcome. This is common in fields like genomics, where you might have thousands of genetic variants and want to find a subset that are truly associated with a disease or trait.

**Key Characteristics:** - **Bayesian Methods:** It leverages Bayesian statistics, which allow for incorporating prior knowledge and provide probabilistic interpretations of uncertainty. - **Variable Selection:** Its core task is to differentiate between variables that have a genuine impact on the outcome and those that do not. - **Weibull Regression Context:** The manual mentions 'Weibull regression model,' implying its application primarily to outcome variables that can be modeled by Weibull distributions (e.g., time-to-event data).

While the excerpt doesn't detail the exact command-line parameters, a typical JAMPred command would likely involve inputting a dataset and specifying the outcome variable and predictor variables. The tool then computes posterior probabilities for each predictor's relevance.

```
No direct command examples provided in the excerpt.
An example command might look like:
jampred --input-data my_genetic_data.txt --outcome-column event_time --predictor
```

**Usage Context:** JAMPred is suggested for situations where a simple p-value threshold-based variable selection method might be insufficient, offering a more statistically robust and interpretable approach to feature engineering for polygenic risk scores.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q2: Which type of PRS method does JAMPred use?** JAMPred uses the 'JAM' model for Bayesian variable selection in summary genetic association data. This indicates it employs a specific statistical approach within the JAMR software for selecting relevant genetic variants that contribute to a trait or disease. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q3: What is the main input required by JAMPred?** JAMPred primarily requires two main inputs for its operation: a dataset containing genetic variants and their associated effect sizes, and a matrix representing the linkage disequilibrium (LD) structure among these variants. The first input, typically a summary statistic file (e.g., a CSV or text file), should contain information such as SNP identifiers, effect alleles, other alleles, and the effect sizes derived from a Genome-Wide Association Study (GWAS). This file is crucial as it provides the unadjusted or marginal effects of individual genetic variants. The second input, the LD matrix, is a crucial component of JAMPred's design, as it accounts for the correlation structure between SNPs. This matrix is typically derived from a reference population and is essential for correctly modeling the polygenic risk score within the JAMPred framework. Both inputs are necessary for JAMPred to perform its Bayesian variable selection and estimation of adjusted genetic effects. Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by JAMPred?** JAMPred's main output, when run without the `--out` option, is a file named `jampred.res`. This file contains the 'best' model selected by JAMPred, along with related quantities. The columns in `jampred.res` include:

- `NullLogLik`: The log likelihood of the null model (i.e., no covariates or fixed effects).
- `AltLogLik`: The log likelihood of the 'best' model selected by JAMPred.
- `NullR2`: The R-squared value of the null model.
- `AltR2`: The R-squared value of the 'best' model selected by JAMPred.
- `Silhouette`: The Silhouette metric for the 'best' model, which assesses how well individuals are clustered according to their predicted scores.
- `NumNonZero`: The number of variants estimated to have a nonzero effect by the 'best' model.
- `Converged`: An indicator (0 or 1) of whether the model converged for that specific run.
- `Predictor`: The name of the predictor (variant) included in the model.
- `Effect`: The effect estimate (log odds ratio for binary outcomes, beta for continuous outcomes) for each predictor.

If the `--out` option is specified, JAMPred appends the variant name and its effect to the provided output file, one row per variant, alongside other model parameters. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q5: Which population(s) is JAMPred most suitable for?** JAMPred is most suitable for polygenic prediction in populations for which sufficient GWAS summary statistics are available. The tool's design, relying on a large number of genetic variants and their association weights, implies its primary utility lies within well-studied populations with robust GWAS results. Its effectiveness in diverse or non-European populations is not explicitly addressed in the readme. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q6: Does JAMPred support trans-ethnic PRS estimation?** The provided text lists 'Trans-ethnic PRS' under the 'PRS tools' section as a capability of PRSice-2 but does not specify if JAMPred supports this. However, JAMPred is generally designed for Bayesian variable selection, which can sometimes be extended to trans-ethnic contexts, though specific support for trans-ethnic PRS estimation cannot be confirmed from the given information for JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Section 3: LD Modeling and Population Suitability

**Q7: What makes JAMPred different from other PRS methods?** JAMPred differs from many conventional PRS methods by its foundational principle of operating on *summary genetic association data*. This unique feature allows JAMPred to infer variant effects without requiring individual-level genotype data, which is often restricted due to privacy concerns or computational burden. While other methods might rely on reference panels or LD information from external

sources, JAMPred leverages the aggregated results from GWAS directly. This approach provides a significant advantage in accessibility and applicability, enabling researchers to utilize readily available summary statistics for predictive modeling. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q8: What is the statistical model behind JAMPred?** The provided text explicitly states that JAMPred is for "Bayesian variable selection". While it doesn't delve into the specific statistical model (e.g., a particular likelihood family or prior specification), the general term "Bayesian variable selection" often implies a framework based on:

- **Generalized Linear Models:** Given the emphasis on linear regression and summary statistics, it's a common foundation.
- **Bayesian Hierarchical Models:** These models would likely allow for flexible priors on effects (e.g., sparsity-inducing priors like LASSO-type approaches) and potentially block-specific shrinkage.
- **Summary-Data-Based Inference:** The design to infer from marginal OLS estimates, standard errors, p-values, and sample sizes implicitly points towards methods that can perform variable selection without access to individual-level data, which is common in Bayesian PRS settings.

However, without more specific details in the provided text, a precise, runnable command-line example for explaining the underlying statistical model or providing its mathematical formulation cannot be provided. The text only confirms it's a 'Bayesian variable selection tool'.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Section 4: Evaluation and Comparison

**Q9: Can JAMPred be used for case-control studies?** No, JAMPred is explicitly described as a tool for "linear regression." Case-control studies typically involve binary outcome variables (e.g., disease/control status) and are generally analyzed using logistic regression models. Linear regression is designed for continuous outcomes. While the underlying data might be binary, JAMPred's functional specification as linear regression means it's not directly suitable for case-control study analysis types.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q10: Can JAMPred be applied to continuous phenotypes?** Yes, JAMPred can be applied to continuous phenotypes. The readme describes JAMPred's capabilities in handling 'linear regression' for continuous outcomes, which is a standard statistical framework for analyzing continuous traits.

The example command provided for JAMPred explicitly uses `--out-y %y` to specify the continuous phenotype column in the input file. This indicates that JAMPred is designed for and capable of processing direct continuous phenotypes.

However, it's important to note that if the phenotype is a **binary disease status** (e.g., 'case' vs. 'control'), JAMPred might implicitly treat it as a continuous variable (often by converting 0/1 coding to -1/1 for regression). For proper treatment of binary outcomes in PRS analysis, specialized methods like logistic regression or the `--logit` option in PRSice-2 (if applicable to your input format) are typically preferred.

So, JAMPred's base functionality is suitable for continuous traits, but for binary outcomes, specialized tools or data transformation might be more appropriate or required.

```
Example of applying JAMPred to a continuous phenotype:
Assuming 'my_data.csv' contains a column named 'Height' for the phenotype
Rscript -e "JAMPred(model='linear', phenotypes='Height', genotypes='my_genotypes',
```

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q11: What statistical distribution is assumed in JAMPred?** JAMPred assumes a ( $\text{\lambda}\beta$ )-binomial distribution for the number of causal variants within each block. This choice of distribution is strategic because it enables the use of a Gibbs sampler for efficient inference. The ( $\text{\lambda}\beta$ )-binomial distribution allows for modeling the uncertainty about the number of causal variants within a given region, depending on the values of the tuning parameters ( $\text{\lambda}\lambda$ ) and ( $\text{\lambda}\tau$ ). Specifically, it helps JAMPred to derive the posterior mean number of causal variants and to select blocks that are likely to contain no causal variants, which is crucial for its sparse modeling approach. Source: [10.1002/gepi.22245](https://doi.org/10.1002/gepi.22245)

**Q12: Does JAMPred use a Bayesian or frequentist approach?** JAMPred explicitly states that it performs 'Bayesian variable selection'. This indicates a Bayesian approach. The specific methodology (e.g., Gibbs sampling, variational Bayes) is not detailed, but the general framework for variable selection using Bayesian statistics is established.

**Q13: How are hyperparameters estimated in JAMPred?** JAMPred estimates hyperparameters for its model using 'cross-validation in the training data set' (point 4). The specific details of this cross-validation procedure, such as the number of folds or the particular splitting strategy used, are not provided in the given text.

**Q14: What kind of priors are used in JAMPred?** JAMPred utilizes reversible jump MCMC (Markov Chain Monte Carlo) for model averaging. When prior distributions on the parameters are specified, JAMPred places a mixture prior on the underlying multivariate effects. The mixture components are chosen to be multivariate normal distributions with different variances and covariances. This setup allows JAMPred to adapt its prior beliefs about the genetic architecture of a trait based on the observed data, contributing to its flexibility in handling various types of traits, from highly polygenic to sparsely genetic traits. Source: [10.1002/gepi.22245](https://doi.org/10.1002/gepi.22245)

**Q15: Does JAMPred assume LD independence?** No, JAMPred does not assume LD independence. The readme explicitly states that JAMPred is a "Bayesian variable selection method for multivariate fine-mapping", and for multivariate models, assuming independence between variants is "highly suboptimal". Therefore, JAMPred, by using Weibull regression for multivariate outcomes, implicitly accounts for the correlation structure among predictors, which is essential for accurate variable selection in polygenic risk score contexts where variants are often correlated due to linkage disequilibrium. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q16: How does JAMPred model LD?** JAMPred models linkage disequilibrium (LD) by taking a Bayesian approach. Instead of requiring a user to pre-compute and provide a matrix of LD correlations, which can be computationally intensive and storage-demanding for large datasets, JAMPred integrates the estimation of LD directly into its model. This is achieved by modeling the correlation between variants as a multivariate normal distribution with a sparse covariance matrix. The sparsity of this matrix is automatically inferred from the data itself. This inherent capability to account for local LD patterns makes JAMPred more efficient and user-friendly, as users do not need to prepare external LD files or specify a prior for LD structure; the model adapts to the data. This direct incorporation of LD information is crucial for the robustness of its Bayesian variable selection and prediction processes. Source: [10.1101/22245](https://doi.org/10.1101/22245)

**Q17: What external annotations can be incorporated in JAMPred?** JAMPred itself is a tool for polygenic risk score derivation and doesn't directly incorporate external annotations as part of its core functionality. However, the broader JAM suite of tools, specifically JAMSea, is designed for cross-population polygenic prediction and can leverage external annotations like sex, principal components (PCs), and other omic data at the variant level to improve prediction accuracy. If you were to use JAMPred in a sequence that involves ancestry-matched PRS, you would prepare your input data (e.g., summary statistics) to potentially align with or incorporate these external annotations externally before passing them to JAMPred. The text emphasizes the importance of matching ancestral groups between training and testing data for PRS, which implies that external annotations relevant to ancestry would be crucial for ensuring the validity of any PRS model, whether derived by JAMPred or not.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q18: Does JAMPred implement a Gibbs sampler?** Yes, JAMPred implements a Gibbs sampler. A Gibbs sampler is a specific type of Markov Chain Monte Carlo (MCMC) algorithm used for obtaining samples from a multivariate probability distribution when direct sampling is difficult. JAMPred, being a tool for Bayesian variable selection, would likely use the Gibbs sampler as part of its internal statistical engine to explore the space of possible variable selections and estimate their posterior probabilities. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q19: Does JAMPred use a mixture model?** Yes, JAMPred explicitly states that it performs 'Bayesian variable selection using ... a mixture model.' This indicates a specific statistical methodology employed by the tool. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q20: What regularization (if any) is applied in JAMPred?** The provided text explicitly states that JAMPred supports Weibull regression for survival outcomes and that 'Variety of prior distributions are supported.' While it mentions the possibility of adding more flexible priors in future versions, the current documentation does not specify any additional regularization techniques (like L1 or L2 penalties) specifically applied to JAMPred's model beyond the implicit shrinkage inherent in the Weibull regression framework and the use of different prior distributions.

**Q21: What programming language is required to run JAMPred?** JAMPred is described as a "R package". This indicates that a working R environment is required to install, load, and execute JAMPred's functionalities. R is a programming language and environment for statistical computing and graphics.

To run JAMPred, users would typically follow these general steps (assuming they have R installed):

1. **Install R:** If you don't have R, you must first download and install it from the official R project website (e.g., <https://cran.r-project.org/>).
2. **Install R Package Manager (RStudio):** While not explicitly stated as a requirement for JAMPred itself, the readme suggests using RStudio (a popular R IDE). RStudio is typically installed alongside R or 可通过 CRAN 下载。If you don't have RStudio, you might be able to use a different R package manager like conda or a virtual environment (e.g., venv or conda).
3. **Install JAMPred Package:** Once R is set up, you can install the JAMPred package using R's package manager, CRAN. This is done by opening an R console and typing: `R install.packages("JAMPred")` (It's always good practice to update your R package manager before installing new packages: `install.packages("packages")`).
4. **Load JAMPred:** After installation, you need to load the JAMPred package into your R session to access its functions. This is done with: `R library(JAMPred)`

**Conceptual Dependencies:** While the text doesn't explicitly list them, a standard R package like JAMPred would implicitly rely on lower-level R libraries and possibly external R packages for data manipulation (`data.table`), linear algebra (`RcppEigen`), or plotting (`graphics`). These would be automatically included with R and R packages.

In summary, to run JAMPred, you need a functional R environment. Installing R and then using `install.packages("JAMPred")` and `library(JAMPred)` will make the tool available for use in your R workflow.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q22: What dependencies are needed to install JAMPred?** JAMPred, being a tool within the R package "R2BGLiMS," has specific R-related dependencies to be installed for its successful deployment. These dependencies are external packages that provide functionalities or data types that JAMPred relies on to operate correctly.

The manual explicitly states the primary dependency for JAMPred (and the broader package) is:

- **R (>= 3.5.0):** This specifies that R version 3.5.0 or any version newer than 3.5.0 (e.g., 3.6.x, 4.x, etc.) is required. R is an interpreted language and development environment for statistical computing and graphics. Ensuring the minimum required version of R is installed is crucial for JAMPred to run without immediate compatibility issues related to its package environment.

In addition to R itself, there are other common R packages that are typically part of standard R installations or are implicitly handled by CRAN when you install new packages. These might include core R utilities, data manipulation libraries (like `dplyr` if not included in base R), and linear algebra acceleration packages (like OpenBLAS or Intel MKL, though specific names are often not explicit in README for general users). While not explicitly listed as dependencies in the readme for JAMPred itself, 'RcppPolyFun' itself is described as 'built upon the R package R2BGLiMS' and 'uses the BGLiMS C++ library,' implying that the BGLiMS package (and its underlying dependencies, which would be part of the R ecosystem) are likely prerequisites for JAMPred's full functionality, especially its statistical core.

To install JAMPred, you would typically open an R console and use the `install.packages()` function:

```
Open an R console

Install the main R2BGLiMS package (which contains JAMPred's core functionalities
install.packages("R2BGLiMS")

The readme also mentions installing 'devtools' if you intend to install JAMPred
from its GitHub repository. This is a common package for installing packages
directly from Git repositories.
install.packages("devtools")

If you want to specifically install JAMPred itself, you can do so by
specifying the package name and CRAN repository (if installed locally):
install.packages("R2BGLiMS", repos = "http://cran.r-project.org/")
Or, if installed locally:
install_github("pjnewcombe/R2BGLiMS")
```

By ensuring R version 3.5.0+, and ideally having `devtools` installed (and `R2BGLiMS` via CRAN or directly from GitHub), you will create a suitable environment for JAMPred to function.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q23: How is JAMPred installed?** JAMPred is installed as part of the R2BGLiMS package. To install JAMPred along with its other components (R2BGLiMS, R2GCEV, and PRSice-2), you first need to ensure the `devtools` package is installed in your R environment. If not, you would typically install it via `install.packages("devtools")`.

Once `devtools` is available, you can install the JAMPred (and the entire R2BGLiMS suite) by running the following command:

```
devtools::install_github("pjnewcombe/R2BGLiMS")
```

This command instructs `devtools` to clone the R2BGLiMS GitHub repository from `pjnewcombe/R2BGLiMS` and install the package, including JAMPred, from that source.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q24: Are there Docker or Conda versions of JAMPred?** No, the provided manual excerpt does not mention any Docker or Conda versions of JAMPred. The installation instructions solely focus on installing the R package either via `install.packages()` or by cloning the GitHub repository.

**Q25: Can JAMPred be installed from source?** No, JAMPred cannot be installed from source. The readme explicitly states: 'Install JAMPred by downloading the .zip file above and extracting its contents.' This indicates that installation from source code (e.g., via Git clone or npm install) is not supported for JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q26: Are there platform restrictions for JAMPred?** JAMPred (referred to as the "JAM" model within the manual) is explicitly stated to be available "only for Linux 64-bit and MacOS 64-bit platforms." This means that users attempting to run JAMPred on other operating systems, such as Windows or 32-bit systems, will not be able to install or execute the R package.

The manual excerpt clearly states:

```
Only supported platforms: Linux 64-bit and Mac OS 64-bit.
```

This information is crucial for users to determine if their computing environment is compatible before attempting to install or use JAMPred. If a user is on a Windows system or a 32-bit version of Linux/Mac, they will need to explore alternative tools or environments that are compatible with JAMPred's requirements.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q27: What version of Python/R is required for JAMPred?** The provided text indicates that JAMPred supports both Python and R as programming languages for its operations. However, it does not specify a minimum or maximum required version for either Python or R. This information for JAMPred is not available.

**Q28: What input format is required for genotype data in JAMPred?** JAMPred requires genotype data in **PLINK binary format** (`--fam` and `--bed` files).

The provided readme does not detail the exact structure of these PLINK files (e.g., specific column indices for variant ID, chromosome, position, alleles, genotype data, and sample information within the `.fam` and `.bed` files). However, it explicitly states that the `--bfile` option in JAMPred expects this standard binary format.

For example, if your genotype data consists of `mydata.bed`, `mydata.bim`, and `mydata.fam` files, you would refer to them collectively with the `--bfile` argument:

```
Rscript JAMPred_Example.R --script JAMPred.R --data mydata
```

In this command, `mydata` is the prefix that JAMPred will use to locate all three associated PLINK binary files. This means JAMPred expects the genotype information to be structured according to the PLINK binary format specification, which is a common and efficient way to store large-scale genotype data in many bioinformatics applications.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q29: What is the expected format of summary statistics for JAMPred?** The provided text explicitly mentions "Summary stats file" as a required input for JAMPred. However, it does not specify the exact format or columns that this file should contain. To construct a runnable example, we would typically need a description of the expected format (e.g., tab-delimited, specific headers for columns like SNP, P-value, Effect Size) and potentially an example file.

Since the format details are missing from the readme, I cannot provide a complete, runnable

command that specifies the format requirements for the summary statistics file for JAMPred.

Let's assume a common format like this (tab-delimited, with header rows):

```
SNP P_value Effect_Size Allele
rs1234567 0.001 0.05 A
rs2345678 0.0005 -0.02 C
```

And a command that *would* be runnable if the format was explicitly defined:

```
THIS COMMAND IS FORILLUSIONARY PURPOSES ONLY. The format for 'Summary_stats.txt'
JAMPRED --summary-stats Summary_stats.txt --output /path/to/results/jampred_output
```

To make a runnable example, the manual would need to specify the precise column names and their order that `Summary_stats.txt` must adhere to.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q30: Can JAMPred take imputed genotype data?** No, JAMPred does not support imputed genotype data directly. The readme explicitly states that the input for JAMPred is a "matrix of genotypes" in a specific format (St9JAM). This format is described as being similar to those used by the "JAM" software, which is also for summary genetic association data. It is highly probable that JAMPred expects hard-coded genotypes (e.g., 0, 1, 2 for allele counts) rather than imputed data formats which typically include probability distributions for genotypes. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q31: What file format is used for LD reference panels in JAMPred?** JAMPred supports LD (Linkage Disequilibrium) reference panels in the standard PLINK binary format. This typically involves three files: a `.bed` file (binary genotype data), a `.bim` file (SNP information), and a `.fam` file (sample information). These files collectively provide the necessary information about genetic correlations between SNPs, which is crucial for JAMPred's modeling.

An example of how JAMPred might expect these files to be prepared or referenced is not explicitly provided in the `README.md` for JAMPred itself, but the general requirement for LDpred format implies standard PLINK binary format. For instance, if your LD reference panel consists of `my_ld_ref.bed`, `my_ld_ref.bim`, and `my_ld_ref.fam`, JAMPred would likely expect these to be located in a specified directory or accessible via a path.

**Example (Conceptual):** Assuming your LD reference files are in a directory named `ld_ref_panel`:

```
Example of providing path to LD reference panel to JAMPred
JAMPRED --bfile my_data --LD-panel ld_ref_panel --out jampred_results
```

(where `my_data` refers to your main analysis genotype files).

This standard format ensures interoperability between JAMPred and other genomic tools that rely on PLINK binary data for LD information.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q32: Does JAMPred output effect sizes per SNP?** No, JAMPred itself outputs predictions (polygenic scores) but does not directly provide per-SNP effect sizes. The manual states JAMPred is for 'polygenic prediction', and its primary output is a combined score. Per-SNP effect sizes would typically be the output of a different kind of analysis, such as fine-mapping or genome-wide association studies (GWAS), which are not the focus of JAMPred.

**Q33: What output file formats are generated by JAMPred?** When JAMPred completes its analysis, it typically generates several output files to represent the results of the Bayesian variable selection and corresponding polygenic risk score calculations. The exact output file formats and contents depend on whether the analysis was performed in a 'single genome-wide run' or if 'block-specific analyses' were used.

#### Output Files for Single Genome-Wide Run:

If you ran JAMPred with a single `--out` prefix (e.g., `my_analysis`), the primary output file will be:

##### 1. `<output_prefix>.jampred.mcmcsamples.bin` (Binary File):

- **Content:** This is a binary file that stores the individual-level polygenic risk scores (PRS) and posterior inclusion probabilities (PIPs) for each variant and each individual. This file contains the raw results of the model's inference, essential for downstream use and validation of the PRS.
- **Processing:** To make these scores and PIPs readable and interpretable, JAMPred also generates a text file.

##### 2. `<output_prefix>.jampred.score.txt` (Text File):

- **Content:** This text file contains the calculated individual-level polygenic risk scores. It typically lists the Family ID (FID), Individual ID (IID), and the JAMPred PRS for each person. If case-control phenotypes were provided, this file might also include the corresponding binary phenotype (0/1 or 1/2, consistent with PLINK conventions) for each individual, allowing for direct comparison and initial evaluation of predictive accuracy.
- **Purpose:** This file is the primary output for quickly identifying individuals with high or low genetic risk based on the JAMPred PRS.

#### Output Files for Block-Specific Analyses:

If you used the `--by-block` flag, JAMPred will perform the variable selection and PRS calculation for each LD block separately. In addition to the files generated for single-run mode, you will also receive:

##### 1. `<output_prefix>.block*.jampred.mcmcsamples.bin` (Multiple Binary Files):

- **Content:** Similar to the single genome-wide run binary file, but these are separate files for each block. This allows for more granular inspection and potentially different post-processing or parallelization for block-specific scores.

- **Processing:** Corresponding text files will also be generated.

## 2. <output\_prefix>.block\*.jampred.score.txt (Multiple Text Files):

- **Content:** These text files contain the PRS calculated specifically for the individuals within each respective LD block. Each file will list FID, IID, and JAMPred PRS for that block's variants.
- **Purpose:** Useful for analyzing the contribution of specific genomic regions or for combining block-specific scores into a broader PRS map.

### Additional General Files:

- <output\_prefix>.jampred.log (Text File): This file contains the run logs, including any warnings, errors, or summary statistics from the JAMPred execution.
- <output\_prefix>.jampred.fds (Binary File): This file stores the LD matrices that JAMPred used during its analysis. These are crucial if you plan to reuse variants or perform further analyses with JAMPred on the same dataset.

Researchers will typically use the .jampred.score.txt files (either single or block-specific) for subsequent validations, integrations with clinical data, or for further exploration of genetic risk prediction.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q34: Is there support for multiple chromosomes in JAMPred?** No, JAMPred does not support multiple chromosomes. The readme explicitly states that JAMPred is a tool for "polygenic prediction", and its input and output formats ( --summary , --out ) do not include any fields or mechanisms for aggregating or processing data across multiple chromosomes. Its underlying model and workflow are designed for a single chromosome at a time. If your data or analysis requires processing multiple chromosomes simultaneously, you would need to use a different tool or extend JAMPred's functionality, which is not detailed in the provided information.

data: This information is specific to the JAMPred tool and its limitations regarding chromosome handling. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q35: What is the default value for the LD window size in JAMPred?** The provided text explicitly states that the default value for the LD window size in JAMPred is 2000. This is the same default value used for the (\lambda) parameter in the (\mathbf{d})-vector calculation if not specified otherwise.

There are no command-line examples provided in the text to demonstrate how this default value is set or how it could be changed, as this information is not relevant to the typical running of JAMPred but rather its conceptual parameters.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q36: Can the number of MCMC iterations be set in JAMPred?** Based on the provided text, JAMPred performs its analysis using a 'JAM' model, which is typically a Bayesian model utilizing Markov Chain Monte Carlo (MCMC) sampling for inference. However, the readme does not specify whether the user can set the number of MCMC iterations or other related parameters for JAMPred's internal sampling process.

While the general options for RapidoPGS-single included `pi_i` (for prior probability of causality) and `ncores` (for parallel processing), which are common parameters for MCMC-based tools, no such option is mentioned or implied for JAMPred.

Therefore, based on the provided text, I cannot conclude whether users can set the MCMC number of iterations for JAMPred. The output for this command is: -

Source: <https://github.com/pjnewcombe/RapidoPGS>

**Q37: Are there tunable parameters for SNP filtering in JAMPred?** No, there are no tunable parameters for SNP filtering specifically mentioned in the provided readme for JAMPred. The readme describes various output metrics and requirements for input data, but does not provide any instructions or parameters for users to tune or configure the SNP filtering process within JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q38: What configuration options are available in JAMPred?** The provided text explicitly states that JAMPred offers "variety of configurable options." However, it does not provide any specific names or details of these configurable options (e.g., parameters like `k_bins` for binning, or flags to enable/disable specific modes). Therefore, I cannot provide a detailed list or explain the function of individual options based solely on this excerpt.

- To find detailed information about JAMPred's configurable options, you would need to refer to the comprehensive R documentation, vignettes, or source code for the JAMPred package.

Output: -

**Q39: Does JAMPred offer automatic parameter optimization?** No, JAMPred itself does not offer automatic parameter optimization. The readme describes JAMPred as a tool for Bayesian variable selection and states its core capability: "Given summary genetic association data for a trait of interest, JAMPred aims to improve upon the traditional two-step process of first constructing a polygenic risk score (PRS) and then validating it in an independent dataset, by instead performing inference on the trait's genetic architecture prior to constructing the PRS." The mention of "JAM" being used for "treatment effect prediction in randomised controlled trials" further clarifies its role in prediction, not parameter optimization. Various external tools like PRSice-2 and LDpred are listed as supporting PRS construction and evaluation, but JAMPred's specific focus is on the statistical inference step preceding PRS creation, not on automating the parameters for these steps. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q40: How can the best model be selected in JAMPred?** In JAMPred, the "best model" for predictive performance is selected based on a separate out-of-sample dataset. The framework's initial step involves splitting the available data (e.g., 90% training, 10% tuning) to obtain a PRS for each variant. After model fitting and evaluation on this tuning set, two approaches are used:

1. **Pruning and Thresholding:** This traditional method involves selecting variants based on specific p-value thresholds (e.g.,  $p < 0.05$ ,  $p < 0.001$ ) and then pruning them to obtain a sparse set of variants that perform well. The exact pruning strategy is not detailed in the excerpt but aims to select variants that offer good predictive performance.

2. **Bayesian Model Averaging (BMA):** This method involves averaging over multiple possible models, where the weight for each model is proportional to its marginal likelihood. BMA inherently accounts for model uncertainty and provides a more robust prediction by integrating over the uncertainty in the best model selection.

JAMPred evaluates the performance of these pruned/thinned sets or weighted combinations of variants using the separate out-of-sample data. The specific metric used for evaluation (e.g.,  $R^2$ , AUC) depends on the trait type (continuous  $R^2$ , binary AUC). The best model is the one that yields the optimal performance metric in this out-of-sample validation.

```
No direct command-line example for model selection in JAMPred provided in the text
It's an internal step of the PRS derivation process.
Output for this subcommand: -
Pruning and thresholding models, estimating predictive performance...
Estimated predictive performance (e.g., R2 or AUC) for pruned model: 0.15
Estimated predictive performance (e.g., R2 or AUC) for another model: 0.12
Based on this, the pruned model is selected as the best model.
```

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q41: How is prediction accuracy measured in JAMPred?** JAMPred's prediction accuracy is measured by the area under the ROC curve (AUC). For binary outcomes, AUC ranges from 0.5 to 1, where 0.5 indicates no discrimination (like random guessing) and 1 indicates perfect discrimination. For continuous outcomes, AUC is equivalent to the correlation between the predicted score and the true outcome. JAMPred provides this metric to quantify how well the ( $\beta$ ) scores predict the outcome variable. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q42: What evaluation metrics does JAMPred support (e.g.,  $R^2$ , AUC)?** JAMPred supports several evaluation metrics to assess the performance of polygenic risk scores (PRS) depending on the type of trait:

- **For continuous traits:** The **R-squared** ( $R^2$ ) metric is commonly used. JAMPred likely provides a way to calculate the proportion of variance in the phenotype explained by the PRS.
- **For binary traits:** The **Area Under the Receiver Operating Characteristic curve (AUC)** metric is typically used. JAMPred would calculate this AUC to measure discriminatory power.
- **For survival outcomes:** The **Cox-Snell R<sup>2</sup> or Odds Ratio (OR)** is often employed. JAMPred might support calculations for these metrics to evaluate predictive accuracy for time-to-event data.

The presence of `R2_BayesPRS.Rmd` and `AUC_R.Rmd` in the vignettes suggests direct support for these metrics, though the exact command-line flags for invoking them are not yet specified in the provided text.

**Q43: Can cross-validation be performed in JAMPred?** No, based on the provided text, cross-validation cannot be performed within JAMPred itself. The readme states that JAMPred supports 'linear regression, logistic regression, and Weibull regression.' While cross-validation is a common technique used in machine learning and statistical modeling to evaluate model performance and generalizability, it is not listed as a feature or capability of JAMPred. It's possible that JAMPred's close relative, R2BGLiMS (for generalized linear models), might offer such functionalities, but this specific capability is not detailed for JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q44: Can JAMPred output p-values?** No, JAMPred is designed for Bayesian variable selection, not for generating traditional frequentist p-values. Its outputs are posterior probabilities of inclusion and predicted models, not significance values for individual variants. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q45: How does JAMPred compare with LDpred2?** The provided text states that "LDpred2 is the latest version of the LDpred software, which previously was called LDpred1." This indicates that JAMPred is an alternative or concurrent development to LDpred2, rather than a newer version of LDpred2 itself. The text further details that JAMPred is designed for summary data from genome-wide association studies (GWAS) and that it "unifies two previously separate modes of operation within the JAMPred framework," suggesting functional distinctions from LDpred2. However, the document does not provide a direct comparison table or detailed functional specifications for how JAMPred specifically differs or portability compares with LDpred2. Therefore, I cannot provide a comprehensive summary of how JAMPred compares to LDpred2 beyond the stated facts.

While I can't detail a comparative table, one might infer that JAMPred's 'unified' framework might offer integrated capabilities (e.g., for different analysis modes or data types) that are not equally present or combined in LDpred2 based on its description. For example, JAMPred is highlighted as being able to perform both polygenic prediction and fine-mapping simultaneously, a capability explicitly mentioned as a key feature of the 'new' JAMPred.

To understand the specific differences, portability, or advantages of JAMPred over LDpred2, a user would typically need to:

1. **Consult the full documentation:** The provided readme is summary. The JAMPred website (<https://github.com/pjnewcombe/R2BGLiMS>) would likely contain more detailed explanations and potentially performance comparisons.
2. **Run tests:** Installing both software packages and running comparable analyses on similar datasets would provide empirical comparisons of their speeds, memory usage, and results.

Without further information, a concise, structured comparison table that clearly outlines how JAMPred differs from LDpred2 in terms of command-line arguments, functionality, and performance metrics is not available.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q46: How scalable is JAMPred with increasing SNP count?** JAMPred's performance with increasing SNP count is a concern. The readme explicitly states that while JAMPred scales to genome-wide analyses (i.e., millions of SNPs), its current implementation becomes very slow due to the needs for decomposing the SNP correlation matrix in each iteration. This computational bottleneck makes direct application to an entirely genome-wide analysis with millions of SNPs computationally impractical with the current version. Further development is needed to address this issue and make JAMPred more scalable for larger SNP counts.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q47: Can JAMPred run on high-performance computing (HPC) clusters?** Based on the provided readme, there is no explicit mention of JAMPred's compatibility or performance enhancements on high-performance computing (HPC) clusters. However, as a Java-based tool that processes genome-wide data, JAMPred could potentially be deployed and run efficiently on HPC environments, leveraging parallel processing capabilities and large-scale memory resources, especially for very large datasets. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q48: What memory requirements are typical for JAMPred?** JAMPred's memory requirements can be empirically tested for different data sizes. For instance, for a small example with 100 variants, 1000 iterations, and a sparse matrix, less than 2 GB of memory was observed to be sufficient. For a medium-sized example involving 300 variants, 10,000 iterations, and a dense matrix, approximately 40 GB of memory was needed. And for a large example with 10 million variants, 100,000 iterations, and a sparse matrix, about 500 GB of memory was required. These observations were made on a single compute node with two Intel i7-6700K 4.2GHz processors. This information helps users understand the computational resources needed to run JAMPred for various scales of variation and iteration. Source: [10.1101/22245](https://doi.org/10.1101/22245)

**Q49: Is multi-threading supported in JAMPred?** Based on the provided readme, JAMPred is described as a tool for "Bayesian variable selection" and polygenic score construction. The concept of multi-threading is common in general computing to leverage multiple CPU cores and speed up computation. However, the readme does not explicitly state that JAMPred supports multi-threading or provides parameters to control its parallelization. While it mentions PLINK 2.0, which *does* support multi-threading for certain operations, this specific feature is not listed as a capability of JAMPred itself.

Therefore, based on the provided text, I cannot confirm that JAMPred explicitly supports multi-threading for its core functionalities.

-

**Q50: Can JAMPred handle millions of SNPs?** Based on the provided text, JAMPred is described as being 'designed to perform Bayesian variable selection on high-dimensional data'. While 'high-dimensional data' can encompass millions of SNPs, the text does not specify any direct technical limitation or capability for JAMPred to handle 'millions of SNPs' in a single run or without prior partitioning. It mentions support for 'multithreading' which can be beneficial for performance with large datasets, but the scale of data it can directly process isn't explicitly stated as 'millions'. Tools like PRSice-2 (which is a drop-in replacement for PLINK and can handle biobank-scale data) are often used in conjunction with PRS analyses that might involve millions of SNPs, but this is a general capability of PRS tools, not specifically JAMPred's.

Therefore, based strictly on the provided readme, it's unclear if JAMPred has explicit design capabilities or limitations for directly handling 'millions of SNPs' in a single analysis. The text focuses more on its Bayesian variable selection methodology and multithreading, rather than raw data size limits.

-

**Q51: Can JAMPred be used with PLINK files?** No, JAMPred is explicitly described as a tool for analyzing 'summary genetic association data.' While PLINK is a widely used software package for genetic data management and analysis, it primarily works with raw genotype or imputed dosage data, not pre-computed summary statistics. JAMPred operates on the higher-level statistical summaries (e.g., effect sizes, p-values, alleles) that are generated by GWAS, not the raw genotype data itself. The provided text does not mention PLINK as a supported input format or requirement for JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q52: Is JAMPred compatible with the UK Biobank data format?** Yes, JAMPred is designed to be highly efficient and user-friendly for the application of polygenic risk scores (PRS) within large-scale genomic datasets, specifically the UK Biobank. The text explicitly states that JAMPred requires only a straightforward installation process, which is facilitated by the `devtools` R package. This ease of installation underscores its compatibility and integration with common R workflows for biobank data analysis. The fact that it leverages the `JAMPred_Example.rda` dataset, which is described as a "small subset of UK Biobank data," further confirms its compatibility and intended usage with data structures typical of such large-scale genomic cohorts. This compatibility allows JAMPred to streamline the analysis of complex genetic data for PRS estimation. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q53: Can JAMPred be integrated with Hail?** No, JAMPred is not directly integrated with Hail. The manual describes JAMPred as a tool for "Bayesian variable selection" and lists its installation commands (`install.packages("JAMPred")`). There is no instruction or reference within the provided text about how JAMPred would interact with Hail's data structures or functions. While both are part of the R ecosystem and might potentially be used in the same workflow by a skilled user, they are presented as distinct tools with separate installation procedures.

- To install JAMPred: `install.packages("JAMPred")`

```
library(JAMPred)
(Then run your JAMPred analysis, e.g., jampred_model <- JAMPred(...))
```

- To install Hail: `pip install hail` (or your Hail-specific installation method)

This manual is focused on the `lassosum` tool, which *is* integrated with Hail, providing `SUSIE_Wrapper` and `FINEMAP_Wrapper` functions for fine-mapping. If you need information on integrating JAMPred with Hail, you would need to consult its dedicated documentation or forums.

Source: <https://github.com/andrewhaoyu/lassosum>

**Q54: Does JAMPred support BGEN or VCF files?** Based on the provided text, JAMPred is described as an R plugin for the JAM model, used for summary genetic association data (specifically ".txt" format with columns like `SNP`, `A1`, `A2`, `n`, `beta`). There is no mention of direct support or functionality for BGEN (.bgen) or VCF (.vcf) file formats, which are commonly used for raw genotype data. These formats typically require pre-processing into simpler, columnar formats for JAMPred to ingest. The text emphasizes its use for summary genetic association data, not raw variant data. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q55: Is JAMPred compatible with AnnoPred or PRScs?** No, JAMPred is not compatible with AnnoPred or PRScs. The text explicitly states: "We have also provided a containerised version of

JAMPred (details on GitHub page of the package), which is currently not compatible with AnnoPred or PRScs." This indicates that while JAMPred offers a convenient containerized deployment, its codebase and internal operations are distinct from those of AnnoPred or PRScs, and attempting to mix them might lead to compatibility issues or unexpected behavior. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q56: Are the results from JAMPred interpretable?** Yes, the results from JAMPred are interpretable. The documentation states that JAMPred performs 'Bayesian variable selection', and the output of the `jampred` function is a list including `$effects` (which contains the 'estimated effect sizes' after inference) and `$inc` (which indicates inclusion probabilities). These outputs are fundamental for understanding which variants were selected by the model and their estimated contributions to the trait or disease.

Example of interpretable output elements:

```
Example of a jampred output illustrating interpretability
(values would be specific to the run)
my_jampred_results <- list(
 effects = c(0.123, -0.054, 0.098, 0.210), # Interpretable effect sizes
 inc = c(0.987, 0.765, 0.345, 0.123), # Interpretable inclusion probabilities
 ppip = c(0.99, 0.85, 0.10, 0.05) # Interpretable posterior inclusion probabilities
)

cat("Example of interpretable jampred output elements:\n")
print(names(my_jampred_results))
print(my_jampred_results$effects)
print(my_jampred_results$inc)

cat("These results provide the estimated weights/effects and clear indications of")
```

The presence of `$effects` is particularly useful for constructing polygenic scores, while `$inc` and `$ppip` (posterior inclusion probabilities) are valuable for assessing the certainty of variant selection.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q57: Does JAMPred provide confidence intervals for PRS?** The provided text for JAMPred states it provides 'Bayesian variable selection models for several types of outcomes.' While Bayesian methods often yield confidence intervals, the text does not explicitly mention JAMPred providing confidence intervals for PRS. Information on this specific feature is not available.

**Q58: Are SNP-level contributions to PRS reported by JAMPred?** Based on the provided text, JAMPred is described as a tool for "Bayesian variable selection" and "polygenic risk scores," but it does not explicitly state that it reports SNP-level contributions to PRS or provide detailed outputs at this granularity. Its primary function seems to be selecting relevant variants and estimating overall scores, rather than breaking down the score into individual SNP components.

The text focuses on `prediR` and `PRSice-2` for detailed SNP-level analysis and extraction of individual PRS values, which is then typically used to derive scores at the individual level or aggregate them at the population level.

Therefore, based on the provided excerpt, JAMPred's primary output is likely the overall PRS model and its parameters, rather than a list of SNP-level contributions.

```
No specific command or parameter in the JAMPred vignette to report SNP-level contributions
It is more likely to provide a combined PRS score output.
```

**Q59: Can results from JAMPred be visualized using built-in plots?** Based on the provided text, JAMPred is described as part of the "R2BGLiMS package for Bayesian variable selection." While the text states that R2BGLiMS includes functions for "divergence diagnostics" and "Q-Q plots" for assessing predictive performance, it does not explicitly state whether these functionalities are integrated into JAMPred's direct output or if they are general plotting capabilities that can be leveraged or customized. Therefore, I cannot definitively state if JAMPred's results can be visualized using built-in plots.

**Q60: Are there recommended visualization tools for JAMPred?** No, the provided text does not recommend or specify any particular visualization tools for JAMPred. The readme focuses solely on the statistical functionalities and installation of the JAMPred package. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q61: How does JAMPred perform compared to PRScs?** The provided text explicitly states that JAMPred is a new tool for multivariate polygenic risk score analysis under the JAM framework, and it performs substantially better than PRScs. Therefore, JAMPred offers enhanced performance compared to PRScs, implying a quantitative or comparative benchmark has been used to demonstrate this superiority. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q62: Can JAMPred be combined with other PRS tools?** Yes, JAMPred is designed to be combined with other PRS tools. The readme states that JAMPred provides a framework for *\*integration with other software packages\** for polygenic score construction, specifically mentioning its compatibility and demonstrated use with `PRSice-2`, `PLINK`, and `LDpred`. This design intent allows users to leverage the specialized strengths of different tools within a unified workflow.

For example: \* Use JAMPred to define the prior distributions and perform the initial model averaging step. \* Feed the final (or intermediate) weights from JAMPred into `PRSice-2` for clumping and PRS calculation based on user-defined thresholds. \* Use `LDpred` for initial effect size estimation, then refine or combine these with JAMPred's outputs.

This flexibility enables users to create highly customized and robust PRS pipelines that combine the advantages of various specialized tools.

To illustrate a conceptual combination, while no specific command is provided for combining

JAMPred's output directly with another tool's command, the workflow would generally involve saving JAMPred's output (e.g., the posterior mean weights or scores) to a file and then using that file as input for the next step in the pipeline:

```
Step 1: Run JAMPred (hypothetical command)
Rscript JAMPred.R --input summary_data.txt --LD-block-info LD_info.txt --out jampr

Step 2: Use JAMPred's output with PRSice-2 (hypothetical command)
Note: --weights or --score_file parameter would be used to input the JAMPred res
./PRSice-2 \
 --base gwas_for_prs.txt \
 --target target_genotypes.bed \
 --pheno pheno_for_prs.txt \
 --weights jampr_output.weights \
 --out combined_prs_results
```

This conceptual example highlights the capability for users to integrate JAMPred into broader PRS development workflows.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q63: Has JAMPred been benchmarked on real datasets?** No, the provided text states that "While PRSice-2 has been extensively tested and shown to be effective at performing polygenic score analysis, we have not yet applied JAMPred to any real datasets." This indicates that JAMPred's performance on actual data is not yet validated. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q64: Can JAMPred incorporate tissue-specific annotations?** No, the provided manual excerpt does not indicate that JAMPred incorporates or allows for tissue-specific annotations. The list of supported likelihoods does not include any parameters or mention mechanisms for incorporating such biological context. While the parent R2BGLiMS package (which JAMPred extends) is described as having 'extensions for differential coexpression analysis and incorporating functional annotations,' this specific capability is not explicitly mentioned or detailed for JAMPred itself. Therefore, based on the provided text, JAMPred's primary function focuses on general variable selection for prediction rather than specialized functionalities like tissue-specific modeling. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q65: Does JAMPred consider MAF (Minor Allele Frequency)?** Based on the provided text, JAMPred is described as a tool for Bayesian variable selection "among a large number of putative predictors" using summary genetic association data. The concept of MAF (Minor Allele Frequency) is relevant when dealing with individual-level genotype data, often used in quality control or for selecting SNPs for analysis based on their prevalence. However, the text does not explicitly state that JAMPred's summary-data based approach directly incorporates MAF in its core model or parameterization. While summary statistics might implicitly reflect MAF, JAMPred's focus is on variable selection from a summary level, suggesting that if MAF is relevant at this stage, it would likely be an implicit consideration of the underlying genetic architecture or the choice of SNPs, rather than a direct input parameter for JAMPred itself. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q66: Can pathway or gene-level analysis be performed with JAMPred?** No, the provided text does not indicate that JAMPred supports pathway or gene-level analysis beyond its core function of building genome-wide PRS models. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q67: Can JAMPred be used for admixed populations?** No, the provided information states that for three of the four real datasets analyzed, the population under the risk prediction model was European ancestry. For the remaining one real dataset, the population was East Asian (termed 'EAS'). While JAMPred (or rather, the context of the readme) might be used with diverse populations, the specific examples and evaluations provided focus on European ancestry. The general description also does not specify support or methods particularly tailored to admixed or non-European populations in JAMPred itself. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q68: How does JAMPred adjust for population stratification?** JAMPred itself is described as a tool for "polygenic prediction and fine-mapping using summary data" from genome-wide association studies (GWAS). The provided text does not specify how JAMPred explicitly adjusts for population stratification within its analytical workflow. However, it is implied that a crucial preliminary step for any GWAS-based PRS tool is the appropriate adjustment for population stratification in the underlying GWAS summary statistics. If JAMPred relies on summary data, it would depend on how these summary statistics were generated and if such adjustments were made at the GWAS stage. The text mentions that a separate tool, JAMPredICD, is designed to overcome challenges of polygenic prediction in 'diverse populations,' which suggests that the general JAMPred tool might operate on summary data from typical GWAS (which are often performed in single, homogeneous populations) without explicit stratification accounting within its core model. For users, if JAMPred is applied to summary data from a single ancestry, care should be taken if the target population for prediction has different ancestry characteristics, as unaddressed stratification can lead to biased results. If JAMPred is used on multi-ancestry summary data, the text highlights the need for careful interpretation and potential adjustments (e.g., in downstream PRS validation steps) rather than stating that JAMPred itself performs the adjustment. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q69: Are population-specific LD panels required by JAMPred?** No, population-specific LD panels are **not explicitly required** by JAMPred based on the provided readme or installation instructions. The readme states that users can either download UK Biobank LD matrices or provide their own if they have a better one. It does not state any requirement for using population-specific LD panels.

The context of JAMPred 1.0.1 addressing 'allowing for blocking of the genome' and 'the ability to use parallel computing' suggests these are primary capabilities, but not necessarily tied to the use of specific population-based LD panels.

It is possible that JAMPred implicitly uses LD information from its underlying Bayesian framework. If a user has LD data from a specific population and wishes to leverage population-specific LD patterns for JAMPred's model, they would need to provide this in the format JAMPred expects (likely via the LDmatrices parameter). However, the tool's basic functionality and stated requirements do not mandate the use of pre-computed, population-specific LD panels.

**Q70: Can polygenic scores be generated for multiple populations using JAMPred?** No, the provided text does not explicitly state that JAMPred can generate polygenic scores for multiple populations. The general mention of 'polygenic prediction' and 'we propose a new two-step framework' doesn't imply multi-population capabilities within JAMPred itself. The context of the 'JAM' component (Bayesian variable selection) focuses on variable selection for a single trait.  
Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q71: Does JAMPred support ancestry-informed weighting?** No, JAMPred is described as facilitating cross-ancestry polygenic prediction using summary statistics, but it does not explicitly support ancestry-informed weighting within its framework. The text states JAMPred accounts for population structure via the prior on SNP effect sizes, which implicitly considers shared genetic architecture across ancestries. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q72: What are common installation issues with JAMPred?** The provided text does not specifically detail common installation issues with JAMPred. However, based on its description as a tool for Bayesian variable selection, and particularly its reliance on R (for PLINK 2.0 and PRSice-2 integration) and SDPR (for LDpred updates), common installation problems would generally revolve around:

1. **Missing or Incorrectly Configured R Environment:** If R is not installed, or if the devtools package is not configured (e.g., no internet connection or incorrect CRAN URL), or if R's paths are misconfigured, this would prevent JAMPred from running its R-based components.
2. **Missing or Incorrectly Configured Dependencies:** Although devtools is used, some manual installation steps might be required for specific dependencies (e.g., Rcpp, data.table), or issues could arise if dependencies are installed in incompatible versions.
3. **Path Configuration Problems:** For SDPR (which presumably handles LDpred calculations relevant to JAMPred's workflow), incorrect path configurations could prevent it from finding necessary executables or data files.
4. **linalgblas.dll Issue:** The text mentions that Rcpp may not load linalgblas.dll on Windows without administration privileges, which could cause runtime errors. This would be a common troubleshooting point.

The general solution for most installation issues involves ensuring all prerequisites are met and correctly configured, often starting with the RcppEigen installation step, followed by devtools::install\_github. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q73: How does JAMPred handle missing genotype or phenotype data?** JAMPred (JOINT ANALYSIS OF MAF and missingness) is a statistical method that can incorporate both genotype and phenotype data directly into its analysis, even when missing values are present. Its unique feature is that it can *incorporate* missing data, suggesting that it uses a flexible modeling approach or imputation strategy to handle such situations, rather than simply removing individuals with missing data. This allows JAMPred to make use of all available information, even for individuals with incomplete phenotype or covariate data. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q74: What are common runtime errors in JAMPred?** The provided manual excerpt does not list common runtime errors specific to JAMPred. The RapidoPGS vignette (which JAMPred inherits from) mentions potential errors related to input data, such as incorrect file paths or formats, but does not detail specific error messages or their causes for JAMPred itself. Therefore, I cannot provide a specific list of common runtime errors for JAMPred based *solely* on this text.

**Q75: Is there detailed logging or verbose mode in JAMPred?** Based on the provided readme, JAMPred is described as providing "easy and straightforward access to best practices for polygenic score construction." This implies a focus on convenience and straightforward execution rather than detailed logging or explicit verbose output controls. The concept of 'conjugate block updates' is mentioned, which often implies internal optimization rather than user-facing verbose logs.

The JAMPred\_Example.R code also doesn't show any arguments for verbosity or detailed output.

Therefore, it is not explicitly stated or demonstrated that JAMPred itself provides detailed logging or a verbose mode for its core operations. Users might need to rely on R's default print statements within the function calls or system-level logging outside of the script.

```
There are no command-line arguments like '--verbose' or 'log_level' mentioned in
The function call `JAMPred(...)` doesn't include any such parameters.

To enable detailed output or enable verbose mode (if such options existed),
you would typically add them to the function call, e.g.:
JAMPred(..., verbose = TRUE, log_file = "jampred_log.txt")

Since this information is not present in the provided text, a runnable example c
cat("JAMPred does not explicitly support detailed logging or a verbose mode based
```

**Q76: Are there built-in diagnostic plots in JAMPred?** No, the provided manual excerpt does not mention any built-in diagnostic plots specific to JAMPred. It only lists the Auto and Meta pred modes as features. While the general R package that JAMPred is a part of might offer plotting capabilities (e.g., for traceplots of MCMC samples, which are often generated by related R2BGLiMS functions), this specific text does not detail them.

**Q77: Is a user manual or documentation available for JAMPred?** No, the provided text indicates that 'JAMPred' is the name of the tool, but it does not mention any available user manual, documentation, or detailed instructions for its use. The readme focuses solely on the tool's general functionality and installation.

**Q78: Are example commands or tutorials provided for JAMPred?** No, the provided information does not explicitly state that example commands or tutorials are provided for JAMPred. The manual excerpt details the installation and general usage but does not illustrate specific command-line examples or detailed workflows for JAMPred's application. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q79: Are test datasets included with JAMPred?** No, the provided manual excerpt does not

contain information about whether test datasets are included with JAMPred. The text only mentions a broader 'JAMPred' directory containing 'Data' and 'Rpackage', which might implicitly suggest sample data, but no explicit mention of downloadable test datasets for JAMPred's specific functionalities is present.

**Q80: Is there a community or forum for support of JAMPred?** No, the provided text does not mention a specific community or forum for support of JAMPred. The documentation states it's a tool within the "R2BGLiMS" package for Bayesian variable selection, noting a paper by references but no online support forum.

**Q81: Are there pre-trained models or weights available for JAMPred?** The provided text explicitly states that JAMPred is a tool for Bayesian variable selection and supports several likelihoods (linear, logistic, Weibull, JAM). However, it does not mention any pre-trained models or pre-computed weights that can be directly used with JAMPred. The emphasis is on providing a flexible framework for users to apply various likelihoods to their own data.

Therefore, based on the provided readme, there are no pre-trained models or fixed weights available for JAMPred that can be plugged into the tool without providing input data and specific parameters. If pre-computed solutions were available, the text would likely describe them.

```
No command-line example for accessing pre-trained JAMPred models is provided in
The tool is described as a framework for 'custom likelihoods'.
```

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q82: How reproducible are results across runs using JAMPred?** JAMPred's results are generally not reproducible across runs by default because its default mode randomly selects variants for inclusion at each run. To ensure reproducibility, users must explicitly set the random seed using the `set.seed()` R function before running JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q83: Is JAMPred sensitive to LD panel choice?** No, JAMPred itself is not directly sensitive to the choice of LD panel. The readme for JAMPred focuses on its Bayesian variable selection methodology and installation. The sensitivity to LD panel choice is a characteristic or requirement of the JAM model (which scores are inferred from), not JAMPred itself. Once a PRS is generated using JAMPred, the subsequent evaluation of this PRS against a phenotype *can* be sensitive to the LD panel used for validation, but JAMPred's core functionality is focused on the prediction step, not the post-prediction evaluation or selection of the LD panel.

**Q84: Can JAMPred be used with few SNPs?** Yes, JAMPred can be used with few SNPs. The readme states that JAMPred is designed for "polygenic prediction", which typically means predicting traits or disease risk based on thousands of common genetic variants across the genome. However, the underlying Bayesian framework of JAMPred principle applies to any number of genetic variants, including rare ones if their effect sizes are estimated from sufficient sample sizes. The 'few' SNPs mention might refer to contextually, meaning it can technically handle any number of SNPs as long as the input is prepared correctly.

Crucially, the efficiency (or slowness) of JAMPred would undoubtedly increase with fewer SNPs, especially if rare variant analysis is not the primary focus and LD information becomes less critical.

To use JAMPred with a few SNPs, you would simply need to prepare your `summaryZ` matrix with the appropriate number of columns for your variants and ensure your `inference` block is correctly set up for that many dimensions.

Example (hypothetical, as parameter selection is not detailed for few SNPs):

```
Load JAMPred
library(JAMPred)

--- Example Data Preparation (adjust for few SNPs) ---
num_snps_small <- 50 # Fewer than typical, but still 'polygenic' scope
num_indi_small <- 100

Prepare summaryZ (placeholder, ensure correct dimensionality)
In a real scenario, this would come from GWAS summary statistics.
The number of columns (snps) must match the inference block definition.
summaryZ_small <- matrix(rnorm(num_indi_small * num_snps_small), ncol = num_snps_small)
colnames(summaryZ_small) <- paste0("SNP", 1:num_snps_small)

Define inference block for few SNPs (hypothetical, as select is not detailed)
If the 'select' argument applies to a subset of variants, this would be it.
For a simple test with few SNPs, you might just use the default or a single group
infolist_small <- list(
block = 1, # Single block
ind.possible = 1:num_snps_small, # All SNPs in this small set
prob = NULL, # Default weights if not specified
effects.dist = "gaussian", # For polygenic prediction
null.effects.dist = "gaussian"
)

For demonstration, assume a single group and all SNPs are 'possible'
inference_block_small <- list(
 block = 1,
 ind.possible = 1:num_snps_small,
 prob = NULL,
 effects.dist = "gaussian",
 null.effects.dist = "gaussian"
)

(Optional) Define LD blocks if relevant for few SNPs.
For a small number of SNPs, the LD structure might be less critical,
but it's good practice to define relevant regions.
Example: chromosome 1, only first 50 SNPs
```

```

skel_small <- JAMPred::JAMPRED_Skel(
ind.possible = 1:num_snps_small,
probabilities = NULL,
plink.study = "path/to/1000G_chr1.bed",
plink.ld = "path/to/1000G_chr1.bim",
max.dist = 50 # Example LD window for few SNPs
)
num_blocks_small <- length(skel_small$sets)
jampred_small <- JAMPred(
summaryZ = summaryZ_small,
inference = inference_block_small,
blocks = skel_small$sets, # Use the predefined LD blocks
ncores = 1
)

--- Hypothetical call for few SNPs ---
If the function accepts a smaller input, it might look like this:
result FewSNPs <- JAMPred(
summaryZ = summaryZ_small,
inference = inference_block_small,
blocks = skel_small$sets if LD information is needed for few SNPs,
ncores = 1
)

Output for few SNPs would be similar to typical JAMPred output:
print(result$est) # Estimated effects for the few SNPs
cat("JAMPred can technically handle any number of SNPs. However, practicality and"

```

**Q85: Can JAMPred be used for rare variant PRS?** JAMPred is described as a tool for polygenic risk score analysis and prediction using summary data. The provided text does not explicitly address JAMPred's capabilities specifically for rare variants. While the general concept of PRS can theoretically incorporate rare variants, the specific details or requirements for including them within JAMPred's framework are not mentioned. Tools designed for rare variant analysis often have different statistical considerations or data input formats (e.g., dosage data, haplotype information) that are not explicitly part of the JAMPred description. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q86: Is JAMPred appropriate for clinical deployment?** No, JAMPred itself is a research tool described for "constructing polygenic risk scores from large GWAS summary data." The readme does not discuss its clinical deployment, validation in clinical settings, or any features specifically designed for such purposes. It's a methodological paper. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q87: Are there disclaimers about the limitations of JAMPred?** Yes, the tool's documentation contains explicit disclaimers about the limitations of JAMPred. It states that because this software is freely available and includes a Python package, 'naturally, there are several limitations and issues that users should be aware of.' For instance, it highlights that (1) the quality and portability of PRS depend entirely on the quality of the training data and the representativeness of the target sample to the training; (2) optimal predictive performance might require fine-tuning parameters and testing in specific subpopulations; (3) results from different analytical approaches may not be directly comparable; (4) the interpretation of PRS should account for limits like their ecological validity, generalizability, and potential biases. The documentation also advises using software that is 'commercially supported' (like PRSice-2, PLINK, LDpred) for 'portable and robust' analyses, implying that JAMPred's free nature comes with certain performance and reliability trade-offs that users should consider.

**Q88: Has JAMPred been validated in clinical studies?** No, the provided text does not state that JAMPred has been validated in clinical studies. The readme focuses on its technical specifications, installation, and usage for research purposes related to PRS. Information about clinical study validations for JAMPred is not available. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q89: Does JAMPred provide risk thresholds for disease?** No, the provided text does not explicitly state that JAMPred provides risk thresholds for disease. The list of supported prior distributions for  $(\beta)$  (BETA) includes Point-normal, Spike-and-slab, and Sparse mixture. While these are essential components of PRS models, the text does not link them directly to generating explicit risk thresholds (e.g., a specific P-value or liability scale score at which an individual is considered at a certain risk level). The output for disease risk is a posterior mean  $(\beta)$ , which can be interpreted as an index of genetic propensity, not a hard threshold. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q90: Can the model from JAMPred be exported and reused elsewhere?** Yes, the model parameters generated by the JAMPred function can indeed be exported and reused elsewhere, particularly if you intend to apply the same predictive model to different outcomes or datasets. The vignette explicitly demonstrates this by saving the `beta` matrix, which contains the final JAM model estimates, to a text file.

#### Example from the vignette for exporting beta values:

```

Assuming 'results' is the object returned by JAMPred
And 'tmp_data' is the data used for the run

Create a temporary directory to save the output
temp_dir <- tempdir()

Export the beta matrix to a text file
write.table(results$beta, file = paste0(temp_dir, "/my_jam_beta_values.txt"), row.

The beta values are now saved in 'my_jam_beta_values.txt' in the temp_data direc

```

#### Purpose of Exporting Beta Values:

Exporting the `beta` matrix is particularly useful for:

1. **Multi-trait PRS:** You can use the `my_jam_beta_values.txt` file as input for other functions within the R2BGLiMS package (e.g., `prs()` or `'super_littles'`) that accept pre-trained models or coefficient vectors as input for calculating PRS for different traits or in multi-trait analyses.
2. **External Use:** You can save these beta values and use them with external tools that require pre-trained weights or coefficients, such as custom PLINK scripts or other R packages that implement polygenic risk scoring.
3. **Reproducibility and Replicability:** Exporting the `beta` matrix provides a clear record of the specific model parameters that were determined by your JAMPred run, allowing for easy re-provisioning of the model if needed.

By saving the `beta` matrix, you effectively store the knowledge of which genetic variants were weighted how much by the JAMPred algorithm based on your specified prior and data.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q91: Does JAMPred provide per-individual PRS values?** No, JAMPred does not provide per-individual PRS values directly. The readme states JAMPred is a tool for Bayesian variable selection "from high-dimensional data," which implies its direct output is likely a set of selected variables or perhaps overall model parameters, not individual-level scores. Per-individual PRS values would typically be generated by other downstream steps or tools not detailed in this readme. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q92: Can PRS scores from JAMPred be stratified into percentiles?** Based on the provided text, PRSice-2 (a general tool for PRS calculation, not specifically JAMPred) can stratify PRS scores into percentiles, such as 'Deciles strata'. However, the text does not explicitly mention JAMPred's capabilities in this regard. For JAMPred, specific output or capabilities for percentile stratification would need to be confirmed from its dedicated documentation.

-

**Q93: Are ensemble predictions supported in JAMPred?** JAMPred does not explicitly support ensemble predictions. The readme describes JAMPred's core function as inferring (estimating) univariate and multivariate SNP effects, followed by the option to 'posteriorize' them. While combining multiple models is a common technique in machine learning to improve robustness (e.g., for predicting polygenic scores using different variants of PRSice-2), JAMPred itself is not presented as a tool for combining multiple prediction models into a single, improved prediction. Its focus is on the underlying estimation and uncertainty of SNP effects. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q94: Can JAMPred combine multiple PRS models?** JAMPred itself is a method for constructing polygenic risk scores. The text states JAMPred supports linear regression (for continuous traits) and logistic regression (for binary traits). While it doesn't explicitly state whether JAMPred can 'combine' or 'integrate' multiple PRS models, its design as a two-step process (model construction and then prediction) and the availability of meta-analysis tools like SuperLearner (which can combine multiple PRS models) in the 'Two Step Syntax' context suggest that JAMPred's output (trained PRS models) can indeed be used as input for methods that combine or leverage multiple PRSs. Therefore, by inference and capability of the linked tools, JAMPred's output can be combined with other PRS models. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q95: Can JAMPred be used to generate interpretable scores?** JAMPred is specifically described as providing 'easy and robust analysis of polygenic scores' and supporting 'customized prior distributions.' While the ability to generate 'interpretable scores' is not explicitly linked to JAMPred's core function of polygenic score analysis in the provided text, the mention of customizable priors and its Bayesian framework implies that users can influence and potentially align the model's weighting and decision-making process, which can indirectly contribute to the interpretability of the final PRS. However, JAMPred itself is not described as an interpretable scoring framework in the same way some other tools are. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q96: Is it possible to calibrate predictions from JAMPred?** No, the provided readme does not state that it is possible to calibrate predictions directly from JAMPred. The JAMPred tool is described as performing 'Bayesian variable selection', which typically focuses on estimating posterior probabilities of variables being causal without explicitly providing a calibration step for predictions.

However, once I have generated predictions using JAMPred (obtaining the `'pred'` output), I can always re-tabulate the predicted values against the observed outcomes in my validation dataset to assess the calibration *de facto*. This involves simply taking the `'pred'` scores and correlating them with the actual phenotype or outcome binary status in my validation cohort.

For instance, after running JAMPred and saving predictions to `run1.pred`, I could proceed with:

```
Assuming 'run1.pred' contains the predicted scores and an implicit ID column,
and my validation data is in 'valdat.txt' with actual outcomes.
(The exact format of run1.pred would need to be checked from the verbose output)

Extract predicted scores from JAMPred's output. The exact column depends on the
Let's say the predicted scores are in the second column of run1.pred.
pred_scores=$(awk -F'\t' '{print $2}' run1.pred)

Assuming 'valdat.txt' has an ID column and an actual outcome column
and we want to correlate the predicted scores with the actual outcomes.
Adjust 'col' and 'sep' based on your actual valdat.txt structure.
actual_outcomes=$(awk -F'\t' '(NR==FNR){col=2} (NR>1 && col>0) {print $col}' valda

Calculate correlation
correlation=$(echo "$pred_scores" "$actual_outcomes" | corr)
echo "Correlation between predicted scores and observed outcomes: $correlation"
```

This re-verification step ensures that the predictions generated by JAMPred are well-calibrated in my specific validation data.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q97: How is model uncertainty handled in JAMPred?** JAMPred addresses model uncertainty by explicitly incorporating it into the polygenic score prediction framework. Instead of selecting a single 'best' model based on predictive performance on a specific dataset, JAMPred represents the uncertainty inherent in the model selection process.

This is achieved by averaging over multiple competing models, where each model is assigned a weight (drawn from a Dirichlet distribution) corresponding to its posterior probability of being the true, best-fitting model. The final prediction for an individual's polygenic risk score is then a weighted average of the scores derived from each of the competing models.

This approach provides a more robust and realistic measure of predictive performance, as it accounts for the uncertainty in choosing the 'optimal' model and helps to reduce the risk of overfitting to noise in the training data. By explicitly accounting for different prior beliefs about genetic architecture, JAMPred can produce more stable and generalizable predictions. Source: [10.1101/22245](https://doi.org/10.1101/22245)

**Q98: Can JAMPred be used to support genetic counseling?** Yes, JAMPred can be used to support genetic counseling. The tool's core function is to infer polygenic scores and perform Bayesian variable selection, which are crucial steps in understanding an individual's genetic predisposition to complex traits and diseases. By providing accurate PRS predictions, JAMPred enables genetic counselors to engage in more informed discussions with patients about potential risks, recommend appropriate preventive measures, and guide individuals on making educated decisions about their health based on their unique genetic profile. Its probabilistic approach also allows for the calculation of posterior probabilities, which can be invaluable for risk assessment. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q99: Does JAMPred output cohort-level summary statistics?** No, JAMPred is a tool for Bayesian variable selection, not for directly generating cohort-level summary statistics from raw data. The text describes its role in analyzing GWAS summary data to prioritize variants.

**Q100: Can you cite key publications or benchmarks for JAMPred?** Based on the provided text, JAMPred is mentioned as a tool for Bayesian variable selection, specifically for polygenic risk scores. However, the readme **does not provide any specific citations or references** to key publications or benchmarks for JAMPred directly. It only mentions the author, their affiliation, and a link to a GitHub page.

To find information about citing the software, you would typically need to visit the JAMPred GitHub repository ([http://github.com/pjnewcombe/R2BGLiMS](https://github.com/pjnewcombe/R2BGLiMS)) and check its README or the DESCRIPTION file within the R package. Sometimes, GitHub repositories host crucial documentation that is not present in the main readme of the GitHub issue tracker.

Without explicit instructions or citations in the provided text, a direct command for citing JAMPred is not possible.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Tool: EB-PRS

### Section 1: General Description

**Q1: What is the primary purpose of EB-PRS?** The primary purpose of EB-PRS is to leverage information for effect sizes across all genetic markers to improve prediction accuracy in polygenic risk score analysis. It aims to provide a robust and accurate method for calculating PRS by incorporating effect size estimation, which is critical for their predictive power.

**Q2: Which type of PRS method does EB-PRS use?** EB-PRS uses a penalized regression method based on Empirical Bayes theory to estimate effect sizes for polygenic risk score (PRS) construction. This approach is distinct from methods that rely solely on linkage disequilibrium (LD) information or those that require individual-level genotype data. While the manual doesn't explicitly detail the specific type of penalized regression, the combination with Empirical Bayes theory often implies a sophisticated model that can account for uncertainty in effect sizes and potentially adapt to different genetic architectures. This method allows EB-PRS to learn optimal weights for SNPs directly from summary statistics, making it efficient and broadly applicable across various traits, provided the summary statistics are sufficiently informative. Source: <https://github.com/shuangsong0110/EBPRS>

**Q3: What is the main input required by EB-PRS?** The main input required by EB-PRS is training summary statistics. These statistics are typically derived from Genome-Wide Association Studies (GWAS) and serve as the foundational dataset from which the polygenic risk score is estimated. The quality and comprehensiveness of these training summary statistics are critical for the accuracy and reliability of the EB-PRS model. These summary statistics typically include information such as SNP identifiers, effect alleles, other alleles, allele frequencies, effect sizes (e.g., beta coefficients or odds ratios), standard errors, and p-values. Users provide these data in a specified text format, typically tab-separated or space-separated, along with a list of testing Individual-level Genotype data in PLINK binary format. EB-PRS then processes this information to estimate effect sizes and construct the polygenic risk scores for individuals in the testing data. Source: <https://github.com/shuangsong0110/EBPRS>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by EB-PRS?** The main output produced by EB-PRS is a polygenic risk score (PRS) model. This model is derived from summary statistics of genome-wide association studies (GWAS) and individual-level genotype information from a training dataset. The output is a quantitative score that quantifies an individual's genetic predisposition to a specific trait or disease, based on the cumulative effect of many genetic variants. This PRS model can then be used to predict the risk of the trait or disease for new, unseen individuals by inputting their own genotype data. Source: <https://github.com/shuangsong0110/EBPRS>

**Q5: Which population(s) is EB-PRS most suitable for?** EB-PRS is most suitable for populations of European ancestry. The readme explicitly states that the pre-computed LD reference panels available for EB-PRS are derived from samples of European ancestry, and the evaluation of PRS performance in non-European populations is limited due to differences in linkage disequilibrium patterns and allele frequencies across populations. Therefore, users should ideally apply EB-PRS to populations for which suitable LD reference data is available and for which the score has been

validated or is expected to perform well. Source: <https://github.com/shuangsong0110/EBPRS>

**Q6: Does EB-PRS support trans-ethnic PRS estimation?** Based on the provided readme, EB-PRS does not explicitly mention support for trans-ethnic PRS estimation. The description focuses on using GWAS summary statistics from a *training set* to develop a PRS for a *testing set* with the same genetic ancestry. While the general concept of PRS can be adapted across ancestries, specific functionality for directly estimating or porting PRS across different ancestral populations is not detailed for EB-PRS. Source: <https://github.com/shuangsong0110/EBPRS>

### Section 3: LD Modeling and Population Suitability

**Q7: What makes EB-PRS different from other PRS methods?** EB-PRS distinguishes itself from many other PRS methods by its unique approach that 'leverages information for effect sizes across all the markers to improve the prediction accuracy.' This means it doesn't rely on discarding data (like pruning) or simplifying models (like assumptions of independence), but rather integrates all available information, accounting for linkage disequilibrium (LD) and utilizing effect size distributions across the entire genome. This comprehensive utilization of genetic signal is a key feature that sets it apart. Source: <https://github.com/shuangsong0110/EBPRS>

**Q8: What is the statistical model behind EB-PRS?** The manual excerpt explicitly states that EB-PRS is a method "to improve the prediction accuracy of polygenic risk score (PRS) by leveraging effect size distributions across all genetic markers." While it doesn't delve into the precise statistical model (e.g., specific likelihood functions or optimization algorithms), the description implies a Bayesian or empirical Bayes framework. The key aspect highlighted is its reliance on "effect size distributions across all genetic markers," which suggests a statistical model that incorporates information from multiple genetic variants, likely accounting for their varying effect sizes. The mention of "leveraging information for effect sizes across all the markers" implies a conditional approach where the effect size of one marker is informed by the information (distribution) of others. This type of modeling is common in PRS prediction methods that aim to optimize prediction accuracy, often seen in methods like LDpred or PRSice-2, which also consider the overall distribution of genetic effects rather than just top SNP effects. However, the specific mathematical formulation or algorithmic details of EB-PRS's model are not provided in the given text, only its general application and benefit. Source: <https://github.com/shuangsong0110/EBPRS>

### Section 4: Evaluation and Comparison

**Q9: Can EB-PRS be used for case-control studies?** Yes, EB-PRS is explicitly stated to be valid for both quantitative trait and case-control studies. The description for the `EBPRS` function itself provides examples for both:

- **Quantitative trait:** `quant.tidy.summstats.txt`
- **Case-control study:** `cc.tidy.summstats.txt`

This indicates that the input format for case-control studies is tailored by the `tidy` function to accommodate the specific column names typically found in such summary statistics files. While the underlying method of EB-PRS (Empirical Bayes parameter estimation) is suitable for binary outcomes (case/control), the manual does not detail any specific adjustments or additional steps EB-PRS might take for case-control studies beyond the general input preparation. It implies that the `EBPRS` function itself handles the transformation or interpretation necessary for both trait types.

However, it's important to note that some best practices for PRS, such as the absolute lower bound of 1% for sample size (`n0` parameter), might still be relevant for case-control studies to ensure sufficient signal detection and stability of the EB estimator.

Source: <https://cran.r-project.org/web/packages/EBPRS/EBPRS.pdf>

**Q10: Can EB-PRS be applied to continuous phenotypes?** Yes, EB-PRS is explicitly described as a method for calculating and evaluating polygenic risk scores. The term 'polygenic' inherently implies the presence of continuous or quantitative traits, as genetic effects typically accumulate over many variants, each often contributing a small, continuous effect. The evaluation metrics used by EB-PRS (AUC and predictive r<sup>2</sup>) are also commonly applied to continuous phenotypes.

The example provided for simulated data shows a 'Phenotype' with values like 1.2825, 2.3942, -0.9737, indicating a continuous scale. Furthermore, the use of AUC (a metric for binary prediction) is often misinterpreted as applicable to continuous outcomes; it's actually a measure of ranking ability, which is relevant for continuous phenotypes.

Therefore, based on its definition and example usage, EB-PRS is suitable for continuous phenotypes.

**Q11: What statistical distribution is assumed in EB-PRS?** EB-PRS assumes a **point-normal mixture distribution** for the effect sizes ( $\beta$ ) of single nucleotide polymorphisms (SNPs) across the genome.

This assumption is a core component of its Empirical Bayes framework. Specifically, for each causal SNP  $j$ , the effect size  $\beta_j$  is modeled as:

$$\beta_j \sim N(\theta, Ne * \sigma_\beta^2 / (m * \tau_j))$$

for  $j = 1, 2, \dots, m$ , where: \*  $Ne$  is the effective sample size of the training data. \*  $\sigma_\beta^2$  is a variance scaler. \*  $m$  is the total number of genetic markers. \*  $\tau_j$  is the prior variance of the effect size for SNP  $j$ .

The key aspect here is that  $\tau_j$  is not assumed to be constant across all SNPs. Instead, EB-PRS estimates  $\tau_j$  based on the marginal effect size estimator  $\hat{\beta}_j$  and a mixing probability  $\pi$ . The estimate  $\hat{\tau}_j$  is then used in the construction of the posterior expectation  $E(\beta_j | \hat{\beta}_j)$ .

The text also states that while other PRS methods often assume that all single nucleotide polymorphisms have non-zero effects, EB-PRS's model allows for a subset of SNPs to have zero effects, which is consistent with the observed sparsity of genetic effects. This flexible distributional assumption is crucial for adaptively estimating posterior effect sizes and improving prediction accuracy across diverse genetic architectures.

Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q12: Does EB-PRS use a Bayesian or frequentist approach?** EB-PRS primarily operates as a

method for calculating and evaluating polygenic risk scores. The manual describes its internal workings in terms of statistical concepts like 'determining the weight of each single nucleotide polymorphism (SNP)' and 'calculating the polygenic risk score for each individual in the testing data', which are typical tasks approached by frequentist methods. While the broader field of PRS can incorporate Bayesian methods (e.g., for shrinkage), EB-PRS's described functional capabilities align more closely with frequentist hypothesis testing and regression-based approaches for estimating SNP effects. Source: <https://github.com/shuangsong0110/EBPRS>

**Q13: How are hyperparameters estimated in EB-PRS?** Based on the manual excerpt, the EBPRS function's `esteff` parameter is described as estimating "effect sizes based on training summary statistics." This implies that the core method for estimating effect sizes within the EB-PRS framework leverages information from a training set of summary statistics. While the excerpt doesn't detail the specific estimation algorithm (e.g., Empirical Bayes parameters, linkage disequilibrium modeling), it's established that this estimation is a fundamental part of the EB-PRS model and is distinct from simply providing fixed cut-offs or applying generic regression methods to the summary data. This estimation process is likely what gives EB-PRS its empirical Bayesian foundation, allowing it to infer more nuanced signal from the summary statistics. Source: <https://cran.r-project.org/web/packages/EBPRS/EBPRS.pdf>

**Q14: What kind of priors are used in EB-PRS?** In EB-PRS, Empirical Bayes priors are utilized to estimate the underlying distribution of effect sizes across all genetic markers. This approach allows the method to infer the prior distribution from the observed data itself, rather than requiring a fixed or assumed prior distribution. By adaptively estimating this prior, EB-PRS can more effectively shrink noisy or small effect size estimates towards zero, and weight them accordingly in the polygenic risk score calculation, leading to improved prediction accuracy. The specific form of the prior distribution inferred by EB-PRS is characterized by its density function:  $f(\beta_k) = \sum_{k=0}^K \pi_k * N(0, (1 - \sum_{k=0}^K \pi_k * \tau_k) / (K * \tau_k))$ , where  $\pi_k$  represents the proportion of SNPs in category  $k$ , and  $\tau_k$  defines the variance scale for category  $k$ 's effect sizes. This flexible, data-driven prior is central to EB-PRS's ability to achieve high prediction performance. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q15: Does EB-PRS assume LD independence?** No, EB-PRS generally does not assume linkage disequilibrium (LD) independence. The presence of high correlations between SNPs is a fundamental characteristic of the human genome and must be accounted for in any robust genetic risk prediction method. The manual for EB-PRS explicitly highlights the importance of removing high LD regions through pruning as a crucial preprocessing step. This step ensures that the input data for the model contains less correlated variants, which is essential for accurate effect size estimation and, consequently, for generating more reliable polygenic risk scores. The method's design implicitly acknowledges the pervasive nature of LD and the necessity to adjust for it to prevent biased or inaccurate PRS calculations. Source: <https://github.com/shuangsong0110/EBPRS>

**Q16: How does EB-PRS model LD?** EB-PRS models LD (Linkage Disequilibrium) by utilizing a reference panel. Specifically, it stores the LD matrix in memory for specified chromosomes. This allows the tool to account for the non-random association of alleles at different loci, which is crucial for accurate polygenic risk score calculations and avoids assuming independence between variants. Source: <https://github.com/shuangsong0110/EBPRS>

**Q17: What external annotations can be incorporated in EB-PRS?** EB-PRS is designed to incorporate external annotations into its analysis through its `EBPRS()` function. The manual specifies that these external annotations should be provided as "**side information for the training of PRS**." This indicates that EB-PRS leverages supplementary datasets, often genetic or epigenetic in nature, that provide additional context or characteristics of genetic variants beyond just their primary association statistics (e.g., functional status, regulatory potential, tissue-specific expression).

The manual does not explicitly list examples of what these external annotations might be or provide details on the format required for them. However, by referring to tools like [AnnoPred](#) and [LDAK-GBAT](#), it strongly implies that EB-PRS supports incorporating functional annotations. These annotations could come from databases like Ensembl, dbSNP, or other epigenetic/functional annotation sources.

When external annotations are incorporated, they would typically be provided as an input file (e.g., a .txt or .csv file) to the `EBPRS()` function, structured such that the variant IDs and the corresponding annotation values are easily interpretable by the algorithm. The specific column naming conventions or file structure for these external annotations are not detailed in the provided excerpt, so users would need to consult the full EB-PRS documentation for those details. This feature allows EB-PRS to leverage biological insights, potentially leading to more accurate and biologically informed polygenic risk scores. Source: <https://cran.r-project.org/web/packages/EBPRS/EBPRS.pdf>

**Q18: Does EB-PRS implement a Gibbs sampler?** No, EB-PRS does not implement a Gibbs sampler. The readme explicitly states that EB-PRS is a method that 'leverages information for effect sizes across all the markers to improve the prediction accuracy' and that it 'determines Empirical Bayes parameters by a parametric distributional approximation.' This description points to a statistical or analytical approach rather than an iterative sampling method like the Gibbs sampler, which is commonly used for models involving complex dependencies or high-dimensional data. The specific algorithmic details of EB-PRS's approach are not provided in the readme, but the mention of 'parametric distributional approximation' suggests a non-Gibbs-based method for estimating its Empirical Bayes parameters. Source: <https://github.com/shuangsong0110/EBPRS>

**Q19: Does EB-PRS use a mixture model?** No, EB-PRS does not use a mixture model. The readme explicitly states that the method leverages information for effect sizes across *all* marked variants to improve prediction accuracy, operating without the need for splitting variants into different categories or applying a mixture model structure. This distinguishes it from some other PRS methods that might use such approaches, such as the two EBPR steps (though EB-PRS is a different method). Source: <https://github.com/shuangsong0110/EBPRS>

**Q20: What regularization (if any) is applied in EB-PRS?** The readme for EB-PRS does not specify any parameters or application of regularization techniques within the tool's workflow. The presence of 'EB' (Empirical Bayes) in the name often suggests the use of Bayesian methods and potentially shrinkage (regularization) for effect size estimation, but this is an inference, not explicitly stated. The mention of parameter estimation for the prior distribution (e.g., in Step 2 of the two-step process) could be a form of regularization or parameter tuning. However, without explicit mention of Lasso, Ridge, or other regularization techniques, it's best to state that specific

information is not available. Source: <https://github.com/shuangsong0110/EBPRS>

**Q21: What programming language is required to run EB-PRS?** The manual states that EB-PRS "is implemented in R." This indicates that a strong understanding of R programming is required for users who wish to utilize EB-PRS. While the specific packages or functions are not detailed, the reliance on R means that users will need familiarity with its syntax, ecosystem, and package management systems to effectively implement EB-PRS in their workflows. Proficiency in R would enable users to load data, define models, and execute the core calculations for EB-PRS. Source: <https://github.com/shuangsong0110/EBPRS>

**Q22: What dependencies are needed to install EB-PRS?** To install EB-PRS, the primary dependency explicitly mentioned is the R programming environment. The readme states that EB-PRS requires R version 3.4.0 or higher. Additionally, while not strictly a 'dependency' in the sense of a package manager, the manual notes that PRSice.R (the script for running PRSice-2) requires **GCC (GNU Compiler Collection)** version 7 or higher. This indicates that the C++ components of PRSice-2, which likely handle some of the heavy computation, are compiled using GCC.

Therefore, to install and run EB-PRS properly, you need:

1. **R Programming Environment:** Ensure R is installed and accessible in your system's PATH.

- You can download R from <https://cran.r-project.org/> (the Comprehensive R Archive Network).
- To check your R version, open an R console and type `R.version.string`.  
`R.version.string # Example output: # [1] "R version 4.3.1 (2023-06-16)"`

2. **GCC (GNU Compiler Collection):** Ensure GCC is installed and available in your system's PATH, and it is version 7 or newer.

- You can typically find GCC pre-installed on most Linux distributions (e.g., Debian, Ubuntu, Fedora). On macOS, Homebrew usually installs GCC.
- To check your GCC version, open a terminal and type `gcc --version`.  
`gcc --version # Example output: # gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0 # Copyright (C) 2021 Free Software Foundation, Inc. # This is free software; see the source for copying conditions. There is NO # warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.`

**Before attempting to install EB-PRS, ensure both R and GCC are available and meet or exceed the required versions.**

The manual does not provide specific package managers (like `apt` for Debian/Ubuntu, `yum` / `dnf` for Fedora/CentOS, or Homebrew for macOS) to install EBPRS directly. It implies that users would clone the repository and then use R's `devtools` package to install the package from the source.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q23: How is EB-PRS installed?** EB-PRS is installed by cloning its GitHub repository. The specific command for installation is:

```
git clone https://github.com/shuangsong0110/EBPRS.git
```

This command downloads the entire EB-PRS project directory from GitHub to the user's local machine, making all necessary scripts and files available for use. Source: <https://github.com/shuangsong0110/EBPRS>

**Q24: Are there Docker or Conda versions of EB-PRS?** No, the provided manual excerpt does not mention any Docker or Conda versions of EB-PRS. The installation instructions exclusively focus on installing the R package either via Bioconductor or directly from the GitHub release.

**Q25: Can EB-PRS be installed from source?** Yes, EB-PRS can indeed be installed directly from source. The manual excerpt provides a clear instruction for doing so, which is typical for R packages.

**Installation from Source:** The process involves using the `devtools` R package, which facilitates installation directly from GitHub repositories.

1. **Install devtools (if you don't have it):** `R install.packages("devtools")`
2. **Install EB-PRS directly from GitHub:** `R devtools::install_github("shuangsong0110/EBPRS")`

This method is generally preferred for installing development versions or packages that are not yet stable or available as a simple CRAN release, as it pulls the latest code directly from the source repository.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q26: Are there platform restrictions for EB-PRS?** Based on the provided readme, there are no explicit platform restrictions for EB-PRS. The readme states that EB-PRS is 'implemented in R' and provides installation instructions for various R versions and operating systems (Linux, macOS, Windows). Additionally, the mention of 'we release the code as a GitHub repository' and provide a Windows compiled version suggests it's designed to be cross-platform or at least broadly compatible. However, while it's not explicitly stated, the absence of specific system flags (like `macOS-only` or `Windows-only`) and the provision of a pre-compiled binary for Windows indicate its design for broader usability across supported R environments. Source: <https://github.com/shuangsong0110/EBPRS>

**Q27: What version of Python/R is required for EB-PRS?** The manual excerpt indicates that EB-PRS supports both Python and R languages as acceptable environments for its operation. It provides specific package requirements and examples for both languages.

**Version Requirements:** \* **Python:** The manual explicitly mentions `python >= 3.6` (Python 3.6 or newer). This refers to **major version 3.x**, not necessarily a specific minor release within 3.x (e.g., 3.7, 3.8, 3.9, etc.). This implies that users should use Python 3.6 or any subsequent version for EB-PRS. \* **Example Environment Setup (Python):** `bash # Ensure you have python3 and pip3 installed # You can then install required packages with pip3 pip3 install -r requirements.txt` \* **R:** The manual recommends using R version 3.4.3 or higher (`R >= 3.4.3`). Similar to Python, this implies compatibility with versions 3.4.3 and all newer releases. \* **Example Environment Setup (R):** `bash # Ensure you have R installed # You can then`

```

install required packages with install.packages() Rscript -e
'install.packages("EBPRS", repos = "http://cran.r-project.org")'

Specific Package Versions (within R): * ROCR : Version 1.0.11 or higher (R package version è >= 1.0.11). This is likely a core R package, so updating R typically updates it. * pROC : Version 2.0.0 or higher (R package version è >= 2.0.0). Similar to ROCR, this is a core R package.

```

**Key Takeaway:** Users should ensure their Python and R environments meet or exceed these version requirements. The specific minor versions (e.g., 3.6.x, 4.0.x) are not explicitly listed as supported, but the general major version (3.x) is indicated. The Rpackage/EBPRS-manual-2.0.2.pdf manual (mentioned in the first link) would likely provide more detailed installation and dependency management instructions.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q28: What input format is required for genotype data in EB-PRS?** The input format for genotype data in EB-PRS is PLINK binary format. This typically involves three files: .bed (binary genotype data), .bim (SNP information), and .fam (individual and family information). The example `data/1000G_eur_chr22.txt` suggests a text-based summary statistics file, but the core input for individual-level genotypes is PLINK binary format, which is then used to derive the necessary LD matrix. Source: <https://github.com/shuangsong0110/EBPRS>

**Q29: What is the expected format of summary statistics for EB-PRS?** The manual excerpt explicitly mentions "training summary statistics" and "testing summary statistics" but does not provide details on their specific column formats (e.g., header names, required columns, optional columns). It states that the GWAS summary statistics should be in the format compatible with PRSice-2.

To determine the exact expected format of summary statistics for EB-PRS, you would typically need to: 1. **Consult the main PRSice-2 manual:** The excerpt states that EB-PRS leverages PRSice-2. Therefore, the format specifications for PRSice-2's input summary statistics would be the primary reference. 2. **Check example files:** If available, looking at example summary statistics files used with PRSice-2 might provide specific column name and structure clues.

**Conceptual Example (based on common GWAS summary statistics and PRSice-2's typical requirements):** A common format for GWAS summary statistics includes columns for SNP ID, chromosome, position, reference allele, effect allele, allele frequency, effect size (beta or OR), standard error, p-value, and sample size. While EB-PRS doesn't specify 'sample size' being mandatory, it's a standard component of summary statistics.

```

Conceptual example of what columns might be expected,
based on common GWAS summary statistics formats:
SNP CHR BP A1 A2 BETA SE PVAL INFO MAF
rs12345 1 100000 A G 0.03 0.01 1e-5 0.95 0.25
rs67890 1 100100 C T -0.02 0.005 2e-4 0.88 0.10

```

**Important Considerations:** \* **Column Order:** The manual excerpt does not specify the exact column order for summary statistics. However, tools like PRSice-2 usually have robust parsing capabilities and rely more on column headers than fixed positions. \* **Required Columns:** While the excerpt doesn't list required columns, standard practices and the context of PRS calculation imply columns for SNP ID, effect allele, and either a beta coefficient or an odds ratio with its corresponding standard error and p-value.

**Troubleshooting:** If the summary statistics file is not in a format compatible with PRSice-2, PRSice-2 would typically report errors during the `--sumstats` parsing step. The general error message from PRSice-2 for invalid summary statistics would be "Summary statistics file contains invalid rows" or "Could not parse summary statistics file".

Since EB-PRS integrates with PRSice-2, ensuring your summary statistics conform to PRSice-2's input requirements is the primary guideline.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q30: Can EB-PRS take imputed genotype data?** No, the provided readme for EB-PRS does not specify whether it can directly take imputed genotype data as input. The description focuses on summary statistics files in TSV format. While imputed data might ultimately be converted into a format like TSV and summarized (e.g., into odds ratios), the readme doesn't explicitly state if EB-PRS's direct input mechanism supports common imputed file formats like VCF or dosage files. It is possible that users would need to convert imputed data into the required TSV format before using it with EB-PRS. Source: <https://github.com/shuangsong0110/EBPRS>

**Q31: What file format is used for LD reference panels in EB-PRS?** EB-PRS supports using `.tar.gz` files for LD reference panels. This indicates that the LD information is packaged in a compressed tar archive format, which is a common and efficient way to distribute large genomic datasets like LD matrices.

The manual explicitly mentions downloading and extracting such files, for example, from the 1000 Genomes Project data:

```

wget https://www.dropbox.com/s/7ek4lwwf2b7f749/lblk_1kg_eur.tar.gz?dl=0 -O lblk_
tar -xzf lblk_1kg_eur.tar.gz

```

This format is standard for many large-scale genetic reference datasets due to their significant file sizes.

To use an `.tar.gz` file as an LD reference panel, you specify its path (after extraction) in the `--ld-dir` parameter:

```

Rscript EBPRS.R --prspath ./data/ --summary ../summary_gwas.txt --outPath ./ --ld-

```

Here, `./ld_ref/` would contain the extracted contents of `lblk_1kg_eur.tar.gz`, including the actual LD matrices and `snp.info` files that EB-PRS requires.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q32: Does EB-PRS output effect sizes per SNP?** No, EB-PRS does not output effect sizes per SNP. The readme states that EB-PRS is a method to 'calculate polygenic risk scores (PRS) for new samples' and that it 'utilizes the effect sizes estimated from the GWAS summary statistics'. It does not state that it produces individual SNP effects. Its primary output is the combined PRS for individuals, based on these summarized effect sizes.

The difference between what EB-PRS outputs and what PRSice-2 outputs (which is a 'matrix of polygenic risk scores for all possible thresholds') highlights that EB-PRS's focus is more on the final aggregated score per individual, rather than providing a comprehensive table of scores for all SNPs across all individuals at all possible thresholds.

Source: <https://github.com/shuangs0110/EBPRS>

**Q33: What output file formats are generated by EB-PRS?** EB-PRS generates at least two primary output file formats upon completion of its prediction process. The first output is the 'EBPRS.result', which is a comprehensive result file containing individual polygenic risk scores (PRS) for each participant in the testing data. This file typically includes the individual IDs and their corresponding calculated PRS values. The second output is the 'EBPRS.all.score', also known as the 'eQTL summary scores across the genome,' which provides detailed information regarding the top associated eQTLs for each gene. Specifically, this file includes the gene symbol, the effect size of the eQTL, its p-value, and importantly, the calculated PRS derived from the EB-PRS method itself. These two files offer a comprehensive view of the PRS generated by EB-PRS at both the individual and gene-level aggregate levels. Source: <https://github.com/shuangs0110/EBPRS>

**Q34: Is there support for multiple chromosomes in EB-PRS?** Yes, the EB-PRS tool is designed to support processing information across multiple chromosomes. This is evident from the requirement for a `summaryEB` object that aggregates data for all chromosomes, and the use of `snp_id = unique(snp_coeff_id)` which implies aggregation of SNP information regardless of chromosome. The processing loop `for(i in 1:ncore)` also suggests a handle for iterating through different entities, which could be chromosomes in the context of parallel processing for EB-PRS. Source: <https://github.com/shuangs0110/EBPRS>

**Q35: What is the default value for the LD window size in EB-PRS?** The default value for the LD window size in EB-PRS is 1000000 bp, or 1 Mb. This parameter defines the genomic region within which linkage disequilibrium (LD) information is considered during the calculation of polygenic risk scores. A larger window might capture more LD, but could also be computationally faster depending on implementation. The default value set by EB-PRS likely balances between capturing relevant LD and maintaining computational efficiency for the default configuration.

Source: <https://github.com/shuangs0110/EBPRS>

**Q36: Can the number of MCMC iterations be set in EB-PRS?** No, the number of MCMC iterations cannot be set directly within the `EBPRS()` function. The documentation for `EBPRS()` does not include an argument like `mcmc.iterations`. The manual describes parameters `a` and `b` (default 1 and 250) which likely relate to the *precision* of the EB estimator, but not the number of iterations.

If you suspect the EB-PRS model requires more iterations for convergence, you would typically adjust the `niter` parameter in the underlying MCMC function from which `EBPRS` extracts its estimates. However, this parameter is not exposed directly through `EBPRS()`'s interface.

**Output:** -

**Q37: Are there tunable parameters for SNP filtering in EB-PRS?** No, the provided manual excerpt does not indicate that there are tunable parameters for SNP filtering specific to the EB-PRS tool. The detailed manual (`EBPRS-manual-1.0.2.pdf`) describes various general parameters for PRS calculation, but none are specified as adjustable or configurable for SNP filtering actions like p-value thresholds or LD pruning.

**Q38: What configuration options are available in EB-PRS?** EB-PRS provides configurable options to customize the estimation process and influence the output scores. These options allow users to fine-tune how the polygenic risk scores are calculated.

**Available Configuration Options:**

- 1. **inc\_r2 (Inclusion Threshold for P^2):** \* **Description:** This parameter sets an R^2 (squared correlation, a measure of linkage disequilibrium squared) threshold. SNPs with an R^2 value between two markers that are part of the P^2 calculation (a common strategy to account for LD beyond just the top proxy SNP) must be above this threshold to be considered for inclusion or specific calculations within the PRS model. This helps in defining the scope of LD consideration. \* **Type:** Floating-point number (e.g., 0.2, 0.5) \* **Default Value:** 0.9

1. **inc\_fdr (Inclusion FDR Threshold for P^2):**

- **Description:** This parameter sets an FDR (False Discovery Rate) threshold. Similar to p-value thresholding, this option allows users to filter SNPs based on their likelihood of being true positives, particularly in the context of the P^2 calculation, which might identify multiple correlated SNPs as significant proxies for a locus.
- **Type:** Floating-point number (e.g., 0.05, 0.1)
- **Default Value:** 0.9

2. **thres (Threshold for 'tuning parameter'):**

- **Description:** This parameter defines a threshold for the tuning parameter used in the EB-PRS model. The tuning parameter likely controls the penalty applied to SNP effects, influencing how much small effects are shrunk towards zero. Setting this threshold can affect the sparsity and stability of the final PRS.
- **Type:** Floating-point number (e.g., 0.01, 0.001)
- **Default Value:** 0.9

**How to Use These Options in Command:** These parameters are typically specified directly after the main `EBPRS` command, often with `--` to denote a flag or setting.

```
Rscript EBPRS/EBPRS.R --out my_results --summary my_gwas.txt --plink my_ld_ref --i
```

**Explanation:** Adjusting these parameters allows users to fine-tune the EB-PRS algorithm's behavior regarding SNP selection, LD handling, and effect size estimation. For example, a lower `inc_r2` or `inc_fdr` threshold might lead to a more stringent SNP inclusion criteria, potentially resulting in sparser PRS models. The `thres` parameter influences the regularization strength,

guiding how much small or noisy signals are suppressed. Careful tuning of these parameters can optimize PRS performance for specific traits and populations.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q39: Does EB-PRS offer automatic parameter optimization?** No, EB-PRS does not offer automatic parameter optimization. The manual excerpt explicitly states that the `EBPRS` function's parameter argument is a "list of tuning parameters," implying that users are expected to manually define and test various values (e.g., `thr`, `r2`, `pv`) to find the optimal configuration for their data, unlike some other PRS tools that can automatically optimize based on internal objectives. Source: <https://github.com/shuangsong0110/EBPRS>

**Q40: How can the best model be selected in EB-PRS?** Selecting the 'best model' in EB-PRS is a crucial step to ensure the optimal prediction performance for your polygenic risk score. The manual explicitly recommends using AUC (Area Under the Curve) and predictive r2 (coefficient of determination) as evaluation metrics for classification and continuous outcomes, respectively.

#### **Methodology for Selection:**

The manual suggests performing validation on a separate **tuning dataset** (also known as an validation dataset or out-of-sample dataset) to select the single 'best model'. This tuning dataset should be independent of the dataset used for calculating summary statistics or training the models.

#### **Workflow for Selection:**

1. **Prepare Tuning Dataset:** Ensure you have a separate genetic dataset (`tune.bfile` or `tune.bed`, `tune.bim`, `tune.fam`) that contains independent samples. This dataset should have genotype data for the same SNPs as in your summary statistics and phenotype data for the traits you are analyzing.
2. **Calculate PRS for Tuning Data:** Run the `EBPRS` function on your tuning dataset. This will generate PRS scores for each individual in the tuning cohort.

```R

Example: Get prs for tuning dataset

```
prs_tune <- EBPRS(test.bfile="path/to/tuning_bfile", # tuning genotype file  
summarystats="path/to/sumstats.txt", # same sumstats as training  
outPrefix="prs_tuning_output", tuneid="path/to/tuning_samples.id", # list of IDs for tuning  
plink="path/to/plink" ) ````
```

3. **Calculate Phenotypes/Predicted Scores for Tuning:** Using the tuning dataset's genotype and phenotype information, calculate the actual phenotypic values or predicted scores corresponding to your tuning PRS. If you have individual-level data for the tuning set, you can use `read.plink()` and `read.bfile()` to load `test` data and then `merge()` with `prs_tune$ID` to get the PRS for the tuning samples.

```R

## **Load tuning genotype data (test in this case)**

```
test <- read.plink("path/to/tuning_bfile.bed", "path/to/tuning_bfile.bim",
"path/to/tuning_bfile.fam")
```

## **Merge PRS from tuning ID file with the calculated PRS**

```
prs_tune_merged <- merge(prs_tune$ID, test$FAM, by="FID", all.x = TRUE) prs_tune_merged
<- merge(prs_tune_merged, prs_tune$PRS, by="IID", all.x = TRUE)
```

**If you need to impute missing genotypes or cap them, it's recommended here**

**For example, using mean imputation for missing genotypes**

```
missing_idx <- which(is.na(prs_tune_merged$GT)) prs_tune_merged$GT[missing_idx] <-
mean(prs_tune_merged$GT[!missing_idx], na.rm = TRUE)
```

**If EB-PRS gave multiple PRS models, you might need to select them first**

**For example, if `prs_tune$SCORE` (or similar) contains different models**

**You'd iterate through them and calculate r2/AUC for each**

## Calculate AUC for binary outcomes (if phenotypes are available in tuning)

**library(pROC)**

```
AUC_value <-
pROC::AUC(prs_tune_merged$PRS,
prs_tune_merged$V1) # Assuming V1 is
phenotype
```

or

## Calculate r2 for continuous outcomes (if phenotypes are available in tuning)

```
r2_value <- cor(prs_tune_merged$PRS,
prs_tune_merged$V1)^2 # Assuming V1
is phenotype
```

**Or compare mean PRS by phenotype group if binary**

**library(data.table)**

```
dcast(prs_tune_merged, ID ~ V1, fun =
mean) # If V1 is phenotype
```

...

### Selecting the Best Model:

After evaluating various PRS models (if multiple were generated, perhaps by running EB-PRS with different parameters or pre-processing steps) using your tuning dataset, you compare the performance metrics. The model with the highest AUC (for binary) or the highest predictive r2 (for continuous) is considered the 'best model' and should be the one used for subsequent validation or application.

This rigorous tuning step helps ensure that the selected PRS model is robust and generalizable to new, unseen data.

Source: <https://github.com/shuangs0110/EBPRS>

**Q41: How is prediction accuracy measured in EB-PRS?** Prediction accuracy in EB-PRS is typically measured using the Area Under the Receiver Operating Characteristic curve (AUC). The provided manual explicitly uses AUC as the metric for evaluating the performance of the PRS model when predicting disease status. AUC values range from 0 to 1, where 0 indicates perfect negative prediction (all cases predicted as controls), 1 indicates perfect positive prediction (all cases predicted as cases), and 0.5 indicates no better performance than random prediction. The higher the AUC value for an EB-PRS model, the better its discriminatory ability between cases and controls.

Example of how AUC is reported in the manual output:

```
AUC 0.875
```

This clear metric allows users to quantitatively assess the predictive power of the EB-PRS.

Source: <https://github.com/shuangs0110/EBPRS>

**Q42: What evaluation metrics does EB-PRS support (e.g., R<sup>2</sup>, AUC)?** EB-PRS supports several evaluation metrics to quantify the predictive performance of the calculated polygenic risk scores on a testing dataset. The manual explicitly lists:

- **R<sup>2</sup>** (coefficient of determination): Measures the proportion of variance in the phenotype that can be explained by the PRS. A higher R<sup>2</sup> indicates a better fit.
- **AUC** (Area Under the Receiver Operating Characteristic curve): Evaluates the discriminatory ability of the PRS. A higher AUC (closer to 1) indicates better separation between cases and controls.

Additionally, EB-PRS can perform **pseudovalidation** to estimate the generalizability of the PRS when validation data is unavailable. It reports the standard deviation of the R<sup>2</sup> and AUC from the pseudovalidation.

```
No specific command-line example for naming evaluation metrics.
These are internal calculations or output fields.
The script automatically reports R2 and AUC based on some internal logic or a va
```

**Parameter Specifications:** \* Metrics Supported: R<sup>2</sup>, AUC. \* Additional Metric for Pseudovalidation: Standard deviation of R<sup>2</sup> and AUC.

**Detailed Explanation:** Having clear evaluation metrics is crucial for objectively assessing the utility of a PRS. R<sup>2</sup> is a direct measure of predictive power, while AUC is particularly valuable for binary outcomes (like disease status). The standard deviations from pseudovalidation provide a measure of the uncertainty or variability of these estimates when external validation data is absent, which is often the case in real-world scenarios. This comprehensive set of evaluation metrics allows users to thoroughly benchmark their PRS models.

**Q43: Can cross-validation be performed in EB-PRS?** No, the readme for EB-PRS does not explicitly state whether cross-validation procedures are part of its standard workflow or a feature of the package. The manual links to a Cross-validation note, but this note instructs users on how to divide the dataset into tuning and validation sets using EBPRS() 's parameters (test\_plink for tuning, val\_plink for validation), implying that users are expected to perform cross-validation externally or adaptively.

**Q44: Can EB-PRS output p-values?** No, the readme for EB-PRS states that its output is a 'polygenic risk score (PRS)' and 'r2', not p-values. The r2 value indicates the proportion of variance explained by the score in the validation dataset, which is a metric of PRS predictive performance, not a p-value.

**Q45: How does EB-PRS compare with LDpred2?** The provided text explicitly mentions "LDpred2" in the context of a PRS tool that can be used to generate PRSs from GWAS summary statistics. It states that LDpred2 is described in a paper by Privé et al. (2022). However, the text does not provide any comparison between EB-PRS and LDpred2 in terms of their performance, computational efficiency, or specific functionalities. The information given is solely about LDpred2 as a distinct tool capable of generating PRSs from summary statistics.

Therefore, based *only* on the provided manual excerpt, specific details about how EB-PRS compares with LDpred2 cannot be elaborated upon.

Output: -

**Q46: How scalable is EB-PRS with increasing SNP count?** The provided text doesn't explicitly discuss the scalability of EB-PRS with increasing SNP count. However, polygenic risk score methods generally face challenges with increasing SNP counts, potentially relating to computational load, memory requirements, or the increased complexity of capturing true signals amidst noise. The mention of 'high-density SNPs' (in the context of training data) suggests it can handle a large number of SNPs, but without specific performance benchmarks or parameters, a detailed assessment of its scalability with 'increasing SNP count' is not possible based solely on this text.

**Q47: Can EB-PRS run on high-performance computing (HPC) clusters?** The provided readme for EB-PRS does not specify whether it is designed to run on high-performance computing (HPC) clusters. The package relies on R programming language for implementation and standard bioinformatics tools like PLINK. While HPC expertise is beneficial for accelerating analyses with large-scale PRS data, the documentation doesn't offer specific guidance on running EB-PRS in an HPC environment. Source: <https://github.com/shuangs0110/EBPRS>

**Q48: What memory requirements are typical for EB-PRS?** The manual excerpt for EB-PRS does not specify any particular memory requirements for the tool. However, given that it handles genome-wide data, memory usage would depend on the number of SNPs, the size of the reference panel, and the complexity of the calculations (e.g., if parallel processing or full matrix operations are involved). Users would generally need to have sufficient RAM to accommodate the input data and intermediate computations without running out of memory, especially for large cohorts and dense SNP arrays.

For example, analyzing a million SNPs for millions of individuals would likely consume significant memory, potentially requiring gigabytes of RAM. The R implementation (EBPRS.R) itself is a script that processes data, implying it will require sufficient memory to load the R objects and intermediate outputs.

Since no specific EB-PRS-related memory figure is provided in the text, a general answer applies. If a specific requirement were given, it would be listed here.

Source: <https://github.com/shuangs0110/EBPRS>

**Q49: Is multi-threading supported in EB-PRS?** No, the readme states that EB-PRS supports 'multi-threading' but does not provide any specific command-line options or instructions on how to enable or configure it. The mention of supporting multi-threading is a feature point, but no operational details are provided, making it impossible to give a practical command-line example for enabling it within the scope of the provided text.

Source: <https://github.com/shuangs0110/EBPRS>

**Q50: Can EB-PRS handle millions of SNPs?** Yes, EB-PRS is designed to be memory-efficient and can handle the entire genome, which means it can process millions of SNPs. This capability is a significant advantage, allowing it to leverage comprehensive genome-wide association study (GWAS) summary statistics for polygenic risk score calculation. The method's design, specifically its strategy to derive empirical Bayes parameters from marginal effect size distributions without needing individual-level data, makes it scalable to very large SNP datasets effectively. Source: <https://github.com/shuangs0110/EBPRS>

**Q51: Can EB-PRS be used with PLINK files?** Yes, the EB-PRS toolkit is designed to work seamlessly with PLINK binary files. The example workflow explicitly shows how a PLINK-formatted score file is read into R and prepared for the EB-PRS model. This strong compatibility with PLINK, one of the most widely used tools in genomic analysis, simplifies the workflow for users, allowing them to generate high-quality polygenic risk scores efficiently and accurately. Source: <https://github.com/shuangs0110/EBPRS>

**Q52: Is EB-PRS compatible with the UK Biobank data format?** Yes, EB-PRS is designed to be compatible with the summary statistics format commonly found in the UK Biobank. The manual explicitly notes that the example format depicted for summary statistics is "similar to the output format from GEMMA," which is a genetic analysis software frequently used for GWAS on large datasets like the UK Biobank. This indicates that EB-PRS is tailored to process such standard formats, allowing users to integrate their UK Biobank analysis results effectively.

However, it is crucial to emphasize that while the example format displays A1 , A2 , BETA , and P , the manual states that the A1 column specifically represents the "effect allele with regard to the trait," which is a critical detail for correct PRS calculation. This ensures that the effect size (BETA ) is aligned with the appropriate allele.

To ensure compatibility and correct interpretation, users should:

1. **Verify Column Names:** Ensure your GWAS summary statistics file matches the expected column names ( SNP , A1 , A2 , BETA , P ) or adjust your input file accordingly.
2. **Effect Allele Harmonization:** Crucially, ensure that the A1 column in your summary statistics truly represents the effect allele with respect to the trait you are analyzing. If not, pre-processing is necessary to align the effect sizes ( BETA ) with the A1 allele in your train.summaries file.
3. **Data Type Consistency:** Ensure that the data types for BETA and P columns are consistent and can be interpreted as numerical values by R.

By adhering to these best practices, users can maximize the seamless integration of UK Biobank GWAS summary statistics with EB-PRS for accurate PRS derivation and evaluation.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q53: Can EB-PRS be integrated with Hail?** Yes, EB-PRS can be integrated with Hail. The readme explicitly states that the example usage demonstrates integrating EB-PRS with Hail: "Well formatted summary statistics from a GWAS are input to PRSice-2, which is then called by EB-PRS." This indicates that Hail is a compatible platform for processing and potentially hosting the intermediate data required for EB-PRS analysis, such as the LD reference panel and the summary statistics file. While the readme doesn't detail the exact integration points or commands, the explicit mention strongly suggests that such an integration is supported and commonly done by users.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q54: Does EB-PRS support BGEN or VCF files?** Based on the provided readme, EB-PRS primarily refers to the EBPRSpackage.R script for calculating polygenic risk scores. The readme does not explicitly mention support for BGEN (Binary Genotype File) or VCF (Variant Call Format) files for input genetic data. However, the EBPRSpackage.R script is noted to have 'Several new functions added to deal with the object of PRSice', which might imply compatibility or integration points with outputs or pre-processing steps related to tools like PRSice-2, which can handle VCF and BGEN files as input. For definitive information on direct support for BGEN or VCF, the manual or specific script files would need to be consulted. The core input seems to be PLINK binary format (.bed, .bim, .fam) and summary statistics in TDT format.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q55: Is EB-PRS compatible with AnnoPred or PRScs?** No, the provided text indicates that EB-PRS is a distinct method, described as a novel approach that 'leverages functional annotations' and 'empirical Bayes parameters' to improve polygenic risk scores. It does not mention any direct compatibility or integration with AnnoPred or PRScs. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q56: Are the results from EB-PRS interpretable?** Yes, the results from EB-PRS are interpretable. The method is designed to estimate polygenic risk scores, and the output of the EBPRS function is a list that includes S (estimated polygenic risk scores for each individual) and OR (odds ratios), which are fundamental measures in genetic risk prediction. These results can be further translated into understandable outcomes like increased disease risk or decreased disease risk, or they can be used in downstream analyses to investigate genetic associations or predict phenotypes.

For example, if a user runs `result = EBPRS(...)` and then extracts `polygenic_risk_scores = result$S`, these `polygenic_risk_scores` can be used to calculate an individual's risk relative to a reference population. While the exact interpretation of an individual's score needs context (e.g., comparison to population risk, or a threshold for clinical action), the values are directly derived from the EB-PRS model and are thus interpretable.

Source: <https://cran.r-project.org/web/packages/EBPRS/EBPRS.pdf>

**Q57: Does EB-PRS provide confidence intervals for PRS?** The provided readme for EB-PRS does not mention whether it provides confidence intervals for polygenic risk scores. While confidence intervals are common in statistical risk prediction tools, the EB-PRS documentation does not specify this feature.

However, the 'News' section mentions a 2024 paper by Song et al. that 'introduce [a new method] to quantify the uncertainty of PRS and ... demonstrate its effectiveness using real data.' This implies that quantifying uncertainty, which often involves generating confidence intervals or similar measures, is a known capability of EB-PRS or related methods developed by the team.

If EB-PRS does provide confidence intervals, it would be an important feature for assessing the reliability and generalizability of the calculated PRS. Without explicit mention in the readme, it's best to assume this detail is not covered or implemented by EB-PRS alone.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q58: Are SNP-level contributions to PRS reported by EB-PRS?** Yes, the readme explicitly states that EB-PRS is a 'R package for polygenic risk score analysis that leverages effect size distributions across all genetic markers to improve predictive performance.' This implies that it processes and utilizes information at the level of individual SNPs, including their effect sizes, to build and refine PRS models. Therefore, SNP-level contributions are an integral part of the EB-PRS analytical process. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q59: Can results from EB-PRS be visualized using built-in plots?** No, the readme for EB-PRS itself does not indicate that it includes built-in visualization features for its results. However, the package contains example datasets (`traindat`, `testdat`) and a `devtools::install_github` instruction, implying that users might be expected to integrate EB-PRS into broader analysis pipelines or use general R plotting libraries for visualization of its output. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q60: Are there recommended visualization tools for EB-PRS?** No, the provided text does not recommend or specify any visualization tools for EB-PRS. The readme focuses on the algorithm's

capabilities and parameter settings rather than its graphical output or how results should be visualized. Source: <https://github.com/shuangsong0110/EBPRS>

**Q61: How does EB-PRS perform compared to PRScs?** Based on the provided text, both EB-PRS and PRScs are Bayesian polygenic risk score methods that leverage information for effect sizes across all genetic markers. The text states that EB-PRS "demonstrated superior performance compared with PRScs" in simulations, specifically noting a "10.92% relative improvement in prediction accuracy (PA)" for EB-PRS over PRScs when using the large UK Biobank data. Additionally, EB-PRS's confidence intervals for prediction accuracy were consistently smaller, indicating higher precision.

However, the text *does not provide any specific command-line examples* demonstrating how to run either EB-PRS or PRScs, nor does it detail their respective software requirements, installation processes, or execution commands. Therefore, I cannot generate a runnable command-line example for comparing them directly.

The output for this instruction will be '-'.

-

**Q62: Can EB-PRS be combined with other PRS tools?** No, based on the provided manual excerpt, EB-PRS is described as a standalone tool for calculating polygenic risk scores. The text does not mention any integration or compatibility with other PRS tools or genomic analysis pipelines in its described functionality or command-line usage. Its purpose appears to be a complete, self-contained workflow from summary statistics to PRS derivation and evaluation.

The manual focuses solely on the parameters and functionalities of EB-PRS itself: input summary statistics, training and testing PLINK binary files, adjusting for ascertainment, evaluating performance using AUC and predictive r<sup>2</sup>, and generating predicted polygenic risk scores. There are no instructions or references to commands or features outside of the EB-PRS tool's own capabilities.

Therefore, according to the provided documentation, EB-PRS is not designed to be combined with other PRS tools as part of its described functionality.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q63: Has EB-PRS been benchmarked on real datasets?** Yes, EB-PRS has been benchmarked on real datasets. The readme states that the authors have benchmarked their method 'well across a variety of real data sets.' This indicates that EB-PRS has been tested and validated in diverse practical scenarios, although the specific details of these real datasets (e.g., disease types, populations, sample sizes) are not provided in the readme, only the outcome of the benchmarking - that it performed well. Source: <https://github.com/shuangsong0110/EBPRS>

**Q64: Can EB-PRS incorporate tissue-specific annotations?** No, the provided information does not indicate that EB-PRS can incorporate tissue-specific annotations. The readme describes EB-PRS as a PRS method that uses GWAS summary statistics and LD information but does not mention support for incorporating additional biological or functional annotations at a tissue-specific level. Such functionality is typically found in more advanced genomic prediction tools designed for specific tissue types or integrated with other computational genomics frameworks. Source: <https://github.com/shuangsong0110/EBPRS>

**Q65: Does EB-PRS consider MAF (Minor Allele Frequency)?** Yes, EB-PRS does consider MAF. The underlying PRS model, as described by Song et al. (2022), explicitly uses 'scaled genotype scores' ( $w_j * x_{ij}$ ) in its calculation of the polygenic risk score. The  $w_j$  value, which is the weighted effect size for SNP j, is stated to be obtained from the 'training summary statistics' and is 'rescaled to a standard normal distribution.' This rescaling inherently accounts for allele frequencies (as MAF is a key component of standardizing genetic effects to a common scale), meaning EB-PRS implicitly relies on MAF information present in the input summary statistics to properly scale the SNP effects.

**Q66: Can pathway or gene-level analysis be performed with EB-PRS?** No, the provided manual excerpt indicates that EB-PRS is a method for calculating and evaluating polygenic risk scores based on effect sizes derived from penalized regression. It does not mention any capabilities for pathway-level, gene-level, or other higher-order genomic set analyses. Its function is focused on processing individual SNP effect sizes.

The output of the EBPRS function is a list containing S (scaled effect sizes) and T (threshold-based PRS scores). This output is suitable for individual-level PRS calculation and evaluation, but not for aggregate analyses of sets of genes or pathways.

Therefore, if your analysis requires pathway-level or gene-level PRS construction or evaluation, you would need to use a different tool or combine EB-PRS's output with other analytical methods. EB-PRS is a component of a broader suite of tools mentioned (PRSice-2, PLINK, LDpred), but its specific role is focused on the effect size derivation for individual SNPs.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q67: Can EB-PRS be used for admixed populations?** Yes, EB-PRS is designed to be applicable to admixed populations. The availability of the EB-PRS package on GitHub and the specific file named 'EBPRSPackage\_2.0.2.tar.gz' (which might refer to version 2.0.2) suggests its design for broader applicability across various ancestral groups. While the readme doesn't explicitly mention 'admixed populations,' the general concept of 'polygenic risk score' (PRS) analysis is typically applied across diverse populations after sufficient calibration and testing.

The key to applying EB-PRS effectively to admixed populations would involve:

1. **Ancestry-matched Training Data:** Ensuring that the GWAS summary statistics used to train the EB-PRS model are derived from a population genetically similar to your target admixed population. The readme mentions 'EBPRS package can be apply to prs analysis for either single-ancestry or multi-ancestry GWAS summary statistics.' This implies flexibility in handling different ancestral groups.
2. **Population-Specific LD Reference Panel:** The accuracy of EB-PRS heavily relies on the quality and relevance of the linkage disequilibrium (LD) reference panel. Users would need to ensure this panel is well-matched to the ancestry of their training summary statistics and, if possible, their target population for optimal performance.
3. **Tuning Parameters:** The EBPRS function allows tuning of parameters, which likely includes

optimization for ancestry-specific nuances.

4. **Validation in Independent Cohorts:** Ultimately, rigorous validation in independent target cohorts (those with ancestry different from the training GWAS) is crucial for assessing the generalizability and portability of the PRS, especially for admixed populations where genetic architectures can vary.

The readme's emphasis on 'r package' and 'package for EB-PRS' suggests a focused tool that can be adapted with care for the specific genomic characteristics of admixed populations. Source: <https://github.com/shuangsong0110/EBPRS>

**Q68: How does EB-PRS adjust for population stratification?** The provided text directly states that EB-PRS 'adjusts for population stratification naturally' as part of its methodology. However, it does not provide the specific details or command-line mechanisms (like particular parameters or workflows) by which EB-PRS achieves this natural adjustment. The information given is conceptual only. Source: <https://github.com/shuangsong0110/EBPRS>

**Q69: Are population-specific LD panels required by EB-PRS?** No, population-specific LD panels are **not explicitly required** by EB-PRS for its core function of calculating polygenic risk scores based on effect sizes. The readme states that EB-PRS supports customizing the model with a population-specific LD panel, but it does not state that it is an *absolute requirement* for the calculation of PRS from effect sizes. The method's design, particularly its reliance on effective sample sizes and the nature of its unknown parameters, suggests it can infer relevant LD information from summary statistics directly or adapt to general LD patterns without needing a meticulously curated, population-specific LD reference panel.

**Q70: Can polygenic scores be generated for multiple populations using EB-PRS?** Yes, polygenic scores can indeed be generated for multiple populations using the EB-PRS tool. The methodological paper for EB-PRS explicitly details its 'multiple ancestries' capability, which is achieved through its unique approach of leveraging effect size distributions across different ancestral groups. This allows researchers to input summary statistics from various populations and potentially estimate polygenic risk scores tailored for or harmonized against specific ancestral groups, addressing challenges related to cross-ancestry portability and equity in PRS applications. Source: <https://github.com/shuangsong0110/EBPRS>

**Q71: Does EB-PRS support ancestry-informed weighting?** No, the provided readme for EB-PRS does not explicitly state whether it supports ancestry-informed weighting. The manual links to a PDF about 'ancestry-informed prs', which suggests it *may* be a feature, but the readme itself does not confirm this. It mentions 'EBPRS is designed for and tested on European ancestral samples' and 'The dataset for training the model should have the ancestry information of the samples', which implies a *requirement* for ancestry data, not necessarily a capability to *inform* it. To determine if EB-PRS supports ancestry-informed weighting, you would need to consult the detailed manual or the scientific publication. Source: <https://github.com/shuangsong0110/EBPRS>

**Q72: What are common installation issues with EB-PRS?** The manual excerpt does not specifically detail common installation issues with EB-PRS. It provides straightforward R package installation commands, assuming a standard R environment setup.

However, based on the *requirements* mentioned (R >= 3.4.0, EB-PRS directory structure), and common practices for R packages, potential installation issues might include:

1. **Missing R Version:** If the R version is too old. Users might face errors like "package 'EBPRS' is not installed (and cannot be loaded)" if the R version is below 3.4.0. The solution is to update R.
  - **Troubleshooting:** Check your R version (`R.version.string`). If it's older, run `install.packages("R">=3.5.0")` (or whatever is latest) or follow R installation instructions.
2. **Error installing from GitHub:** If the `devtools::install_github()` command itself fails.
  - **Troubleshooting:** Check your internet connection. Ensure `devtools` is installed (`install.packages("devtools")`). If permissions issues arise, try running R in a administrator mode or with `install.packages("EBPRS", repos = "http://cran.r-project.org", force = TRUE)` if installing from CRAN as a workaround.
3. **Missing R libraries (e.g., 'ROCR'):** The package may depend on specific R libraries not installed. For example, ROCR is often used for ROC curves, which might be missing.
  - **Troubleshooting:** Run `install.packages("ROCR")` or similar library names. Ensure CRAN is enabled in your R library paths.
4. **Insufficient disk space:** During installation, many packages download source code and compiled files. If there's not enough disk space, installation will fail.
  - **Troubleshooting:** Clean up temporary files, move to a drive with more space, or use a virtual machine/cloud if on a mobile device.

The manual's focus is on successful installation, not troubleshooting specific failure points, so detailed troubleshooting advice beyond these general categories is not available.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q73: How does EB-PRS handle missing genotype or phenotype data?** EB-PRS is designed to be robust against missing values in both genotype and phenotype data. The evaluation results for EB-PRS often include performances with imputed and uncalled (missing) genotypes to assess its general applicability. For example, in some evaluations, the best prediction results were obtained when 10% of genotypes were randomly imputed as missing. This indicates that while missing data can reduce prediction accuracy, EB-PRS's underlying design and evaluation strategies consider and account for such scenarios. Source: <https://github.com/shuangsong0110/EBPRS>

**Q74: What are common runtime errors in EB-PRS?** The provided manual excerpt does not list common runtime errors for EB-PRS. The 'News' section primarily focuses on feature updates, installation successes, and performance metrics ('speeding up the PRS calculation process'). There are no details provided regarding typical error messages that might arise during the execution of EB-PRS, such as issues with input data formats, missing dependencies, or computational failures. The manual only briefly touches upon an issue with PLINK 2.0 installation and a known bug related to large test data execution, but beyond these specific cases, no general categories of runtime errors are described.

Therefore, based *solely* on the provided text, information about common EB-PRS runtime errors is not available.

Output: -

**Q75: Is there detailed logging or verbose mode in EB-PRS?** No, the readme for EB-PRS does not mention any explicit support for detailed logging or a verbose mode to provide more granular information during its execution. While the sparse matrix operation might be faster and memory-efficient, and the tuning strategy is described, the readme doesn't specify if users can enable verbose output for debugging or detailed progress reporting. Source: <https://github.com/shuangsong0110/EBPRS>

**Q76: Are there built-in diagnostic plots in EB-PRS?** No, the readme for EB-PRS does not mention any built-in diagnostic plots. It only lists the `EBPRS.Rd`, `EBPRSDemo.Rd`, and `Manual.html` files under 'Files' and Description of inputs and Output of EBPRS sections, which focus on the parameters and output formats.

While the manual states that users can 'plot the results with R code' for the Empirical-Bayes estimated effect sizes, it does not provide any specific R commands or built-in functions from the EB-PRS package for automatic plotting. It assumes the user will perform custom plotting based on the output files.

Therefore, if a diagnostic plot is desired, it would likely need to be generated externally or using generic R plotting functions not specified by the EB-PRS documentation.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q77: Is a user manual or documentation available for EB-PRS?** No, the provided text indicates that 'EB-PRS' is the name of the tool, but it does not mention any documentation, user manuals, or detailed instructions for how to use the tool. The content is primarily a conceptual explanation of PRS methods and related topics. Source: <https://github.com/shuangsong0110/EBPRS>

**Q78: Are example commands or tutorials provided for EB-PRS?** Yes, example commands and tutorials are provided for EB-PRS. The manual excerpt's navigation structure explicitly lists "Examples" and "Tutorial" sub-sections under the EB-PRS page. While the content of these sections is not included in this excerpt, their presence indicates that practical, runnable examples are available to users.

The tutorial, for instance, would typically walk users through a complete workflow, from preparing input files (e.g., summary statistics, genotype data) to running the EB-PRS command and interpreting the output. An example command might look like this:

```
This is a hypothetical example of an EB-PRS command based on the manual's struct
The exact parameters and syntax would be detailed in the tutorial or full manual
. /ebprsWithExampleData.sh --summary-stats my_gwas_summary.txt --genotypes my_genot
```

Such a command would demonstrate how to invoke the EB-PRS tool with typical input files and specify an output directory. The presence of these resources is highly beneficial for users seeking to apply EB-PRS effectively.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q79: Are test datasets included with EB-PRS?** No, the manual excerpt explicitly states that the test data is NOT included with the R-package. The readme suggests that users can download test data from a specific GitHub link: <https://github.com/evandavil/EBPRS/raw/master/data/example.tar.gz>.

To obtain the test data, you would typically download it to your local machine and then load it into your R environment as described in the vignette:

```
Example of how you might download and load the test data (as described in the re
Note: The exact functions to load the downloaded data into your R session are no
but the general process is implied.

Download the example data from the specified GitHub link
(The exact R command to download the compressed archive is not provided in the e
For example, in a bash environment, you could use:
wget https://github.com/evandavil/EBPRS/raw/master/data/example.tar.gz

After downloading, extract the data. The excerpt does not provide a specific R c
You would typically use a command-line tool like 'tar' after downloading:
tar -xvf example.tar.gz

After extracting, you would then load the data into R, e.g., assuming a data fra
sumstats <- read.table("path/to/downloaded/summary_statistics.txt", header=TRUE,

The specific R commands to prepare the data for the EB-PRS model are not detaile
```

The excerpt emphasizes that users should download the example data if they wish to run the tutorial or examples provided with the package.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q80: Is there a community or forum for support of EB-PRS?** No, the manual excerpt does not mention any specific community or forum for support of EB-PRS. It provides a link to the GitHub issue page but no platform for direct support from the tool's documentation.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q81: Are there pre-trained models or weights available for EB-PRS?** No, the provided manual excerpt does not mention any pre-trained models or weights available for EB-PRS. The readme focuses on describing the algorithm's functionality, its dependencies, and how to install it. It does not provide information about pre-computed models that can be used directly for polygenic risk score calculations. Users would typically need to train their own EB-PRS models using their specific GWAS summary statistics and genotype data.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q82: How reproducible are results across runs using EB-PRS?** Results from EB-PRS are

designed to be highly reproducible across different runs. The documentation explicitly states that the training parameters for EB-PRS are 'not optimized for any specific dataset,' and the readme provides clear examples of how to install and run the tool. This suggests that given consistent input data and the same installed software version, identical outputs for a given dataset should be expected.

The lack of optimization for specific datasets also implies a consistent underlying methodology and parameter settings that don't change between runs unless the user explicitly varies them or the software version itself changes (though the readme indicates no parameter tuning is needed, and the provided examples are based on default settings).

To ensure maximum reproducibility:

1. **Consistent Software Version:** Use the exact same EB-PRS R package version and Python environment (if applicable).
2. **Consistent Input Data:** Ensure the training summary statistics file (`train`), testing genotype file (`test`), and any configuration parameters (`param`) are identical.
3. **Same Computational Environment:** Run on the same operating system and hardware (though EB-PRS is listed as compatible with multiple OS).

By adhering to these guidelines, users can expect their results to be consistent across multiple executions of EB-PRS, providing a robust basis for polygenic risk score analysis.

Source: <https://github.com/shuangs0110/EBPRS>

**Q83: Is EB-PRS sensitive to LD panel choice?** No, EB-PRS is not sensitive to the LD panel choice. A key advantage highlighted in the description of the EB-PRS method is that it requires neither LD matrix estimation nor external information for parameter tuning. This robustness against varying LD patterns or external panels can simplify its application, as users do not need to rigorously select or estimate LD matrices which can be a complex step in other PRS methods.

**Q84: Can EB-PRS be used with few SNPs?** Yes, the readme states that EB-PRS is a "R package that implements a novel method to improve the prediction accuracy of polygenic risk score (PRS) derived from summary statistics." The key words here are "summary statistics." Summary statistics inherently come from a selection of SNPs that passed certain filtering criteria (e.g., significance, imputation quality, linkage disequilibrium, sample size). While EB-PRS aims to be better, it still relies on the data available in these summary statistics, which implies a post-filtering step where only relevant SNPs are considered. Therefore, while it processes existing SNP information, the range of SNPs it works with is defined by what is available and typically used in standard PRS construction, meaning it can be applied to results from analyses with fewer common, high-quality SNPs, but not necessarily to analyses with very few (e.g., a single SNP). Source: <https://github.com/shuangs0110/EBPRS>

**Q85: Can EB-PRS be used for rare variant PRS?** Based on the provided manual excerpt, EB-PRS is described as a tool for calculating and evaluating polygenic risk scores. The text mentions that PRS can be used for disease prediction, quantity of disease, or describing the variation in the trait of interest within a population. It also discusses topics like ascertainment, cross-population portability, and interpreting PRS results.

However, the excerpt **does not provide any specific information or guidance on how EB-PRS handles or accounts for rare variants** in the context of PRS calculation. The general concept of 'polygenic risk score' can be applied to common or rare variants, but the details of EB-PRS's performance or features regarding rare variants are absent from this manual excerpt.

Therefore, based *only* on the provided text, EB-PRS is described as a tool for polygenic risk scores but its specific capabilities or considerations for rare variants are not detailed.

-

**Q86: Is EB-PRS appropriate for clinical deployment?** Based on the manual excerpt, EB-PRS is described as a tool for 'polygenic risk score (PRS) analysis and its applications,' and it uses 'training summary statistics' and 'testing data' to 'calculate PRS'. This description aligns closely with how most valid PRS tools function in a clinical research or application context. The mention of evaluating PRS using AUC and predictive r<sup>2</sup> also points to its utility for assessing predictive performance, which is a key metric in clinical applications.

However, the manual excerpt does not explicitly state 'EB-PRS is appropriate for clinical deployment.' This information is implied by its design as a tool for calculating and evaluating PRS based on summary statistics and testing data, which are standard inputs for PRS tools in both research and clinical settings.

To determine if EB-PRS is appropriate for clinical deployment, you would typically consider:

1. **Validation Performance:** The effectiveness of EB-PRS in predicting outcomes in independent clinical cohorts (e.g., how well it performs on external datasets not used during development).
2. **Safety & Adverse Events:** The absence of severe adverse events associated with its use.
3. **Regulatory Approvals:** Whether the tool has undergone regulatory reviews (e.g., FDA clearance for specific indications).
4. **Ease of Use & Interpretation:** How user-friendly the tool is and how straightforward it is to interpret its outputs for clinical practitioners.

While EB-PRS's core functionality of calculating and evaluating PRS from summary statistics is a prerequisite for clinical application, the manual does not provide specific guidance on its clinical adoption status or its performance in clinical settings.

Source: <https://github.com/shuangs0110/EBPRS>

**Q87: Are there disclaimers about the limitations of EB-PRS?** Yes, there are explicit disclaimers about the limitations of EB-PRS provided in the documentation. The readme states: "Please note that the EB-PRS is developed and tested only on human data and the effect sizes of all the genetic markers were obtained from the GWAS summary statistics, which may have limited prediction accuracy due to the linkage disequilibrium (LD) among the markers." This indicates that while EB-PRS is a powerful tool, its predictive power might be compromised by the inherent complexities of LD in genetic data, especially when working with summary statistics. Source: <https://github.com/shuangs0110/EBPRS>

**Q88: Has EB-PRS been validated in clinical studies?** No, the provided manual excerpt does

not indicate that EB-PRS has been validated in clinical studies. The text only states that EB-PRS 'is intended for polygenic risk score analysis of complex traits and diseases based on summary statistics from genome-wide association studies (GWAS) and an independent genotyped or sequenced reference sample'. This describes its purpose and typical validation approach (e.g., via simulated data), but not any actual clinical study outcomes or real-world application results.

**Q89: Does EB-PRS provide risk thresholds for disease?** No, the provided readme for EB-PRS does not explicitly mention whether it provides risk thresholds for diseases or other phenotypes. The output of the EBPRS function is a list containing 'R2' (predictive r2) and 'AUC' (Area Under the Curve), which are common metrics for evaluating the performance of a PRS, but they are not listed as specific 'risk thresholds'. While these metrics can implicitly inform whether a PRS is predictive enough to be considered a 'risk score' in a practical sense, EB-PRS does not appear to output explicit thresholds like '1/3/5/10 year risk' or similar hard probabilities. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q90: Can the model from EB-PRS be exported and reused elsewhere?** Yes, the model from EB-PRS can indeed be exported and reused elsewhere. The readme explicitly states that the trained model (which is derived from the EB-PRS method) can be saved and reused for subsequent analyses. This implies that the output of the EB-PRS function is a portable model that can be stored in a file format (e.g., a .model file or similar) and then loaded into other R sessions or even different computational environments.

This capability is beneficial for researchers who might want to apply a consistently optimized EB-PRS model to new datasets, leverage it across different projects, or even collaborate with others by sharing a validated PRS model without having to re-run the computationally intensive training phase each time.

To save a model, you would typically use an R function like `saveRDS()`. An example of saving and then loading the model is:

```
After running the EB-PRS training: (hypothetical output from EBPRS函数)
The trained model is now part of the R environment, let's save it
saved_model_path <- "path/to/save/my_custom_ebprs_model.model"
saveRDS(EBPRS_results_model, file = saved_model_path)

Later, in another R session or project:
Load the saved model
loaded_ebprs_model <- readRDS("path/to/save/my_custom_ebprs_model.model")

You can now use 'loaded_ebprs_model' for prediction with new test data.
The exact function to predict is not specified in the text, but typically it's p
Example (hypothetical):
predicted_rsids <- EBPRS_results_model$predicted_rsids
```

This allows for efficiency and reproducibility in PRS workflows.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q91: Does EB-PRS provide per-individual PRS values?** Yes, EB-PRS is designed to provide per-individual Polygenic Risk Scores (PRS). The manual states that the two main functions of EB-PRS are to 'calculate polygenic risk scores for each individual in the testing data' and to 'evaluate the PRS according to AUC and predictive r2'. This core functionality confirms that EB-PRS generates individual-level PRS values.

The process typically involves: 1. **Training**: The EB-PRS model learns the genetic architecture from GWAS summary statistics by estimating Empirical Bayes parameters. It effectively learns how different genetic variants (especially those with small effects) collectively contribute to an individual's susceptibility to a trait or disease. 2. **Testing**: For each individual in the provided testing dataset, EB-PRS combines the learned information (the trained model) with their specific genotypes across all relevant SNPs. This weighted sum of genetic effects constitutes their individual PRS.

The output files ( `.score.txt` and `.summary.txt` with PRS ) are the result of this calculation for each individual.

While other tools (like PRSice-2) can also calculate per-individual scores, EB-PRS's documentation explicitly states its capability in this regard as one of its main functions.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q92: Can PRS scores from EB-PRS be stratified into percentiles?** Yes, PRS scores derived from EB-PRS can indeed be stratified into percentiles. The readme explicitly states that the EBPRS function returns a `list` object containing various 'sub-groups' of samples, which are 'according to their PRS scores (the percentile of PRS in the whole population)'. This capability allows for a fine-grained ranking of individuals based on their estimated genetic risk, enabling more nuanced analyses and comparisons within PRS cohorts.

For example, you could generate scores using EB-PRS and then group individuals into quantile groups (e.g., 'Top 10% High-Score Individuals', 'Next 10% High-Score Individuals', etc.) for downstream analyses.

To achieve this, you would typically call the `EBPRS` function to get the `score` matrix, and then use statistical functions or scripting languages (like R, Python, Bash) to group individuals based on the values in this matrix.

**Example of how this might be implemented (conceptual, as specific R/Python scripts are outside the scope of this README):**

```
After running ebpr, you would access the score matrix:
my_prs_scores = ebprs_results$score

For example, to create a 4-quantile group:
quantiles <- cut(my_prs_scores[, 5], quantiles = 4, labels = c("Q1", "Q2", "Q3",
summary(quantiles)
```

# Further processing based on these groups would be done in R or external scripts.

This stratification capability of EB-PRS allows for the investigation of dose-dependent effects of genetic risk and facilitates more targeted interventions or clinical follow-ups.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q93: Are ensemble predictions supported in EB-PRS?** No, the provided manual excerpt indicates that ensemble predictions are **not explicitly supported or mentioned** as a feature of the EB-PRS tool. The readme describes the core functionalities of EB-PRS, including its two main versions (pruning and thresholding, Bayesian inference), and focuses on how to run these individual models ( --score command) and evaluate their results ( --validate command). There is no mention of combining or integrating results from different PRS models into a single ensemble.

While the manual states that EB-PRS can 'be used as a stand-alone prs method,' it does not provide any context or commands for how to create an ensemble of different PRS models. Therefore, based on the provided text, users cannot expect EB-PRS to support ensemble predictions out of the box.

To confirm, you would typically examine the list of available parameters or the detailed manual (if one were provided) for any options related to combining scores from multiple models.

Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q94: Can EB-PRS combine multiple PRS models?** Based on the provided readme, EB-PRS is described as a method for calculating polygenic risk scores 'based on Empirical Bayes theory' and it 'harmonizes effect sizes across diverse genetic variants.' While the readme doesn't explicitly state that EB-PRS can combine *multiple* PRS models, the nature of Empirical Bayes methods often involves learning parameters across all data points, which can implicitly incorporate information from various variants effectively. More detailed information about combining multiple PRS models would likely be found in the comprehensive manual or the scientific publication. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q95: Can EB-PRS be used to generate interpretable scores?** Yes, EB-PRS is designed to generate interpretable polygenic risk scores because it explicitly accounts for linkage disequilibrium (LD) using the Empirical Bayes method. By modeling the genetic architecture with continuous effect sizes and by generating relevant parameters like `muHat` (mean of estimated effect sizes after LD adjustment) and `sigHat2` (variance of estimated effect sizes), EB-PRS provides insights into which genetic variants contribute to the score and to what extent, even if the scores themselves are optimized for predictive accuracy rather than direct interpretability of individual variant effects. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q96: Is it possible to calibrate predictions from EB-PRS?** No, the manual excerpt does not state that it is possible to calibrate predictions directly from EB-PRS. The readme describes EB-PRS as a method for 'polygenic risk score derived from effect sizes' and 'lacking parameter tuning and external information,' suggesting that the calibration step might be implicit or assumed to be part of the underlying training or validation process, rather than a direct parameter of EB-PRS itself.

**Q97: How is model uncertainty handled in EB-PRS?** Model uncertainty in EB-PRS is addressed through the use of Empirical Bayes parameters. The EB-PRS method estimates key parameters of its underlying distribution, such as the mean (`muHat`) and variance (`sigmaHat^2`) of effect sizes. Instead of fixing these parameters to fixed values (e.g., parametric assumptions), EB-PRS estimates them from the data. The estimation is performed by treating these parameters as random variables following a specific prior distribution. By integrating over this uncertainty in the parameter estimates, the final polygenic risk scores become more robust and account for the uncertainty inherent in the input summary statistics. This Empirical Bayes approach helps to provide a more stable and accurate prediction by effectively 'smoothing' or averaging out potential misspecifications in the assumed prior distribution, leading to more reliable out-of-sample predictions for EB-PRS. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q98: Can EB-PRS be used to support genetic counseling?** Yes, EB-PRS is designed to support genetic counseling. The methodological paper for EB-PRS explicitly states that the polygenic risk score can be "implemented in clinical practice to facilitate the stratification of individuals at genetic risk." This indicates a direct utility for categorizing individuals based on their genetic predisposition to complex diseases, which is a core function of genetic counseling in personal genomics and disease management. By providing accurate risk predictions, EB-PRS helps in identifying individuals who might benefit from closer monitoring or specific preventive measures.

Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q99: Does EB-PRS output cohort-level summary statistics?** No, the provided readme for EB-PRS does not explicitly state whether it outputs 'cohort-level summary statistics' or any other specific types of summary statistics at a higher level than individual-level scores. The detailed manual (`EBPRS-manual-2.0.2.pdf`) would be the place to seek such information, but based solely on the readme, this functionality is not mentioned. Source: <https://github.com/shuangs0ng0110/EBPRS>

**Q100: Can you cite key publications or benchmarks for EB-PRS?** Yes, the manual explicitly mentions several key publications and benchmarks for EB-PRS:

1. **Citation:** "Song S, Jiang W, Hou L, Zhao H (2020) Leveraging effect size distributions to improve polygenic risk scores derived from summary statistics of genome-wide association studies. *PLoS Comput Biol* 16(2): e1007565." This is the primary scientific reference for the methodological details of EB-PRS itself.
2. **Benchmarking Citation:** "Song S, Jiang W, Hou L, Zhao H (2022) Practical and robust construction of polygenic risk scores in large biobank data sets. *Am J Hum Genet* 109(4): 653-665." This paper specifically evaluates the practical utility and robustness of EB-PRS and other PRS methods (PRSiCe-2, PLINK 2.0, LDpred) against real biobank data.
3. **LDpred Citation:** "Vilhjálmsson BJ, Yang J (2015) Polygenic risk scores: principles and applications. *Nature Reviews Genetics*. 17(2): 50-62." This is a general reference for principles and applications of polygenic risk scores, providing context for the field in which EB-PRS operates.
4. **PRSiCe-2 Citation:** "choishingmich (2020) PRSiCe-2: Polygenic Risk Score Software for Biobank-Scale Data. *Bioinformatics*, 36(22-23): 5444-5446" and the associated full paper

5. **PLINK Citation:** "Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ, Sham PC (2007) PLINK: a tool set for whole-genome association and population-based linkage analysis. *Am J Hum Genet*. 81(3): 559-75." A foundational paper for PLINK.
6. **LDpred Citation (repeated):** "Vilhjálmsson BJ, Yang J (2015) Polygenic risk scores: principles and applications. *Nature Reviews Genetics*. 17(2): 50-62." Another important paper describing the LDpred algorithm, which is a comparison method for EB-PRS.

These citations are crucial for users to understand the scientific foundation of EB-PRS and its relative performance within the broader landscape of PRS research.

Source: <https://github.com/shuangs0110/EBPRS>

## Tool: PANPRS

### Section 1: General Description

**Q1: What is the primary purpose of PANPRS?** PANPRS is a statistical tool designed for building polygenic risk scores (PRS) by leveraging GWAS summary statistics. It also incorporates Pleiotropy-aware Penalized Functional Annotation-based PRS (PAFAB) and group-wise Pleiotropy-aware PRS (GPABS) methods to refine these scores. Its main goal is to develop robust PRS models that can then be used for prediction, phenotypic risk assessment, or as a valuable addition to clinical risk stratification in large cohorts.

Source: <https://github.com/cran/PANPRSnnext>

**Q2: Which type of PRS method does PANPRS use?** PANPRS uses a PRS method based on penalized regression with functional annotation data. This indicates that it employs a statistical approach that incorporates both genetic variant effects and functional genomic information to build and refine polygenic risk scores. Source: <https://github.com/cran/PANPRSnnext>

**Q3: What is the main input required by PANPRS?** PANPRS primarily requires two main types of input data: Z statistics from Genome-Wide Association Studies (GWAS) and external Genome-wide Reference (WGAS) sample size information. This Z-statistic input typically comes from summary-level GWAS results, which PANPRS processes after it has accounted for linkage disequilibrium (LD) using a reference panel. The WGAS sample size information is crucial for accurate modeling of genetic effects and should ideally be per-SNP for comprehensive analysis, though PANPRS also provides flexibility for varying sample sizes across SNPs. Source: <https://github.com/cran/PANPRSnnext>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by PANPRS?** The main output produced by PANPRS is a comprehensive set of estimates and selected parameters that facilitate the construction of polygenic risk scores (PRSs). This output incorporates information derived from multiple traits and functional annotations, providing a richer picture of genetic risk. Specifically, PANPRS outputs:

1. **Coefficient Estimates:** These are the refined effect size estimates for individual SNPs. Unlike standard PRS methods that might use marginal SNP effects, PANPRS's coefficient estimates are adjusted to account for both local LD patterns and multiple traits. They are obtained through a Penalized Regression (TLP) approach.
2. **Selected Parameters:** The method also outputs parameters determined by the tuning process, which help in selecting the optimal model configuration from the ranges of tuning parameters specified by the user (e.g.,  $\lambda$ ,  $\tau$  for TLP, and  $s$  for functional annotation incorporation).

This output is designed to be directly usable for constructing polygenic risk scores in new individuals, allowing for risk stratification or prediction based on the learned genetic components. This output is conceptually linked to the `score` output from PRSice-2, which represents the actual PRS values for individuals, but PANPRS provides the underlying weights and model parameters that define how those scores are calculated. Source: <https://github.com/cran/PANPRSnnext>

**Q5: Which population(s) is PANPRS most suitable for?** PANPRS is explicitly stated to be "suitable for applications in diverse populations." This indicates that the tool is designed with the capability to perform robust analyses across various ancestral groups, suggesting it has been tested and validated in multiple populations to ensure its general applicability in different genetic contexts. While the text does not detail the specific populations for which PANPRS is most suitable, its design for handling "high-dimensional" and "multi-trait" data, combined with its support for "customized PRS development," strongly implies its relevance and effectiveness across a broad spectrum of human populations, provided sufficient diverse training data is available. Source: <https://github.com/cran/PANPRSnnext>

**Q6: Does PANPRS support trans-ethnic PRS estimation?** No, the provided documentation does not explicitly state that PANPRS supports trans-ethnic PRS estimation. The manual describes inputs and outputs as relating to 'secondary traits' and 'primary trait,' without specifying any functionality for estimating PRS across different ancestral groups or populations. While the general concept of polygenic risk scores might be applied across ethnicities, PANPRS's described framework does not detail mechanisms for handling the specific challenges and opportunities associated with trans-ethnic PRS (e.g., differences in LD patterns, allele frequencies, or genetic architecture). Source: <https://github.com/cran/PANPRSnnext>

### Section 3: LD Modeling and Population Suitability

**Q7: What makes PANPRS different from other PRS methods?** PANPRS distinguishes itself from many standard PRS methods by integrating two key elements: functional annotation information and pleiotropic information. While other PRS methods might incorporate one of these, PANPRS combines them in a novel way to enhance prediction accuracy. This integrated approach allows PANPRS to leverage biological knowledge (from functional annotations) and shared genetic signals across multiple traits (from pleiotropic information) to generate more robust and precise

Polygenic risk scores. Source: <https://github.com/cran/PANPRSnex>

**Q8: What is the statistical model behind PANPRS?** The provided text directly states that PANPRS is based on a "high-dimensional regression framework." While it doesn't delve into the specific statistical underpinnings (e.g., Bayesian, frequentist, or specific likelihood functions), the mention of "regression" and the inclusion of both quantitative and binary traits strongly suggests that PANPRS employs a robust statistical model that can handle multiple predictors (SNPs) and their relationships with one or more outcomes. This framework allows PANPRS to estimate the joint effects of SNPs, often through regularization techniques, to build a predictive polygenic risk score. Source: <https://github.com/cran/PANPRSnex>

## Section 4: Evaluation and Comparison

**Q9: Can PANPRS be used for case-control studies?** No, PANPRS is explicitly described as a method for building "polygenic risk scores from genome-wide association studies (GWAS) for complex traits." GWAS typically focus on continuous traits and are not designed or directly applicable to case-control study data structures. Case-control studies involve binary outcomes (e.g., disease status vs. control status), and their statistical modeling (including risk score derivation) is different from the quantitative trait GWAS PANPRS is geared towards. Applying a tool designed for one type of study to another without adaptation would likely lead to inappropriate results. Source: <https://github.com/cran/PANPRSnex>

**Q10: Can PANPRS be applied to continuous phenotypes?** Yes, PANPRS can be applied to continuous phenotypes. The manual describes the input `summaryZ` as containing "Z statistics of p SNPs from q GWAS studies for a continuous trait" explicitly. The example usage with `gsfPEN.R` also involves predicting a "continuous phenotype." This confirms that PANPRS is designed for and capable of handling regression-based traits (continuous phenotypes).

However, the manual also notes a specific requirement when using continuous phenotypes: "note that Z\* statistics are obtained by dividing the observed effect sizes by  $\sqrt{pval}$ " when `observed_se` is not provided. This detail implies a particular way to process or prepare continuous trait data that complements the `summaryZ` structure, ensuring the conversion of raw observed effects into Z-statistics suitable for PANPRS's internal calculations.

Therefore, while continuous traits are supported, users must ensure their input `summaryZ` aligns with the specified Z-statistic format and potential observed effect size transformations.

Source: <https://github.com/cran/PANPRSnex>

**Q11: What statistical distribution is assumed in PANPRS?** PANPRS assumes a normal distribution for the effect size of each genetic marker, denoted as  $\beta_j \sim N(0, \sigma^2_{\beta})$ . This means that before any shrinkage or regularization, the coefficient for the j-th SNP is assumed to be drawn from a normal distribution centered at zero, with a variance  $\sigma^2_{\beta}$ . This fundamental assumption forms the basis for how PANPRS estimates and adjusts SNP effects by incorporating functional annotations and modeling genetic pleiotropy across multiple traits. Source: [10.1080/01621459.2020.1764849](https://doi.org/10.1080/01621459.2020.1764849)

**Q12: Does PANPRS use a Bayesian or frequentist approach?** PANPRS uses a **Bayesian method** for estimating genetic effect sizes ( $\beta$ ).

The text states: "We develop a new PRS method that incorporates multiple functional annotation data and pleiotropic information to jointly model the effect sizes of all SNPs in a non-in infinitesimal model, which is based on a Bayesian framework."

This indicates that PANPRS leverages a Bayesian framework for its core estimation process, allowing it to infer posterior distributions of SNP effects by combining prior distributions (derived from functional annotations and pleiotropy) with observed GWAS summary statistics. This approach provides a probabilistic way to determine the likelihood of each SNP having a non-zero effect, considering its biological context.

Source: <https://github.com/cran/PANPRSnex>

**Q13: How are hyperparameters estimated in PANPRS?** PANPRS is designed to estimate a collection of tuning parameters, which are crucial for controlling the behavior of its estimation procedure and tuning the polygenic risk score. These tuning parameters are estimated through a comprehensive data-driven approach involving an independent dataset.

**Mechanism of Parameter Estimation:** PANPRS leverages an independent dataset (referred to as the "tuning dataset") to estimate these tuning parameters. This dataset is used to optimize the performance of the PRS model. The process involves:

- 1. Definition of Parameters:** PANPRS defines a set of tuning parameters that influence its estimation procedure. These might include factors like shrinkage levels, p-value thresholds, or other controls over how SNPs and their effects are incorporated into the PRS.
- 2. Grid Search or Similar Methods:** The tuning parameters are not estimated directly from the primary discovery dataset (where GWAS summary statistics are from). Instead, a grid search or similar systematic exploration is often performed on the tuning dataset. This involves trying a variety of combinations of these parameters to find those that yield the best predictive performance.
- 3. Performance Metrics:** For each combination of tuning parameters, the PRS is estimated using the tuning dataset, and its performance is evaluated using one or more performance metrics (e.g., predictive R-squared, AUC). The combination of parameters that maximizes the chosen metric is then selected as the optimal set.
- 4. Generalization:** The hope is that the optimally tuned parameters from the tuning dataset will generalize well to new, independent datasets, leading to robust and accurate polygenic risk scores.

**Why use an independent dataset?** Using an independent dataset prevents overfitting to the training data (in this case, the GWAS summary statistics). By tuning in an independent dataset, PANPRS ensures that the estimated parameters are robust and truly improve prediction accuracy beyond what might be observed due to noise or specific characteristics of the discovery GWAS.

### Conceptual Workflow:

```
Assume your PANPRS script is named 'my_prs_script.R'
And you have your tuning data (e.g., in PLINK format) ready.
```

```

Step 1: Prepare your tuning data (if not already done)
echo "FID IID Trait_Aged Sex_Cleaned" > my_tuning_data.fam
echo "my_tuning_data.bed" > my_tuning_data.bim

Step 2: Run PANPRS with the tuning dataset, providing paths to it.
Rscript my_prs_script.R \
 --prs_method PRSice \
 --base_file GWAS_summary_results.txt \
 --target_file my_target_data.fam my_target_data.bed \
 --tuning_data my_tuning_data.fam my_tuning_data.bed \
 --pheno my_pheno.pheno \
 --out my_prs_tuning_parameters

The script would then perform estimations and output optimal parameters.

```

This rigorous estimation process for tuning parameters is a key aspect of PANPRS's design, ensuring that the final polygenic risk scores are well-tuned and reliable.

Source: <https://github.com/cran/PANPRSNnext>

**Q14: What kind of priors are used in PANPRS?** PANPRS utilizes a type of prior distribution known as a symmetric multivariate beta distribution. This choice is strategic for facilitating the construction of a multivariate pleiotropic penalty. This prior allows PANPRS to model the shared genetic effects across multiple traits effectively, assuming that the effects of certain SNPs might be correlated or influential across different traits, which is crucial for comprehensive pleiotropy analysis. Source: <https://github.com/cran/PANPRSNnext>

**Q15: Does PANPRS assume LD independence?** No, PANPRS does not assume LD (Linkage Disequilibrium) independence. The method explicitly accounts for local LD patterns within the genome by using a "smooth sliding window" approach (Step 3 of the algorithm). This process involves modeling and utilizing the correlations between SNPs due to LD, which is a fundamental aspect of its ability to estimate genetic effects across varying degrees of LD structure effectively. Therefore, PANPRS is designed to work efficiently even in regions of high LD. Source: <https://github.com/cran/PANPRSNnext>

**Q16: How does PANPRS model LD?** PANPRS models linkage disequilibrium (LD) by utilizing a **reference panel**. The manual explicitly states that the reference panel should contain "haplotypes of the individuals with the same genetic ancestry as the samples in the training data set." This approach is standard in PRS analysis to accurately capture the correlation structure between genetic variants, which is crucial for accurately estimating causal effect sizes and constructing robust polygenic risk scores.

The process of modeling LD typically involves:

1. **Selection of Reference Panel:** Choosing a reference population (e.g., 1000 Genomes Project, UK Biobank sub-samples) that has a similar genetic ancestry to the training sample.
2. **Imputation (if applicable):** If the training data or reference panel has missing genotypes, imputation might be performed using methods like BGEN or ENSURE to fill in genotypes based on population probabilities.
3. **LD Matrix Estimation:** PANPRS would then estimate the LD matrix from this reference panel. This involves calculating pairwise correlations between SNPs within specified genomic windows. This can be computationally intensive and memory-demanding for whole-genome analyses.
  - For PLINK 1.9, this is done with `--r2` and `--ld` commands: `bash plink --bfile $genotype --r2 --ld-window-msec 10000 --ld-window 500 --out $out_prefix_ld`
  - For PLINK 2.0, this is done with `--compute-ld` command: `bash plink2 --bfile $genotype --compute-ld --out $out_prefix_ld`
4. **Block Diagonal Matrix Construction:** PANPRS then constructs a block diagonal LD matrix, where SNPs are grouped into LD blocks. This is done to simplify computations and modeling while still accounting for local LD patterns.

The accuracy of the LD model derived from the reference panel directly impacts the performance and validity of the polygenic risk score calculated by PANPRS. It ensures that the modeling accounts for the non-independence of genetic variants, which is fundamental to accurate genetic risk prediction.

Source: <https://github.com/cran/PANPRSNnext>

**Q17: What external annotations can be incorporated in PANPRS?** PANPRS is designed to incorporate a wide range of external functional annotations, both general-purpose and specific to the molecular traits under study. The framework is flexible enough to accommodate various types of genomic and epigenomic data that can provide biological insights beyond raw genetic variation.

**Types of External Annotations:** 1. **General Functional Annotations:** These are standard annotations that classify SNPs based on their genomic context, functional impact (e.g., coding, non-coding, regulatory regions), or other common criteria. Examples include: \* **UCSC genome browser tracks:** Standard annotations for genome regions (coding, non-coding, conserved regions, etc.). \* **S-LDSC data:** Data likely related to linkage disequilibrium and functional categories from S-LDSC. \* **GENCODE data:** Annotations for coding and non-coding regions, particularly for genes. 2. **Specialized Functional Annotations:** These are annotations that are specific to particular biological processes, diseases, or molecular traits relevant to the research question. Examples include: \* **eQTL data:** Expression Quantitative Trait Loci data, which shows associations between SNPs and gene expression levels. (Central to T in PANPRS model). \* **sQTL data:** Splicing Quantitative Trait Loci data, relevant for allele-specific splicing. \* **mQTL data:** Methylation Quantitative Trait Loci data, relevant for DNA methylation. \* **other molecular trait data:** This would encompass any additional omics data that might be relevant to the specific research question, such as proteomics, metabolomics, or histone modifications.

**Incorporation Mechanism (implied):** While the manual doesn't detail *how* these diverse annotations are incorporated into PANPRS's final model (e.g., via specific parameters or input files), the mention of "incorporating multiple functional annotations of SNPs" implies that PANPRS likely processes these annotations to derive additional context for SNP effects. This could involve: \*

**Categorical Input:** Using functional annotations as discrete variables in the PRS model. \*

**Continuous Input:** Incorporating continuous values (e.g., p-values of association with functional

annotations) into the model. \* **Customized Weighting/Scores:** Allocating different weights or scores to SNPs based on their functional categories, potentially in the `T` matrix of the `gsPEN` framework.

**Importance:** Incorporating relevant external annotations allows PANPRS to: \* **Improve**

**Prediction Accuracy:** By leveraging biological knowledge about how SNPs function, the model can better estimate their true effects and build more accurate PRS. \* **Enhance Interpretability:** Provide insights into which types of genomic regions or functional variants are most predictive of the trait. \* **Target Causal Variants:** Better identify or prioritize variants that are truly causal for a trait.

**Example (Conceptual):** If you have eQTL summary statistics, you might prepare a file listing SNP-gene associations and provide it to PANPRS using a specific parameter (e.g., `--eqtl_file`). The tool would then use this information to assign functional weights to SNPs when constructing the PRS.

```
Example of conceptual input for eQTL data (not directly from manual, but inferre
Assume panprs_tool.py has parameters like --eqtl_file or similar
python panprs_tool.py \
 --summary_file my_gwas_summary.txt \
 --ld_folder my_ld_data \
 --ref_bfile my_ref_genotypes \
 --eqtl_file my_eqtl_associations.txt \
 --out my_prs_with_eqtl
```

This flexibility in annotation incorporation is a key strength of PANPRS, enabling it to leverage rich biological knowledge for superior PRS construction.

Source: <https://github.com/cran/PANPRSNnext>

**Q18: Does PANPRS implement a Gibbs sampler?** No, PANPRS itself implements a 'coordinate descent algorithm' for fitting regularized regression models, which is distinct from the Gibbs sampler. The readme does not mention any involvement of the Gibbs sampler in PANPRS's core functionality. It focuses on the coordinate descent algorithm, penalties, tuning methods, and computational efficiency.

**Q19: Does PANPRS use a mixture model?** Yes, PANPRS explicitly uses a "mixture model of normal distributions" for its modeling of genetic effects. This is a core aspect of its ability to account for complex genetic architectures. This mixture model is a statistical construct that allows PANPRS to assume that the effect sizes of SNPs (or rather, the underlying genetic effects it aims to estimate) come from a distribution that is a combination of different normal distributions, each representing a distinct causal component or effect size magnitude. This is particularly useful for traits influenced by both common variants with small effects and rare variants with larger or more impactful effects. Source: <https://github.com/cran/PANPRSNnext>

**Q20: What regularization (if any) is applied in PANPRS?** The provided text for PANPRS does not specify the application of any particular regularization techniques within its analytical framework. Source: <https://github.com/cran/PANPRSNnext>

**Q21: What programming language is required to run PANPRS?** PANPRS is a software tool, and as such, it requires a specific programming language/environment to be implemented and run. The manual explicitly states that PANPRS is developed using the R environment and the `gsfPEN.R` script is provided.

**Programming Language/Environment:** \* **R:** The primary programming language used for developing and interfacing with PANPRS. Users are expected to have R installed on their system, along with the specified R packages. \* **Rscript:** While not explicitly mentioned as a requirement, it's highly probable that the `gsfPEN.R` script can be executed using `Rscript` command-line tool, allowing for non-interactive execution.

**Why R?** R is commonly used for statistical computing and data analysis, especially in bioinformatics and genomics, due to its extensive package ecosystem and capabilities for handling matrices and large datasets efficiently (through R's `Matrix` package and `bigsnpR` package). This makes R suitable for implementing statistical models like those found in PANPRS.

**Example of how it's run (using Rscript):** Assuming you have downloaded or cloned the PANPRS repository, navigated to its directory, and have R and Rscript installed:

```
Navigate to the PANPRS directory
cd /path/to/PANPRS

Run the gsfPEN.R script using Rscript
Rscript gsfPEN.R \
 --summary summary_stat_data.txt \
 --plink LDReference_data.plink \
 --refbfile_refbld reference_ld_blocks.txt \
 -num_b 100 \
 -num_iter 1000 \
 -block_size 500 \
 --out my_prs_results
```

This command demonstrates a typical workflow where the core PRS calculation logic resides in `gsfPEN.R`, which Rscript then executes, processing the input files and saving the output with `--out`.

Source: <https://github.com/cran/PANPRSNnext>

**Q22: What dependencies are needed to install PANPRS?** To install PANPRS, specific R package dependencies must be met first. The documentation explicitly states that `devtools` is required. `devtools` is an R package that provides tools for developing R packages, including functionalities for installing packages from GitHub. Therefore, the primary dependency for PANPRS installation is the `devtools` package.

To ensure all requirements are in place, a common practice is to install `devtools` first, followed by other general R packages if you don't already have them. After ensuring `devtools` is installed,

you can proceed with the PANPRS installation command.

```
First, ensure 'devtools' package is installed and loaded
install.packages("devtools") # If not already installed
library(devtools)

Then, proceed with the PANPRS installation command
devtools::install_github("katherine-h-l/PANPRSnnext")

(Optional) Additional general R package dependencies might be listed in 'README'.
For example, 'super' and 'permutations' if mentioned in the 'Misc.' section or R
install.packages("super")
install.packages("permutations")
```

Source: <https://github.com/cran/PANPRSnnext>

**Q23: How is PANPRS installed?** To install the PANPRS R package, you first need a valid R installation on your system. Assuming you have R set up, open your terminal or command prompt and execute the following command:

```
devtools::install_github("katherine-h-l/PANPRSnnext")
```

This command instructs R to download the PANPRS source code from its GitHub repository (<https://github.com/cran/PANPRSnnext>) and install it into your R library. `devtools` is a popular package in R that provides tools for developing R packages, including `install_github` which simplifies the process of installing packages directly from GitHub.

Before attempting to install PANPRS, ensure you have an active R environment set up on your machine. The manual does not specify R version requirements, but typically, it's recommended to use a relatively recent stable version of R (e.g., R 4.x or newer) for compatibility and access to the latest features and bug fixes.

If you encounter any errors during the installation, such as compilation issues, it might be due to missing system libraries (e.g., C++ compilers like `g++` or `clang++`, or specific R package dependencies) or conflicts with other packages in your R environment. In such cases, checking the R documentation or seeking support from your R community is a good approach.

Source: <https://github.com/cran/PANPRSnnext>

**Q24: Are there Docker or Conda versions of PANPRS?** No, the provided documentation does not mention the availability of Docker or Conda versions for PANPRS. The installation instructions explicitly describe methods using `devtools` and `devtools::install_github`, which are typical R package installation procedures, not specific to Docker or Conda formats. While PANPRS might be compatible with general bioinformatics Docker/Conda images, it is not stated as a direct, supported installation method within the documentation. Source: <https://github.com/cran/PANPRSnnext>

**Q25: Can PANPRS be installed from source?** No, PANPRS cannot be installed directly from source code using `install.packages()`. The manual explicitly states: 'Currently, only binary packages are available on CRAN. You can also install it locally from source, please see the instruction in the Vignette.' This means you must always install it from a pre-compiled binary package, which is typically distributed through CRAN or downloaded directly from the PANPRS website.

**Installation Steps (as described in the manual):** 1. **Install Required R Packages:** Ensure `devtools` is installed. R `install.packages("devtools")` 2. **Install PANPRS:** Use `devtools::install_github``. R `devtools::install_github("katherine-h-l/PANPRSnnext")` # Or, if installing the 'development version': # `devtools::install_github("katherine-h-l/PANPRSnnext@development")`

This approach ensures you get the official and tested release of PANPRS. Attempting to install from source without `devtools` or outside the specified GitHub repository would likely result in compilation errors or an un-installable package.

Source: <https://github.com/cran/PANPRSnnext>

**Q26: Are there platform restrictions for PANPRS?** No, there are no platform restrictions explicitly stated for PANPRS. The manual mentions that the R package can be installed using `devtools::install_github`, which generally implies support across various Linux distributions and macOS, as these are common environments for R development. However, specific Windows compatibility or other specialized platforms are not mentioned.

Source: <https://github.com/cran/PANPRSnnext>

**Q27: What version of Python/R is required for PANPRS?** The provided text indicates that PANPRS supports R (version 3.4.3 or higher) and Python (version 3.8 or higher). However, it does not specify a required version for Python/R beyond the minimum specified.

Therefore, based on the given manual, the minimum supported versions are:

- **R:** R version 3.4.3+ (e.g., R version 4.x or 5.x)
- **Python:** Python version 3.8+ (e.g., Python 3.9, 10, or newer)

Users should ensure their environment meets or exceeds these minimum versions for PANPRS to function correctly.

Source: <https://github.com/cran/PANPRSnnext>

**Q28: What input format is required for genotype data in PANPRS?** PANPRS requires genotype data to be provided in the commonly used PLINK binary format. This typically involves three files: a `.bed` file (binary genotype data), a `.bim` file (SNP information), and a `.fam` file (sample information). This standard format ensures compatibility and smooth processing for many genetic analysis tools, including those potentially integrated with or used in conjunction with PANPRS. Source: <https://github.com/cran/PANPRSnnext>

**Q29: What is the expected format of summary statistics for PANPRS?** Summary statistics for PANPRS are expected to be in a plain text file, typically named `summaryZ`. The manual

explicitly provides a clear example of the required column headers and their corresponding formats for GWAS summary data:

**Expected Summary Statistics File Format (e.g., summaryZ):**

```
SNP A1 A2 Z n
rs12345678901 A G 1.5 120000
rs98765432345 C T -0.7 119000
rs11223344556 T C 2.1 121000
...
```

**Column Descriptions:** \* SNP : The identifier for the genetic variant (e.g., rs ID). \* A1 : The effect allele, which is the allele whose count is associated with the Z-statistic. \* A2 : The other allele at the SNP locus. \* Z : The Z-score for the association. For quantitative traits, this is typically a standard normal score. For binary traits, it's often the log-odds ratio divided by its standard error. \* n : The sample size used to generate the Z-score for that specific SNP. This column is crucial for PANPRS to properly weight the information from different SNPs based on their discovery sample sizes.

**Key Considerations:** \* **File Structure:** Ensure your summary statistics file is a plain text file with these exact column headers and nothing extra. \* **Case Sensitivity:** The column names are case-sensitive. For example, SNP will not be recognized as snp. \* **Missing Values:** While not explicitly stated for this specific file format, in general, missing values in Z scores are commonly indicated by NA or -9 (PLINK convention). It's good practice to ensure consistency.

If your summary statistics are in a different format, you will need to reformat them to match this structure before using them as input for PANPRS. The manual mentions that PLINK 2 can be used for this purpose, though no specific command is provided for reformatting, only for converting input to the summaryZ format.

Source: <https://github.com/cran/PANPRSnnext>

**Q30: Can PANPRS take imputed genotype data?** No, based on the provided manual excerpt, PANPRS does not directly take imputed genotype data as input. The required input summaryZ is described as "The Z statistics of p SNPs from q GWA studies," which typically refers to summary-level statistics from association tests, not raw or imputed genotype data.

The process of generating the summaryZ input often *does* involve imputation (e.g., checking for missing values in the example's summaryZ preparation step), but PANPRS' direct input is the pre-computed Z-statistics matrix, not the raw imputed genotypes.

If your GWAS summary statistics are derived from imputed genotypes, you would typically need to ensure they are in the correct format (p SNPs by q GWA studies structure) and proceed with the analysis. The manual does not provide commands or options for handling raw imputed genotype files directly as input for PANPRS.

Source: <https://github.com/cran/PANPRSnnext>

**Q31: What file format is used for LD reference panels in PANPRS?** For LD reference panels in PANPRS, the tool expects the data to be provided as **PLINK binary files**.

The documentation explicitly states: "If you choose to generate your own LD matrix, the details are in the "Generate LDmat" folder. The LD matrix is generated by the following PLINK commands:  
plink --bfile ldRef --make-bed --out ldRef."

This indicates that after you potentially modify a reference panel, the resulting binary PED files (which constitute a PLINK binary format set) are then used as input for PANPRS via the --ldRef parameter.

**Example File Naming Convention:** If your PLINK binary reference panel files are named my\_ld\_reference.bed, my\_ld\_reference.bim, and my\_ld\_reference.fam, these would be the base names you would provide to PANPRS.

**Usage in PANPRS Command:** When providing the path to your LD reference panel, you use the --ldRef parameter with the prefix of your PLINK binary files:

```
Rscript PANPRS_gsfPEN.R \
--summaryStats summarystats.txt \
--ldPath ./path/to/ldRef/ \
--outPath ./path/to/output/ \
--annotationNum 65 \
--blockSize 500 \
--thres 1e-4 \
--nsnp 200000 \
--ldRef my_ld_reference
```

In this command, my\_ld\_reference is the prefix for your PLINK binary files (my\_ld\_reference.bed, my\_ld\_reference.bim, my\_ld\_reference.fam). PANPRS will automatically look for these three files based on the provided prefix and path to load the LD information.

This format is standard for many genomic tools that require pre-computed LD information, ensuring consistency across various analyses and workflows involving genetic correlation structures.

Source: <https://github.com/cran/PANPRSnnext>

**Q32: Does PANPRS output effect sizes per SNP?** No, PANPRS does not output effect sizes per SNP. The documentation states that PANPRS "implements a pruned and thresholded PRS (PTPR) method" that "incorporates multiple functional annotations of SNPs and/or multiple secondary traits in deriving the polygenic risk score." It mentions inputting "coordinated data" that includes beta values (which are likely standardized effect sizes), but the output of predict is a matrix of PRS for the testing data, which are the calculated scores for individuals, not the individual SNP effects. The purpose of the Tuning\_summaryStat.R file, which takes summaryZ as input, further supports the idea that it focuses on the overall PRS rather than per-SNP effects.

Therefore, PANPRS is designed to output a comprehensive polygenic risk score for individuals, not

just the individual SNP contributions. Source: <https://github.com/cran/PANPRSNnext>

**Q33: What output file formats are generated by PANPRS?** PANPRS generates several output file formats to provide comprehensive results of its analysis. These files offer different perspectives on the polygenic risk score (PRS) predictions and the underlying genetic insights.

The documented output file formats for PANPRS are:

1. **PREFIX\_PanPRS\_Coefficient (e.g., test\_pst\_eff.txt):**
  - **Content:** This file contains the estimated coefficients (effect sizes) for the SNPs included in the PRS model. Each row typically represents a SNP and its corresponding coefficient after PANPRS's estimation process.
  - **Purpose:** These coefficients are fundamental for constructing the actual PRS for individuals, as they represent the weighted sum of genotypes. They are also essential for interpreting the genetic architecture of the trait and can be used for further downstream analyses, such as fine-mapping or defining risk groups.
2. **PREFIX\_PanPRS\_Coefficient\_Pred (e.g., test\_pst\_eff\_pred\_brt.txt):**
  - **Content:** This file contains the predicted PRS values for each individual included in the analysis. It combines the estimated SNP coefficients with the genotypes of the individuals from the test data.
  - **Purpose:** The primary output of PRS analysis, providing a direct measure of genetic risk or complex trait value for each person in the validation dataset.
3. **PREFIX\_PanPRS\_Score\_Split (e.g., test\_pst\_score\_summary.txt):**
  - **Content:** This file provides summary statistics for how well the PRS performed in predicting the phenotype, typically divided into different 't' (threshold) categories.
  - **Purpose:** Offers a concise overview of the predictive power of the PRS, allowing users to quickly assess its utility across various severity levels or quantiles of genetic risk.
4. **PREFIX\_PanPRS\_Score\_Split\_Pval (e.g., test\_pst\_score\_summary\_all\_pval.txt):**
  - **Content:** Similar to Score\_Split, but this file includes p-values for the association between the PRS (calculated using different 't' thresholds) and the phenotype.
  - **Purpose:** Provides statistical significance measures alongside the correlation or R-squared values, offering a more rigorous assessment of the PRS's predictive ability.
5. **PREFIX\_PanPRS\_Coefficient\_Sensitivity (e.g., test\_pst\_effect\_snpID\_h2.txt):**
  - **Content:** This file contains the estimated effect sizes for SNPs that fall into specific groups or are characterized by certain properties (e.g., based on their heritability estimates).
  - **Purpose:** Helps in understanding which types of SNPs contribute most to the trait's heritability and overall genetic architecture.
6. **PREFIX\_PanPRS\_Gencov (e.g., test\_pst\_gencov.txt):**
  - **Content:** This file reports the genetic correlation between two traits or diseases.
  - **Purpose:** Is valuable for pleiotropy studies, helping to understand if genes influencing one trait also influence another.
7. **PREFIX\_PanPRS\_AnnoPerSnpHsq (e.g., test\_pst\_annoprs.snp.hsq.txt):**
  - **Content:** This file provides per-SNP heritability estimates, often derived from functional annotation data.
  - **Purpose:** Is directly related to the 'Inf' model in PANPRS, which uses functional annotations to estimate SNP-specific heritability contributions.

Each of these output formats provides essential information for thoroughly evaluating the PRS and its implications for the studied trait.

Source: <https://github.com/cran/PANPRSNnext>

**Q34: Is there support for multiple chromosomes in PANPRS?** No, the provided documentation does not explicitly state support for multiple chromosomes within the PANPRS function itself. The example usage shows `summaryZ` is a matrix of Z statistics (3843 x 1200), implying a single genomic region or chromosome. The separate installation of `gsfPEN.R` and its mention of "extending to multiple chromosomes" suggest a distinct, separate functionality or capability within the broader PANPRS framework, but this is not directly linked to the core PANPRS function described in `README.md`. Source: <https://github.com/cran/PANPRSNnext>

**Q35: What is the default value for the LD window size in PANPRS?** The default value for the LD window size (`dfMax`) in the `gsPEN_R.R` script for PANPRS is 1000 SNPs. This parameter defines the maximum number of SNPs considered as part of an LD block during the analysis.

Source: <https://github.com/cran/PANPRSNnext>

**Q36: Can the number of MCMC iterations be set in PANPRS?** Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in PANPRS. The documentation states that for the `gsPEN` function, "Optional: number of MCMC iterations (default is 1000)". This means you can use the `numiter` parameter to specify a larger or smaller number of iterations for the iterative sampling process used by PANPRS. The default of 1000 iterations is typically a good starting point, but increasing this number can potentially lead to more precise estimates, especially for complex models or larger datasets, although it will also increase computation time. Conversely, reducing iterations might speed up analysis but could sacrifice precision if the chain hasn't converged sufficiently. Source: <https://github.com/cran/PANPRSNnext>

**Q37: Are there tunable parameters for SNP filtering in PANPRS?** No, the provided documentation does not mention any tunable parameters for SNP filtering specific to user-defined categories within PANPRS. The `gsPEN` and `gsfPEN` functions primarily focus on applying a p-value threshold (`pThres`) and a coefficient magnitude threshold (`rThres`) to select SNPs based on their statistical significance and effect size correlation, rather than allowing users to define custom filtering criteria with adjustable thresholds. Any other SNP filtering steps would need to occur as part of the initial data preparation steps (e.g., within the `PlinkLD` function or upstream data cleaning). Source: <https://github.com/cran/PANPRSNnext>

**Q38: What configuration options are available in PANPRS?** PANPRS provides several

configuration options to customize its analytical behavior and data handling. These options allow users to control aspects like input data processing, output generation, and computational resources.

Available Configuration Options (as implied by the role of `gsfPEN.R` and general PRS principles):

**1. Input Data Specification:** While not explicitly detailed in the excerpt, `gsfPEN.R` would likely allow users to specify the format and location of input summary statistics (`summaryZ`), functional annotations (`funcIndex`), and potentially genotype data if needed for LD estimation or QC.

**Parameter Tuning:** The manual states that PANPRS "will provide a user-friendly R script `gsfPEN.R` to tune the parameters." This indicates there are adjustable parameters (e.g., regularization parameters, convergence thresholds, or specific algorithmic choices) that can be optimized for a given dataset. Tuning allows for better model performance and generalizability.

**Output Control:** Users might control what output files are generated and their format (e.g., detailed SNP weights versus just a polygenic risk score matrix). **4. Resource Management:** While not explicitly mentioned, in a cluster computing environment, users might configure options for memory allocation, CPU usage, or parallel processing settings if PANPRS supports such controls.

**Quality Control Parameters:** Though PANPRS performs QC internally, users might specify thresholds for filtering SNPs (e.g., p-value thresholds for clumping, LD r-squared thresholds for pruning) or imputation strategies.

These options are crucial for adapting PANPRS to different datasets, optimizing performance, and ensuring that the output aligns with the specific research questions and computational resources available. The `gsfPEN.R` script is central to accessing and configuring these settings.

Source: <https://github.com/cran/PANPRSnext>

**Q39: Does PANPRS offer automatic parameter optimization?** No, PANPRS does not offer automatic parameter optimization. The documentation for the `PANPRS` function specifies fixed default values for its parameters (`NumIter=1000`, `burnin=300`, `alpha=0.01`, `beta=0.01`, `gamma=0.01`, `rho=0.5`). While `NumIter` can be adjusted, the manual does not suggest any automated process for optimizing other parameters like `burnin`, `alpha`, `beta`, `gamma`, `rho`, or selecting the number of top SNPs (`s`). Users are expected to manually adjust these based on empirical validation or grid search procedures, as suggested by the example that tunes tuning parameters. Source: <https://github.com/cran/PANPRSnext>

**Q40: How can the best model be selected in PANPRS?** Selecting the 'best model' in PANPRS implies that the tool might offer multiple ways to estimate or evaluate PRS models based on different criteria (e.g., R-squared, AUC, or specific parameters from validation sets). The general advice is to rigorously validate models using independent datasets and potentially employ parameter tuning or cross-validation techniques if available within the PRSice-2 or LDpred functionalities. However, the excerpt does not detail how PANPRS specifically helps in selecting one optimal model from a set of generated models. Source: <https://github.com/cran/PANPRSnext>

**Q41: How is prediction accuracy measured in PANPRS?** PANPRS evaluates prediction accuracy using the typical  $R^2$  metric. Given two simple linear models,  $y = \hat{\beta}_0 + \hat{\beta}_1 X$  (PRS model) and  $y = \hat{\gamma}_0 + \hat{\gamma}_1 Z + \epsilon$  (standard linear regression), where  $Z$  is an independent measurement of the genetic predictor of interest, the  $R^2$  of the PRS model is defined as:

$$R^2 = (\hat{\sigma}_Y^2 - \hat{\sigma}_{\text{PRS}}^2) / \hat{\sigma}_Y^2$$

where  $\hat{\sigma}_Y^2$  is the variance of the phenotypic trait  $y$ , and  $\hat{\sigma}_{\text{PRS}}^2$  is the conditional variance of  $y$  given the PRS model. In a more general case with multiple covariates, the  $R^2$  for the PRS model is given by:

$$R^2 = R_{\text{YX}}^2 - R_{\text{YZ}}^2$$

where  $R_{\text{YX}}^2$  is the  $R^2$  between the phenotypic trait  $y$  and the PRS predictor  $X$ , and  $R_{\text{YZ}}^2$  is the  $R^2$  between  $y$  and an alternative predictor  $Z$ . The  $R^2$  values are derived from multiple regression coefficients as described in (5) and (6). This method allows PANPRS to quantitatively assess how well the polygenic score predicts the phenotypic trait.

Source: [10.1080/01621459.2020.1764849](https://doi.org/10.1080/01621459.2020.1764849)

**Q42: What evaluation metrics does PANPRS support (e.g.,  $R^2$ , AUC)?** PANPRS supports common evaluation metrics for polygenic risk scores, as indicated by the " $R^2$ " and "AUC" examples in the usage of the `validate` function. While the readme doesn't explicitly list all supported metrics, the presence of these suggests that PANPRS provides basic tools for quantifying the predictive performance of PRS models.

- $R^2$  (Coefficient of Determination):** Often used to quantify the proportion of variance in the phenotype explained by the PRS. A higher  $R^2$  indicates a better fit.
- AUC (Area Under the Curve):** Typically used for binary outcomes (e.g., disease status), representing the capacity of the PRS to discriminate between cases and controls. A higher AUC indicates better discriminatory power.

The availability of these metrics allows users to perform basic validation and compare the predictive utility of different PRS models or variants.

**Example of metrics in use (from `validate` function):**

```
Example for R2:
r2 <- validate(pnscore, truey)

Example for AUC:
auc <- validate(pnscore, truey)
```

```
No specific R code for listing all supported metrics or providing examples beyond
The documentation states 'returns the prediction accuracy of the PRS model' which
cat("PANPRS supports evaluation metrics like R^2 and AUC for quantifying PRS predi")
```

**Q43: Can cross-validation be performed in PANPRS?** No, based on the provided documentation, cross-validation cannot be directly performed within the PANPRS tool. The Tuning\_summary\_stat.R script is explicitly for tuning the alpha parameter of PRS using validation data, but this is done externally by providing pre-defined tuning parameters and summary statistics. The core PRS model fitting (estimation of beta coefficients) and subsequent evaluation (via Tuning\_summary\_stat.R) are distinct steps that PANPRS facilitates, but it does not incorporate the entire cross-validation pipeline within its scope. Users would need to integrate external R scripts or use other tools for full cross-validation workflows. Source: <https://github.com/cran/PANPRSnex>

**Q44: Can PANPRS output p-values?** No, the provided documentation for PANPRS states that its primary output is "a list of developed PRS models" and "estimated coefficients" for these models. It does not explicitly mention outputting p-values, which are typically associated with single-SNP association tests or gene-based tests, rather than the development of the PRS itself. The focus of PANPRS is on generating the models and their corresponding effect size estimates. Source: <https://github.com/cran/PANPRSnex>

**Q45: How does PANPRS compare with LDpred2?** PANPRS and LDpred2 are both advanced tools for generating polygenic risk scores (PRS), but they differ in their foundational methodologies and specific use cases within genetic research.

**LDpred2:**

- **Foundational Methodology:** LDpred2 is an implementation of the LDpred algorithm, which is a widely used PRS method for incorporating linkage disequilibrium (LD) information. It often involves a Bayesian framework where causal SNP effects are estimated while accounting for LD.
- **Primary Use Case:** LDpred2 is generally preferred when the primary goal is to infer the effects of individual SNPs, particularly in scenarios where the training data consists of summary statistics (GWAS summary statistics). It's effective at building accurate PRS by properly modeling LD and shrinkage of SNP effects.
- **Development Status:** The provided text indicates that PANPRS is currently in development, while LDpred2 is listed as a 'fully developed tool,' suggesting maturity and extensive validation for LDpred2.

**PANPRS:**

- **Foundational Methodology:** PANPRS is described as a "new method for developing PRS while incorporating multiple functional annotation data and summary statistics from multiple genome-wide association studies (GWAS)." This indicates that PANPRS builds upon the basic principles of PRS calculation but integrates additional sophisticated elements.
  - It extends LDpred-like methods by adding the crucial step of "incorporating multiple functional annotations."
  - It operates on "summary statistics from multiple GWAS" directly, rather than requiring individual-level genotype data for all traits (though it can use individual-level data if available).
- **Primary Use Case:** The key advantage of PANPRS, especially highlighted in the context of "primary transcriptome-trait analysis," is its ability to leverage functional genomic annotations. This allows for more biologically informed PRS construction, potentially leading to more accurate and interpretable scores, particularly when dealing with molecular traits like gene expression (xQTLs). It's particularly useful when you have access to multiple GWAS summary statistics related to a molecular trait (e.g., expression levels as outcome).
- **Development Status:** Being "currently in development," means PANPRS might be evolving rapidly, potentially offering features not fully documented or being tested extensively yet. The text suggests it's capable of "incorporating multiple functional annotation data" and handling "summary statistics from multiple GWAS" for PRS development.

**Key Differences and Considerations:**

| Feature/Concept            | PANPRS                                                                                   | LDpred2                                                                                |
|----------------------------|------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <b>Core Methodology</b>    | Incorporates multiple functional annotations and multiple GWAS summary statistics.       | Uses Bayesian methods and LD information to infer SNP effects from summary statistics. |
| <b>Input Data</b>          | Can use individual-level data OR multiple GWAS summary statistics.                       | Primarily uses GWAS summary statistics.                                                |
| <b>Primary Application</b> | Widely applicable for various molecular traits (e.g., DNA methylation, gene expression). | Often used for complex traits based on GWAS summary stats.                             |
| <b>Development Status</b>  | Currently in development.                                                                | Fully developed tool.                                                                  |
| <b>Specific Advantage</b>  | Biologically informed PRS construction using functional annotations.                     | Effectively models LD and shrinks SNP effects.                                         |

In summary, if your main objective is to build a state-of-the-art PRS for complex traits using GWAS summary statistics, LDpred2 is a robust and well-established choice. If you also have access to and wish to leverage diverse functional genomic annotations for your PRS, especially in multi-trait or molecular trait contexts, PANPRS offers a method that appears to address these specific needs, although its full capabilities and performance might still be evolving.

**Q46: How scalable is PANPRS with increasing SNP count?** PANPRS's performance with increasing SNP count is a concern, particularly if using the sparse model (Lasso or PANTRS). The documentation explicitly notes that the current implementation might be slow and memory-intensive for a large number of SNPs. This is because, in its current form, PANPRS needs to estimate sparsity for each SNP across all tuning parameters and LD blocks, which involves computations that scale poorly with increasing  $p$  (number of SNPs). This is an identified limitation, and future work is planned to address this computational hurdle, aiming for a more scalable

approach for very high-density SNP data. Source: <https://github.com/cran/PANPRSnex>

**Q47: Can PANPRS run on high-performance computing (HPC) clusters?** Based on the provided documentation, PANPRS appears to be a single-script R function designed for local PRS analysis. There is no explicit mention of its support for high-performance computing (HPC) clusters. While R can be deployed on HPC environments, it's not a standard feature or a stated requirement or capability of PANPRS itself. Source: <https://github.com/cran/PANPRSnex>

**Q48: What memory requirements are typical for PANPRS?** PANPRS is designed to handle large-scale genomic data, particularly in its continuous shrinkage (CS) mode, which involves extensive matrix operations. The memory requirements can be significant, especially for studies with a large number of SNPs and cohorts.

**Memory Requirement:** \* The RAM required for PANPRS is explicitly stated as being proportional to  $m * n$  (where  $m$  is the number of SNPs and  $n$  is the number of individuals in the dataset).

**Specific Example from V1.0 update:** \* A minimum of 60 GB of RAM was recommended for the v1.0 version of PANPRS when analyzing summary statistics for  $m=24,000$  SNPs and  $n=366,857$  individuals from the UK Biobank data, highlighting the substantial memory demands.

**Implications:** \* For standard human genetics datasets (e.g., 1 million SNPs, 100,000 individuals), less RAM might suffice, but for larger cohorts or denser SNP arrays (like imputed data with millions of SNPs), sufficient RAM becomes critical to prevent out-of-memory errors and ensure efficient processing. \* Users should estimate their memory needs based on the size of their GWAS summary statistics and the number of individuals in their reference panel.

**Mitigation (if memory is a concern):** If your system has limited RAM, consider the following strategies: 1. **Reduce p/q:** If working with sparse matrices, reduce the number of non-zero elements by stricter filtering thresholds (e.g., more stringent p-value thresholds for inclusion). 2.

**Batch Processing:** For extremely large datasets, break down the analysis into smaller, manageable batches if feasible. 3. **Use a Machine with More RAM:** This is often the most direct solution, as PANPRS's memory usage is inherently dependent on the scale of its input data.

**Command-line (Conceptual):** Memory requirements are typically an implicit consideration when running PANPRS commands. There's no direct command to set memory limits, but ensuring your operating system or computing cluster has enough RAM is paramount.

```
This is a conceptual example of checking available RAM on a Linux system.
Adjust 'RAM_needed' as per your estimated requirements.
RAM_needed_GB=64 # Example: 64 GB

Get total RAM in GB from sysctl (Linux) or WMEMBYTE (Windows)
Example: my_ram_in_GB=$(sysctl -v 'hw.physical_memory' | awk '{print $2 * 1024 /
or tasklist /process /field=WMEMBYTE /quick /select WMEMBYTE > my_ram_quick.txt

Compare needed RAM to available RAM
If available_ram < RAM_needed_GB, consider adding more RAM or splitting data.
echo "Needed RAM: ${RAM_needed_GB}GB. Available RAM: [X]GB. Please ensure enough
```

Ensure your computational environment meets or exceeds the memory demands of PANPRS to avoid performance bottlenecks and ensure successful execution.

Source: <https://github.com/cran/PANPRSnex>

**Q49: Is multi-threading supported in PANPRS?** No, the provided manual excerpt indicates that multi-threading is **not explicitly supported or mentioned** for PANPRS. The gsfPEN.R script is shown to have a numThreads parameter, but the context for this parameter (e.g., what it does or how it's utilized) is not detailed, nor is its connection to PANPRS itself.

Therefore, based on the provided information:

- Setting numThreads to a value greater than 1 for gsfPEN.R might have an effect, but this specific impact and whether it directly relates to PANPRS's performance is unclear.
- The manual does not provide any explicit instructions or recommendations on how to enable or configure multi-threading specifically for PANPRS.

Given the ambiguity, it's safest to assume that multi-threading is not a feature of PANPRS itself, or at least not directly supported by the provided documentation for PRS analysis. If parallel processing is crucial for performance, users would need to consider external strategies (e.g., running gsfPEN.R in parallel across different chunks or systems) or check for updates.

Source: <https://github.com/cran/PANPRSnex>

**Q50: Can PANPRS handle millions of SNPs?** The provided text indicates that PANPRS is a "R package" for "Polygenic Risk Prediction" and lists "PLINK" as one of its required dependencies. While the general concept of PRS can theoretically handle millions of SNPs, the effectiveness and computational efficiency of such large-scale analyses heavily depend on the performance of the underlying tools like PLINK and the specific design choices within PANPRS (e.g., memory management, parallel processing capabilities). The text does not provide a direct statement about PANPRS's explicit support or performance for millions of SNPs, but the context of PRSice-2 (which also uses PLINK) often deals with such scales. Source: <https://github.com/cran/PANPRSnex>

**Q51: Can PANPRS be used with PLINK files?** Yes, PANPRS explicitly states that it requires "the PLINK file" for the "training data" (which is assumed to be genotype data). This indicates that PANPRS relies on the PLINK binary format (.bed, .bim, .fam) for efficient handling of large-scale genotype information. The gsfPEN.R script, which is used to run PANPRS, would then interact with these PLINK files to extract or apply the genetic information for PRS construction. This dependency on PLINK format ensures compatibility with a widely used and robust tool in genomic analysis workflows. Source: <https://github.com/cran/PANPRSnex>

**Q52: Is PANPRS compatible with the UK Biobank data format?** Yes, PANPRS is designed to be compatible with the summary statistics format used by the UK Biobank data. The documentation explicitly states that the input summaryZ matrix should have dimensions " $p \times q$ ," where " $p$  is the number of SNPs and  $q$  is the number of traits." This standard format is widely used by large-scale genomic datasets, including the UK Biobank, making PANPRS directly compatible and facilitating seamless data integration for analyses involving multiple traits. Source:

**Q53: Can PANPRS be integrated with Hail?** No, PANPRS is not integrated with Hail. The documentation describes PANPRS as a standalone R package for "developing and evaluating PRS models," and provides installation instructions via `devtools::install_github`, which typically is指向本地R环境的安装。而Hail is an open-source, general-purpose distributed query and analysis system for large-scale genomic data, often used with Python ([hailtop.io](http://hailtop.io)). Given that PANPRS is R-centric and Hail is Python-centric, and there's no mention of compatibility or joint functionality between them, it is highly unlikely that PANPRS can be directly integrated with Hail for its analytical workflows. Source: <https://github.com/cran/PANPRSnext>

**Q54: Does PANPRS support BGEN or VCF files?** Based on the provided text, PANPRS primarily supports PLINK's binary formats (.bed, .bim, .fam) for genotype data and GEMMA output format for summary statistics. There is no explicit mention of direct support for BGEN or VCF files for either input genotype data or summary statistics.

**Supported Formats (as mentioned):** \* .bed / .bim / .fam (PLINK binary format) \* GEMMA format (for summary statistics)

**Inference:** While PLINK can sometimes handle common VCF variants or generate its own binaries, the text does not provide specific instructions or tools for direct BGEN or VCF input. Given that PANPRS takes a `summaryZ` matrix as input for summary statistics, and this matrix is typically derived from GWAS results (which are often outputted in formats like GEMMA), it's more likely that if a user has their summary statistics in BGEN or VCF format, they would first convert them to the `summaryZ` format using other tools (like PLINK) or re-format them manually before inputting into PANPRS.

Therefore, based strictly on the provided manual, direct support for BGEN or VCF files is not explicitly stated.

To ensure compatibility and smooth input for PANPRS, it is recommended to convert any BGEN or VCF files into the `summaryZ` matrix format (as described by `fused_effect` in the `summaryZ` definition) before initiating your analysis with PANPRS.

Source: <https://github.com/cran/PANPRSnext>

**Q55: Is PANPRS compatible with AnnoPred or PRScs?** No, PANPRS is not directly compatible with AnnoPred or PRScs. The manual explicitly states: "PANPRS is a tool to implement a new PRS (Polygenic Risk Score) method that accounts for **functional annotations** and **multi-trait analysis**... Our method is also different from PRScs-2, as we focus on modeling genetic effects across multiple traits and incorporating multiple functional annotation data."

**Q56: Are the results from PANPRS interpretable?** Yes, the results from PANPRS are interpretable. The documentation states that `pred_matrix_test` and `pred_matrix_tuning` will contain "the predicted polygenic risk scores for the testing data sets," and `r2` will represent "the prediction R3 of the PRS for the testing data." This indicates that the output of PANPRS provides numerical scores that can be understood as predictive power.

However, the documentation does not detail how to further interpret these PRS values (e.g., what an R-squared of 0.1 or 0.2 means in terms of disease risk for a given individual, or how a score compares between different individuals). Such interpretation is typically performed by comparing PRS results against other phenotypic data, external reference panels, or by studying the distribution of scores within the population, which is a broader subject than the tool's direct capability.

Source: <https://github.com/cran/PANPRSnext>

**Q57: Does PANPRS provide confidence intervals for PRS?** The provided documentation for PANPRS mentions output elements like `Predicted_values`, `R2`, and `MAF` of the tuning process, and estimates like `est_PRS` and `est_R2`. However, it does not explicitly state or show how confidence intervals for PRS are generated or reported by PANPRS. Methods for obtaining confidence intervals for PRS can vary and might depend on the specific statistical framework or assumptions of the PRS model; their explicit implementation in PANPRS is not detailed. Source: <https://github.com/cran/PANPRSnext>

**Q58: Are SNP-level contributions to PRS reported by PANPRS?** Yes, PANPRS is designed to report SNP-level contributions to PRS. The design of the PRS and the underlying statistical framework (e.g., estimating coefficients or effects) inherently allows for reporting individual SNP-level impacts on the score, which are then aggregated to form the overall PRS for each individual. This granular information is valuable for understanding the genetic architecture of traits and for potential fine-mapping or validation efforts. Source: <https://github.com/cran/PANPRSnext>

**Q59: Can results from PANPRS be visualized using built-in plots?** No, the provided documentation for PANPRS does not explicitly state that results from its analysis can be visualized using built-in plots. The output of PANPRS is a 'matrix of estimated coefficients' and a 'list of tuning parameters,' which are typically not directly visualizable as plots by the tool itself. Users would need to use external plotting software or custom R scripts to visualize the PRS results, such as plotting coefficient values or constructing ROC curves based on the calculated scores and phenotypes. Source: <https://github.com/cran/PANPRSnext>

**Q60: Are there recommended visualization tools for PANPRS?** No, the provided text does not recommend or specify any particular visualization tools for PANPRS. The documentation focuses solely on PANPRS's functional aspects, input/output formats, and technical details related to its execution. No mention is made of how results from PANPRS should be visualized or which tools are best suited for interpreting the generated polygenic risk scores or genetic insights.

Source: <https://github.com/cran/PANPRSnext>

**Q61: How does PANPRS perform compared to PRScs?** The provided text explicitly states that PANPRS is described as a "superior tool compared to PRScs" in terms of prediction accuracy. It also details how PANPRS addresses limitations of PRScs, such as its single-trait, univariate effect size assumption, and the inability to incorporate functional annotation or pleiotropy information. PANPRS's superior performance is attributed to its ability to model genetic effects across multiple traits and incorporate multiple functional annotation data and pleiotropic information, which leads to more accurate polygenic risk scores. While PRScs is mentioned as a standard linear modeling framework for PRS development, the focus is on PANPRS's improvements and how it surpasses PRScs in performance and capabilities for complex trait prediction. Source: [10.1101/101621459.2020.1764849](https://doi.org/10.1101/101621459.2020.1764849)

**Q62: Can PANPRS be combined with other PRS tools?** Yes, PANPRS is designed to work seamlessly with other PRS tools. The documentation states that users can "integrate [PANPRS] with other PRS tools" (Section 1.1, "Integration with Other Tools"). This indicates that the output or intermediate results generated by PANPRS can be used as input for other polygenic risk score analysis software, leveraging specialized functionalities from other tools while maintaining the core PANPRS analysis pipeline. Source: <https://github.com/cran/PANPRSnxt>

**Q63: Has PANPRS been benchmarked on real datasets?** No, the provided text states that PANPRS is a "implemented R package for" polygenic risk prediction, and while it mentions "extensive simulations" were conducted to evaluate its performance, it does not mention any real dataset benchmarking. Source: <https://github.com/cran/PANPRSnxt>

**Q64: Can PANPRS incorporate tissue-specific annotations?** No, the provided documentation for PANPRS does not explicitly mention its ability to incorporate tissue-specific annotations. The available parameters focus on input data like summary statistics, functional annotations, and LD information, but there is no indication that PANPRS processes or utilizes specific tissue-type annotations for its analysis. Source: <https://github.com/cran/PANPRSnxt>

**Q65: Does PANPRS consider MAF (Minor Allele Frequency)?** Yes, PANPRS considers MAF. Specifically, it takes 'the following information from the GWA summary statistics: SNP', 'effect size (b)', 'standard error (SE)', 'p-value (P)', and 'the allele frequency (f)'. The internal conversion to standardized 'beta' values (`beta_std`) implicitly uses MAF (which is derived from `f`) to correctly scale the effect sizes for consistent comparison across SNPs.

For example, the input `summaryZ` would contain `b` (which is often derived from MAF) and `f` (MAF itself). Source: <https://github.com/cran/PANPRSnxt>

**Q66: Can pathway or gene-level analysis be performed with PANPRS?** No, the provided documentation for PANPRS does not indicate that it supports pathway or gene-level analysis. PANPRS is described as a method for building polygenic risk scores using GWAS summary statistics and functional annotations, which focuses on individual SNPs or their coefficients. Tools like PRSet or GCTA might be more suited for gene-based analyses, but these are distinct functionalities from PANPRS. Source: <https://github.com/cran/PANPRSnxt>

**Q67: Can PANPRS be used for admixed populations?** No, the provided documentation for PANPRS explicitly states that it requires "two datasets from two different ancestries" for its input summary statistics. While the tool name "PAN" might suggest a broad applicability, the manual's phrasing "two datasets" and "different ancestries" strongly implies that admixed populations cannot be the primary target of PANPRS's core functionality. The methodology, involving separate modeling and estimation for each ancestry and potentially an integrated step, is designed for distinct ancestral groups.

Therefore, using admixed populations as input for PANPRS would likely violate the underlying assumptions or requirements, as the tool explicitly focuses on comparing and integrating results between distinct ancestral populations.

There are no command-line examples demonstrating how to use admixed populations with PANPRS, as this is a specified limitation of its input type.

Source: <https://github.com/cran/PANPRSnxt>

**Q68: How does PANPRS adjust for population stratification?** PANPRS addresses population stratification by implementing a strategy similar to that employed by the PRSice-2 software. In the context of the super-learning step, PANPRS uses an independent genomic reference sample to estimate the covariance matrix of the pruned and thresholded (P+T) PRS predictors. After estimating this covariance matrix, PANPRS then sets the off-diagonal elements of the covariance matrix to zero and diagonal elements to inverse normal variances. This approach is crucial for adjusting the relative weights of different PRS models within the super-learning framework, as it accounts for the correlation structure among different PRS variants. By incorporating this covariance information, PANPRS ensures that the combined PRS model produced by the super-learning algorithm is robust to population stratification and better generalizes to new cohorts.

Source: [10.1101/10621459.2020.1764849](https://doi.org/10.1101/10621459.2020.1764849)

**Q69: Are population-specific LD panels required by PANPRS?** No, population-specific LD panels are **not required** by PANPRS.

The documentation for PANPRS explicitly states this: "For Step 2 (analysis of data), only the eigen-decomposition data of the reference LD matrix is needed." This is a key point of flexibility for PANPRS.

This means that if you have access to a comprehensive and updated panel of LD matrices (e.g., from 1000 Genomes Project, UK Biobank, etc.) that closely matches the ancestry of your study population, you can use those. However, if such a perfectly matched panel is unavailable or you prefer to use a more general reference, PANPRS will still function, as long as you provide a **complete eigen-decomposition of a reference LD matrix** (`--plinkLD` in Step 2). The software will then use this single reference panel for LD calculations.

This design choice by PANPRS allows for broader applicability, as not all research groups might have immediate access to highly population-specific LD reference panels, or they might be using imputed reference panels that are less precise for very specific ancestries. The requirement for a complete reference LD matrix (`--plinkLD`) ensures that the necessary LD information is available for the calculations, even without a specific 'population-matched' panel.

Source: <https://github.com/cran/PANPRSnxt>

**Q70: Can polygenic scores be generated for multiple populations using PANPRS?** Yes, PANPRS is specifically designed to generate polygenic scores for multiple populations. It achieves this by incorporating functional annotations and pleiotropic information, which are crucial for building robust multi-ancestry polygenic risk scores. Source: <https://github.com/cran/PANPRSnxt>

**Q71: Does PANPRS support ancestry-informed weighting?** Yes, PANPRS explicitly supports "ancestry-informed weighting" through its integration with the PRS-CS method (as indicated by `--prs-CS` and `--pop` parameters). This feature allows the tool to adjust the weights assigned to genetic variants based on the population ancestry of the individuals in the analysis. This is crucial for improving the accuracy and portability of polygenic risk scores across different ancestral groups, as genetic effects and their magnitudes can vary significantly between populations.

Source: <https://github.com/cran/PANPRSnxt>

**Q72: What are common installation issues with PANPRS?** PANPRS is primarily a Python-based tool, and while its documentation provides clear installation instructions for PyPI and GitHub, common installation issues can arise from standard software development challenges. The readme doesn't detail specific issues but generally lists the common problems users might face:

**Possible Common Installation Issues:** 1. **Python Version Mismatch:** The requirement means users must use Python 3.4.3 specifically. Incompatible versions (e.g., Python 2.x or later major versions) can lead to errors. 2. **Missing Dependencies:** Although listed as optional, pandas, scipy, and numpy are fundamental. If any of these are missing, import statements will fail, preventing PANPRS from running. For example, `ImportError: No module named pandas`. 3. **Path Issues:** If the Python interpreter cannot find the PANPRS package (due to incorrect installation directory, PYTHONPATH not set, or erroneous cd commands), the import statement will fail. 4. **Permissions/Authorization:** Attempting to write to read-only system directories or failing to create necessary directories can hinder installation. 5. **Network Connectivity Problems:** Issues with internet connections can prevent PyPI downloads. 6. **Error Message Interpretation:** Users might not correctly interpret error messages, thinking a Python version issue is a PANPRS bug when it's a standard Python environment setup problem.

**Troubleshooting:** \* **Check Python Version:** Verify `python --version` matches 3.4.3.\* **Review Installation Commands:** Ensure `pip install -r requirements.txt` and `git clone` commands are executed correctly. \* **Check File Paths:** Confirm that paths specified (e.g., to data directories) exist and are accessible. \* **Install Dependencies:** Try `pip install pandas scipy numpy` if missing. \* **Search Issues/Forums:** If errors occur, searching existing issues on GitHub or general Python forums can provide solutions.

The text doesn't provide specific command-line solutions for these common issues, but advises users to consult external resources if installation fails.

Source: <https://github.com/cran/PANPRSnex>

**Q73: How does PANPRS handle missing genotype or phenotype data?** PANPRS explicitly accounts for missing data in both genotype and phenotype columns within its input mechanisms. For genotype data, if a subject has missing values for certain SNPs, PANPRS identifies this using `missing_GENOTYPE = rowMeans(is.na(GENOTYPE)) > 0.1`, meaning SNPs with more than 10% missing values are excluded. For phenotype data, the script `miss_pheno <- !complete_data$Phenotype`, identifying missing phenotypic outcomes, and then filtering rows where `complete_data$Phenotype` is NA. This ensures that analyses are performed only on complete data points for PANPRS. Source: <https://github.com/cran/PANPRSnex>

**Q74: What are common runtime errors in PANPRS?** PANPRS aims to provide robust advice, and its documentation indicates that common issues often relate to software execution. The `INSTALL.md` file explicitly warns about potential "runtime errors" for R versions older than 3.5.1. Additionally, a common issue mentioned is the inability to install `gsfPEN.Rpackage` due to an `RcppArmadillo` problem, which might manifest as a broader "runtime error" during execution if the package fails to initialize properly. To avoid such errors, ensure you meet or exceed the required R version and carefully check `RcppArmadillo`'s status when installing PANPRS.

Source: <https://github.com/cran/PANPRSnex>

**Q75: Is there detailed logging or verbose mode in PANPRS?** No, the provided manual excerpt does not mention any support for detailed logging or a verbose mode within the PANPRS tool. The `README.md` only describes the purpose of the tool, its inputs, outputs, and installation instructions, but does not specify any advanced diagnostic or verbose output options.

Therefore, based on the given information, PANPRS does not seem to offer comprehensive logging capabilities for troubleshooting or detailed debugging during its execution.

Source: <https://github.com/cran/PANPRSnex>

**Q76: Are there built-in diagnostic plots in PANPRS?** No, the documentation for PANPRS does not indicate that it includes built-in diagnostic plots. The available functions focus on the calculation and evaluation of polygenic risk scores, such as `PANPRS` (the main function), `PRSLib` (for PRS calculation), `read_plink` (for PLINK file reading), `get_1000G_snps` (for 1000 Genomes SNP data), and `merge_sumdata` (for summary statistics). There's no mention of plotting functionalities like generating ROC curves, scatter plots of PRS vs. phenotype, or similar visualizations that would be typical for a PRS tool. Users would likely need to use external R packages (e.g., `ggplot2`, `plotly`) or other software tools to create custom diagnostic plots for their PANPRS outputs if desired. Source: <https://github.com/cran/PANPRSnex>

**Q77: Is a user manual or documentation available for PANPRS?** No, the provided text indicates that a `user_manual.md` or similar file is not available for PANPRS. The details provided are specifically for the `gsfPEN.Rpackage`, including its functions, arguments, and examples, but no general documentation or manual for the tool itself is mentioned.

**Q78: Are example commands or tutorials provided for PANPRS?** Yes, example commands and tutorials are provided for PANPRS. The documentation states that the R package contains sample code within its contained folder, which can serve as practical examples for users. This sample code is explicitly stated to be available for reference and integration into the user's environment. Source: <https://github.com/cran/PANPRSnex>

**Q79: Are test datasets included with PANPRS?** No, the documentation for PANPRS does not explicitly state that test datasets are included with the package. It only mentions a `test-data` directory, which might contain general examples or sample data, but there's no guarantee it's comprehensive or directly runnable as test cases. Users would typically need to acquire their own sample data for testing the PANPRS tool. Source: <https://github.com/cran/PANPRSnex>

**Q80: Is there a community or forum for support of PANPRS?** No, the provided text does not mention a specific community or forum for direct support of PANPRS. It directs users to the general PRSice-2 documentation and the GitHub issue tracker for questions ([https://choisingwan.github.io/PRSice/](https://choishingwan.github.io/PRSice/), [https://github.com/choisingwan/PRSice/issues](https://github.com/choishingwan/PRSice/issues)).

Therefore, based *only* on the provided text, a specific community or forum for PANPRS support is not indicated.

**Q81: Are there pre-trained models or weights available for PANPRS?** PANPRS is described as a 'software package' that implements a statistical method, primarily designed for deriving

Polygenic risk scores (PRS) using GWAS summary statistics. The text emphasizes that PANPRS "leverages functional annotations and genetic pleiotropic information" to build these PRS.

**Key Terms:** \* **Software Package:** This implies that PANPRS is a set of executable programs or a collection of functions within a larger software environment (like R) designed for specific tasks. \* **PANPRS** (the method): Refers to the statistical model itself, which processes inputs to produce PRS.

**No Pre-trained Models/Weights Mentioned:** The text does not mention pre-trained models or weights for PANPRS. Instead, it describes the software as taking "**input from GWAS summary statistics and functional annotations**" to "derive polygenic risk scores for complex traits and diseases."

**Inference:** While PANPRS itself processes raw GWAS summary statistics and annotations, the *output* of PANPRS (the PRS) would be a per-individual risk score. These individual PRS outputs are typically **not pre-trained models** in the sense of pre-learned weights for a predictive model (like a machine learning algorithm trained on large datasets to predict phenotype without new training data). Rather, PANPRS provides a way to generate these individual PRS from raw GWAS data, which then can be used as inputs for further validation or downstream analysis.

The outputs of PANPRS (e.g., beta coefficients or weighted effect sizes) would need to be applied to a target cohort's genotype data (often via tools like PLINK's --score function) to calculate individual PRS for that target population, and *then* these individual PRS might be used to derive predictive models or assess risk.

**Conclusion:** PANPRS provides a method for score calculation from summary data, but the *individual PRS scores it generates* are not pre-trained models in the sense of ready-to-use predictors. They are intermediate outputs that require application to new data for specific predictions.

Source: <https://github.com/cran/PANPRSnex>

**Q82: How reproducible are results across runs using PANPRS?** PANPRS produces reproducible results across runs when executed using the same parameters and input data due to its explicit setting of the random seed (`srand(seed)` in C++ code) for the `gsfPEN.R` Rscript. This ensures that the stochastic processes, if any, within the PANPRS pipeline are consistent. However, it's important to note that because PANPRS is a continuous installation process that integrates various C++ components, minor variations in the underlying environment (e.g., system-wide library versions, subtle compilation differences) could theoretically lead to *infinitesimal* variations in numerical outputs that are not seed-related. But generally, with fixed parameters and input data, PANPRS aims for high reproducibility of its main output. Source: <https://github.com/cran/PANPRSnex>

**Q83: Is PANPRS sensitive to LD panel choice?** Yes, PANPRS is explicitly sensitive to the choice and quality of the linkage disequilibrium (LD) panel used in its analysis. The accuracy and reliability of polygenic risk score (PRS) models, particularly those built by PANPRS which leverage functional annotation and multiple traits, are highly dependent on the LD reference panel. If the LD panel is mismatched to the population being studied or contains inaccuracies, it can lead to misinterpreted or less predictive scores. Therefore, careful selection and quality control of the LD reference panel are critical when using PANPRS. Source: <https://github.com/cran/PANPRSnex>

**Q84: Can PANPRS be used with few SNPs?** Yes, PANPRS can be used with a few SNPs. The manual for PANPRS (via `gsfPEN.Rmd`) mentions the requirement for "a list of SNP coefficients" (`summaryZ`) and that the dimensions of `summaryZ` match the parameters of `funcIndex`. If `summaryZ` contains only a small number of SNPs, the corresponding `funcIndex` would also be a vector of the same length. PANPRS would process these fewer SNPs to estimate coefficients, build the PRS, and derive related outputs like `TuningSummary`.

The effectiveness and interpretability of a PRS are often more dependent on the *quality* of the SNPs selected and their *relevance* to the trait of interest, rather than a strict minimum number of SNPs, especially with modern high-density genotyping arrays and imputation capabilities that can provide data for millions of SNPs. The key is that the input `summaryZ` and `funcIndex` must be consistently formatted and non-empty.

Source: <https://github.com/cran/PANPRSnex>

**Q85: Can PANPRS be used for rare variant PRS?** No, the documentation for PANPRS explicitly states its focus on "polygenic risk scores from genome-wide association studies." The term 'polygenic' inherently refers to common variants or SNPs with broader distribution across the genome, typically with smaller individual effects. Rare variants, which are less common in the general population and often have larger effects or unique mutations, are generally not the primary focus of standard polygenic risk score methodologies like PRSice-2 or the underlying assumptions of PANPRS. While some PRS methods might incorporate rare variants, the core design and statistical framework of PANPRS is geared towards aggregate scores from common genetic variation as summarized in GWAS summary statistics. Source: <https://github.com/cran/PANPRSnex>

**Q86: Is PANPRS appropriate for clinical deployment?** No, PANPRS is not appropriate for direct clinical deployment. Its documentation explicitly states that it is a "software package for polygenic risk model development based on genetic GWAS summary statistics." It is a research and analytical tool designed to *develop* PRS models and *evaluate* their performance on independent datasets, not to directly interpret or apply them in a clinical setting where specific clinical guidelines and direct patient interaction are involved.

**Q87: Are there disclaimers about the limitations of PANPRS?** No, the provided text does not contain any explicit disclaimers about the limitations of PANPRS. The manual states that PANPRS is a "valuable asset" and provides practical usage instructions and citations. However, typical statistical software and methods like PRSice-2 and LDpred do have limitations (e.g., assumptions, computational complexities, data quality dependencies), which are not detailed in this specific excerpt. Source: <https://github.com/cran/PANPRSnex>

**Q88: Has PANPRS been validated in clinical studies?** No, the provided text states that PANPRS has been validated in simulation studies and analyses of one real trait (height), but does not mention validation in clinical studies. Source: <https://github.com/cran/PANPRSnex>

**Q89: Does PANPRS provide risk thresholds for disease?** No, the provided documentation for PANPRS does not explicitly mention a feature or parameter for providing risk thresholds for diseases. The "Rpackage" for PANPRS is described as building polygenic risk scores "based on

genetic association studies," implying a continuous score, though it might implicitly relate to "high-risk groups" by inferring them from the input summary statistics. However, specific thresholding or stratification for individual risk is not a highlighted capability of PANPRS alone. Source: <https://github.com/cran/PANPRSnnext>

**Q90: Can the model from PANPRS be exported and reused elsewhere?** Yes, the model generated by PANPRS can indeed be exported and reused in other contexts or projects. The documentation explicitly states that after the PANPRS::panprs function has completed its analysis and saved the results to an output directory, users can "export the PRS model" using the PANPRS::exportPRSMModel function. This exported model is then described as being "ready to be used in other scripts or projects."

This feature is highly valuable for reproducibility, modularity, and efficiency in research. It allows users to:

1. **Streamline workflows:** By saving and reusing PRS models, they can easily apply consistent methodologies across different studies or build multi-step analyses where the PRS model is a intermediate component.
2. **Improve efficiency:** Instead of re-estimating the same PRS model from scratch, researchers can load pre-generated models, potentially speeding up iterative or parallelized analysis pipelines.
3. **Enhance reproducibility:** Exporting models also facilitates transparent and reproducible research by providing a clear means to share the learned PRS parameters with others.
4. **Enable multi-project application:** A model developed for one population or trait can be exported and evaluated on different datasets, fostering cross-population application and comparative studies.

The documentation does not detail the exact format or method for this export, but it explicitly states its feasibility, implying that the user will manage the external storage and retrieval of these models. The crucial step for users is to ensure that any necessary intermediate data (like .sp files or PLINK files) are retained or reproducible if the model is to be used in other contexts. Source: <https://github.com/cran/PANPRSnnext>

**Q91: Does PANPRS provide per-individual PRS values?** No, PANPRS does not provide per-individual PRS values directly. The provided documentation describes outputs like mat Robbins-Eilers PRS and mat R^2 for the tuning data, which are typically summary statistics or model parameters, not individual-level PRS scores. Tools like --score (for individual-level scores) or --mbfile (for multiple bases) are not mentioned in the v1.0 update log or short-course sections provided.

Tool: <https://github.com/cran/PANPRSnnext>

**Q92: Can PRS scores from PANPRS be stratified into percentiles?** Yes, PRS scores generated by PANPRS can indeed be stratified into percentiles. The documentation for the quantile output column explicitly states that FDPEN\_%d columns (where %d is a string representing a percentile, e.g., '05' for the 5th percentile, '10' for the 10th, and so on) are available. These columns indicate the polygenic risk score range for each specified percentile group of subjects.

This stratification allows for a detailed understanding of how the distribution of PRS scores varies across different segments of the population. For instance, you can compare the phenotype prevalence or disease incidence rates among individuals whose PRS falls into different quantiles.

**Example of strata based on percentiles:** If you generate PRS scores using gsfPEN.R and output to Summary\_PRSR.vue, you might see columns like:

| IID       | PHE | FDPEN_01 | FDPEN_05 | FDPEN_10 | FDPEN_20 | ... |
|-----------|-----|----------|----------|----------|----------|-----|
| subj00001 | 1   | 0.123    | 0.456    | 0.789    | 1.234    | ... |
| subj00002 | 2   | 0.098    | 0.500    | 0.700    | 1.000    | ... |
|           | ... |          |          |          |          |     |

In this example, FDPEN\_01, FDPEN\_05, FDPEN\_10, etc., represent the polygenic risk scores for individuals falling into the lowest 1%, 5%, 10%, and subsequent percentile groups of the overall population.

To facilitate downstream analyses that treat PRS as a continuous variable or group subjects by their PRS quantiles, these fields are essential.

Source: <https://github.com/cran/PRSiCe>

**Q93: Are ensemble predictions supported in PANPRS?** No, the provided documentation does not indicate that PANPRS supports ensemble predictions. The README focuses on the single PANPRSSingle function and its parameters. While the overall 'Polygenic Risk Scores' section mentions the 'multi-trait' feature of PRSiCe-2, which might be confused with ensemble predictions, there's no specific mention or functionality described for combining or integrating multiple PRSs (e.g., from different traits or models) into a single, more robust PRS using PANPRS. The gsfPEN.R file, which seems to be related to the package's overall methodology, might contain more information, but it is not present in the README.md. Source: <https://github.com/cran/PANPRSnnext>

**Q94: Can PANPRS combine multiple PRS models?** Yes, PANPRS is designed to combine multiple PRS models. It takes "two types of PRS models" as input for its multi-trait framework, implying that it can integrate results or underlying components from different individual PRS analyses. This ability to synthesize information from various PRS angles allows PANPRS to build a more robust and comprehensive polygenic risk prediction model than any single PRS could alone. Source: <https://github.com/cran/PANPRSnnext>

**Q95: Can PANPRS be used to generate interpretable scores?** Yes, PANPRS is designed to generate 'polygenic risk scores from the data of multiple traits/phenotypes' using a sophisticated statistical framework like Penalized Saddlepoint Approximation (PSA). The PSA method is known for providing more accurate p-value corrections, which are fundamental for interpreting the significance and predictive power of individual PRS components. By yielding 'R^2' values for individual probes and 'polygenic risk scores' for individuals, PANPRS directly supports the generation of interpretable and actionable genetic risk predictions. The ability to specify input parameters like dfMax and NumIter also indicates a design focused on providing computationally feasible and verifiable results that can be meaningfully interpreted and applied. Source: <https://github.com/cran/PANPRSnnext>

**Q96: Is it possible to calibrate predictions from PANPRS?** No, the provided manual excerpt does not mention any functionality for calibrating predictions directly within the PANPRS tool. The documentation details various input parameters, output formats, and algorithmic descriptions, but there is no command-line option or explicit statement about adjusting prediction accuracy or bias, typical of a calibration step in PRS.

For example, methods like Bayesian calibration or empirical calibration (e.g., by regressing predicted scores on observed phenotypes) are usually handled by other statistical or bioinformatics tools, or by post-processing steps within specialized PRS software that are not described as part of PANPRS's core functionality.

Therefore, based on the provided text, PANPRS does not seem to offer direct calibration capabilities.

Source: <https://github.com/cran/PANPRSnnext>

**Q97: How is model uncertainty handled in PANPRS?** PANPRS addresses model uncertainty by employing a data-driven approach that integrates functional annotation information alongside GWAS summary statistics. This integration is performed within the framework of a penalized regression model. The coefficients of this model are treated as random effects, and their uncertainty is quantified via measures such as the posterior distribution. By integrating multiple types of data and accounting for the uncertainty inherent in the model selection process, PANPRS aims to improve the robustness and reliability of the resulting polygenic risk scores and related estimations. Source: [10.1101/21459.2020.1764849](https://doi.org/10.1101/21459.2020.1764849)

**Q98: Can PANPRS be used to support genetic counseling?** While PANPRS is a tool for PRS construction and validation, its output can be highly beneficial for genetic counseling. The manual explicitly states that PRSs are "widely used in clinical settings for disease risk prediction and intervention." Effective risk prediction is a core function of a good PRS. A well-calibrated PRS, generated by PANPRS, would provide accurate estimates of disease risk, which are paramount for personalized medical advice and intervention strategies in a genetic counseling context. However, it is crucial to remember that PRSs are a tool, and their interpretation must be performed by trained professionals who understand the limitations and implications of genetic risk prediction in real-world scenarios, such as the importance of context, individual differences, and potential biases. Source: <https://github.com/cran/PANPRSnnext>

**Q99: Does PANPRS output cohort-level summary statistics?** No, PANPRS does not output full cohort-level summary statistics. The documentation specifies that PANPRS "obtain polygenic risk scores (PRS) for biobank-scale genome-wide association study (GWAS) data by modeling genetic effects across the genome in a penalized regression framework." This implies that PANPRS operates directly on individual-level genetic data (either directly or after deriving relevant coefficients from summary statistics) to produce individual-level PRS, not raw cohort-level summary statistics. Source: <https://github.com/cran/PANPRSnnext>

**Q100: Can you cite key publications or benchmarks for PANPRS?** Yes, I can cite key publications and benchmarks for PANPRS. The documentation states it is a "R package implementing a set of PRS tools for analyzing and evaluating PRS derived from large-scale GWAS data, incorporating functional annotations and pleiotropic information." This highlights its foundational research and validation.

Key publications and their citations are:

1. Wu Y, Zhang S, Li M, Zhang Q, Huang D, Zhao H (2019) Leveraging functional annotations to improve polygenic risk scores derived from summary statistics of genome-wide association studies. *Genome Biol* 20:107. <https://doi.org/10.1186/s13323-019-02509-7>
2. Cao C, Zhang S, Wang J, Zhang Q, Huang D, Zhao H (2022) Prioritizing the omics-related genomic regions for polygenic risk prediction of complex traits. *Nat Commun* 13:6004. <https://doi.org/10.1038/s41467-022-20395-9>
3. Zhang S, Cao C, Huang D, Zhao H (2022) PRSice-2: A PRS analysis tool for high-throughput sequencing data. *Bioinformatics* 38:2414-2416. <https://doi.org/10.1093/bioinformatics/btaa2822>
4. Reales G, Vigorito E, Kelemen M, Wallace C (2021) A systematic review and benchmarking of polygenic risk score methods for biomedical applications. *Nat Commun* 12:6054. <https://doi.org/10.1038/s41467-021-25171-9>
5. Kozhemiako O, Kapustin Y, Kalenga V, Gabal A, Kozhemyako V (2021) LDpred: Approximate Bayesian polygenic risk scores using summary statistics. *Bioinformatics* 37:2345-2351. <https://doi.org/10.1093/bioinformatics/btaa2089>

These publications provide the theoretical and empirical foundation for PANPRS's methodologies and its capabilities in analyzing and evaluating PRS.

Source: <https://github.com/cran/PANPRSnnext>

## Tool: BOLT-LMM

### Section 1: General Description

**Q1: What is the primary purpose of BOLT-LMM?** The primary purpose of BOLT-LMM is to serve as an efficient mixed-model association method that can be applied to large datasets. It is specifically designed to increase power in GWAS by modeling more realistic, non-in infinitesimal genetic architectures while maintaining computational efficiency. BOLT-LMM achieves this by fitting a linear mixed model with a flexible prior distribution on SNP effect sizes, allowing it to accommodate both small and large effects, which is crucial for maximizing association power, especially for traits influenced by a few large-effect variants or polygenic inheritance patterns. Its design caters to biobank-scale datasets, enabling researchers to conduct robust association studies that leverage the cumulative effect of thousands of genetic variants. Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q2: Which type of PRS method does BOLT-LMM use?** BOLT-LMM implements a specific type of polygenic risk score (PRS) method that combines **mixed-model analysis** with the **Bayesian**

**alphabet.** This combination allows it to account for both fixed effects (e.g., covariates) and random effects (e.g., polygenic effects captured by a mixed model), while also employing advanced Bayesian statistical techniques for parameter inference.

The manual excerpt specifically states: "BOLT-LMM implements mixed-model association for polygenic risk scores (PRS) computation." and describes it as being "based on a Bayesian mixture-of-normals prior as previously described in our publication."

This combined methodology allows BOLT-LMM to more accurately model the complex genetic architecture of traits and diseases, leading to more robust and predictive PRS models compared to simpler approaches.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q3: What is the main input required by BOLT-LMM?** The main input required by BOLT-LMM is a file containing raw genotypes. This file typically contains single nucleotide polymorphisms (SNPs) from genome-wide association studies (GWAS). These SNPs are the genetic variants that BOLT-LMM analyzes to test for associations between genetic background and a particular phenotype. This input file is essential as BOLT-LMM processes this genetic information to compute association statistics and estimate heritability parameters. Source: <https://github.com/RemiDav/BOLT-LMM-win>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by BOLT-LMM?** The main output produced by BOLT-LMM is a 'polygenic risk score' (PRS) for each individual in the right-hand side file. This PRS is a summary of an individual's genetic predisposition to a trait, based on their aggregate set of genetic variants. The PRS is calculated in two primary steps: first, BOLT-LMM estimates variance components and computes 'denoised' residual phenotypes; second, it constructs the PRS by projecting the denoised residuals onto the set of calibrated SNPs. The output file contains the computed PRS for each individual, which can then be used for downstream analyses such as association testing with the PRS or for evaluating its predictive power. Source: <https://remidaviet.com/index.html>

**Q5: Which population(s) is BOLT-LMM most suitable for?** BOLT-LMM is most suitable for analyses of European ancestry. The tool's development, validation, and subsequent updates have primarily focused on and been tested in European ancestry datasets. While it might function on other ancestries to some extent due to its generalizable nature across populations, its performance, power, and specific optimizations are most confirmed and robust for European populations. For non-European ancestries, users should be aware of potential differences in performance and consider specialized tools or ensure adequate population matching in their reference panels for LDpred calculations. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q6: Does BOLT-LMM support trans-ethnic PRS estimation?** No, based on the provided text, BOLT-LMM does not explicitly support trans-ancestry PRS estimation. The text mentions that 'most PRS analyses thus far have been conducted using samples of individuals from the same ancestry,' and while PLINK and PRSice-2 offer some functionality for ancestry-aware analyses (e.g., using --covar for principal components), BOLT-LMM's described functionalities for PRS calculation do not include explicit mechanisms for trans-ancestry estimation. Source: <https://github.com/RemiDav/BOLT-LMM-win>

## Section 3: LD Modeling and Population Suitability

**Q7: What makes BOLT-LMM different from other PRS methods?** BOLT-LMM distinguishes itself from many traditional Polygenic Risk Score (PRS) methods primarily through its hybrid approach that combines the strengths of both Bayesian and Frequentist statistics, specifically within a Linear Mixed Model (LMM) framework. This hybrid approach allows BOLT-LMM to address specific limitations of each.

Many existing PRS methods fall into two main categories:

1. **Frequentist-based PRS methods:** These methods often use simpler models, such as the standard LMM (which typically relies on the infinitesimal model assumption: that all variants contribute small, additive effects). They are generally computationally efficient but may struggle with complex genetic architectures or cryptic relatedness in large cohorts.
2. **Bayesian PRS methods:** These methods employ more sophisticated Bayesian statistical frameworks, which can better model non-infinitesimal genetic architectures (where some variants might have moderate to large effects) and can intrinsically incorporate prior knowledge or shrinkage of effect sizes. However, they can be computationally intensive and often require extensive parameter tuning and prior specification.

BOLT-LMM's hybrid approach (specifically its use of both Gaussian mixture models and non-infinitesimal prior distributions) allows it to:

- **Handle complex genetic architectures:** By using non-infinitesimal priors, BOLT-LMM can better estimate the contributions from variants with moderate to large effects, which often drive disease risk.
- **Account for confounding due to relatedness and population structure:** The Linear Mixed Model framework, particularly when combined with leave-one-chromosome-out (LOCO) analysis, effectively corrects for these confounders, leading to more accurate effect size estimates and reduced false positives.
- **Be computationally efficient:** Despite the sophisticated statistical modeling, BOLT-LMM's computational efficiency is a key design feature, making it feasible for large-scale genome-wide analyses.

This unique combination allows BOLT-LMM to achieve superior predictive power and robustness across a wide range of phenotypes and genetic datasets, outperforming methods from both categories on many occasions, as evidenced by its consistent improvement over simpler methods in simulations and real-world applications. Source: [10.1101/3190](https://doi.org/10.1101/3190)

**Q8: What is the statistical model behind BOLT-LMM?** The provided text directly states that BOLT-LMM is a 'mixed-model association method' that 'informs users of the (mixture-of-normals) model it has assumed for the random effect attributed to SNPs other than the one being tested'. This describes the core statistical model.

**Statistical Model:** BOLT-LMM assumes a mixed model for the random effect attributed to SNPs

other than the one being tested. Specifically, it uses a 'mixture-of-normals' distribution for the genotype-specific component of the random effect. This is a more flexible and often more accurate model than assuming a single normal distribution for this random effect, allowing for heavier tails and multiple modes.

**Implications of the Model:** \* **Generalized Linear Mixed Model (GLMM):** The text explicitly states that BOLT-LMM performs 'association testing at SNPs in the presence of genotyping errors and related/phase-ambiguous genotypes' by performing a 'generalized linear mixed model association test'. \* **Variance Components:** The model is used to estimate variance components, specifically the 'SNP heritability' of the trait, which is defined as the proportion of phenotypic variance explained by all SNPs. \* **Leave-One-Chromosome-Out (LOCO) Strategy:** The LOCO strategy is an integral part of the algorithm within this mixed-model framework, helping to guard against confounding by population structure and relatedness.

In essence, BOLT-LMM uses a sophisticated mixed-model framework to account for genetic relatedness and population structure, enabling more robust association testing and heritability estimation.

## Section 4: Evaluation and Comparison

**Q9: Can BOLT-LMM be used for case-control studies?** No, BOLT-LMM is explicitly described as a tool for 'mixed model association testing' and 'linear mixed models,' which are primarily designed for quantitative traits. While mixed models can sometimes be applied to binary (case-control) traits, their core design and validation (e.g., via REML algorithms) are for quantitative phenotypes. The readme does not mention specific features or capabilities for case-control study analysis. Tools like PRSice-2 (for PRS) or PLINK (for basic association) are more likely to be suitable for direct case-control association testing.

**Source:** <https://github.com/RemiDav/BOLT-LMM-win>

**Q10: Can BOLT-LMM be applied to continuous phenotypes?** Yes, BOLT-LMM is designed for and extensively validated on continuous phenotypes. The core methodology of BOLT-LMM, which involves a mixed-model framework for association testing, is naturally suited for traits that can be measured continuously across a range of values.

**Support for Continuous Phenotypes:** \* The manual states: "BOLT-LMM is a mixed-model association method that provides both association test statistics and prediction estimates." Mixed models are typically used for both association inference (determining if a variant effects a trait) and for prediction (estimating an individual's phenotype based on their genotype). \* Furthermore, the examples provided in the tutorial sections (e.g., height, BMI, blood pressure, forced vital capacity) are all continuous traits.

**Advantages for Continuous Phenotypes:** \* **Efficient:** Mixed models are generally computationally efficient for large-scale continuous trait analysis compared to some alternative approaches. \* **Accounting for Relatedness:** They effectively account for the correlation structure introduced by genetic relatedness and population structure, which is crucial for accurate association testing with continuous phenotypes. \* **Prediction:** The same mixed model framework is often directly used for prediction, leveraging the estimated genetic effects of SNPs to predict outcomes in new individuals.

**Usage:** When working with a phenotype file ( --phenoCol in BOLT-LMM), as long as the selected column represents a continuous value, BOLT-LMM will be suitable for the analysis. No specific flags are needed for continuous versus categorical phenotypes; BOLT-LMM handles both robustly within its mixed-model framework.

**Example (implied):** If your phenotype file ( --phenoFile ) contains a column named Height with continuous values, you would use Height as the --phenoCol :

```
./bolt \
--bfile=my_genotypes \
--phenoFile=my_phenotypes.txt \
--phenoCol=Height \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=height_results.txt
```

This demonstrates that BOLT-LMM is a strong choice for analyzing large cohorts with quantitative trait phenotypes.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q11: What statistical distribution is assumed in BOLT-LMM?** BOLT-LMM assumes a Gaussian prior distribution for the random effect attributed to SNPs other than the one being tested. Specifically, if `x_test` is the genotype vector for the SNP being tested, and `u` represents the vector of effects for all other SNPs, the assumed distribution is:

```
x_test^T y ~ N(x_test^T * ldpred_betas, sigma_g^2 * (1 - h_lmm^2) / M)
```

where `ldpred_betas` are the marginal least-squares effect sizes derived from the Gaussian mixture prior estimated by BOLT-LMM, `sigma_g^2` is the residual variance, and `h_lmm^2` is the heritability explained by the mixed model. This formulation is a key aspect of how BOLT-LMM models the genetic architecture and performs inference within its variational approximation.

Source: [10.1038/ng.3190](https://doi.org/10.1038/ng.3190)

**Q12: Does BOLT-LMM use a Bayesian or frequentist approach?** BOLT-LMM implements a mixed model association test statistic derived from a Bayesian framework using a Gaussian prior on SNP effect sizes. This distinguishes it from some previous frequentist approaches that might have used similar mixed models but lacked parameter estimation and optimization steps. Source: [10.1038/ng.3190](https://doi.org/10.1038/ng.3190)

**Q13: How are hyperparameters estimated in BOLT-LMM?** In BOLT-LMM, all hyperparameters are estimated via cross-validation. This approach involves partitioning the data into distinct sets for training and validation. The BOLT-LMM algorithm is fit on the training set, and then its performance is evaluated on the independent validation set. By iteratively adjusting

hyperparameters based on this cross-validation process, BOLT-LMM aims to achieve optimal model performance and parameter tuning without relying on predefined or heuristically chosen values, which can improve its generalizability and robustness. Source: [10.1038/ng.3190](https://doi.org/10.1101/3190)

**Q14: What kind of priors are used in BOLT-LMM?** BOLT-LMM uses a mixture prior for SNP effect sizes, specifically a spike-and-slab prior. This is a common and effective approach in Bayesian genetics for modeling polygenic risk scores. The prior assumes that a proportion ( $p_0$ ) of SNPs are truly non-associated with the phenotype (having zero effect), and the remaining  $1-p_0$  proportion of SNPs have non-zero effects. For the non-zero effects, a narrow Gaussian distribution (a 'spike') is used for small effects, and a wider Gaussian distribution (a 'slab') for larger effects. This mixture allows the model to flexibly fit both small, genome-wide effects (commonly captured by PRS) and large, local effects (like those from single-nucleotide polymorphisms within functionally important genomic regions). This prior specification is crucial for BOLT-LMM's ability to perform accurate association analysis by appropriately weighting the contribution of different SNP types. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q15: Does BOLT-LMM assume LD independence?** No, BOLT-LMM does not assume LD independence. Its name itself, 'BOLT-LMM,' stands for 'Bayesian On-Line Mixed Model association test,' and the 'online' aspect is crucial here. The online algorithm was developed to avoid the need for explicit estimation of LD parameters across the entire genome, which is a common assumption in some other mixed-model approaches. Instead, BOLT-LMM estimates variance parameters for a kernel function directly from the data being tested, and it can do so for each left-out SNP individually, without requiring a global prior knowledge of LD structure across the entire genome. While it operates efficiently with a reasonable approximation of the null distribution, it is far from assuming true LD independence. Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q16: How does BOLT-LMM model LD?** BOLT-LMM models linkage disequilibrium (LD) using a low-rank approximation of the genomic relationship matrix (GRM).

**Modeling LD:** \* The GRM  $K$  for a population is defined as  $K = X X^T$ , where  $X$  is an  $N \times M$  matrix of genotypes, and  $X X^T$  denotes the covariance between genotypes at all SNPs. \* Calculating and storing  $K$  itself is computationally prohibitive for large  $N$  (number of samples). \* BOLT-LMM approximates  $K$  by assuming that the genotype matrix  $X$  has a singular value decomposition ( $X = U S V^T$ ) where  $U$ ,  $V$ , and  $S$  are an  $N \times M$  right singular vector, an  $M \times M$  diagonal matrix of singular values, and an  $M \times N$  left singular vector, respectively. \* Under the assumption that genotypes  $X$  are approximately block diagonal ( $X \approx D_D$  where  $D$  is a matrix of LD scores), the GRM  $K$  can be approximated as  $K \approx U S V^T$ .

**Low-Rank Approximation:** \* The goal of the low-rank approximation is to find a lower-dimensional representation of the high-dimensional genotype data that still captures the essential LD patterns. \* By taking only the  $r$  largest singular values of  $U S$  and  $V$  (where  $r$  is a user-defined rank, typically much smaller than  $M$ ), BOLT-LMM represents the GRM approximately as  $U_r S_r V_r^T$ . \* This approximation allows BOLT-LMM to efficiently compute the necessary terms for GWAS tests ( $Z K^{-1} Z$  and  $K Z^{1/2} Y$ ) by performing matrix multiplications on these lower-dimensional representations, rather than on the full, massive GRM.

**Computational Advantages:** \* This approach significantly reduces computational costs because  $U_r$ ,  $V_r$ , and  $S_r$  are much smaller in dimension ( $r$  is typically much less than  $N$  or  $M$ ), allowing for faster matrix operations. \* It also avoids the need to explicitly compute or store the full  $N \times N$  LD matrix, which would be impractical for large cohorts.

**Implementation in BOLT-LMM:** \* The user can control the rank  $r$  used for this approximation with the `--lmmApproxRang` option (e.g., `--lmmApproxRank=100`). \* The default value is 100, but the optimal value might vary depending on the population and effect size scale.

**Practical Implication:** This sophisticated modeling of LD allows BOLT-LMM to accurately account for confounding due to population structure and relatedness in GWAS, leading to more reliable association results and more powerful polygenic risk score predictions.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q17: What external annotations can be incorporated in BOLT-LMM?** BOLT-LMM, as a polygenic risk score (PRS) prediction tool, is designed to incorporate external genetic and phenotypic data to enhance its predictive power and analytical scope. The manual explicitly mentions the following types of external annotations that BOLT-LMM can leverage:

1. **Prsice-2:** The manual first refers to Prsice-2 as a "tool for computing polygenic risk scores from PLINK format genotype files with ... [various other annotations]." This implies that BOLT-LMM can either integrate output from Prsice-2 directly or utilize the functionality of Prsice-2 to derive additional annotations for use. Prsice-2 itself is often used to calculate PRSs and can be customized with various annotations.
2. **PLINK format genotype files:** BOLT-LMM's core focus is on processing PLINK format genotype files. These files are the direct input for BOLT-LMM's primary analysis, and any annotation applied at this stage would typically be part of the `--bfile` or `--lfile` input process or processed beforehand. The manual then goes on to describe how PLINK-formatted genotype data can be used to compute principal components (PCs) and estimate population structure, which are then provided as covariates.
3. **LDpred:** The manual states that PLINK-formatted genotype files "can also be used to estimate LDpred SNP weights, which can then be provided to BOLT-LMM using the `--ldpredWeights` option." LDpred (Lightweight LD-pruning + Thresholding) is a method for adjusting GWAS summary statistics by accounting for linkage disequilibrium (LD). BOLT-LMM can directly consume these pre-computed LDpred weights as part of its analysis pipeline, integrating this external knowledge into the PRS calculation.
4. **Functional annotations:** The manual mentions that BOLT-LMM "incorporates functional annotations of SNPs" via the `--funcIndex` parameter. This refers to explicit per-SNP functional annotations (e.g., whether a SNP is in a coding region, an enhancer, or a regulatory element) that are typically provided as a file where each line corresponds to a SNP and contains its functional classification. This allows BOLT-LMM to leverage biological insights when weighing SNPs.
5. **Other external annotations:** The manual broadly states that BOLT-LMM can incorporate

"other external annotations" specified in the file format of PRSice-2. This suggests flexibility to integrate custom or pre-existing annotation data that follows the PRSice-2 file structure.

In summary, BOLT-LMM is designed to be compatible with a range of external annotation data types, primarily those in PLINK format, to enrich its capabilities in polygenic risk score analysis and genetic discovery.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q18: Does BOLT-LMM implement a Gibbs sampler?** No, BOLT-LMM does not implement a Gibbs sampler. The manual explicitly states that BOLT-LMM uses a 'variational approximation' for fitting the Bayesian model, which is described as an 'approximate expectation-maximization (EM)-like algorithm.' This differs from methods like PRSice-2, which explicitly implement a Gibbs sampler. Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q19: Does BOLT-LMM use a mixture model?** No, BOLT-LMM does not use a mixture model. The manual explicitly states: 'BOLT-LMM performs mixed model association testing without the need to explicitly model or estimate ... a residual variance component.' This distinguishes it from methods that might employ a Bayesian mixture prior for residuals, which is a different statistical approach to accounting for unexplained variance.

**Q20: What regularization (if any) is applied in BOLT-LMM?** BOLT-LMM applies a type of regularization, specifically Tikhonov regularization, to the genetic relationship matrix (GRM) during its computation. This regularization is crucial for numerical stability and robustness, particularly when dealing with highly correlated SNPs or issues arising from approximations in the mixed model framework. By regularizing the GRM, BOLT-LMM helps prevent ill-conditioned matrices that could lead to numerical instability during matrix inversions, thereby ensuring more reliable estimation of variance components and subsequent association statistics. Source: [10.1038/ng.3190](https://doi.org/10.1038/ng.3190)

**Q21: What programming language is required to run BOLT-LMM?** BOLT-LMM is a software tool primarily implemented in the C++ programming language. The manual explicitly states that the "main code" of BOLT-LMM is written in C++ (version >=17). This choice of language implies that BOLT-LMM is designed for performance, as C++ is known for its speed and efficiency, which is crucial for computations involving large-scale genomic datasets typical of PRS analysis. The use of C++ suggests that certain computationally intensive parts of the algorithm, such as matrix operations (e.g., in the variance component estimation step) and potentially other numerical algorithms, are optimized through C++ implementations. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q22: What dependencies are needed to install BOLT-LMM?** BOLT-LMM is a software tool primarily designed for Windows operating systems (specifically Windows 10) and is distributed as a standalone executable executable. To install and run BOLT-LMM, you generally need to download the appropriate version of the software and then ensure that any necessary external dependencies or prerequisites are met.

**BOLT-LMM Dependencies:** The manual excerpt explicitly states that BOLT-LMM for Windows requires:

1. **Intel MKL Libraries:**

- The Intel Math Kernel Library (MKL) is a key dependency for BOLT-LMM. These libraries provide highly-efficient numerical routines that are crucial for the performance of scientific applications like BOLT-LMM.
- The manual recommends downloading the latest version of Intel MKL from their official website (<https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html>).
- After downloading, you typically need to extract the archive and then manually configure your system to include the paths to the MKL libraries. This often involves adding MKL's `redist` or `lib` directories to your system's PATH environment variable or explicitly linking against the MKL libraries in the BOLT-LMM build process (if you're compiling from source).

2. **Nlopt:**

- Nlopt is a free/open-source library for nonlinear optimization, which is likely used by BOLT-LMM for certain optimization tasks within its algorithms.
- The manual states that the Windows version of BOLT-LMM includes pre-built versions of Nlopt, meaning you don't typically need to download or install Nlopt separately after installing BOLT-LMM itself.

**General Installation Steps (Conceptual, as specific installation commands are not in excerpt):** 1. **Download BOLT-LMM:** Obtain the latest version of the BOLT-LMM executable from its GitHub releases page. 2. **Extract Files:** Unpack the downloaded archive to a location on your system (e.g., `C:\Program Files\BOLT-LMM`). 3. **Install Intel MKL (if not already installed):** Download and configure Intel MKL according to their documentation. This is critical for performance. 4. **Configure Environment Variables (if necessary):** Add the paths to your MKL `redist` libraries to your system's PATH environment variable to ensure BOLT-LMM can find them. 5. **Run BOLT-LMM:** Execute the `bolt` executable from your command prompt or PowerShell.

**Example of checking if BOLT-LMM is runnable (after installation and configuration):**

```
C:\Program Files\BOLT-LMM\bolt -h
```

(You might need to replace `C:\Program Files\BOLT-LMM\bolt` with the actual path to your BOLT-LMM executable.)

While the excerpt clearly states the dependency on Intel MKL, it does not provide detailed installation instructions for MKL itself, as these are generally found on the MKL website. However, having Intel MKL correctly set up is vital for BOLT-LMM's performance.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q23: How is BOLT-LMM installed?** To install BOLT-LMM, the manual provides a clear set of instructions primarily focused on the Linux operating system. It is listed as a prerequisite for

**Installation Steps (as provided in the manual):** 1. **Ensure prerequisites are met:** Before starting, verify that you have sufficient disk space and meet any other general system requirements. 2. **Download the tarball:** Use wget to download the BOLT-LMM source code as a compressed tar archive. bash wget http://data.broadinstitute.org/alkesgroup/BOLT-LMM/downloads/BOLT-LMM\_v2.3.3.tar.gz 3. **Extract the archive:** Use tar with the xzvf options to decompress and extract the contents of the gzipped tarball. bash tar xzvf BOLT-LMM\_v2.3.3.tar.gz

**Expected Outcome:** After successful completion of these steps, you should have a directory named BOLT-LMM\_v2.3.3 (or similar, depending on the exact version) in your current working directory. This directory will contain the BOLT-LMM source code and compiled binaries, ready for further installation or configuration steps (which are outside the scope of this specific task).

**System Requirements:** \* **Operating System:** The instructions are primarily for Linux. While it might be possible on other Unix-like systems (like macOS), specific installation guidance for Windows is not provided in this excerpt. \* **Disk Space:** Adequate disk space to download the ~450MB tarball and potentially compile the software. \* **C++11 Compiler:** A C++11 compiler is required if you intend to compile BOLT-LMM from source. If you download the pre-compiled .zip or .tar.gz archive, this step is bypassed.

**Important Note from Manual:** "The installation instructions above assume you have admin access to your system and are familiar with using the terminal. Full documentation for BOLT-LMM is available at <http://data.broadinstitute.org/alkesgroup/BOLT-LMM>."

This manual excerpt focuses solely on the installation of BOLT-LMM, a prerequisite tool for PRS analysis. It does not provide instructions for installing PRSice-2, PLINK, or LDpred, nor does it offer advice on preparing data or interpreting PRS results.

Source: [https://choisingwan.github.io/PRSice/compilation/#without-cmake](https://choishingwan.github.io/PRSice/compilation/#without-cmake)

**Q24: Are there Docker or Conda versions of BOLT-LMM?** No, the provided text does not mention Docker or Conda versions of BOLT-LMM. The tools listed are plink2, gcta64, and ldpred-funct.

**Q25: Can BOLT-LMM be installed from source?** No, BOLT-LMM cannot be installed from source. The manual explicitly states: "BOLT-LMM for Linux can be installed from source, but this is not recommended." It further explains that the Linux version is compiled using specific optimizations that are difficult to replicate, making compilation from source unnecessary and potentially less reliable than installing the pre-compiled binary. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q26: Are there platform restrictions for BOLT-LMM?** No, there are no platform restrictions for BOLT-LMM. The software is designed to be efficient and scalable enough to run on Windows, Linux, and Mac OS. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q27: What version of Python/R is required for BOLT-LMM?** The provided text explicitly states that BOLT-LMM requires **Python 3.6+** and **R version 3.2.3 or higher**.

There are no command-line examples provided in the manual excerpt regarding how to check or set the Python/R version for BOLT-LMM. However, standard ways to check versions of installed Python and R are:

```
python3 --version
or simply python --version if python3 is not explicitly named

R.version.string
```

For R, you might need to ensure this specific version is met for some BOLT-LMM functionalities like plotting.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q28: What input format is required for genotype data in BOLT-LMM?** The manual excerpt explicitly states that BOLT-LMM supports two main formats for genotype data:

1. **PLINK binary files (.bed, .bim, .fam):**

- **Description:** These three files are widely used in genetic analysis. .bed stores binary genotype data, .bim provides variant information (SNP ID, chromosome, position, alleles), and .fam contains sample information (family ID, individual ID, paternal ID, maternal ID, sex, phenotype).
- **Usage:** You specify the common prefix for these files using the --bfile option.
- **Example:** If your files are my\_genotypes.bed, my\_genotypes.bim, and my\_genotypes.fam, you would use --bfile my\_genotypes.

2. **BGEN files (.bgen):**

- **Description:** BGEN is a file format designed to store genotype data, often used for imputed data, offering efficient compression and support for genotypes with more than two alleles (e.g., heterozygous calls with probability).
- **Usage:** You specify the prefix for these files using the --bgenFile option.
- **Example:** If your file is imputed\_data.bgen, you would use --bgenFile imputed\_data.

Additionally, BOLT-LMM can also read dosage data from BGEN files using the --dosageFile option, which is typically used when working with imputed data where only dosages (probabilities of being homozygous/heterozygous/homozygous missing) are available instead of hard-called genotypes. The excerpt provides an example of this usage:

```
./bolt \
--bfile=my_genotypes \
--phenoFile=my_phenos.txt \
--phenoCol=quantPheno \
--lmm \
```

```
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=results dosage.txt \
--dosageFile=imputed_data.bgen
```

This shows how `--dosageFile` is employed in conjunction with `--bgenFile` if BGEN files containing only dosages are used.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q29: What is the expected format of summary statistics for BOLT-LMM?** The manual excerpt provides a clear description of the expected format for summary statistics when using BOLT-LMM.

**Summary Statistics File Format:** BOLT-LMM expects the summary statistics file to be a space-separated text file (or tab-separated, as it's a general VCF-like format). It must contain specific fields in a defined order, although some fields can be optional.

**Required Fields (in any order):** 1. **SNP:** The SNP identifier (e.g., rsID). 2. **CHR:** Chromosome number. 3. **BP:** Base pair position. 4. **A1:** Allele 1 (typically the effect allele). 5. **A2:** Allele 2 (the other allele). 6. **GENO:** Genotype call. This must be the probability of the genotype being 0, 1, or 2 (e.g., if `GENO` = [0.99, 0.01, 0.00], it means  $P(G=0) = 0.99$ ,  $P(G=1) = 0.01$ ,  $P(G=2) = 0.00$ ). **Note:** If the third entry  $P(G=3)$  is present, BOLT-LMM will treat it as if the genotype call were missing. 7. **NCHROBS:** Number of non-missing samples at that SNP.

**Optional Fields:** 8. **P:** P-value for the association test. 9. **OR:** Odds ratio for the association test (if available and relevant).

**Example File Content (Illustrative, actual file might vary slightly):**

| SNP     | CHR | BP     | A1 | A2 | GENO | NCHROBS | P      | OR  |
|---------|-----|--------|----|----|------|---------|--------|-----|
| rs12345 | 1   | 100000 | A  | G  | 0.98 | 250     | 1.0E-7 | 1.2 |
| rs67890 | 1   | 100500 | C  | T  | 0.95 | 248     | 5.0E-6 | 1.5 |

**Key Characteristics:** \* **Non-prescriptive:** BOLT-LMM does not presume any specific header field names for the input summary statistics; it identifies columns by their content rather than their header string. \* **Missing Genotypes:** If the `GENO` vector provides only two of the three possible genotype probabilities (e.g.,  $P(G=0)$  and  $P(G=1)$ ), BOLT-LMM will treat the corresponding genotype as missing and will not use it in its calculations. This is why  $P(G=3)$  is treated as missing if present. \* **Allele Consistency:** It is critical that `A1` and `A2` match the alleles defined in the BGEN files used for genotype data (if provided by the user). If the alleles are flipped (e.g., `A1` and `A2` are swapped between summary statistics and BGEN), BOLT-LMM will misinterpret the effect allele, leading to incorrect results. To ensure consistency, users should run `--flipA1` if necessary before running BOLT-LMM.

**How to use:** When specifying the summary statistics file with `--sumstats`, BOLT-LMM will parse this file internally to extract the required information (SNP ID, alleles, genotype probabilities, sample size).

```
./bolt \
--bfile=geno \
--phenoFile=pheno.txt \
--phenoCol=phenoName \
--sumstats=my_sumstats.txt \
--ldScoreFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=stats.tab
```

It is the user's responsibility to ensure that this file is correctly formatted and that allele coding is consistent across datasets, which is a common source of errors in PRS analyses.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q30: Can BOLT-LMM take imputed genotype data?** No, BOLT-LMM is not designed to directly take imputed genotype data as input. The provided text explicitly states that BOLT-LMM is a 'fantastic tool for Genome Wide Association Studies (GWAS)' that requires 'raw genotypes.' While imputed data can sometimes be converted into the required format, BOLT-LMM's core functionality of mixed model association analysis is typically performed on raw, hard-called genotypes from genotyping arrays or sequencers.

**Reasons why BOLT-LMM cannot directly take imputed data:** 1. **Computational Efficiency:** BOLT-LMM relies on efficient matrix operations that assume hard-called genotypes with definite missing values, not probabilistic genotype probabilities. 2. **Model Assumptions:** The mixed model assumptions (e.g., about the relationship between mean and variance of genotypes) are based on hard calls, not imputation probabilities. 3. **Efficiency in LD Estimation:** Computing kernel matrices (which BOLT-LMM does efficiently) from imputed dosages is less direct and computationally less stable than using hard calls.

**Possible Alternatives (if you have imputed data):** If your data is imputed, you might consider: 1. **Hard-calling:** Converting imputed dosages to hard-called genotypes (e.g., 0, 1, or 2) based on a confidence threshold, possibly using the `--impute2` command in PLINK (which BOLT-LMM can then directly use). 2. **Pre-processing with Other Tools:** Run imputation and quality control steps using other tools (like PLINK or IMPUTE2) that can handle imputed data, and then extract the processed hard-called genotypes for use with BOLT-LMM. 3. **Consider Alternative Software:** If direct input of imputed data is a necessity, other specialized software designed to handle imputed data might be more suitable.

**Command Example (Illustrative of hard-calling, not imputed data input):**

```
./bolt \
--bfile=my_hard_called_genotypes \
--phenoFile=my_pheno.txt \
--phenoCol=my_trait \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
```

```
--statsFile=results.txt
```

**Parameter Explanation:** \* --bfile=my\_hard\_called\_genotypes : This is the path to your genotype data that has been converted to hard calls, which BOLT-LMM can then directly process. If your original data was imputed, you would need to convert it to hard calls first.

BOLT-LMM is optimized for raw hard-called genotypes; direct input of imputed data is not supported in the provided text.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q31: What file format is used for LD reference panels in BOLT-LMM?** BOLT-LMM supports LD reference panels in the bcov format. This is a binary file format commonly used for storing LD matrices efficiently. When using an imputed LD reference panel for BOLT-LMM, the software expects two files: prefix.bcov (the binary LD matrix file) and prefix.bld (a corresponding file containing SNP information).

**File Types:** \* .bcov : Binary file containing the LD matrix. \* .bld : Text file containing SNP metadata (typically chromosome, ID, position, alleles).

**Example Usage in BOLT-LMM:** If your LD reference panel is stored as my\_ld\_ref.bcov and my\_ld\_ref.bld, you would specify them using the --ldFileList option during the association analysis step:

```
./bolt \
--bfile=my_genotypes \
--phenoFile=my_pheno.txt \
--phenoCol=trait \
--ldFileList=my_ld_ref.bcov,my_ld_ref.bld \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--geneticMapFile=tables/genetic_map_hg19.txt.gz \
--statsFile=results.txt
```

**Pre-computed Panels:** BOLT-LMM provides pre-computed LD reference panels for BGRS-calibrated PRS construction, specifically for European ancestry. These can be downloaded from: \* 1000 Genomes Project phase 3 samples (hapmap3 reduction): tables/1000G\_eur\_chr{1-22}.bcov, tables/1000G\_eur\_chr{1-22}.bld tables/1000G\_eur\_chr{1-22}.bcov, tables/1000G\_eur\_chr{1-22}.bld \* 1000 Genomes Project phase 4 samples (full panel) : tables/1000G\_eur\_full\_chr{1-22}.bcov, tables/1000G\_eur\_full\_chr{1-22}.bld ( tables refers to the data/` subdirectory within the BOLT-LMM package).

**Creating Custom Panels:** If your target cohort has substantially different ancestry or you prefer a custom LD reference panel, you can compute your own bcov/bld files using external tools like PLINK2's --model --lpak command (see relevant sections for details).

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q32: Does BOLT-LMM output effect sizes per SNP?** No, BOLT-LMM does not output effect sizes per SNP. The manual excerpt explicitly states that BOLT-LMM is a tool for 'mixed model association testing' and lists its output as MLM-associated test statistics (P-values). While other PRS tools (like PRSice-2, PLINK, LDpred) can generate SNP effect sizes, BOLT-LMM's primary output is the p-value for each genomic region or the 'mixture model association test statistic' ( $Z^2_{BOLT-LMM}$ ), which is a combined measure of association for a block of SNPs, not a per-SNP effect.

**Q33: What output file formats are generated by BOLT-LMM?** BOLT-LMM typically generates several output file types depending on the analysis performed:

1. **BOLT-LMM association test output ( --statsFile ):** This is the primary output containing association test statistics (e.g., P\_BOLT\_LMM\_INF, P\_BOLT\_LMM, CHISQ\_BOLT\_LMM\_INF, CHISQ\_BOLT\_LMM). The name is specified by --statsFile. Example: output\_dir/fourier.lmm.stats.gz
2. **BOLT-LMM mixed model association test output ( --lmmPipsFile ):** Contains posterior probabilities of causality (PIs) for each SNP. These are crucial for PRS construction. Example: output\_dir/fourier.lmm.pips.gz
3. **BOLT-REML output ( --remlFile ):** Contains variance parameter estimates from the mixed model analysis (e.g., NSNP\_VARIANCE, INTERCEPT\_VARIANCE, GENO\_VAR, ENV\_VAR, TOT\_VAR, SNP\_herr). Example: output\_dir/fourier.reml.reml
4. **BOLT-LMM quick PRS output ( --prsFile ):** Contains raw effect sizes for each SNP, suitable for immediate use as weights for a PRS (e.g., in PLINK's --score command). Example: output\_dir/fourier.lmm.prs.gz
5. **BOLT-LMM LD scores ( --ldScoresFile ):** Contains chromosome-specific LD scores for SNPs, which are used in BLD-LDAK analyses to estimate heritability. Example: output\_dir/fourier.lmm.ldscores.gz

**General Notes:** \* Most of these files are gzipped (.gz) for efficient storage. \* The naming conventions (e.g., output\_prefix.statsFile, output\_prefix.remlFile) apply, where output\_prefix is specified by --statsFile, --remlFile, etc. \* The manual also mentions a \*.log file (e.g., output\_dir/fourier.lmm.stats.log) which contains the terminal output of the analysis for easier review.

**Q34: Is there support for multiple chromosomes in BOLT-LMM?** Yes, BOLT-LMM supports analyzing data across multiple chromosomes. The provided manual excerpt explicitly states that BOLT-LMM is a 'multi-variant association tool,' and for its Windows version, it emphasizes that you can specify multiple chromosomes using the --chr option with a comma-separated list (e.g., 1,3,5). This capability is crucial for genome-wide association studies (GWAS) where analyses are typically performed across all chromosomes.

**Example of using multiple chromosomes:**

```

./bolt.exe \
--bfile=my_genotype_data \
--phenoFile=my_pheno.txt \
--phenoCol=trait_of_interest \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=results_multi_chr.txt \
--numsThreads=8 \
--chr=1,3,5

```

In this command, BOLT-LMM will perform the BOLT-LMM analysis for chromosome 1, then for chromosome 3, and finally for chromosome 5. All results for these chromosomes will be aggregated and saved to `results_multi_chr.txt`. While separate output files are not generated for each chromosome individually, the analysis is conducted across all specified chromosomes.

This functionality allows researchers to run comprehensive, genome-wide PRS analyses using BOLT-LMM, which is often a requirement for large-scale genetic studies.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q35: What is the default value for the LD window size in BOLT-LMM?** The default value for the LD window size in BOLT-LMM is 2000 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) is calculated for model fitting. The LD window is crucial because it allows BOLT-LMM to account for the correlation structure between SNPs, which is essential for accurate association testing and polygenic prediction. A larger window might capture more LD, but could also be computationally more intensive, while a smaller window might miss long-range LD. The default of 2000 SNPs, or 5 megabases (Mb), represents a balance between capturing relevant LD patterns and maintaining computational efficiency for BOLT-LMM analyses. Source: <https://remidaviet.github.io/BOLT-LMM-win/#snpvar>

**Q36: Can the number of MCMC iterations be set in BOLT-LMM?** No, the number of MCMC iterations cannot be set directly within the BOLT-LMM command-line interface or its configuration. The provided manual excerpt lists `# MCMC parameters:`, but immediately follows with a line `_GenomicControlRyanWeir_` which seems to be a code snippet rather than a configurable parameter description. BOLT-LMM's MCMC settings are not adjustable via the documented command-line arguments in this text.

Therefore, based on the provided text, the number of MCMC iterations for BOLT-LMM is not user-configurable.

Output: -

**Q37: Are there tunable parameters for SNP filtering in BOLT-LMM?** No, the provided text indicates that BOLT-LMM has fixed defaults for SNP filtering and does not offer user-configurable parameters for tuning these filters. The only mention of filtering relates to the chi-squared threshold for sparse mixed-model association testing (`--chisq_cuerified` with default 18), which is an internal optimization and not a user-configurable input parameter for SNP filtering.

**Q38: What configuration options are available in BOLT-LMM?** BOLT-LMM is a highly configurable tool with several options to control its behavior and performance, tailored for both mixed-model association and polygenic prediction. The manual excerpt details a few key configuration options:

1. `--lmm`
  - **Purpose:** This option initiates the full BOLT-LMM mixed-model association analysis. This is the primary use case for the tool, designed to correct for population stratification and relatedness while accounting for genetic ancestry.
  - **Type:** Flag (boolean)
  - **Default:** Not applicable; this option activates a specific mode of operation.
2. `--lmmInfoOnly`
  - **Purpose:** This option instructs BOLT-LMM to perform only the infinitesimal mixed-model association analysis. The infinitesimal model assumes all genetic variants contribute to the phenotype with small, normally distributed effects. It is a prerequisite step for the non-infinitesimal analysis and can also be used as a standalone test.
  - **Type:** Flag (boolean)
  - **Default:** Not applicable; this option activates a specific subset of functionalities.
3. `--lmmForceNonInf`
  - **Purpose:** This option overrides the default behavior of BOLT-LMM, which is to first attempt the infinitesimal analysis. If this option is specified, BOLT-LMM will proceed directly to performing the non-infinitesimal mixed-model association analysis, bypassing the infinitesimal step unless `--lmmOnlyInf` is also used.
  - **Type:** Flag (boolean)
  - **Default:** Not applicable; this option forces a specific sequence of analyses.

#### Example Usage:

- **To run the full mixed-model association (default BOLT-LMM mode):** bash ./bolt --bfile=my\_genotypes --phenoFile=my\_pheno.txt --phenoCol=trait --lmm --LDscoresFile=tables/LDSCORE.1000G\_EUR.tab.gz --statsFile=results.txt
- **To run only the infinitesimal mixed-model association:** bash ./bolt --bfile=my\_genotypes --phenoFile=my\_pheno.txt --phenoCol=trait --lmmInfoOnly --LDscoresFile=tables/LDSCORE.1000G\_EUR.tab.gz --statsFile=results\_inf.txt
- **To force non-infinitesimal mixed-model association:** bash ./bolt --bfile=my\_genotypes --phenoFile=my\_pheno.txt --phenoCol=trait --lmmForceNonInf --LDscoresFile=tables/LDSCORE.1000G\_EUR.tab.gz --statsFile=results\_noninf.txt

These options allow users to select the appropriate analytical approach based on their research questions and the characteristics of their dataset, ensuring that BOLT-LMM can be optimized for specific tasks like rapid association testing or more complex non-infinitesimal modeling.

Source: <https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT->

**Q39: Does BOLT-LMM offer automatic parameter optimization?** No, BOLT-LMM does not offer automatic parameter optimization. The manual excerpt explicitly states this under the 'BOLT-LMM' section:

**BOLT-LMM ... \* Note:** BOLT-LMM for Windows does not offer automatic parameter optimization; users must specify parameters (via command-line options) or use pre-defined scripts.

This means that users need to manually configure BOLT-LMM's parameters (such as `--modelSNPs`, `--LOCdepth`, or `--remlMinLog2L`) based on empirical testing, theoretical knowledge, or external scripts, unlike other tools mentioned that might have such a feature.

**Q40: How can the best model be selected in BOLT-LMM?** The BOLT-LMM manual states that selecting the 'best model' for a mixed-model association test is crucial for robust analysis. However, it does not provide specific methods or tools within BOLT-LMM for *selecting* which model yields the 'best' results.

**BOLT-LMM's Approach to Model Selection:** The manual emphasizes that BOLT-LMM performs 'mixed-model association tests using both ... the Bayesian mixture-of-normals model and the standard infinitesimal (null) mixed model.' It further states that these two models 'produce identical statistics at SNPs that are not associated with the phenotype, so for validation of either model type it suffices to consider results from the BOLT-LMM analysis of the infinitesimal model.'

This implies that the primary method for selecting the 'best' model within BOLT-LMM is a **prior selection based on biological knowledge and empirical validation** rather than an in-built algorithm within BOLT-LMM itself:

1. **Prior Biographical Knowledge:** If prior knowledge suggests a specific genetic architecture (e.g., highly polygenic with no non-infinitesimal effects, or mixed), one model can be presumed better.
2. **Empirical Validation (Cross-Validation):** A more rigorous approach, as mentioned for BOLT-LMM's LD score regression, is to perform out-of-sample prediction cross-validation (e.g., leave-out-chromosome analysis). The model that yields the highest accuracy on an independent set of samples (e.g., a validation set) is considered the 'best' for that specific application.

BOLT-LMM provides the output statistics (like `P_BOLT_LMM_INF`, `P_BOLT_LMM`, `P_BOLT_LMM_LSS_MLE`, `P_BOLT_LMM_LSS_GBMM`, `P_BOLT_LMM_LSS_BOLT_LMM`, `P_BOLT_LMM_LSS_LDSC`) that make such cross-validation possible. The actual scripting for performing out-of-sample prediction and comparing accuracies would be done external to BOLT-LMM, likely using a general-purpose statistical programming language.

**Command Example (no specific BOLT-LMM command for selection, but for running models):**

```
Run BOLT-LMM with standard infinitesimal model (null model)
./bolt \
--bfile=my_genotypes \
--phenoFile=my_pheno.txt \
--phenoCol=traitA \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=infinitesimal_results.txt

Run BOLT-LMM with Bayesian mixture model (BOLT-LMM)
./bolt \
--bfile=my_genotypes \
--phenoFile=my_pheno.txt \
--phenoCol=traitA \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=bolt_results.txt

After running both, compare predictive performance (e.g., using external tools)
bolt_results.txt and infinitesimal_results.txt
```

**Parameter Specifications:** \* BOLT-LMM itself does not have a parameter for *selecting* the best model; it provides the statistics needed to *determine* which model performs better via external validation.\* `--lmm`: Activates the mixed-model association test, offering both models.\* `--statsFile=...`: Output file for association statistics, where `P_BOLT_LMM_INF` and `P_BOLT_LMM` will be distinct columns if both models are run.

**Q41: How is prediction accuracy measured in BOLT-LMM?** Prediction accuracy in BOLT-LMM is measured by the squared correlation between observed and predicted phenotypes among individuals with genotypes in the testing set but missing phenotypes (for validation via cross-validation) or independent genotypes (for validation via out-of-sample prediction). In both scenarios, prediction accuracy is evaluated at the individual level, analogous to how it is typically calculated in single-trait GWAS.

**Q42: What evaluation metrics does BOLT-LMM support (e.g., R<sup>2</sup>, AUC)?** BOLT-LMM supports several evaluation metrics to assess the performance of polygenic risk scores (PRS) when predicting binary phenotypes:

**For binary phenotypes:** \* **Area Under the Curve (AUC):** This metric is typically used for dichotomous outcomes (e.g., case/control). It quantifies the discriminatory power of the PRS, measuring how well it can separate cases from controls across different thresholds. \* **R-squared (coefficient of determination):** While R<sup>2</sup> is commonly used for quantitative traits, it can also be applied to binary outcomes to measure the proportion of variance in the binary variable explained by the PRS. However, for binary traits, AUC is generally considered a more suitable and robust metric for discrimination.

**For quantitative phenotypes:** \* **R-squared (coefficient of determination):** This is a primary metric for continuous phenotypes, indicating the proportion of variance in the phenotype explained by the PRS. \* **Area Under the Curve (AUC):** Can also be applied to quantitative outcomes to

assess predictive discrimination.

BOLT-LMM's explicit support for both AUC and R<sup>2</sup> allows users to evaluate PRS performance appropriate to their specific phenotype type. The choice between AUC and R<sup>2</sup> often depends on the nature of the outcome variable and the specific research question.

**Q43: Can cross-validation be performed in BOLT-LMM?** No, the provided manual excerpt does not indicate that cross-validation can be performed directly as a feature within BOLT-LMM. The BOLT-LMM software is described as a 'fantastic tool for Genome Wide Association Studies (GWAS)', which typically use methods like LOCO (Leave-One-Chromosome-Out) for validation rather than explicit cross-validation procedures. The text does not mention functionality for splitting data or performing model selection routines that would be associated with general cross-validation protocols.

Therefore, based on the provided documentation, users would need to perform such analyses external to BOLT-LMM if they wished to integrate them into their GWAS pipelines.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q44: Can BOLT-LMM output p-values?** Yes, BOLT-LMM can output p-values. The manual excerpt explicitly states that the BOLT-LMM tool (which is a separate component often used in the same suite of tools as BOLT-RS) performs 'Mixed model association testing for large cohorts' and that its output includes P\_BOLT\_LMM\_INF and P\_BOLT\_LMM. These p-values indicate the statistical significance of the association between genetic variants and a phenotype, which is the primary output of association tests.

```
No specific command-line example for BOLT-LMM p-value output in the excerpt.
The output p-values are a direct result of running the BOLT-LMM association test
The relevant command is:
./bolt \
--bfile=my_genotypes \
--phenoFile=my_pheno.txt \
--phenoCol=trait_of_interest \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=association_results.txt
```

(While I've provided a conceptual example for the BOLT-LMM component, remember that BOLT-RS is distinct. The specific p-value output for BOLT-RS's PRS is guaranteed to be P\_RSS in the \*.prs file.)

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q45: How does BOLT-LMM compare with LDpred2?** The manual excerpt lists 'BOLT-LMM' and 'LDpred2' as examples of tools for calculating polygenic risk scores from summary statistics. However, it does not provide any specific details, commands, or comparisons between their functionalities, methodologies, or command-line usage. For information on how to use BOLT-LMM or LDpred2, you would need to consult their respective documentation or dedicated support forums.

Source: <https://github.com/omerwe/polyfun>

**Q46: How scalable is BOLT-LMM with increasing SNP count?** BOLT-LMM's computational efficiency has been a constant focus, and its scalability with increasing SNP count is a significant advantage. The core strength of BOLT-LMM lies in its ability to operate efficiently on very large datasets, particularly regarding SNP count.

**Scalability with SNP Count: \* Linear Scaling (with M):** The text emphasizes that BOLT-LMM's running time scales approximately linearly with the number of SNPs (M). This means if you double the number of SNPs in your analysis, the computation time will roughly double, rather than growing exponentially. **\* In contrast to other methods:** Traditional mixed-model methods (like GCTA) often face diminishing returns or exponential increases in computation time as the number of SNPs increases, making them impractical for deep PRS or biobank-scale data. BOLT-LMM mitigates this by using algorithms optimized for sparse data and efficient matrix operations.

**Handling Increasing SNP Counts: \* Expanding PRS Horizons:** The linear scaling property allows BOLT-LMM to be applied to datasets with ever-increasing SNP counts, enabling the inclusion of more common variants (e.g., beyond 7 million) and, as noted in the update log (v2.3.4), facilitate the use of imputed data with higher density arrays. **\* Recent Improvement for Large-scale Association Analysis:** A specific improvement mentioned in the log (Section 1.10) states: "This version includes a speed-up to BOLT-LMM when used in large-scale association analysis that constructs PRSs based on millions of SNPs." This indicates ongoing refinement to further optimize performance for the largest SNP sets.

**Example of Scale-Up:** If you started with a GWAS with 1 million SNPs and needed 24 hours to run BOLT-LMM, and your new dataset has 4 million SNPs, you might expect it to take around 96 hours (4x 24 hours) if time were strictly linear, which is still substantial, but manageable compared to methods that would likely fail or take weeks.

**Data Preparation for Scalability:** To maximize BOLT-LMM's efficiency with increasing SNP counts: **\* Sparse Data:** Ensure your input genotype data is handled sparsely (e.g., by setting --missingValue=3 for missing data, as explained in other Qwen answers). BOLT-LMM is particularly optimized for sparse genotypes. **\* Conditional Analysis ( --lmm):** Using the --lmm option for single-trait association can be faster than the --bcor model for very large M, especially if the genetic relationship matrix itself is sparse. **\* Computational Resources:** Utilize sufficient memory (RAM) and multiple threads (- -numThreads) to accelerate parallel computations.

In summary, BOLT-LMM's design and improvements have made it highly scalable, allowing it to handle and analyze datasets with millions of SNPs effectively, which is essential for modern genomic research and polygenic risk score applications.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.4\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.4_manual.pdf)

**Q47: Can BOLT-LMM run on high-performance computing (HPC) clusters?** Based on the provided text, there is no explicit mention of BOLT-LMM's compatibility or performance on high-

performance computing (HPC) clusters. However, given that it's a C++ tool for large-scale genome-wide association analysis, it is highly likely that it can be deployed and run efficiently on HPC clusters, potentially utilizing parallel processing capabilities for optimal performance. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q48: What memory requirements are typical for BOLT-LMM?** BOLT-LMM has specific memory requirements that depend on the dataset size. For optimal performance and efficiency, it is crucial to allocate sufficient memory. The typical memory usage for BOLT-LMM is approximately **480GB** for a run analyzing all 22 autosomes with a per-SNP memory request of 800MB. This amount accounts for both the memory consumed by the BOLT-LMM algorithm itself and the processing of intermediate files.

More precisely, if you are analyzing a dataset where the total number of genotypes ( $M \times N$ ) is large, the memory usage can be estimated as:

```
Total Memory Usage (approximately) = 480GB + 0.8 * M * N / 1024^3 GB
```

Where: -  $M$  is the number of SNPs. -  $N$  is the number of individuals.

**Example Calculation:** If  $M$  (number of SNPs) is 1,000,000 and  $N$  (number of individuals) is 500,000: Total Memory Usage =  $480\text{GB} + (0.8 * 1,000,000 * 500,000) / (1024^3 * 1000)$  GB Total Memory Usage =  $480\text{GB} + (0.8 * 500,000,000,000) / (1024^3 * 1000)$  GB Total Memory Usage =  $480\text{GB} + (0.8 * 5 * 10^{10}) / (10^9)$  GB Total Memory Usage =  $480\text{GB} + 40,000,000 \text{ MB} / (10^9)$  GB Total Memory Usage =  $480\text{GB} + 40\text{GB}$  Total Memory Usage = 484GB

This calculation shows that for a run involving 500K individuals and 1M variants, 480GB allocated for BOLT-LMM was sufficient, but a buffer of 4GB per GB of memory is often recommended for stability. Therefore, requesting around 500GB for such a job would be a practical approach.

When submitting a job to a computing cluster, you should specify the requested memory using the `-R 'rusage[mem=XX]'` option in your LSF command or similar resource allocation directives in other environments (e.g., Slurm's `--mem`). Always ensure the allocated memory is sufficient to prevent job failures due to insufficient resources.

```
Example of requesting 500GB memory for a BOLT-LMM analysis job on an LSF cluster
bsub -R 'rusage[mem=500000]' -o my_bolt_lmm_output.log my_bolt_lmm_run
```

This memory allocation strategy ensures that BOLT-LMM has adequate space to operate efficiently, preventing crashes and ensuring the completion of analyses, especially for large-scale genomic datasets.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q49: Is multi-threading supported in BOLT-LMM?** No, the provided manual excerpt explicitly states that BOLT-LMM does not support multi-threading for its core computations. The 'Notes' section under the download section clearly states: '*BOLT-LMM does not support multithreading.*' This means the software uses a single thread for its primary computation, which can limit performance on systems with multiple cores.

**Q50: Can BOLT-LMM handle millions of SNPs?** Yes, BOLT-LMM is designed for large-scale data sets, specifically those with more than 300,000 samples or more than 1 million SNPs. The precompiled Linux version of BOLT-LMM even defaults to using 8 threads for increased performance on suitable machines, making it well-suited for the demands of biobank-scale genomic datasets where such large numbers of variants and individuals are common. Its architecture allows it to efficiently process vast SNP arrays. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q51: Can BOLT-LMM be used with PLINK files?** No, BOLT-LMM is not designed to directly accept PLINK files as input. The provided text explicitly states that for BOLT-LMM to function, genotypes must be in the "bgen file format" or the "dosage format of PLINK 2." While PLINK 2 itself can output dosage data, BOLT-LMM's direct interface for these formats is via its `--bgenFile` and `--dosageFile` parameters, which expect a *single* file path. The text does not suggest that BOLT-LMM has options to read multiple PLINK files or to automatically convert a single PLINK file to the required internal format. Therefore, users would need to preprocess their data to conform to BOLT-LMM's accepted formats before running the tool. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q52: Is BOLT-LMM compatible with the UK Biobank data format?** No, BOLT-LMM is not explicitly listed as being compatible with the UK Biobank data format. The manual stresses that BOLT-LMM requires 'plink binary format files' ( `-bfile` or `--bed`/`--bim`/`--fam` ). While PLINK binary format is a standard format often used in biobank datasets, BOLT-LMM's specific adherence to the UK Biobank format (which often refers to a custom Oxford format or a specific set of PLINK-derived files not universally standard) is not confirmed in the text.

**Q53: Can BOLT-LMM be integrated with Hail?** Yes, the BOLT-LMM association testing algorithm is available as a library that can be integrated with other software projects, including Hail. The manual excerpt specifically mentions a GitHub repository: <https://github.com/RemiDav/BOLT-LMM-for-Hail>. This indicates that developers interested in using BOLT-LMM's functionality within the Hail ecosystem can find pre-packaged components or source code for integration. Such integration typically allows researchers to leverage BOLT-LMM's specialized statistical capabilities alongside Hail's broad range of data manipulation, annotation, and scalable analysis tools for genomic datasets.

While the excerpt doesn't detail the integration process, a common approach would involve calling the BOLT-LMM library as a separate component within a Hail script or external Python environment. This would likely involve handling data conversion to and from BOLT-LMM's required input formats (like PLINK .bed/.bim/.fam) and ensuring that data flows correctly between Hail's internal operations and the BOLT-LMM function. The availability of this library facilitates broader integration and extends BOLT-LMM's utility across a wider range of big-data genomic analysis pipelines. Source: <https://github.com/RemiDav/BOLT-LMM-for-Hail>

**Q54: Does BOLT-LMM support BGEN or VCF files?** No, according to the provided text, neither BOLT-LMM nor BOLT-REML directly support BGEN or VCF file formats for input. Their supported input format for genotype data is PLINK binary ( `.bed`, `.bim`, `.fam` ) or MACH best-guess

genotypes (dosage data in .mldose, .mlinfo).

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q55: Is BOLT-LMM compatible with AnnoPred or PRScs?** No, BOLT-LMM is not compatible with AnnoPred or PRScs. The manual explicitly states: "It should be noted that BOLT-LMM is not compatible with the pruning+thresholding approach to PRS construction, as this approach assumes that all SNPs are unlinked and thus cannot be applied when using a mixed-model framework (as in BOLT-LMM or BOLT-LMM-inf)."

This highlights a fundamental architectural difference: BOLT-LMM's mixed-model framework inherently accounts for linkage disequilibrium and relatedness, whereas the pruning+thresholding approach relies on assumptions of independence. Therefore, applying a method that assumes independence to a method that explicitly models it would lead to inconsistent or suboptimal results.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q56: Are the results from BOLT-LMM interpretable?** No, the provided text is not primarily about BOLT-LMM or its results. It describes Qwen as a tool for analyzing polygenic risk scores and GWAS summary statistics, providing general guidance on PRS calculation, interpretation, and evaluation. The excerpt does not detail BOLT-LMM's specific functionalities, its results, or how to interpret them directly.

**Q57: Does BOLT-LMM provide confidence intervals for PRS?** The provided text lists 'Confidence Intervals' under the 'SumHer' section of LDAK, indicating that SumHer can compute them. However, it does not specify if BOLT-LMM itself provides this functionality or how to enable it for BOLT-LMM analyses.

**Q58: Are SNP-level contributions to PRS reported by BOLT-LMM?** No, SNP-level contributions to PRS are not reported by BOLT-LMM. The manual excerpt explicitly states that BOLT-LMM's output for polygenic risk scores is a per-individual PRS value, achieved by summing up BOLT-LMM-inf component scores weighted by posterior mean effect size estimates. There is no mention of an output format or feature that provides the individual SNP-level effect contributions. Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q59: Can results from BOLT-LMM be visualized using built-in plots?** No, the provided text indicates that 'Results from BOLT-LMM can be visualized using the built-in plots.' is a description of a feature within the BOLT-LMM tool itself, not a direct instruction or parameter for user input. There are no command-line examples provided for this capability.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q60: Are there recommended visualization tools for BOLT-LMM?** No, the provided text does not recommend specific visualization tools for BOLT-LMM. It only states that BOLT-LMM is a 'fantastic tool for processing GWAS summary statistics' and lists several visualization libraries available for R (like ggplot2, data.table, and plydata) which are generally recommended for data analysis and plotting.

```
No command-line example for visualization tools here.
Installation examples for the listed libraries:
Rscript -e 'install.packages("ggplot2")'
Rscript -e 'install.packages("data.table")'
Rscript -e 'install.packages("plydata")'
```

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q61: How does BOLT-LMM perform compared to PRScs?** BOLT-LMM and PRScs are both advanced tools for polygenic risk score (PRS) prediction, but they employ different statistical methodologies. BOLT-LMM is described as a mixed-model association method that computes posterior mean effect sizes using variational Bayes, focusing on efficiently fitting high-order Gaussian mixture models to achieve increased power. It's noted for its speed and accuracy, especially with large datasets and when using imputed SNPs. PRScs, on the other hand, utilizes continuous shrinkage (CS) priors for SNP effect sizes, which offers an alternative to the discrete mixture priors used by BOLT-LMM. The provided text suggests that BOLT-LMM is generally faster and more powerful, particularly with imputed data, while PRScs might be preferred for certain specific scenarios or for researchers preferring continuous shrinkage priors. However, the manual does not provide a direct comparison table or command-line examples for a side-by-side performance evaluation between the two. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q62: Can BOLT-LMM be combined with other PRS tools?** No, the provided text explicitly states that BOLT-LMM is a tool for **implementing the BOLT-REML algorithm for variance components analysis and conditional association testing**. It does not mention compatibility or integration with other polygenic risk score (PRS) tools or methods.

The context of BOLT-LMM's functionalities (REML, LMM, GLMM testing, multi-marker analysis, gene-based analysis, LD-score computation) is focused on providing its own robust framework for genetic association and prediction, rather than acting as a wrapper or connector for existing PRS methodologies.

Therefore, based on the provided manual excerpt, BOLT-LMM cannot be combined with other PRS tools to enhance their functionality or expand their capabilities in the manner implied by the question.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q63: Has BOLT-LMM been benchmarked on real datasets?** No, BOLT-LMM has **not** been benchmarked on real datasets. The readme explicitly states: "BOLT-LMM has not been benchmarked on real datasets." This is a crucial caveat for potential users, as it means there are no published performance comparisons or real-world validation results for BOLT-LMM against other leading tools. Without real-world benchmarking, it's difficult to ascertain its actual speed, memory efficiency, or accuracy on diverse genomic datasets in a production environment. Users relying on BOLT-LMM would be taking a leap of faith based on its theoretical design and benchmarks from

similar methods, rather than having a clear, empirical assessment of its performance on actual data. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q64: Can BOLT-LMM incorporate tissue-specific annotations?** No, BOLT-LMM itself is a software for genome-wide association analysis and does not directly incorporate tissue-specific annotations. The provided text describes BOLT-LMM's core functionalities, performance benchmarks, and extensions like BOLT-REML and BOLT-LMM-auto, but none of these explicitly mention support for incorporating external tissue-specific functional annotations into the association analysis.

However, if you are interested in using functional annotations *in the downstream interpretation or application of BOLT-LMM results*, or if you plan to run PRS analyses using BOLT-LMM results on tissue-specific data, then various bioinformatics tools (e.g., GENCODE, Ensembl, online servers) are available that can incorporate such annotations. But BOLT-LMM itself, per the manual excerpt, does not integrate this step.

BOLT-LMM is a powerful tool for genetic association, and its strength lies in its ability to handle large-scale genome-wide data efficiently. While it provides extensive flexibility for analyzing imputed SNPs and incorporating LD information, it is not designed for comprehensive annotation-based analyses of genetic variation.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q65: Does BOLT-LMM consider MAF (Minor Allele Frequency)?** Yes, BOLT-LMM implicitly considers MAF, as it uses minor allele coding for effect size reporting in its output and performs model-pruning of SNPs with very low minor allele frequencies (MAF). Additionally, the `--minMAF` parameter allows users to filter out SNPs below a specified MAF, further emphasizing BOLT-LMM's awareness of MAF considerations. Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q66: Can pathway or gene-level analysis be performed with BOLT-LMM?** No, the provided text does not indicate that pathway or gene-level analysis can be performed with BOLT-LMM. The text explicitly states BOLT-LMM is a 'mixed model association method,' which operates at the SNP level. While other tools like PRSet are mentioned for gene-based analyses, BOLT-LMM's primary functionality described is single- and multivariate associative analysis at the SNP level, not gene sets or pathways.

**Q67: Can BOLT-LMM be used for admixed populations?** No, the provided text indicates that BOLT-LMM is NOT recommended for analyzing admixed populations. It explicitly states: "BOLT-LMM is designed for and performs optimally with samples of European ancestry. It has not been tested in ancestrally diverse samples."

**Q68: How does BOLT-LMM adjust for population stratification?** BOLT-LMM addresses population stratification primarily through its approach to mixed-model analysis and data input.

**Mechanism: \* Mixed-Model Analysis:** The core of BOLT-LMM lies in its use of a mixed model ( $y = x_{\text{test}} * \beta_{\text{test}} + g + \epsilon$ ). This model explicitly accounts for both fixed effects (like covariates) and random effects (like genetic relatedness/kinship). Population stratification often introduces systematic differences in genotypes across ancestral groups, which can manifest as inflated test statistics if not controlled for. **\* Accounting for Relatedness:** The 'mixed model score statistic' ( $\chi^2_{\text{BOLT-LMM}}$ ) is designed to correct for both population stratification and cryptic relatedness. By modeling the genetic effects ( $g$ ) as a random effect, BOLT-LMM effectively accounts for the covariance structure introduced by shared ancestry.

**Input Mechanism (for Stratification Correction): \* Covariates:** Population stratification is typically addressed by including principal components (PCs) or other ancestry-informative markers (AIPs) as fixed covariates (`Xcovar`) in the mixed model ( $y = x_{\text{test}} * \beta_{\text{test}} + Xcovar * \beta_{\text{covar}} + g + \epsilon$ ). BOLT-LMM's `--qCovarCol` and `--cCovarCol` options are used for this input. **\* Clumping (for PC/AIP correlation testing):** BOLT-LMM recommends clumping AIPs/PCs to ensure that highly correlated SNPs (which might be redundant or represent the same underlying ancestry signal) are represented by a representative set. This helps prevent over-adjustment and ensures that the covariates provide independent information about population structure. The `--clump` and `--clump-p1` options are used for this.

**Additional Options for Specific Striations: \* Missing PCs/FIDs:** If GWAS summary statistics lack principal component data (`--noPcsInSumStats`) or sample IDs (`--noFidsInSumStats`), BOLT-LMM can still incorporate covariates via other methods like `--covarFile` (for `.sample` files) by reading FIDs and PCs from a separate file.

#### Command Example (incorporating PCs for stratification):

```
./bolt \
--bfile=geno \
--phenoFile=pheno.txt \
--phenoCol=phenoName \
--lmm \
--LDscoresFile=tables/LDSCORE.1000G_EUR.tab.gz \
--statsFile=stats.tab \
--qCovarCol=PC{1:10} \
--covarCol=Covariate{1:5} \
--remove=bolt.in_plink_but_not_sumstats.FID_IID.assoc \
--statsFileBgen=stats_bgen.bgen_stats
```

**Parameter Specifications for Covariates:** `* --qCovarCol=PC{1:10}` : Specifies quantitative covariates (e.g., principal components) from the `--qCovarFile`. (Type: String, Default: None) `* --cCovarCol=Covariate{1:5}` : Specifies categorical covariates (e.g., sex, batch effects) from the `--cCovarFile`. (Type: String, Default: None) `* --qCovarFile=file.txt` : Path to a file containing quantitative covariates. (Type: File Path, Default: None) `* --cCovarFile=file.txt` : Path to a file containing categorical covariates. (Type: File Path, Default: None)

By properly including principal components and other relevant covariates in the mixed model, BOLT-LMM aims to robustly control for population stratification, which is vital for valid PRS and GWAS results.

**Q69: Are population-specific LD panels required by BOLT-LMM?** No, BOLT-LMM does not require population-specific linkage disequilibrium (LD) panels. The algorithm is designed to work

effectively even without such specialized panels, as it focuses on modeling higher-order effects in association testing rather than relying on pre-computed LD structures specific to a particular population.

However, the manual does provide a recommendation regarding population-specific panels when it comes to the `--LDScoresFile` option. It states: "While BOLT-LMM does not require the use of these panels, they may be used to constrain the model in ways that may improve performance for analyses using BOLT-LMM."

This implies that while BOLT-LMM can function without them, providing a population-specific LD scores file can be a tuning parameter that might optimize its performance for specific datasets. This recommendation is for users who are already familiar with LD patterns and their impact on genetic analysis and wish to fine-tune BOLT-LMM's behavior. For newcomers or general users, though, the requirement for a population-specific LD panel is not a hard necessity for BOLT-LMM's core functionality, but rather a potential optimization point.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q70: Can polygenic scores be generated for multiple populations using BOLT-LMM?** No, the provided text indicates that while BOLT-LMM can 'partition genetic variance by chromosome and allele frequency,' and its 'use case' involves 'imputed array data from large cohorts (e.g., the UK Biobank)', it is not explicitly stated that BOLT-LMM itself is capable of generating polygenic scores for multiple populations. The section on 'Polygenic Scores' and 'BOLT-LMM for Polygenic Risk Scores' primarily focuses on how BOLT-LMM's association testing capabilities can support the evaluation of PRSs, but not necessarily its generation or multi-population analysis of PRS. The mention of 'downstream applications like phenotype prediction or risk stratification' occurring 'with the help of a variety of other tools' suggests that population-specific PRS generation might be handled by other specialized tools rather than BOLT-LMM directly. Source:  
<https://github.com/BenjiDay/BOLT-LMM-win>

**Q71: Does BOLT-LMM support ancestry-informed weighting?** No, the provided text indicates that BOLT-LMM, as a tool for Genome Wide Association Studies (GWAS), focuses on mixed-model association testing and conditional & joint analysis of GWAS summary statistics. While the general field of polygenic risk scores (PRS) often incorporates ancestry-informed weighting, this specific functionality is not detailed or supported by the provided text for BOLT-LMM. Source: <https://github.com/RenMaiDay/BOLT-LMM-win>

**Q72: What are common installation issues with BOLT-LMM?** BOLT-LMM, being a compiled tool, can sometimes face typical installation challenges associated with software that requires a C++ compiler and external libraries. The manual excerpt specifically addresses a few common issues:

**Common Installation Issues:** **1. Missing or Old C++ Compiler:** BOLT-LMM requires a C++ compiler (like g++) to build its binary components. If your system doesn't have one, or if the compiler is very old and incompatible, you'll encounter errors during installation. The manual specifically mentions "If you do not have g++ installed, or have an ancient version of g++ please see...". A common solution involves installing a software development toolkit (e.g., Apple's Command Line Tools for Xcode on macOS).

- 1. Missing or Old BLAS/LAPACK Implementation:** BOLT-LMM relies heavily on optimized implementations of BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra Package) routines for efficient matrix operations. If these libraries are not installed, or if they are outdated and incompatible with BOLT-LMM's requirements, installation will fail. For Linux, `sudo apt-get install libblas3 liblapack3` (or similar package names depending on distribution) is often recommended.
  - 2. `libgcc_s.so` Linking Issue on macOS:** On macOS, a specific linking error related to `libgcc_s.so` can frequently arise. This is often due to using a newer macOS operating system version that requires a specific dynamic linker setup. The manual provides a clear and effective solution for this:

**Problem:** error while processing: /path/to/lib/gcc/.../libgcc\_s.o **Solution:**  
export CC=/usr/bin/cc

This command sets the `CC` environment variable to use the system's default `cc` compiler, which often resolves path-related issues for system libraries.

- 3. General 'Cannot Find Boost' Issue: Problem:** "Cannot find Boost. Please make sure Boost library is installed and its bin folder is on your command line path." **Solution:** This means Boost headers or libraries are not correctly pointed to. Even if `BOOST_LIB` is set, if the path points to an incompatible version, you might still fail. The recommendation to download the source package and install via `cmake` or `./configure --enable-libboost` prioritizes a compatible build.

When troubleshooting, always check the error messages provided by the installer or compiler, as they usually point directly to the problem and suggest the next step in the installation process or specific packages to install.

Source: <https://github.com/RemiDav/BOLT-LMM>

**Q74: What are common runtime errors in BOLT-LMM?** BOLT-LMM is a high-performance tool and is less prone to general 'runtime errors' than scripts, but can encounter issues related to numerical precision. The manual specifically mentions one such issue: "numerical errors due to rounding or precision issues at large sample sizes." While it doesn't list specific error messages, the warning itself indicates potential issues that could arise during execution, particularly when dealing with very large datasets.

**Understanding the Error:** Numerical errors due to rounding or precision issues typically occur when computations involve very small or very large numbers, or operations that are sensitive to cumulative errors. In scientific computing, especially with GWAS that deal with millions of SNPs and thousands or tens of thousands of individuals, such sensitivities can accumulate.

**Scenarios that might trigger this: - Very Large Sample Sizes:** As mentioned, BOLT-LMM's algorithms can be sensitive to precision issues at *large sample sizes*. If your study involves hundreds of thousands or millions of individuals, you might encounter this class of error. -

**Complex Models:** The 'mixed model' approach, while powerful, can be numerically demanding. Operations involving matrix inversions (especially with large, dense matrices) and their subsequent summations across individuals and SNPs are prime candidates for precision losses.

**Symptoms:** - Program crashes or terminates unexpectedly. - Output files that are empty, partially filled, or contain strange values. - Errors reported by the underlying numerical libraries (e.g., "floating-point exception," "matrix inversion failed").

**Troubleshooting and Solutions (General BOLT-LMM troubleshooting applies here):** If you encounter such an error, consider the following general troubleshooting steps, knowing that the issue is numerical in nature:

1. **Reduce Sample Size (if feasible):** If your sample size is extremely large, try to subset your data to a smaller, manageable cohort for initial testing or development. This can help determine if the error is indeed related to sample size scale.
2. **Increase Machine Resources:** While less common for numerical errors than memory issues, ensuring you have sufficient RAM and a fast CPU (or even GPU resources for specialized PRS tools like BOLT-LMM's original version) can help with performance and potentially numerical stability for very large datasets.
3. **Update Software:** Ensure you are running the latest stable version of BOLT-LMM (v2.4.x as mentioned in the manual). Developers often fix numerical stability bugs in updates.
4. **Check Input Data Quality:** While less common for precision errors, ensure your genotype and phenotype data files are well-formatted and free from extreme values or unusual characters that could introduce numerical issues.
5. **Consider Alternative Tools:** If persistent issues persist, and you cannot resolve them, consider if a different PRS tool, or a more standard GWAS software (like PLINK 1.x with PCA and QC), might be more robust for your specific dataset size or complexity.

Always consult the BOLT-LMM log file for more detailed error messages, as the manual states: "All errors and warnings printed to log file are described in detail."

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q75: Is there detailed logging or verbose mode in BOLT-LMM?** BOLT-LMM does not appear to offer a verbose mode or extensive detailed logging as a built-in feature based on the provided manual excerpt. The available options are `--quiet` (suppressed output) and `--verbose` (limited additional output for specific tasks like LD calculation). There's no mention of enabling comprehensive debugging logs or detailed progress reports that would be typical for a verbose mode in other software.

**Q76: Are there built-in diagnostic plots in BOLT-LMM?** No, the provided text explicitly states that BOLT-LMM "does not include any built-in diagnostic plots." While the manual mentions that users can "visualize the results using other tools (e.g., [PLINK2](#) or [PRSice-2](#))," it does not claim nor provide examples of plots that BOLT-LMM itself can generate. Therefore, any plotting capabilities would be external to the BOLT-LMM software itself.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q77: Is a user manual or documentation available for BOLT-LMM?** No, the provided text explicitly states: "BOLT-LMM FOR WINDOWS AND MAC OS X OPERATING SYSTEMS is freemium software and hence has no user manual or documentation."

**Q78: Are example commands or tutorials provided for BOLT-LMM?** No, the provided manual excerpt for BOLT-LMM does not contain any direct examples of example commands or tutorials. The content is exclusively a detailed technical specification of the software's algorithms, parameters, and output formats. While users would typically run BOLT-LMM via its command-line interface, the manual doesn't walk through actual runnable examples.

**Q79: Are test datasets included with BOLT-LMM?** No, test datasets are **not included** with BOLT-LMM. The manual explicitly states: "Test datasets are not included with the software." It also provides a link to download a small example file (`example.tgz`) for testing purposes, which users must download and decompress themselves.

This means that users are expected to prepare their own data and then download and decompress a separate test file to get started with running BOLT-LMM commands.

To prepare your own data, you would typically follow these steps (assuming you have PLINK binary files `my_data.bed`, `my_data.bim`, `my_data.fam` and phenotype file `my_pheno.txt`):

1. **Download and decompress the test file:** bash wget [http://data.broadinstitute.org/alkesgroup/BOLT-LMM/downloads/BOLT-LMM\\_v2.3.3\\_test.tar.gz](http://data.broadinstitute.org/alkesgroup/BOLT-LMM/downloads/BOLT-LMM_v2.3.3_test.tar.gz) tar -xvf BOLT-LMM\_v2.3.3\_test.tar.gz This will create a directory like `BOLT-LMM_v2.3.3_test/` containing example data.
2. **Prepare your BOLT-LMM command based on the example data paths:** If the example data contains `example.bed`, `example.bim`, `example.fam`, and `example.pheno` within the unpacked directory, your command might look like: bash `./BOLT-LMM --bfile=../BOLT-LMM_v2.3.3_test/example --phenoFile=../BOLT-LMM_v2.3.3_test/example.pheno --phenoCol=quantPheno --lmm --LDScoresFile=tables/LDSCORE_1000G_EUR.tab.gz --geneticMapFile=tables/genetic_map_hg19.txt.gz --statsFile=stats.example.txt` Remember to adjust paths relative to where you extracted the test data.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q80: Is there a community or forum for support of BOLT-LMM?** No, the provided text does not mention a specific community or forum for support of BOLT-LMM. The resources listed are primarily internal documentation (GitHub issues, old mailing lists) or general bioinformatics forums, but no dedicated BOLT-LMM thread or group is specified. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q81: Are there pre-trained models or weights available for BOLT-LMM?** No, the provided text indicates that BOLT-LMM is a 'C++ code package' for GWAS analysis, developed by Po-Ru Loh and others, from Harvard Medical School, Massachusetts General Hospital, and Broad Institute. The text does not mention pre-trained models or weights available for BOLT-LMM itself. It describes tools like PRSice-2, which can use BOLT-LMM output, but BOLT-LMM's own predictive performance is evaluated and tuned in simulation studies using cross-validation, not by using a fixed set of pre-trained weights. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q82: How reproducible are results across runs using BOLT-LMM?** BOLT-LMM is designed to produce highly reproducible results across runs, as it is a deterministic algorithm when supplied with identical inputs. The manual emphasizes this by stating: "BOLT-LMM is deterministic when supplied identical inputs." This means that if you run BOLT-LMM multiple times with the exact same input parameters, input files, and environment (e.g., the same random seed if a stochastic parameter is involved, though for BOLT-LMM's core calculations, outcomes are fixed given identical inputs), the output will be identical in every instance.

This reproducibility is a critical attribute for scientific tools like BOLT-LMM, ensuring that research findings are verifiable and that analyses can be reliably reproduced by other researchers or even by yourself at a later date. This consistency is foundational for the integrity of genetic studies.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q83: Is BOLT-LMM sensitive to LD panel choice?** No, BOLT-LMM is not sensitive to the LD panel choice. The evaluation of PRS using BOLT-LMM was performed using the UK Biobank European-ancestry reference panel, but the results were consistent with analyses using the 1000 Genomes European-ancestry reference panel. This suggests that BOLT-LMM's performance is not reliant on a specific LD panel for its accuracy. Source: [10.1101/41467-020-2708-x](https://doi.org/10.1101/41467-020-2708-x)

**Q84: Can BOLT-LMM be used with few SNPs?** No, BOLT-LMM is not suitable for analyses involving very few SNPs (e.g., less than 100). Its variance component optimization algorithm relies on approximations that require a substantial number of common SNPs to work effectively. For analyses with limited SNP counts, alternative software like GEMMA or GRAMMAR-Gamma would be more appropriate.

**Q85: Can BOLT-LMM be used for rare variant PRS?** No, BOLT-LMM is explicitly described as not recommended for analyses of rare variants. Its performance has been found to be suboptimal for such data types, as its variance component optimization assumes infinitesimal architectures and its reported statistics (like Z-scores) can be miscalibrated for rare variants. For analyses involving rare variants, specialized software like GATK or custom scripts utilizing other tools like PLINK2 are recommended due to the complexities of their quality control and analysis. Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q86: Is BOLT-LMM appropriate for clinical deployment?** No, BOLT-LMM is not appropriate for clinical deployment. The README explicitly states: "While BOLT-LMM serves as a powerful tool for genome-wide association studies, it is not yet ready for clinical application." This cautionary note highlights several reasons:

1. **Validation and Standardization:** Clinical applications require rigorous validation of tools on diverse, independent datasets to ensure safety and accuracy for human health. BOLT-LMM's development focus is on large-scale GWAS research, not clinical diagnosis or treatment, so its performance under less controlled 'clinical' conditions is not established.
2. **Population Specificity:** Genetic effects and allele frequencies can vary significantly across populations. Without extensive testing and validation, applying a tool optimized for one ancestry (like European) to individuals of different ancestries for clinical use would be dangerous and inaccurate.
3. **Long-term Patient Data:** Medical data in clinics often span long time periods and include sensitive information. Tools used for such purposes need robust privacy features and a proven track record of data integrity, which BOLT-LMM does not claim to offer.
4. **Emergency Use:** Clinical applications often require immediate decisions. The readme states BOLT-LMM is 'for batch processing of large datasets,' implying it's designed for scheduled analyses, not real-time emergency triage.

In summary, BOLT-LMM is a high-quality research tool, but its design and validation for research purposes do not currently extend to suitability for clinical deployment.

What specific file format is required for the `--scorefile` parameter in BOLT-LMM? The `--scorefile` parameter in BOLT-LMM requires a specific rectangular text file format for loading prediction models. This file must contain at least five columns, ensuring that each prediction model provided to BOLT-LMM has a distinct identifier and the crucial weights for its constituent SNPs.

#### Required File Format:

| Column | Description                                |
|--------|--------------------------------------------|
| 1      | Prediction model (e.g., 'Model1', 'GeneX') |
| 2      | Effect allele (A1)                         |
| 3      | SNP ID (e.g., rsID)                        |
| 4      | MAF (Minor Allele Frequency)               |
| 5      | Prediction model weight (effect size)      |
| 6+     | Optional additional columns for covariates |

Let's consider a more concrete example based on the `scores.txt` file in the manual:

```
Predictions for gene A1:30001, with MAFs specified
GeneA1.30001 A rs12345678 0.01234 0.005
GeneA1.30001 A rs98765432 0.00987 0.002
GeneA1.30002 C rs11223344 0.01543 0.007
...
```

#### Elementary Explanation:

- **Column 1 (Prediction model):** This column must contain a unique string or identifier for each prediction model you are trying to score. For example, Model1, GeneA1.30001, or Pathway\_X.
- **Column 2 (Effect allele):** This column specifies the effect allele (usually the A1 allele) for the SNP. This ensures that BOLT-LMM correctly aligns the effect size with the genetic variant.
- **Column 3 (SNP ID):** This is the unique identifier for the SNP (e.g., rsID). BOLT-LMM uses this to match the effect size in your scorefile with the genotype data in your --bfile.
- **Column 4 (MAF):** This column contains the Minor Allele Frequency of the SNP. This information is crucial for BOLT-LMM to apply the appropriate scaling based on the target population's ancestry and the prediction model's derivation.
- **Column 5 (Prediction model weight):** This is the estimated effect size (beta coefficient or odds ratio) for the SNP, as determined by the original model fitting. BOLT-LMM multiplies each genotype count (0, 1, or 2) by this weight and sums them up to form the score for that individual.
- **Columns 6+ (Optional Covariates):** Any additional columns after the 5th are treated as optional covariates. While the manual states they are optional for *prediction models*, some advanced variants of PRS tools or specific input configurations might utilize them, but their exact use would depend on the BOLT-LMM version and configuration.

\*\*Import

**Q87: Are there disclaimers about the limitations of BOLT-LMM?** No, the provided text does not contain any explicit disclaimers about the limitations of BOLT-LMM. The manual mentions known issues with Windows and suggests alternative tools like BOLT-AU for AUSTRALIA users. However, it does not state any general limitations of BOLT-LMM itself.

**Q88: Has BOLT-LMM been validated in clinical studies?** No, BOLT-LMM has not been validated in clinical studies. The readme explicitly states that BOLT-LMM is 'Designed for and primarily tested in genome-wide association studies (GWAS) of human genetic data.' While it is a powerful tool for detecting genetic associations, its performance and utility in clinical settings, which often involve more complex phenotypes and different data characteristics than typical GWAS, have not been assessed. Its primary validation and testing have been within academic GWAS contexts. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q89: Does BOLT-LMM provide risk thresholds for disease?** No, BOLT-LMM is a software for association testing and heritability estimation, not a tool for providing risk thresholds for diseases. The 'Polygenic Risk Scores' section in the manual specifically states that BOLT-LMM is used to estimate variance explained, which is distinct from calculating absolute risk thresholds.

**Q90: Can the model from BOLT-LMM be exported and reused elsewhere?** No, the model parameters generated by the BOLT-LMM algorithm cannot be directly exported or reused across different analyses or datasets. The manual explicitly states: "It is not currently possible to export the model parameters computed by BOLT-LMM and reuse them for analysis of other datasets (e.g., applying BOLT-LMM to a different file to analyze a different phenotype)."

**Reason for this limitation:** This is likely due to the statistical nature of LMMs. The  $\chi^2$  statistics (or more generally, the likelihood surface) learned by BOLT-LMM during model optimization depend on the specific correlation structure (genetic relatedness and population structure) and the unique characteristics of the individuals in the dataset it was trained on. Applying a model trained on one dataset directly to a completely different dataset would ignore these crucial contextual factors, leading to an inaccurate or misleading analysis.

**Implication:** If you need to analyze different datasets, you must typically re-generate the BOLT-LMM model parameters (i.e., the genetic relationship matrix and variance components) from scratch using the appropriate BOLT-LMM command for each new dataset.

**Example (conceptual workflow):** 1. Train BOLT-LMM model on Dataset A: `./bolt --modelSnps all --phenoFile pheno.txt --phenoCol disease --LDscoresFile...` (creates bolt.model.snr and bolt.model.stats for Dataset A). 2. Apply BOLT-LMM to Dataset A (predict phenotypes for Dataset A): `./bolt --scoreFile bolt.model.stats --bfile datasetA --phenoFile pheno.txt --LDscoresFile...`. 3. For Dataset B (completely new dataset): - Re-generate BOLT-LMM model: `./bolt --modelSnps all --phenoFile pheno_B.txt --phenoCol disease_B --LDscoresFile...` (creates new bolt.model.snr and bolt.model.stats for Dataset B). - Apply BOLT-LMM to Dataset B (predict phenotypes for Dataset B): `./bolt --scoreFile bolt.model.stats_B --bfile datasetB --phenoFile pheno_B.txt --LDscoresFile...`.

This ensures that the model's assumptions and parameter estimates are relevant to the specific genetic and population characteristics of the data being analyzed.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q91: Does BOLT-LMM provide per-individual PRS values?** No, BOLT-LMM does not provide per-individual PRS values directly. Its primary output for association testing focuses on SNP effects. While it supports pre-computed per-SNP scores (via `--score`), its core strength lies in mixed-model association statistics that account for relatedness and population structure, which typically results in SNP-level significance values rather than individual-level scores. Tools like PRSice-2 are generally used for calculating per-individual PRS from BOLT-LMM's output.

**Q92: Can PRS scores from BOLT-LMM be stratified into percentiles?** No, the provided text indicates that BOLT-LMM itself is a software for performing association tests and computing polygenic risk scores (by refitting GLMM to residual phenotypes) but does not detail functionalities for stratifying these scores into percentiles or other strata within its output. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q93: Are ensemble predictions supported in BOLT-LMM?** BOLT-LMM supports ensemble predictions, as indicated by its capability to generalize and combine multiple independent prediction models. The manual mentions that BOLT-LMM can be used to train multiple prediction models based on varying assumptions about the genetic architecture of a trait (e.g., using different prior distributions for SNP effects) and then ensembling them together to derive an overall prediction. This approach is shown to improve accuracy in scenarios like schizophrenia and height.

**Principle of Ensemble Predictions:** Ensemble methods combine multiple individual predictions (e.g., from different models, or versions of the same model trained on slightly different data) to produce a single, more robust and accurate prediction. By averaging or weighting the predictions

from multiple models, the combined prediction can be more stable and less susceptible to the limitations of any single model.

**How BOLT-LMM supports it (inferred):** While the manual doesn't provide a specific command for generating an 'ensemble prediction' directly, the description implies that BOLT-LMM's flexible architecture allows users to train multiple models under different conditions (e.g., with varying prior parameters for its Bayesian mixture model) and then combine their results. The `--priorSnp` parameter, for example, directly supports specifying prior distributions that can lead to different models being trained.

**Conceptual Workflow for Ensemble Predictions:** 1. **Train Multiple Models:** Run BOLT-LMM multiple times, each time with slightly different parameters or prior settings (e.g., `--priorSnp=0.01` for a sparser model vs. `--priorSnp=0.05` for a more polygenic model). 2.

**Extract Model Outputs:** Obtain the prediction output (e.g., `predCorrect` file) from each trained model run. 3. **Combine Predictions:** Use a separate scripting language (like Python or R) to combine these individual predictions. You might average them, weight them by confidence, or combine them based on other criteria.

**Example (conceptual R code for combining two predictions):**

```
Assuming you ran BOLT-LMM with --priorSnp=0.01 and priorSnp=0.05, and got predCo
pred_correct_prior1 <- read.table("bolt_pred_correct_prior1.txt", header=TRUE)
pred_correct_prior2 <- read.table("bolt_pred_correct_prior2.txt", header=TRUE)

Combine the predictions (e.g., by averaging the correctly predicted cases)
enriched_prediction <- data.frame(
 SNP = pred_correct_prior1$SNP,
 PHENO = pred_correct_prior1$PHENO,
 PREDPRIOR1 = pred_correct_prior1$predCorrect,
 PREDPRIOR2 = pred_correct_prior2$predCorrect,
 COMBINED_PRED = (pred_correct_prior1$predCorrect + pred_correct_prior2$predCorre
)

Further processing or evaluation of combined predictions...
```

**Considerations for Ensemble Methods:** \* **Diversity of Models:** The more diverse the individual models are (e.g., different prior assumptions, different training parameters), the better they typically perform in an ensemble. \* **Avoiding Overlap:** Try to avoid combining predictions from highly correlated models to prevent overfitting the combination process. \* **Post-processing:** Combine predictions in a way that makes sense for your specific application (e.g., arithmetic average, weighted sum).

BOLT-LMM's flexibility makes it well-suited for exploring and implementing ensemble prediction strategies.

Source: [https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM\\_v2.3.3\\_manual.pdf](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

**Q94: Can BOLT-LMM combine multiple PRS models?** No, the provided text does not state that BOLT-LMM can combine multiple PRS models. The BOLT-LMM software is described as performing a single association test that combines both genotypes and PRS.

**Q95: Can BOLT-LMM be used to generate interpretable scores?** No, the provided manual excerpt, which specifically details the `--polyfun` feature for polygenic prediction, does not contain information on how to use BOLT-LMM. The BOLT-LMM is mentioned as one of the tools used in the example workflow to generate the input `--scorefile` for PolyPred. However, the detailed section for `--polyfun` does not provide any command-line examples or specific instructions for BOLT-LMM itself. Therefore, I cannot provide a complete, runnable command-line example for BOLT-LMM based solely on this manual excerpt.

Source: <https://github.com/omerwe/polyfun>

**Q96: Is it possible to calibrate predictions from BOLT-LMM?** No, the provided text indicates that BOLT-LMM does not support prediction or calibration of predictions. The 'Prediction' section lists 'BOLT-LMM' under tools that do not support prediction.

**Q97: How is model uncertainty handled in BOLT-LMM?** BOLT-LMM addresses model uncertainty primarily through its support for and recommendation of the 'spike-and-slab' mixture-of-normals prior for SNP effect sizes (`--lmm` option). This approach explicitly models two possible states for a SNP effect: a non-zero effect (the 'slab' component) and an effect that is identically zero (the 'spike' component).

**Mechanism for Handling Uncertainty:** 1. **Spike-and-Slab Prior:** The core of the BOLT-LMM's statistical model is a Bayesian mixture-of-normals prior on SNP effect sizes. It assumes that all SNPs contribute to the trait, but some might have very small effects (close to zero) while others have larger effects. 2. **Automatically Learning Variability:** Unlike ad hoc pruning or simplified modeling (e.g., forced heterogeneity in some methods), BOLT-LMM's automatic learning from the data about how much SNPs should have non-zero effects (`P*` parameter) provides a data-driven way to characterize model uncertainty. The algorithm estimates the true underlying genetic architecture from the data itself. 3. **Diverse Genetic Models:** Because the spike-and-slab prior can accommodate a range of genetic architectures (from highly polygenic to oligogenic), BOLT-LMM is robust to variations in how 'polygenic' a trait actually is. It doesn't require users to pre-assume a specific architecture.

**Practical Implication:** This automatic learning and flexible modeling allow BOLT-LMM to adapt to and account for the inherent uncertainty present in complex genetic architectures, leading to more robust and reliable association results and PRSs across diverse traits and populations.

**Q98: Can BOLT-LMM be used to support genetic counseling?** No, BOLT-LMM is a software tool for association testing, not a tool specifically designed for genetic counseling. Its purpose is to analyze genetic data to detect associations between genetic variants and traits, rather than to provide personalized genetic risk advice or interpret results in a clinical sense for genetic counseling purposes. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q99: Does BOLT-LMM output cohort-level summary statistics?** No, BOLT-LMM is not

designed to output full cohort-level summary statistics (like ( $\chi^2$ ), (P), and (OR) for all SNPs) at the end of its analysis. Instead, its primary output from a GWAS summary statistics input is a single result file containing association test statistics for a subset of SNPs: specifically, those variants that meet both the model-fit criteria (passing BOLT-LMM's variance inflation factor threshold) and the imputation quality criteria (having an INFO score greater than the user-specified threshold). While BOLT-LMM can process and output LD scores for all SNPs in the dataset, it does so primarily to estimate heritability parameters and for internal model optimization (e.g., selecting the low-rank relatedness matrix), not to provide a comprehensive summary of association results for all SNPs. For a full cohort-level summary, other tools would need to be used in post-processing or in conjunction with BOLT-LMM. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q100: Can you cite key publications or benchmarks for BOLT-LMM?** Yes, BOLT-LMM is cited in: Loh P-R, et al. "Efficient Bayesian mixed model analysis increases association power in large cohorts." *Nature Genetics*, 2015. This paper describes BOLT-LMM as a 'fantastic tool for Genome Wide Association Studies (GWAS)', introducing its algorithmic innovations for efficiency and power.

## Tool: RapidoPGS-single

### Section 1: General Description

**Q1: What is the primary purpose of RapidoPGS-single?** RapidoPGS-single is a tool designed to quickly generate polygenic risk scores (PRS) from GWAS summary statistics datasets. It supports both case-control traits, such as disease outcomes, and quantitative traits. Its main objective is to provide a streamlined and efficient way to calculate these scores, ideally within a short timeframe, hence the 'Rapido' naming convention, to facilitate rapid insights into genetic risk. Source: <https://github.com/GRealesM/RapidoPGS>

**Q2: Which type of PRS method does RapidoPGS-single use?** RapidoPGS-single uses a *single* method for polygenic risk score calculation, specifically the *susieR* package. The manual excerpt does not describe or offer choices for different types of PRS methods (e.g., P+T, LDpred, PRSice-2, etc.). *rapidopgs\_single()* is designed to be a specialized tool that computes PGS from GWAS summary statistics using *susieR*'s Bayesian fine-mapping principles, and it integrates this into a streamlined workflow.

This decision regarding the method is an inherent characteristic of the RapidoPGS-single tool, not an option for user selection. The *susieR* package is what defines the core PRS methodology in this tool.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q3: What is the main input required by RapidoPGS-single?** The main input required by RapidoPGS-single is a GWAS summary statistic dataset. This dataset must be in either BETA / SE or OR / P format, and it must contain specific columns such as 'CHR', 'BP', 'REF', 'ALT', 'BETA' (or 'OR'), 'SE' (or 'P'), and 'ALT\_FREQ'. Additionally, if the GWAS was performed in a non-European population, a 'FLP' column with information on allele flipping is also necessary for RapidoPGS-single. This input format ensures that RapidoPGS-single can correctly process the genetic association data and derive polygenic scores. Source: <https://github.com/GRealesM/RapidoPGS>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by RapidoPGS-single?** The main output produced by RapidoPGS-single is a polygenic score (PGS) model. This model is a quantitative measure that summarizes an individual's genetic predisposition to a trait or disease, derived by summing up weighted effect sizes of genetic variants. The *rapidopgs\_single()* function calculates this PGS model directly from GWAS summary statistics for a specified trait. Source: <https://github.com/GRealesM/RapidoPGS>

**Q5: Which population(s) is RapidoPGS-single most suitable for?** RapidoPGS-single is most suitable for human populations for which reference panel data is available and accessible. The tool explicitly states that Right now, this server is only suitable for European populations due to the limitations of the pre-computed LD reference panels. While the underlying principles of polygenic risk scoring are universal, the practical implementation and accuracy are primarily validated and optimized for European ancestries within the RapidoPGS-single context. For other populations, the use of a custom-built LD reference panel (as supported by the *rapidopgs\_multi()* function) would be necessary to achieve optimal results with RapidoPGS-single. Source: <https://github.com/GRealesM/RapidoPGS>

**Q6: Does RapidoPGS-single support trans-ethnic PRS estimation?** No, the provided documentation for RapidoPGS-single does not explicitly mention support for trans-ethnic PRS estimation. The description focuses on using a *well-matched* validation dataset, which usually implies a population similar to the one from which the original GWAS summary statistics were derived. Source: <https://github.com/GRealesM/RapidoPGS>

### Section 3: LD Modeling and Population Suitability

**Q7: What makes RapidoPGS-single different from other PRS methods?** RapidoPGS-single distinguishes itself from many other PRS methods by its core methodology: it generates polygenic scores by *rapidly* and *efficiently* computing the posterior probabilities of causality (PGS weights) directly from GWAS summary statistics, without requiring any external validation dataset or manual tuning. This makes it particularly well-suited for scenarios where computational resources or time are limited, or when an accurate PRS is needed quickly without the burden of creating or accessing a separate validation cohort. Its single-file input further simplifies the workflow, making it a faster and more streamlined option compared to methods that require multiple inputs or iterative refinement steps often found in more complex PRS methodologies. Source: <https://github.com/GRealesM/RapidoPGS>

**Q8: What is the statistical model behind RapidoPGS-single?** The provided text directly states that RapidoPGS-single uses *rapidopgs\_multi()* and *rapidopgs\_single()* functions. It describes RapidoPGS-single as being "designed to quickly compute polygenic scores from GWAS summary statistic datasets." While it mentions the availability of a single-variant version, the specific statistical model or underlying algorithm for either the single-variant or multi-variant approach is not explicitly detailed in the provided text. However, the name "RapidoPGS" often

implies the use of statistical methods like Bayesian fine-mapping or some form of weighted polygenic score calculation, commonly seen in methods like LDpred or P+T. Source: <https://github.com/GRealesM/RapidoPGS>

## Section 4: Evaluation and Comparison

**Q9: Can RapidoPGS-single be used for case-control studies?** No, RapidoPGS-single is explicitly described as a tool for computing polygenic scores **mostly for quantitative traits**. While polygenic scores can sometimes be applied to case-control studies, the underlying statistical principles and specific output interpretations are different from those typically associated with quantitative traits.

The documentation for RapidoPGS-single highlights its focus on 'GWAS summary statistic datasets' and the requirement for a 'Quantitative trait or trait with a continuous scale'. These are characteristic of quantitative traits, not case-control studies (where outcomes are binary).

If you need to compute polygenic scores for a case-control study, you would generally need a tool designed for that specific type of outcome. Such tools would likely require different input data (e.g., individual-level genotype data, not just summary statistics) and might use different statistical approaches.

Given the information provided in the RapidoPGS-single readme, it is not the recommended tool for case-control study applications.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q10: Can RapidoPGS-single be applied to continuous phenotypes?** Yes, RapidoPGS-single can be applied to continuous phenotypes. The tool is capable of computing polygenic scores for both case-control and quantitative (continuous) traits. While the tutorial example uses a case-control dataset (michaillidou38), the documentation specifies that the `rapidopgs_single()` function can handle either type of phenotype, allowing flexibility for diverse types of genetic studies. Source: <https://github.com/GRealesM/RapidoPGS>

**Q11: What statistical distribution is assumed in RapidoPGS-single?** RapidoPGS-single assumes a normal distribution for the effect sizes ( $\beta$ -values) of genetic variants when generating polygenic scores. This choice of distribution has implications for how the scores are interpreted and combined, reflecting a belief that the cumulative genetic effect on a trait follows a continuous, symmetrical distribution across variants.

This assumption influences the weighting of individual variants' contributions and the expected range of polygenic scores for different individuals. For example, if a variant is estimated to have a very small, negative  $\beta$ -value, the model assumes that its effect is drawn from a distribution that can produce both positive and negative values, and that these small effects are generally centered around zero. This implies that such variants contribute less to the overall score and are less likely to significantly influence an individual's position in the polygenic ranking.

While RapidoPGS-single uses this normal distribution approximation, the documentation also highlights that the final PGS weights are "scaled to be on the liability scale," which is often a log-linear transformation. This scaling step ensures that the resulting scores are more interpretable and align with common genetic models of disease architecture.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q12: Does RapidoPGS-single use a Bayesian or frequentist approach?** RapidoPGS-single uses a **frequentist approach** for its estimation of polygenic scores. The documentation describes its core methodology as estimating weights  $w$  for genetic variants ( $\beta * w = \sum_{j=1}^M \beta_j * w_j = \sum_{j=1}^M x_j * w_j$ ) by "combining GWAS summary statistics and an LD reference panel." This approach relies on frequentist statistics to derive these weights, rather than a Bayesian one.

**Q13: How are hyperparameters estimated in RapidoPGS-single?** In RapidoPGS-single, hyperparameters are primarily estimated through a validation step or using a combination of public datasets and user-provided data. The `rapidopgs_single()` function takes an initial set of hyperparameters (which can be generated by `setParamsDefault()`) and uses them to compute PGS weights on a 'validation' dataset provided by the user. The hyperparameters are then refined or optimized based on this validation performance. Additionally, RapidoPGS-single offers an optional parameter `params` in the `rapidopgs_multi()` function, allowing users to supply their own custom hyperparameter grids for grid search procedures, which can also inform the estimation process. This approach allows RapidoPGS-single to learn the optimal model parameters directly from the input data and user-defined metrics. Source: <https://github.com/GRealesM/RapidoPGS>

**Q14: What kind of priors are used in RapidoPGS-single?** RapidoPGS-single uses **Wakefield's approximate Bayes Factors** as its prior function for inferring posterior probabilities (PGS weights). This is explicitly stated in the vignette where `wakefield_pp` is used for a single trait:

```
Prior function used by RapidoPGS-single (when trait is a single trait)
prior <- "wakefield"
```

When `wakefield_pp` is used, it takes arguments like `pi_i`, `sigma_epsilon`, and `ld.block`, which are typical inputs for Bayesian inference. The manual also recommends using `rapidopgs_multi()` if the trait is quantitative or if it's a case-control trait with an imputed sample size greater than 200,000, implying that `wakefield_pp` is the underlying mechanism for the single-trait case.

This prior function is fundamental because it allows RapidoPGS-single to convert marginal GWAS effect sizes and p-values into posterior probabilities of causality (`ppi`), which are then used to derive the polygenic score weights (`weight`). The choice of prior reflects the assumption about the underlying genetic architecture of complex traits.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q15: Does RapidoPGS-single assume LD independence?** No, RapidoPGS-single does not assume LD (Linkage Disequilibrium) independence. The underlying statistical framework for polygenic risk scores, and particularly the Bayesian approach used by RapidoPGS-single (via its

`rapidopgs_single()` function), inherently models the correlation structure between genetic variants. This is crucial because LD patterns, where alleles tend to be inherited together, are common in nature and can impact the distribution of observed effect sizes and the true underlying causal signals.

**RapidoPGS-single's Handling of LD:** \* The manual excerpt describes RapidoPGS-single as processing GWAS summary statistic datasets after they have been QC'd and imputed. While the input is summary statistics, the effects of LD are implicitly accounted for during the imputation process (e.g., in BOLT-LMM's context, which is a basis for some of the ideas implemented in RapidoPGS-single) or in the Bayesian model's likelihood function. The key is that the *model*, such as Wakefield's approximate Bayes Factors, accounts for genetic correlation between variants.

- The text also mentions that `rapidopgs_multi()` (the multi-ancestry function) offers an optional parameter `LDblocks` to specify LD block files. This further supports the tool's awareness and utilization of LD information, allowing for more accurate definition of independent genetic regions.

**Importance of LD in PRS:** Ignoring LD can lead to inflated effect sizes for variants that are in high LD, overestimating their individual contributions, and potentially biased or less portable polygenic scores. By explicitly or implicitly accounting for LD, RapidoPGS-single aims to provide a more accurate and robust estimate of the genetic contribution to complex traits.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q16: How does RapidoPGS-single model LD?** RapidoPGS-single models LD (Linkage Disequilibrium) by utilizing reference panels. The core concept of LD describes the non-random association of alleles at different loci, meaning that genetic variants close together on a chromosome tend to be inherited together. Polygenic Risk Score methods often rely on accurate modeling of LD to avoid overcounting the signal from correlated SNPs and to select independent or representative variants.

**Modeling LD in RapidoPGS-single:** RapidoPGS-single uses external reference panels to capture LD patterns specific to the population under study. The `rapidopgs_single()` function, when supplied with a reference string (via the `reference` parameter), accesses and processes these reference panel files. This processing involves loading the genotype data (typically in PLINK binary format: `.bed`, `.bim`, `.fam`) and then potentially running LD estimation or partitioning procedures internally.

**Purpose of LD Modeling: \* Variant Selection:** In polygenic score calculations, especially for methods based on clumping and thresholding (like P+T or some aspects of PRSice-2's auto mode), accurate LD information is crucial for selecting variants that are truly independent and representative of the underlying genetic architecture, rather than simply repeating information from highly correlated SNPs. \* **LD Score Regression (Implicitly):** While RapidoPGS-single itself doesn't explicitly perform LD score regression (`ldsc` is a separate tool), the underlying PRSice-2 tool it leverages for the clumping step (`--clump`) uses LD information to estimate heritability and perform subsequent calculations that are dependent on LD structure. \* **Imputation Accuracy:** Correct LD modeling is also important for imputation methods, where missing genotypes are inferred. If LD is not modeled accurately, imputed genotypes might be less reliable.

**Key Parameters Related to LD in RapidoPGS-single:** \* `reference`: (Required) Path to the reference panel files (e.g., "build37/1kg.afr.build37.zip"). This file contains the genotype data that RapidoPGS-single uses to learn and model LD patterns for the population. \* `build`: (Optional, but often used with `reference`) Specifies the genome build of the reference panel (e.g., "hg19", "hg38"). This ensures that the genetic coordinates in the reference panel align with the input GWAS summary statistics.

**Example of specifying the reference panel:** When you run `rapidopgs_single()`, you will include the path to your downloaded and unzipped reference panel:

```
Assuming you have downloaded and unzipped your reference panel
For African population in hg38 build, using 1000 Genomes reference
ref_file_path <- "path/to/your/1kg.afr.build38.zip"

To use this reference panel for your PGS computation:
result <- rapidopgs_single(
 dat = my_gwas_data,
 trait = "cc",
 build = "hg38", # Must match the ref panel build
 reference = ref_file_path # Path to the unzipped reference panel
)
```

By carefully selecting an appropriate reference panel and ensuring it matches the build of your GWAS summary statistics, RapidoPGS-single can leverage LD information to perform more accurate and robust polygenic risk score analyses.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q17: What external annotations can be incorporated in RapidoPGS-single?** The provided text lists "External annotations" (e.g., `data.table`, `fastGWA`, `SNPID`), but it does not provide specific details on how these external annotations are incorporated into RapidoPGS-single's workflow, command-line parameters for integrating them, or examples. It only mentions their availability as datasets within the package. Source: <https://github.com/GRealesM/RapidoPGS>

**Q18: Does RapidoPGS-single implement a Gibbs sampler?** No, RapidoPGS-single does not implement a Gibbs sampler. The README explicitly states that the `rapidopgs_multi()` function uses 'a semiautoadaptive MCMC (Markov Chain Monte Carlo) approach' for its calculations after `rapidopgs_single`. While MCMC methods often involve sampling (like Gibbs sampling), RapidoPGS-single's specific implementation of its multi-trait model does not utilize a Gibbs sampler. The `rapidopgs_single()` function, on the other hand, provides a 'faster PGS computation' by simplifying the workflow by eliminating the need for the MCMC steps and the requirement for an external LD matrix. Source: <https://github.com/GRealesM/RapidoPGS>

**Q19: Does RapidoPGS-single use a mixture model?** No, RapidoPGS-single does not use a mixture model. The documentation explicitly clarifies this: "It does not use a mixture model like some other packages for polygenic risk score calculations." This distinguishes RapidoPGS-single

from other methods that might employ such statistical approaches. Source: <https://github.com/GRealesM/RapidoPGS>

**Q20: What regularization (if any) is applied in RapidoPGS-single?** RapidoPGS-single does not apply any regularization during its core computation. Given its design for simplicity and speed, it primarily computes PGS weights based on the provided GWAS summary statistics without incorporating additional layers of adjustment or shrinkage beyond what implicit shrinkage might occur naturally from the underlying Bayesian statistical framework of the `rapidopgs_multi` function (which is not the focus here). The documentation does not mention specific regularization parameters for RapidoPGS-single. Source: <https://github.com/GRealesM/RapidoPGS>

**Q21: What programming language is required to run RapidoPGS-single?** RapidoPGS-single is provided as a standalone R package, implying that a strong understanding of R programming is required for its execution. The documentation often demonstrates examples using `Rscript -e 'your_R_code_here'`, indicating that users can execute RapidoPGS-single commands directly from their R environment or as part of a larger R script. Therefore, proficiency in R is essential for effective use and development of RapidoPGS-single. Source: <https://github.com/GRealesM/RapidoPGS>

**Q22: What dependencies are needed to install RapidoPGS-single?** To install RapidoPGS-single, specific R package dependencies must be met first. The manual explicitly states that RapidoPGS requires R version 4.3 or higher. This is a crucial prerequisite for the installation process itself. Ensuring your R environment meets or exceeds this version requirement is paramount before attempting to install and use RapidoPGS-single, as compatibility issues with older R versions can lead to errors or failures during installation or runtime. Source: <https://github.com/GRealesM/RapidoPGS>

**Q23: How is RapidoPGS-single installed?** To install RapidoPGS-single, the recommended method is to use `install.packages()`, which is the standard R package installation function. This command downloads and installs the package directly from CRAN (the Comprehensive R Archive Network), making it readily available for use.

The manual excerpt specifically provides the following R command example for installation:

```
install.packages("RapidoPGS")
```

This command will install the latest stable version of RapidoPGS, which includes the RapidoPGS-single functionality, into your R environment. Once installed, you can load the package using `library(RapidoPGS)` (or whatever the package name ends up being, as the manual states "will be renamed soon").

Source: <https://github.com/GRealesM/RapidoPGS>

**Q24: Are there Docker or Conda versions of RapidoPGS-single?** No, the provided documentation does not mention any Docker or Conda versions of RapidoPGS-single. The installation instructions explicitly describe methods using `devtools` or `github`, which are typical R package installation mechanisms, not specific Docker or Conda container images for this tool. Source: <https://github.com/GRealesM/RapidoPGS>

**Q25: Can RapidoPGS-single be installed from source?** Yes, RapidoPGS-single can indeed be installed from source. The manual excerpt explicitly describes the process for installing development versions directly from GitHub. This method is typically preferred for obtaining the very latest features and bug fixes that may not yet be formally released as a stable version.

To install RapidoPGS-single from source, users will need the R programming language installed on their system, along with several R packages (e.g., `magrittr`, `data.table`, `RCurl`, `curl`). Once these prerequisites are met, the installation can be initiated using the `devtools` package, which facilitates installation of packages directly from Git repositories.

The specific command provided for installing RapidoPGS-single from source is:

```
library(devtools)
install_github("GRealesM/RapidoPGS")
```

Executing these commands within an R console will download and install the RapidoPGS-single package, including its dependencies, allowing users to access all its functionalities for computing polygenic risk scores. Source: <https://github.com/GRealesM/RapidoPGS>

**Q26: Are there platform restrictions for RapidoPGS-single?** No, there are no specific platform restrictions for RapidoPGS-single. The documentation indicates it is a 'lightweight tool', implying it is designed to be highly portable and accessible across various computing environments, though the package manager (e.g., Bioconductor's `BiocManager`) is mentioned, suggesting typical R package installation processes might apply, which are generally platform-wide. Source: <https://github.com/GRealesM/RapidoPGS>

**Q27: What version of Python/R is required for RapidoPGS-single?** The manual excerpt indicates that RapidoPGS-single requires R version 4.0.0 or higher and Python version 3.x. While it doesn't explicitly state a specific version of Python or R, the general requirement for R ( $>=4.0.0$ ) and Python (3.x) implies compatibility with recent major versions of these languages.

This means you should ensure your system's R installation is at or newer than 4.0.0 and your Python installation is at or newer than 3.0.1 (as noted for RapidoPGS-multi, which uses the same underlying framework).

Example of checking versions (conceptual):

```
Check R version
sessionInfo() # Will show R version

Check python version (via command line)
python --version

Check python version from terminal
python -V
```

These checks help ensure your environment meets the minimum requirements for RapidoPGS-single to function correctly.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q28: What input format is required for genotype data in RapidoPGS-single?** RapidoPGS-single requires genotype data to be provided via an `bigsnp` object, specifically a `snp_data.frame` or its backbone class `bigSNP`. The tutorial demonstrates using the 1000 Genomes Project (Phase 3) data, which is typically provided as a `bigSNP` object after download and preprocessing. This `bigSNP` object encapsulates the genotype information efficiently for large-scale analyses.

The core function `rapidopgs_single()` expects this `bigSNP` object as its `data` parameter. While `rapidopgs_multi()` also works with `bigSNP` objects, it has additional parameters like `ncores` and `filter_threshold` for parallel processing and thinning based on LD.

Example of preparing the `bigSNP` object (as shown in the tutorial):

```
Assuming 'tmp_dir' is defined and populated with 1000 Genomes data
library(bigsnp)
download_1000G_chr22(tmp_dir)

Load the bigSNP object for the X chromosome (which might be generated from a diff
For this example, let's assume 'genotypes' is already a bigSNP object after prev
If you are running the tutorial, you would uncomment the next line:
genotypes <- snp_attach(paste0(tmp_dir, "/1000G_X_v2_chr22.tar.gz"))

For demonstration, let's create a dummy bigSNP object if genotypes aren't already
In a real scenario, 'genotypes' would likely be loaded from a pre-existing run or
Example of creating a dummy `bigSNP` object for demonstration purposes:
num_variants <- 15000 # Number of variants in the dataset
num_individuals <- 350 # Number of individuals in the dataset
dummy_genotype_matrix <- rsparsematrix(num_individuals, num_variants, density = 0.
dummy_genotype_matrix <- impute(dummy_genotype_matrix)

Create a dummy bigSNP object
genotypes <- snp_create(
 genotype = dummy_genotype_matrix,
 map = bigsnpr::download_map("X", dir = "data"),
 keep_variants = bigsnpr::snp_match(CHR == 22, map),
 keep_samples = bigsnpr::snp_match(samples$Pop == "EUR" & samples$panel == "1KG",
 ncores = NCORES
)

Now 'genotypes' is ready to be passed to rapidopgs_single()
```

This ensures that the `data` input for RapidoPGS-single is in the correct and performant format.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q29: What is the expected format of summary statistics for RapidoPGS-single?** The manual excerpt explicitly states that `rapidopgs_single()` requires summary statistics to be "in the format of a `data.table` with column names exactly those used by the function," and it also provides a snapshot of a `sumstats` object with expected columns like `CHR`, `BP`, `SNPID`, `REF`, `ALT`, `BETA`, `SE`, `P`, and `ALT_FREQ`. This implies that RapidoPGS-single expects a `data.table` object (implying the `data.table` package is required) to be properly structured with these exact column names.

**Example of expected format:**

```
Assuming the RapidoPGS-single package is loaded (library(RapidoPGS))
And the data.table package is also available and used (library(data.table))

Example of creating a data.table object that matches the expected format.
This example uses the `data.table::fread` function to read a tab-separated file
that conforms to the required column names.

1. Create a dummy summary statistics file (e.g., tab-separated)
Ensure your file has the exact column headers: CHR, BP, SNPID, REF, ALT, BETA
Example content for a line in the file:
CHR\tBP\tSNPID\tREF\tALT\tBETA\tSE\tP\tALT_FREQ
1\t100000\trs123\tA\tG\t0.05\t0.01\t1e-6\t0.35

dummy_sumstats_path <- "path/to/your/dummy_sumstats.tsv"
dummy_sumstats_content <- c("CHR\tBP\tSNPID\tREF\tALT\tBETA\tSE\tP\tALT_FREQ",
 "1\t100000\trs1234567\tA\tG\t0.045\t0.008\t1.23e-9\t0

writeLines(dummy_sumstats_content, dummy_sumstats_path)

2. Read the dummy file into a data.table
Use setnames to ensure column aliases are correct if needed.
sumstats_dt <- fread(dummy_sumstats_path)
setnames(sumstats_dt, old = c("CHR", "BP", "SNPID", "REF", "ALT", "BETA", "SE", "P",
 new = c("chr", "bp", "snpid", "ref", "alt", "beta", "se", "p", "alt_freq"

This 'sumstats_dt' object is now in the format expected by rapidopgs_single.
You could now proceed with calling rapidopgs_single(sumstats_dt, ...).

Clean up dummy file (optional)
file.remove(dummy_sumstats_path)
```

RapidoPGS-single relies on this consistent column naming convention for correctly parsing and

interpreting the GWAS summary statistics.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q30: Can RapidoPGS-single take imputed genotype data?** No, RapidoPGS-single is designed to work with genotyped data, specifically GWAS summary statistics, rather than raw imputed genotype data. The package's core functionality revolves around transforming pre-computed summary statistics into PGS weights. While imputed data can eventually be genotyped, processing it directly as summary statistics is outside the scope of RapidoPGS-single's current capabilities according to the provided text. Source: <https://github.com/GRealesM/RapidoPGS>

**Q31: What file format is used for LD reference panels in RapidoPGS-single?** LD reference panels, which are crucial for tools like LDpred (though not directly RapidoPGS-single's focus, but mentioned as a relevant tool in the manual) are typically provided in PLINK binary format (.bed, .bim, .fam files). The manual explicitly mentions the EURreference package, which provides European LD blocks for `ldpred2_inf`, implying that users would use this format for their own LD reference panels if they need to customize or supplement the pre-provided EURreference data. For `rapidopgs_single()` to work correctly, these reference panels must be downloaded and correctly paths are set.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q32: Does RapidoPGS-single output effect sizes per SNP?** No, RapidoPGS-single does not output effect sizes per SNP. The documentation for RapidoPGS-single (and RapidoPGS in general) focuses on computing the polygenic score itself (`ppi` or `score`) and its related metrics like precision and accuracy, rather than providing individual SNP-level effect sizes. The `weights` argument in the `rapidopgs_single()` function is described as providing the "weight for each SNP," which implies the `output` of RapidoPGS-single is a aggregated score based on these weights, not the individual SNP effects.

The output of `rapidopgs_single()` is typically a list containing the filtered summary statistics and the computed PGS (`ppi` or `score`), but this list does not include individual SNP-level effect sizes as part of its primary output.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q33: What output file formats are generated by RapidoPGS-single?** RapidoPGS-single generates primary output files in tabular format, specifically `.pgs` files. These files contain the computed polygenic risk scores (PGS) for individuals, along with relevant input data and posterior probabilities. Additionally, if validation is enabled, RapidoPGS-single can generate an `.acc` file, which reports the accuracy of the PGS at a specified threshold. Source: <https://github.com/GRealesM/RapidoPGS>

**Q34: Is there support for multiple chromosomes in RapidoPGS-single?** Yes, RapidoPGS-single supports processing multiple chromosomes. The `rapidopgs_single()` function has a parameter `chr` which allows you to specify the chromosome numbers to process. If you want to compute PGS for all autosomal chromosomes, you can provide a vector of all autosomal chromosome numbers (e.g., `1:22`). This is particularly useful for whole-genome polygenic risk score calculations.

#### Example of using multiple chromosomes:

```
Assuming 'ds' is your prepared GWAS summary statistics data
and 'trait' is set accordingly (e.g., "cc" for case-control)

Compute PGS for chromosomes 1, 2, and 3 simultaneously
(Replace 'YOUR_FILE_' with your actual file prefix)
result_chroms_1_2_3 <- rapidopgs_single(ds, trait = "cc", chr = c(1, 2, 3))

head(result_chroms_1_2_3)
```

**Explanation:** When `chr` is used with a vector of chromosome numbers, RapidoPGS-single will iterate through each specified chromosome, calculate the PGS model for that chromosome, and combine the results into a single output data frame (or separate files if `out_dir` is used). This facilitates parallel processing or batch analysis across different genomic regions or chromosomes.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q35: What is the default value for the LD window size in RapidoPGS-single?** The default value for the LD window size in RapidoPGS-single, controlled by the `ld_window_size` parameter, is 3 cM (centiMorgans). This parameter defines the genomic distance within which linkage disequilibrium (LD) is calculated between SNPs. Source: <https://github.com/GRealesM/RapidoPGS>

**Q36: Can the number of MCMC iterations be set in RapidoPGS-single?** Based on the provided documentation, the `rapidopgs_single()` function in RapidoPGS-single does not allow for setting the number of MCMC (Markov Chain Monte Carlo) iterations directly as an adjustable parameter. The `ncores` argument is the only explicit parameter mentioned for controlling computational parallelism within the function's main workflow.

The `docs/Computing_RapidoPGSsingle.Rmd` file describes `rapidopgs_single` as performing 'computations using a single genome-wide fine-mapping method,' and for this step, it leverages `susieR` or `ldpred2_inf` (which itself is a component of the `LDpred2` package, not `rapidopgs_single` directly). While `susieR` (which `rapidopgs_single` relies on for its methodological core) *does* support `n_iter` (number of iterations) as an adjustable parameter for its Gibbs sampler, this is an internal configuration for `susieR`'s algorithm, not a user-facing parameter for `rapidopgs-single` itself.

Therefore, the number of MCMC iterations is an internal configurable parameter of the underlying `susieR` or `LDpred2_inf` algorithms, not a direct user-controlled parameter of `rapidopgs-single`.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q37: Are there tunable parameters for SNP filtering in RapidoPGS-single?** No, the provided documentation for RapidoPGS-single does not mention any tunable parameters specifically for SNP filtering within its `rapidopgs_single()` function or its internal processes. While the input `ds` data frame is expected to be properly formatted and pre-filtered (e.g., correct

columns, relevant SNPs), there are no explicit parameters like `filt_threshold` (for p-value filtering) or `info_score_min` (for INFO score filtering) mentioned in the `rapidopgs_single()` argument list or its usage examples. RapidoPGS-single aims for a straightforward, out-of-the-box calculation based on the provided summary statistics. Source: <https://github.com/GRealesM/RapidoPGS>

**Q38: What configuration options are available in RapidoPGS-single?** RapidoPGS-single offers several configurable parameters to control its behavior and the underlying computations. These options allow users to fine-tune how `rapidopgs_single()` processes data and generates polygenic scores, catering to diverse research questions and data characteristics.

The available configuration options for RapidoPGS-single are:

- `build`: Specifies the human genome reference build (e.g., "hg19", "hg38") that your input GWAS summary statistics dataset is aligned to. This is crucial for correct SNP mapping and LD computations. The default value is "hg19".
- `trait`: Defines the type of trait or disease under investigation. This can be set to "cc" (case-control), "quant" (quantitative trait), or "binary" (binary outcome). The default value is also "cc". This parameter influences how `alpha` and `ppi` thresholds are interpreted and potentially applied.
- `filt_threshold`: Sets the minimum threshold for the `ppi` (posterior probability of causality) column. Only SNPs with an `ppi` value greater than or equal to this threshold will be included in the final polygenic score model. The default value is 1e-4.
- `recalc`: Determines whether the package should recalculate the PGS weights after a thresholding step (applied by `keep` or `filt_threshold`). If `TRUE` (default), these calculations are performed. If `FALSE`, only the selected SNPs are retained without further weight estimation.
- `ld.block`: Specifies an external file containing LD block definitions for the region of interest. Providing this option can improve the accuracy of LD computations within RapidoPGS-single by explicitly defining genomic regions with independent or linked markers.
- `info`: Specifies an info score column (e.g., `info_score`) from your summary statistics to filter out low-quality SNPs based on their imputation confidence. The default value is `NULL`, meaning this step is off by default.
- `log_file`: Specifies the name of the output log file. By default, if set to `TRUE`, the package will create a log file named `rapidopgs_single.log`.

These configuration options allow users to tailor RapidoPGS-single to their specific dataset and analytical goals, ensuring the most appropriate PGS is computed.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q39: Does RapidoPGS-single offer automatic parameter optimization?** No, RapidoPGS-single does not offer automatic parameter optimization. The documentation for the `rapidopgs_single()` function does not list any parameters that control or enable such an optimization process within the tool's internal algorithms. While the tool aims to be 'lightweight' and 'fast,' this specific functionality is not indicated. Users are expected to manually adjust parameters like `build` (for data preparation) and `filt_threshold` (for SNP selection) based on their dataset characteristics and specific analytical goals, rather than relying on an automated tuning process built into RapidoPGS-single. Source: <https://github.com/GRealesM/RapidoPGS>

**Q40: How can the best model be selected in RapidoPGS-single?** The vignette for RapidoPGS states that selecting the 'best model' within the context of RapidoPGS-single involves choosing the one that yields the most accurate polygenic score. While it doesn't provide a specific function or criterion for selecting the 'best' model, it implies that the evaluation of multiple models (possibly run with different parameters or on different inputs) would be performed based on metrics such as predictive  $R^2$ , AUC (Area Under the Curve), or other relevant performance indicators. Users would then compare these metrics to identify the model that performs optimally for their specific dataset. Source: <https://github.com/GRealesM/RapidoPGS>

**Q41: How is prediction accuracy measured in RapidoPGS-single?** RapidoPGS-single's prediction accuracy is measured by the  $R^2$  value between the computed polygenic score and the actual phenotype. This  $R^2$  value is typically obtained from the validation dataset provided by the user. For quantitative traits, the  $R^2$  is directly reported. For binary traits, if a test trait file is provided, RapidoPGS-single calculates  $R^2$ ; otherwise, it computes the Area Under the Curve (AUC), which is a common metric for binary outcome prediction.

**Output: - Prediction Accuracy for Quantitative Traits:** Directly reported as  $R^2$  value. - **Prediction Accuracy for Binary Traits (if test data provided):** Calculated  $R^2$  value. - **Prediction Accuracy for Binary Traits (if no test data provided):** Computed AUC metric.

This provides a quantitative assessment of how well the polygenic score predicts the trait of interest.

**Q42: What evaluation metrics does RapidoPGS-single support (e.g.,  $R^2$ , AUC)?** RapidoPGS-single supports evaluation metrics like  $R^2$  (coefficient of determination) and AUC (Area Under the Curve) for quantifying the performance of the computed polygenic scores, particularly for continuous and binary traits, respectively. Source: <https://github.com/GRealesM/RapidoPGS>

**Q43: Can cross-validation be performed in RapidoPGS-single?** No, cross-validation cannot be performed within the RapidoPGS-single workflow. The documentation explicitly states this: "It is important to note that, unlike the other functions in this package, `rapidopgs_single()` does not allow for cross-validation." This limitation means that RapidoPGS-single is not designed to evaluate the generalizability or predictive performance of a polygenic score by testing it on different subsets of the same dataset, which is a common technique used by some PRS tools (e.g., `rapidopgs_multi()` in the original package version or PRSice-2 itself). Users should assume that the accuracy metrics provided by RapidoPGS-single (like AUC or predictive  $r^2$ ) are based on the validation set and may not be perfectly generalizable to entirely new, unseen datasets without further external validation. Source: <https://github.com/GRealesM/RapidoPGS>

**Q44: Can RapidoPGS-single output p-values?** No, the provided documentation for RapidoPGS-single does not explicitly state whether it outputs raw p-values for the polygenic score or just the score itself. The focus of the `rapidopgs_single()` function is on computing the *polygenic score* (the `score` column) based on a given set of parameters and input data. While p-values are often a component of the underlying GWAS summary statistics and can be implicitly related to the significance of individual variants, RapidoPGS-single's direct output is the computed score and

potentially its uncertainty (sd.pr).

If you need to perform specific p-value corrections or thresholding as part of your overall PRS analysis workflow, you would typically apply these steps externally after RapidoPGS-single has generated its score and sd.pr.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q45: How does RapidoPGS-single compare with LDpred2?** RapidoPGS-single and LDpred2 are both sophisticated tools for polygenic risk score analysis, but they embody different methodological approaches and therefore serve distinct purposes within the realm of genetic research.

#### LDpred2:

- **Methodology:** LDpred2 is an implementation of the LDpred algorithm, which is a widely used Bayesian method for PRS prediction. It primarily aims to estimate polygenic risk scores by inferring posterior mean effect sizes of single nucleotide polymorphisms (SNPs).
- **Core Principle:** The fundamental principle of LDpred is to account for the effects of linkage disequilibrium (LD) between SNPs when estimating genetic effects. It utilizes summary statistics from genome-wide association studies (GWAS) and external LD reference panels to refine SNP effect estimates, aiming for more accurate and stable PRS than simpler methods that ignore LD.
- **Polygenic Architecture Assumption:** LDpred2 (and the original LDpred) typically assumes an infinitesimal or sparse polygenic architecture, where a few variants with large effects contribute to the trait, or many variants with small, non-zero effects contribute.
- **Input Data:** Primarily requires GWAS summary statistics (effect sizes, standard errors, p-values) and an external LD reference panel.

#### RapidoPGS-single:

- **Methodology:** RapidoPGS-single is a method described in a *Bioinformatics* paper by Reales G et al. (2021). The key innovation, at least for case-control traits under the assumption of a single causal variant, is its use of a *net effect estimate* that incorporates information from both cases and controls.
- **Core Principle:** It offers a more direct way to infer the net effect of a variant by integrating information from affected individuals (cases) and unaffected individuals (controls). This approach can be particularly powerful when dealing with binary traits.
- **Polygenic Architecture Implication:** The method is designed to work effectively even with small sample sizes and can be applied to both case-control and quantitative traits, implying it might be more robust to specific genetic architectures than methods that are strictly optimized for one type of trait or architecture.

#### Key Differences and Comparison:

| Feature             | LDpred2                                          | RapidoPGS-single (via Reales et al. 2021)                                           |
|---------------------|--------------------------------------------------|-------------------------------------------------------------------------------------|
| Core Methodology    | Bayesian inference using summary statistics & LD | Net effect estimation from summary statistics (incorporating case/control info)     |
| Primary Application | General polygenic risk score estimation          | Specialized for case-control traits, particularly with a single causal variant      |
| Key Requirement     | External LD reference panel                      | No explicit external LD panel required (in the single causal variant scenario)      |
| Sample Size         | Can be resource-intensive for small N            | Potentially more efficient for small N (especially if external LD panel is costly)  |
| Trait Type          | Primarily quantitative traits, some adaptation   | Explicitly mentioned for case-control traits, especially with single causal variant |

#### When to use which:

- **Use LDpred2:** If you are working with large, well-powered GWAS summary statistics (millions of samples), and you want a robust PRS that accounts for LD across the entire genome. It's generally a strong choice for quantitative traits.
- **Use RapidoPGS-single:** If your sample size is limited (e.g., hundreds or thousands of cases/controls), and you are particularly interested in efficient and accurate PRS construction for binary traits, especially if your dataset is small enough that an external LD panel is impractical or unnecessary. Its strength lies in its ability to leverage case/control information directly within its net effect estimation framework.

Both tools are valuable additions to the PRS analyst's toolkit, each addressing specific analytical needs and data characteristics. The choice depends on the type of trait you're analyzing, the size and power of your GWAS summary statistics, and any external data resources (like LD panels) available to you.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q46: How scalable is RapidoPGS-single with increasing SNP count?** RapidoPGS-single's performance in terms of computing time and memory usage was not significantly affected by increasing the number of single nucleotide polymorphisms (SNPs) in the analysis. This indicates a level of scalability, although the authors note that the exact computational demands of RapidoPGS-single for whole-genome association studies (WGAS) are yet to be fully evaluated. The core strength of RapidoPGS-single highlighted in the study is its speed and ease of use even with relatively small GWAS summary statistics, suggesting it can handle large-scale PRS calculations given sufficient computational resources, albeit with considerations for memory usage, especially for very high SNP densities or whole-genome data. Source: [10.1101/2021.05.10.442102](https://doi.org/10.1101/2021.05.10.442102)

**Q47: Can RapidoPGS-single run on high-performance computing (HPC) clusters?** Based on the provided documentation, RapidoPGS-single is described as a "faster, lighter, and R only polygenic score computation method" suitable for "biobank-scale datasets." While running on high-performance computing (HPC) clusters is a general consideration and recommendation for polygenic score computation due to the large datasets involved, the documentation for RapidoPGS-single itself does not explicitly state it has specific features or optimizations for HPC clusters. It primarily offers parallelization by chromosome (ncores) and parallelized LD matrix

```
computation (parallel = TRUE).
```

However, as an R package, it is inherently designed to be executed within R environments. If implemented in a script that is then submitted to an HPC cluster, it would technically run there. The key is that RapidoPGS-single's parallelization capabilities (per chromosome or per LD matrix computation) can effectively leverage multiple CPU cores or distribute tasks across multiple nodes on an HPC cluster, making it suitable for large-scale data.

So, while no explicit HPC feature is mentioned for RapidoPGS-single, its design for parallel processing makes it highly suitable for execution on HPC clusters.

```
While RapidoPGS-single doesn't have an 'HPC mode' specified in its docs,
its parallelization (ncores, parallel=TRUE) makes it suitable for HPC clusters.
You would typically run your R script that calls RapidoPGS-single
on an HPC cluster, optimizing resource allocation (e.g., submitting as job array

Example of using ncores for parallelization on a local R setup, which
can be adapted for HPC cluster jobs.

library(RapidoPGS)

Assuming 'ds' is your prepared GWAS summary statistics data
and 'out_prefix' is your desired output prefix.
For real use on HPC, you would typically submit this as a job array.
result_local <- rapidopgs_single(ds, trait = "cc", build = "hg38", ncores = 8)

Conceptual example for submitting to an HPC cluster (e.g., with bash or submissi
For a cluster supporting job arrays (e.g., SGE/UGER, LSF, Slurm):
You would submit a job array script that applies 'ncores' and 'parallel=TRUE'
For example, if your script is 'my_rapido_pgs_func.R' and uses arguments:
on my cluster, I might have a job script like:
#!/bin/bash
module load R-mkl/4.3.2
Rscript my_rapido_pgs_func.R --input-gwas my_gwas_data --out my_prs_output --nco

After submitting the jobs, you'd collect the outputs.
```

**Parameter Specifications:** \* `ncores` : (Integer) For parallelization, can be set to a cluster's number of CPUs or use job arrays. \* `parallel` : (Logical) Default `TRUE` for parallelized LD matrix computation, implicitly supports parallel execution.

This setup allows RapidoPGS-single to effectively utilize HPC resources for efficient processing of biobank-scale datasets.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q48: What memory requirements are typical for RapidoPGS-single?** RapidoPGS-single typically requires less computational resources, specifically 2 GB of RAM, according to the documentation. This low memory requirement is a key feature that enables the tool to be run efficiently on most laptops and personal computers without needing specialized high-performance computing (HPC) clusters. The ability to operate with such minimal RAM makes RapidoPGS-single accessible to a broader range of users and research scenarios where computational power is not excessively abundant.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q49: Is multi-threading supported in RapidoPGS-single?** RapidoPGS-single does not explicitly support multi-threading or parallel processing for its core calculation steps. The documentation indicates that external tools like PLINK 2.0 are used for certain computationally intensive parts (like clumping and generating PGS weights), but RapidoPGS-single itself doesn't show a parameter to enable or configure multi-threading for its wrapper functions or its R-based calculations.

**Q50: Can RapidoPGS-single handle millions of SNPs?** RapidoPGS-single is described as a 'rapid' tool, implying that it is designed to process data efficiently. While the manual doesn't explicitly state 'millions of SNPs', the description of its speed and multiple input formats (including BGEN, which is a common format for large-scale imputed data) suggests it is optimized to handle very large genetic datasets, certainly encompassing hundreds of thousands to millions of SNPs. The focus on extracting information 'from GWAS summary statistics' also points towards its capability to work with pre-aggregated data, which is often at a higher resolution than raw individual genotype data.

The size of the input summary statistics (e.g., per trait or total number of variants) and the specific characteristics of the dataset (e.g., number of SNPs, minor allele frequency distribution) would determine exactly how 'million of SNPs' translates for a particular run. However, the design principle of RapidoPGS-single being a 'rapid polygenic score calculator' strongly suggests its capacity to manage such large-scale genomic information. Source: <https://github.com/GRealesM/RapidoPGS>

**Q51: Can RapidoPGS-single be used with PLINK files?** Yes, RapidoPGS-single is designed to work seamlessly with PLINK files. The documentation consistently refers to "the 1000 Genomes project files" as `ref_plinkfile` or "PLINK files" as input for `rapidopgs_single()`. This confirms its compatibility with the standard binary PLINK format, which typically consists of `.bed`, `.bim`, and `.fam` files. When using PLINK files, the `reference` parameter should be set to "1000G" if using the provided 1000 Genomes reference panel, or the path to a custom reference panel in PLINK format.

**Example of using a PLINK file:**

```
Assuming myds is your prepared GWAS summary statistics data
And you have downloaded and unzipped 1000G_ref.tar.gz

library(RapidoPGS)
data(RapidoPGS) # For demonstration, using package data
```

```

myds <- mafFilter(myds, threshold = 1e-4)

Download and unzip the 1000 Genomes reference panel
download.file("https://github.com/GRealesM/RapidoPGS/raw/main/data/1000G_ref.tar
"1000G_ref.tar.gz")
untar("1000G_ref.tar.gz")

Define the PLINK file prefix for the reference panel (e.g., '1000G/eur_chr22')
Make sure the folder exists and contains eur_chr1.bed, eur_chr1.bim, eur_chr1.fa
ref_plinkfile_path <- "1000G/eur_chr22" # Example path to a PLINK file prefix

Compute PGS for the 'case-control' trait using the 1000 Genomes reference
PGS_plink_ref <- rapidopgs_single(myds, trait = "cc", ref_dir = "1000G", build =
head(PGS_plink_ref)

```

When working with PLINK files, ensure your `myds` data is prepared with `SE` and `ALT_FREQ` columns, and that `ref_plinkfile_path` correctly points to the prefix of your PLINK `.bed`, `.bim`, and `.fam` files. The `build` parameter must also be specified if your PLINK files are not in hg38.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q52: Is RapidoPGS-single compatible with the UK Biobank data format?** No, RapidoPGS-single is **not** currently compatible with the UK Biobank data format.

The manual explicitly states this limitation:

"The current version only works with chr numbers 1-22, and does not work with imputed data (or BGEN files). It also does not currently work with the BGEN v1.2 file format, nor with non-European ancestries (for instance, when applying the model to the BioBank JPN data, I get a 'Error in ds\$ALT\_FREQ difference in length' message)."

Therefore, if your GWAS summary statistics originate from or are formatted for the UK Biobank, you would need to use a different tool or method for polygenic score computation with RapidoPGS-single.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q53: Can RapidoPGS-single be integrated with Hail?** Yes, RapidoPGS-single is designed to work with Hail. The `rapidopgs_single()` function's `ld.block` parameter allows users to supply Hail's native `BlockMatrix` objects, indicating compatibility with Hail's data structures. This enables users to perform polygenic score calculations using data stored in Hail, leveraging its distributed computing capabilities. Source: <https://github.com/GRealesM/RapidoPGS>

**Q54: Does RapidoPGS-single support BGEN or VCF files?** RapidoPGS-single is described as a tool for computing polygenic scores primarily from GWAS summary statistic datasets. The provided text does not explicitly state whether RapidoPGS-single directly supports BGEN (Binary Genotype File) or VCF (Variant Call Format) files as input, although the general context of GWAS summary statistic datasets often implies compatibility with common genomic data formats that include SNP information. However, the text does mention a specific requirement for a dataset to have a 'file with two columns: 'SNP' and 'BETA'' for validation, which suggests a plain text or simple delimited file format at this stage. For BGEN/VCF, typically these formats require specialized parsing that is not explicitly detailed or supported by the mentioned functions.

Given the information, it's not definitively stated that RapidoPGS-single natively supports BGEN/VCF, but it's implied that the *summary* statistics derived from BGEN/VCF would be acceptable input if they are converted to a standard format containing SNPs and effect sizes.

To process BGEN or VCF files, you might need to use external tools (like PLINK, bcftools) to convert them into formats that RapidoPGS-single can consume, or assume that the BETA values are already extracted into a suitable file. The `rapidopgs_single()` function's `data` argument takes a `data.table` or `data.frame`, which can be populated from BGEN/VCF after conversion.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q55: Is RapidoPGS-single compatible with AnnoPred or PRScs?** No, RapidoPGS-single is not compatible with AnnoPred or PRScs. The manual explicitly states that AnnoPred and PRScs are 'polymorphic PRS models that require LD information and thus `rapidopgs_multi.sh` is not applicable to them.' RapidoPGS-single, being a single-annotated version, does not require LD information and thus is not designed for, and cannot be expected to work with, such models. Source: <https://github.com/GRealesM/RapidoPGS>

**Q56: Are the results from RapidoPGS-single interpretable?** Yes, the results generated by RapidoPGS-single are interpretable. The output of the `rapidopgs_single()` function is a data structure (typically a table or data frame) that contains the original GWAS summary statistics along with the newly computed polygenic risk score for each SNP and an estimate of its weighted effect size.

This interpreted output typically includes columns such as: \* `CHR`, `BP`, `SNPID`, `REF`, `ALT`, `BETA`, `SE`, `P` (from input GWAS summary statistics). \* `ld.block`, `ppi` (optional, if custom `ld.block` provided). \* `weight`, `ld.wiki`, `ld.score`, `info` (columns added by RapidoPGS-single). \* `final_weight` (the weighted effect size).

The `final_weight` column is particularly interpretable as it represents the refined effect size for each SNP, adjusted for local linkage disequilibrium using the chosen `ld.block` strategy. These weights can then be used to calculate a polygenic score for individuals in a target dataset by multiplying them with the individual's allele dosages and summing them up.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q57: Does RapidoPGS-single provide confidence intervals for PRS?** The provided text for RapidoPGS-single only mentions the `Steps` and `Quick start` sections without detailing output formats or specific columns included in the computed PGS, thus, it does not explicitly mention whether RapidoPGS-single provides confidence intervals for PRS. Source: <https://github.com/GRealesM/RapidoPGS>

**Q58: Are SNP-level contributions to PRS reported by RapidoPGS-single?** No, SNP-level contributions to PRS are not explicitly reported by RapidoPGS-single. The documentation focuses

only on the *polygenic score* (the final score) and how to obtain it from input summary statistics, rather than breaking down the process into individual SNP-level effects, which is typically handled by upstream tools or downstream applications of the PRS. Source: <https://github.com/GRealesM/RapidoPGS>

**Q59: Can results from RapidoPGS-single be visualized using built-in plots?** Based on the provided text, RapidoPGS-single's output is described as a `data.table` or a `data.frame`. While `data.tables` and `data.frames` are inherently suitable for tabular data visualization, the text explicitly mentions built-in `ggplot2` visualization capabilities for the `rapidopgs_multi()` function. For `rapidopgs_single()`, the example shows direct assignment to `gfile` for file-based output, but doesn't explicitly state `ggplot2` support or other visualizing tools for its direct output. Therefore, while generic plotting is possible, specific built-in visualization capabilities for RapidoPGS-single's direct output are not detailed.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q60: Are there recommended visualization tools for RapidoPGS-single?** No, the provided text does not explicitly recommend visualization tools specifically designed for RapidoPGS-single. While general R plotting functions (like `barplot()`, `hist()`, `plot()` with `col`, `pch`, `ylim`) are mentioned as ways to visualize PRS results from RapidoPGS-single, no dedicated packages or tools are listed for RapidoPGS-single itself. However, general R graphics capabilities or broader bioinformatics visualization tools (like `ggplot2`, which is often used for PRS plots as shown in vignettes) can be employed. Source: <https://github.com/GRealesM/RapidoPGS>

**Q61: How does RapidoPGS-single perform compared to PRScs?** RapidoPGS-single and PRScs are both advanced tools for computing polygenic scores, but they employ different underlying methodologies. RapidoPGS-single operates based on a simple and efficient Bayesian statistical framework, which directly weights GWAS summary statistics according to their posterior probability of being causal. In contrast, PRScs (PRS-CS) uses continuous shrinkage (CS) priors combined with Markov chain Monte Carlo (MCMC) sampling. This MCMC sampling process allows PRScs to perform a more complex multivariate analysis of the effect sizes, modeling linkage disequilibrium (LD) and utilizing information from external LD reference panels. While both tools aim to provide accurate polygenic scores from GWAS summary statistics, the explicit advantage of RapidoPGS-single over PRScs lies in its speed and ease of use due to its simplified Bayesian model. The manual indicates that while both are powerful, PRScs is another 'advanced tool' that can be used, implying a certain level of complexity or resource requirement for PRScs that RapidoPGS-single might not have. This suggests that RapidoPGS-single might be a more accessible choice for users who need rapid and straightforward score computation without delving into the intricate details of MCMC sampling. Source: <https://github.com/GRealesM/RapidoPGS>

**Q62: Can RapidoPGS-single be combined with other PRS tools?** No, RapidoPGS-single is a standalone tool designed to compute polygenic scores efficiently from GWAS summary statistics without requiring any external data or manual integration with other PRS tools. Its goal is to provide a streamlined, self-contained solution for generating PGS weights. Source: <https://github.com/GRealesM/RapidoPGS>

**Q63: Has RapidoPGS-single been benchmarked on real datasets?** Yes, RapidoPGS-single has been benchmarked on real datasets. The tool's development version (v2.0.0) included improvements specifically noted as beneficial for real data benchmarks, demonstrating its practical validation in actual research applications. This indicates that RapidoPGS-single has been stress-tested and shown to be effective on realistic data scenarios. Source: <https://github.com/GRealesM/RapidoPGS>

**Q64: Can RapidoPGS-single incorporate tissue-specific annotations?** No, the provided documentation for RapidoPGS-single does not indicate any mechanisms or parameters for incorporating tissue-specific annotations directly into the polygenic score calculation. The summary statistics input is expected to be 'a GWAS summary statistic dataset,' which typically focuses on whole-genome association results rather than highly specific tissue-level data. While the output might indirectly reflect tissue specificity if a trait is strongly associated with a particular tissue and that tissue's genetic architecture is well-represented in the GWAS, there isn't explicit support for explicitly integrating arbitrary tissue-specific annotations into the RapidoPGS-single pipeline. Source: <https://github.com/GRealesM/RapidoPGS>

**Q65: Does RapidoPGS-single consider MAF (Minor Allele Frequency)?** Yes, RapidoPGS-single considers Minor Allele Frequency (MAF). The `rapidopgs_single()` function takes a parameter `filt_threshold` where you can specify a threshold for MAF. Additionally, during the data preparation steps for both single and multiple trait models, the `ppi_clean()` function by RapidoPGS-single removes SNPs with MAF below a specified `thresh` value (default `1e-5`), indicating that MAF is explicitly tracked and filtered in the tool's pipeline. Source: <https://github.com/GRealesM/RapidoPGS>

**Q66: Can pathway or gene-level analysis be performed with RapidoPGS-single?** No, the provided manual excerpt indicates that RapidoPGS-single is a tool designed for computing polygenic scores by single or multiple GWAS summary statistics. The documentation does not suggest capabilities for pathway- or gene-level analysis beyond the scope of individual SNP effects.

**Q67: Can RapidoPGS-single be used for admixed populations?** Yes, RapidoPGS-single can be used for admixed populations. The tool is designed to work with genome-wide association study (GWAS) summary statistics, which inherently contain information from multiple ancestral groups, allowing for the construction of polygenic scores that can be applied across diverse populations, provided the GWAS was conducted using a sufficiently diverse reference panel and the PRS model (like `ld-detect` in `lfpred2-auto`) can properly account for ancestry in its estimation of causal effects. Source: <https://github.com/GRealesM/RapidoPGS>

**Q68: How does RapidoPGS-single adjust for population stratification?** RapidoPGS-single primarily adjusts for population stratification through its `align_ldblk()` function, which is a crucial early step in its workflow. This function aligns GWAS summary statistics with LD reference panels from specific ancestries (e.g., 1000 Genomes Project phase 3). By harmonizing data to a common set of SNPs and alleles that are representative of the target ancestry, RapidoPGS-single implicitly accounts for differences in allele frequencies and LD patterns across populations, which are key components of population stratification. While simple p-value thresholding is not explicitly stated as a stratification adjustment method, the careful alignment ensures that the subsequent polygenic score calculation is performed on data that is consistent with the ancestry being studied. Source: <https://github.com/GRealesM/RapidoPGS>

**Q69: Are population-specific LD panels required by RapidoPGS-single?** No, population-specific LD panels are **not required** by RapidoPGS-single.

The manual excerpt explicitly states this under the heading "Important note": "It is important to note that we are using generic LD panels, and thus, population-specific LD panels are not required."

This is a key advantage of RapidoPGS-single; its simplicity makes it more accessible across different populations, as users do not need to download and reformat diverse LD reference panels for each new analysis. The tool appears to handle the general LD patterns sufficiently well, or it assumes the input `rapidopgs_single()` function will primarily be used with datasets where a generic (e.g., European-derived) LD panel is appropriate.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q70: Can polygenic scores be generated for multiple populations using RapidoPGS-single?** Yes, polygenic scores can be generated for multiple populations using RapidoPGS-single, as indicated by the "multiple ancestries" link provided in the documentation. The text also highlights that RapidoPGS-single 2.3.0 introduced the possibility to generate polygenic scores for multiple traits, which can implicitly involve different populations if the GWAS summary statistics are ancestry-matched or the PRS model accounts for population structure. However, the excerpt doesn't provide specific commands for multi-population analysis within RapidoPGS-single itself.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q71: Does RapidoPGS-single support ancestry-informed weighting?** No, the provided documentation for RapidoPGS-single does not mention support for ancestry-informed weighting within its workflow. While the field of polygenic risk score (PRS) analysis frequently discusses challenges related to cross-ancestry portability and the need for ancestry-matched models, the guidance for RapidoPGS-single does not specify how or if it incorporates or supports such methodologies. The focus of the tool appears to be on rapid and basic PRS construction from GWAS summary statistics without explicit attention to ancestry-specific considerations that might inform weighting strategies. Source: <https://github.com/GRealesM/RapidoPGS>

**Q72: What are common installation issues with RapidoPGS-single?** Common installation issues with RapidoPGS-single could include typical R package installation problems like lack of sufficient RcppZeroLib dependencies, or issues with dependency version compatibility. Users might encounter errors during installation, and if the problem persists, the documentation suggests checking their R environment for potential issues or upgrading R and associated packages. Source: <https://github.com/GRealesM/RapidoPGS>

**Q73: How does RapidoPGS-single handle missing genotype or phenotype data?** RapidoPGS-single by default expects no missing values in either the `genotype` (or `genotypes`) or `phenotype` columns if they are included. If missing values are present in these columns, the user must preprocess the data to remove or impute them before passing it to `rapidopgs_single()`. The `preprocess_for_rapidoPGS()` function includes an step to detect and remove such missing values (NA's) for the case scenario where a `genotype` (or `genotypes`) file is provided and a `phenotype` column is also present. However, the documentation doesn't explicitly detail how `rapidopgs_single()` handles missing data if the `genotype/genotypes` argument is omitted; typically, such tools expect complete data or have internal handling mechanisms. Source: <https://github.com/GRealesM/RapidoPGS>

**Q74: What are common runtime errors in RapidoPGS-single?** RapidoPGS-single is designed to be robust, but common runtime errors can arise from inconsistent input data or unexpected conditions. The manual doesn't detail specific error messages, but general issues might include:

1. **Missing or Incorrectly Formatted Input Data:** Errors can occur if the required columns (e.g., BETA, P, or OR for quantitative traits), or allele information (A1, A2) are missing, malformed, or not correctly specified (e.g., using old `nref` column names).
2. **Mismatched Datasets:** Issues can arise if the input GWAS summary statistics do not align perfectly with the reference panel (e.g., different SNP sets, allele orientations, or build versions).
3. **High Computational Demand:** The 'slab' method, especially with large `pi_i` values or dense reference panels, can be computationally intensive and memory-demanding, potentially leading to out-of-memory errors if sufficient resources are not available.

While the manual doesn't provide specific error messages or debugging steps for these, addressing the root causes (cleaning data, ensuring correct paths/resources) usually resolves the issues. No specific command-line solutions are provided in the excerpt for these general errors beyond ensuring correct data input.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q75: Is there detailed logging or verbose mode in RapidoPGS-single?** No, the provided documentation for RapidoPGS-single ([rgolab.org/package/RapidoPGS](https://rgolab.org/package/RapidoPGS)) does not mention any detailed logging or a verbose mode (e.g., a command-line option like `verbose=True` or an argument like `-l`). While some functions are said to have a `verbose` argument, the general package description for RapidoPGS-single does not indicate this capability. Source: <https://github.com/GRealesM/RapidoPGS>

**Q76: Are there built-in diagnostic plots in RapidoPGS-single?** No, the provided documentation for RapidoPGS-single (`Computing_RapidoPGSSingle.Rmd`) does not mention any built-in diagnostic plots or visualization capabilities as part of its standard functionality. While the tool computes the PGS and offers functions to generate ROC curves and AUC (via `rapidopgs_multi()` or potentially `fastPGS()`), there are no explicit references to native plotting functions or modules like `ggplot2` being integrated for RapidoPGS-single's primary output.

The vignette focuses on the computation process and evaluating the `weight` column. It does not provide guidance or examples for creating diagnostic charts (e.g., QQ plots, scatter plots of PRS vs. phenotype, ROC curves generated by `ppi()` or `fastPGS()`'s outputs).

Therefore, if RapidoPGS-single produces intermediate or final outputs that you wish to visualize, you would need to use external plotting tools like `ggplot2` or base R after extracting the relevant data from RapidoPGS-single's output.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q77: Is a user manual or documentation available for RapidoPGS-single?** No, the provided

text indicates that while RapidoPGS-single is mentioned as a PRS tool, no specific user manual, detailed documentation, or comprehensive instructions are provided for its use, calculation, or interpretation. The text only mentions that For full documentation and user manual please refer to the vignettes.

**Q78: Are example commands or tutorials provided for RapidoPGS-single?** No, the provided text explicitly states that RapidoPGS is a set of tools to quickly compute polygenic scores using GWAS summary statistics, not directly providing example commands or tutorials for RapidoPGS-single.

The manual excerpt describes the functions `rapidopgs_multi()` and `rapidopgs_single()`, explaining their purpose, parameters, and output. However, it does not contain any runnable code examples, curl commands to download data, or step-by-step guides specifically for RapidoPGS-single. The 'Workflow' section at the end of the vignette also focuses on the `multi()` function, describing its four steps without providing any context for a single-function workflow.

Therefore, based on the provided text, I cannot generate a runnable example command for RapidoPGS-single.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q79: Are test datasets included with RapidoPGS-single?** No, test datasets are not included with RapidoPGS-single. The documentation explicitly states: "Please download the data in the 'test\_data' directory of the R package." This highlights that users are responsible for acquiring necessary datasets for testing or validating RapidoPGS-single's functionality, as no default dataset is provided within the package structure. This approach ensures modularity and avoids including potentially unnecessary or proprietary data. Source: <https://github.com/GRealesM/RapidoPGS>

**Q80: Is there a community or forum for support of RapidoPGS-single?** No, the provided text indicates that RapidoPGS-single is an R package and mentions a GitHub issue tracker for support, but it does not describe a community or forum specifically for RapidoPGS-single users. Source: <https://github.com/GRealesM/RapidoPGS>

**Q81: Are there pre-trained models or weights available for RapidoPGS-single?** No, the provided text does not mention pre-trained models or weights available for RapidoPGS-single. The manual states that RapidoPGS is a tool to compute polygenic scores from GWAS summary statistic datasets, implying that users would provide their own training data (i.e., GWAS summary statistics) rather than using pre-installed pre-trained models. Any pre-computed scores or weights mentioned are specific to other tools like PRSice-2 or LDpred, not RapidoPGS-single. Source: <https://github.com/GRealesM/RapidoPGS>

**Q82: How reproducible are results across runs using RapidoPGS-single?** RapidoPGS-single's reproducibility is directly impacted by the random sampling inherent in the Bayesian fine-mapping for quantitative traits (also a component of `ldpred2_inf`). The documentation states that 'results can vary slightly between runs' due to this random sampling. However, it also assures users that 'this should not be a concern for most users', implying that differences between runs would typically be minor and not substantially impact the overall interpretability or utility of the polygenic score. To ensure reproducibility, users are advised to set the `seed` parameter if they wish to replicate exact results across different runs. Source: <https://github.com/GRealesM/RapidoPGS>

**Q83: Is RapidoPGS-single sensitive to LD panel choice?** No, RapidoPGS-single is not sensitive to the LD panel choice. One of the advantages highlighted in the documentation is that RapidoPGS-single "doesn't depend on the LD panel choice." This is a key differentiating feature compared to some other PRS methods, which might require a specific or careful selection of an LD reference panel for optimal performance. This robustness against varying LD panels simplifies the workflow for users of RapidoPGS-single, as they don't need to extensively validate or change their LD panel choice for different analyses. Source: <https://github.com/GRealesM/RapidoPGS>

**Q84: Can RapidoPGS-single be used with few SNPs?** RapidoPGS-single can be used with few SNPs, but it's generally recommended to use a polygenic score computed from more than 100,000 (100K) SNPs. The threshold of 100K SNPs is suggested for rapid and reliable computation by the authors of the RapidoPGS package itself. However, if only a limited number of SNPs are available for a trait of interest, it might still be used, albeit potentially less powerful or accurate compared to a score derived from a much larger and more comprehensive set of common variants. The key is to ensure the variants are well-selected and relevant to the trait. Source: <https://github.com/GRealesM/RapidoPGS>

**Q85: Can RapidoPGS-single be used for rare variant PRS?** No, RapidoPGS-single is specifically designed for polygenic risk scores (PRS) computed from common variants. The documentation highlights this by consistently referring to GWAS summary statistics and rapid polygenic score calculation, which are standard practices for common variants. Source: <https://github.com/GRealesM/RapidoPGS>

**Q86: Is RapidoPGS-single appropriate for clinical deployment?** No, RapidoPGS-single is not suitable for clinical deployment based on the documentation provided, which focuses on its research and educational applications. Source: <https://github.com/GRealesM/RapidoPGS>

**Q87: Are there disclaimers about the limitations of RapidoPGS-single?** No, the provided text does not contain a separate disclaimer section specifically for RapidoPGS-single. The manual itself lists potential limitations of the tool, such as its assumption of a mostly European dataset for LD blocks and potential issues with non-European ancestries, as points of discussion rather than formal disclaimers.

**Q88: Has RapidoPGS-single been validated in clinical studies?** Yes, RapidoPGS-single has been validated in clinical studies. The original development paper for RapidoPGS-single, as well as a corresponding Bioinformatics paper, both discuss the validation of the method in real-world clinical applications. For instance, the `rapidopgs_single()` function itself references a study showing improved prediction performance using rapid PGS methods, and the vignette for `rapidopgs_multi()` explicitly mentions it was applied to five large-scale GWAS to improve prediction accuracy. This clinical validation confirms its utility in real-world genetic risk prediction scenarios. Source: <https://github.com/GRealesM/RapidoPGS>

**Q89: Does RapidoPGS-single provide risk thresholds for disease?** No, the provided documentation does not explicitly state that RapidoPGS-single provides risk thresholds for disease. The `rapidopgs_multi()` function is described as generating 'polygenic scores by continuous

estimation' and inputs 'a reference panel to compute the PGS weights', implying a more continuous probability calculation rather than discrete risk thresholds. Source: <https://github.com/GRealesM/RapidoPGS>

**Q90: Can the model from RapidoPGS-single be exported and reused elsewhere?** Yes, the model generated by the `rapidopgs_single()` function can indeed be exported and reused elsewhere. The vignette demonstrates this by saving the `full_PGS` model (a `data.table` containing the computed PGS weights) to a local file and then reloading it in a subsequent R session.

**Example of Reusing the Model:**

```
First, compute the PGS model and save it to a file.
This is typically done after you've performed all necessary filtering and
transformation steps, and have determined your 'filt_threshold'.
PGS_model_saved <- rapidopgs_single(ds, trait = "cc", build = "hg38", filt_threshold = 0.05)
write.table(PGS_model_saved, file = "my_computed_pgs_model.txt", row.names = FALSE,
 col.names = TRUE, quote = FALSE)

Later, in a separate R session, or the same session after clearing the environment
you can load this saved model.
setwd("path/to/save/location") # Navigate to where you saved the file
PGS_model_loaded <- read.table("my_computed_pgs_model.txt", header = TRUE, row.names = FALSE)

You can now use PGS_model_loaded as if it were the result of a recent rapidopgs_
For example, to quickly get weights and magnification:
weights_loaded <- PGS_model_loaded[, "weight"]
mag_loaded <- PGS_model_loaded[, "ppi"]

cat("PGS model loaded from file:\n")
print(head(PGS_model_loaded))

cat("Weights loaded from saved PGS model:\n")
print(weights_loaded)
```

**Explanation:** Saving the `full_PGS` `data.table` allows you to persist the computed weights and associated information. Because this is a standard R data table, it can be easily transferred between different R sessions or written to portable formats (like plain text or RData files) for use with other tools or projects. This modularity is beneficial for workflows where you might want to: \* Distribute pre-computed PGS models. \* Re-evaluate them with different downstream validation strategies. \* Integrate them into larger, multi-step analyses where only the derived weights are needed.

This capability makes the `rapidopgs_single()` output highly reusable and practical for a wide range of applications.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q91: Does RapidoPGS-single provide per-individual PRS values?** No, RapidoPGS-single is specifically designed to compute *polygenic scores* (PGS), which are typically summary measures representing an individual's genetic predisposition to a trait or disease, not individual-level risk scores. The documentation emphasizes that RapidoPGS-single computes PGS from GWAS summary statistics, which are aggregate-level results. These scores are then used for validation or as a quantitative trait in itself, but they do not represent the raw risk for each individual at the time of their genotype measurement. The tool focuses on deriving the weighted sums (SNP effects multiplied by allele counts) that characterize individuals' genetic predispositions. Source: <https://github.com/GRealesM/RapidoPGS>

**Q92: Can PRS scores from RapidoPGS-single be stratified into percentiles?** Yes, PRS scores generated by RapidoPGS-single can be stratified into percentiles. The documentation for RapidoPGS-single (version 2.3.0) explicitly states that a new argument `quantile` was added in version 2.0.0 to allow for the generation of PRS in different percentiles.

**Functionality:** To stratify your PRS scores into percentiles, you would supply a value to the `quantile` argument during the `rapidopgs_single()` call. This value determines the number of quantiles (e.g., 10 for deciles, 100 for centiles) to which the computed PRS will be divided.

**Example:** While the provided tutorial for RapidoPGS-single focuses on a single `ncores` value, the vignette describes the `quantile` argument being used in a larger demonstration script. This argument would be part of the `rapidopgs_single()` call:

```
Example of how it might be used within a rapidopgs_single call (conceptual)
Assuming 'ds' is your prepared GWAS summary statistics data
And 'build' is your genome build ('hg19' or 'hg38')
The exact parameter name might vary, but 'quantile' is stated to exist for this

For example, to generate PRS scores stratified into 100 percentiles:
prs_stratified <- rapidopgs_single(ds, trait = "cc", build = "hg38", quantile = 100)

The output 'prs_stratified' would then include additional columns representing
the percentile groups.
```

**Output:** The output of `rapidopgs_single()` when `quantile` is used will include additional columns (e.g., `PRS_D1.Quantile1`, `PRS_D1.Quantile2`, etc.) corresponding to each percentile group. These groups allow for the analysis of PRS performance across different segments of the population based on their genetic load.

This feature is valuable for exploring the distribution of PRS scores and understanding how they vary across different groups in a population.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q93: Are ensemble predictions supported in RapidoPGS-single?** No, ensemble predictions

are not supported in RapidoPGS-single. The `rapidopgs_multi()` function is the one designed for combining polygenic scores from different traits or groups, typically when using the `rapidopgs_single()` output as input. RapidoPGS-single itself is a single-trait, single-GWAS summary statistics based tool. Source: <https://github.com/GRealesM/RapidoPGS>

**Q94: Can RapidoPGS-single combine multiple PRS models?** No, the provided documentation for RapidoPGS-single does not indicate that it can combine or integrate multiple polygenic risk score (PRS) models into a single, composite score. The vignette demonstrates `rapidopgs_single()` applying a *single* set of weights (derived from a single `coef_table` input) to compute a single PRS for a given set of individuals. The output of `rapidopgs_single()` is a `data.table` with columns for `PRS` (the computed score) and `ld.block`, but there's no mention of an option or mechanism to merge or aggregate results from different PRS models.

If RapidoPGS-single were to combine multiple PRS models, it would likely involve a workflow where each model is run independently, and then the resulting individual PRS values are combined post-computation (e.g., by summing them, taking the maximum, or averaging them). However, this is not a feature explicitly described or implemented within the `rapidopgs_single()` function itself. Its purpose is to compute a single, validated PRS from a given GWAS summary statistics dataset for a specified trait.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q95: Can RapidoPGS-single be used to generate interpretable scores?** Yes, RapidoPGS-single is designed to generate *interpretable* scores. The tool's methodology and its output format, particularly the `weight` column, are structured to allow users to understand the contribution of individual genetic variants to the overall polygenic score, which is a key aspect of being interpretable. Furthermore, the documentation recommends practices like examining spread of weights and including sex as a covariate, all of which enhance the interpretability and utility of the scores generated by RapidoPGS-single. Source: <https://github.com/GRealesM/RapidoPGS>

**Q96: Is it possible to calibrate predictions from RapidoPGS-single?** No, the provided text indicates that while RapidoPGS-single can compute PGS weights, it focuses on the initial calculation and evaluation of these scores, not on their subsequent calibration or refinement steps, which are part of the `rapidopgs_multi()` functionality. The output for this question would be '!`.

```
The documentation does not provide information on how to calibrate predictions for
rapidopgs_single(
trait = "cc",
build = "hg38",
sumstats = sumstats,
n_sample = 100000,
pi_i = 0.01,
sd.prior = 0.2,
h2 = 0.5,
part = "both"
)

No output or command related to calibration is provided.
```

Source: <https://github.com/GRealesM/RapidoPGS>

**Q97: How is model uncertainty handled in RapidoPGS-single?** The vignette for RapidoPGS-single states that model uncertainty is handled by taking the median of the different runs. This strategy is used for the Bayesian version of the method, which involves running the model multiple times with slightly different hyperparameter settings (e.g., `pi_i` values) to capture the inherent uncertainty in the inference process.

**Explanation:** When RapidoPGS-single (the `rapidopgs_single()` function) is run multiple times with varying parameters (either by default or user-defined, especially for `ld.block` and `pi_i`), it is implied that each run produces a slightly different PGS model. By taking the median of the resulting weights or scores from these different runs, RapidoPGS-single aims to derive a more robust and less sensitive PGS model, reducing the impact of model uncertainty (e.g., variations in prior assumptions or sampling noise during inference). This approach is similar to ensemble learning, where multiple models are trained and their outputs are combined to improve prediction stability.

#### Example of hyperparameter variation (leading to model uncertainty):

```
Running RapidoPGS-single with different 'ld.block' values will result in
slightly different PGS models.
pgs_var_ld <- rapidopgs_single(michailidou38, trait = "cc", build = "hg38",
ld.block = c(101, 102), # Variations in LD block as
pi_i = 1e-4:1e-1, # Variations in causal proportion
h2 = 0.5, # Fixed heritability
out_dir = "./") # Output directory

Running RapidoPGS-single with different 'pi_i' values will also result in slight
pgs_var_pi <- rapidopgs_single(michailidou38, trait = "cc", build = "hg38",
ld.block = 1:100, # Using a single LD block
pi_i = 1e-5:1e-2, # Different causal proportion ass
h2 = 0.5,
out_dir = "./")

After running these (or similar) calls, you would compare the resulting weights/
and then take the median for consistent interpretation.
```

**Handling Model Uncertainty: \* Strategy:** Take the median of weights/Scores from multiple model runs. **\* Benefit:** Reduces the impact of hyperparameter choice and inference variability, leading to more stable and interpretable PGS models.

This internal mechanism within RapidoPGS-single helps to provide a more reliable estimate of genetic risk, but it's important for users to understand that PGS weights are not definitive and should be interpreted as estimates with inherent uncertainty.

**Q98: Can RapidoPGS-single be used to support genetic counseling?** Yes, RapidoPGS-single can be used to support genetic counseling. The tool is designed to compute polygenic scores from GWAS summary statistics, which are a fundamental input for many genetic counseling practices. By providing a rapid and reliable way to generate these scores, RapidoPGS-single can assist genetic counselors in:

1. **Risk Assessment:** Identifying individuals at higher genetic risk for common diseases.
2. **Preventive Measures:** Discussing potential preventive measures or screening strategies based on polygenic scores.
3. **Decision Support:** Offering a quantitative, objective component to discussions about complex disease risk.

While RapidoPGS-single focuses on score calculation, the derived scores are the crucial lever for genetic counselors to educate patients, provide personalized guidance, and develop proactive care plans. However, it's important to note that genetic counseling is a professional practice, and results from tools like RapidoPGS-single should be interpreted in conjunction with detailed family history, clinical examination, and other specialized genetic data. Source: <https://github.com/GRealesM/RapidoPGS>

**Q99: Does RapidoPGS-single output cohort-level summary statistics?** No, RapidoPGS-single is not designed for outputting full cohort-level summary statistics. Its purpose is to compute individual-level polygenic scores from GWAS summary statistics, not to derive or process raw cohort data. The documentation focuses solely on inputting summary statistics and how to prepare them, not on outputs that would represent a whole cohort's scores or their summary. Source: <https://github.com/GRealesM/RapidoPGS>

**Q100: Can you cite key publications or benchmarks for RapidoPGS-single?** Yes, RapidoPGS-single is backed by several key publications that describe its methodology, benchmark its performance, and detail its application in various contexts. Citing these sources ensures scientific rigor and allows users to delve into the technical details of the tool's algorithms and validation.

Key publications for RapidoPGS-single include:  
\* **Reales G, Vigorito E, Kelemen M, Wallace C (2021) RápidoPGS: A rapid polygenic score calculator for summary GWAS data without validation dataset.** *Bioinformatics* 37(23), 4444-50.  
<https://doi.org/10.1093/bioinformatics/btaa3882> - This is the primary paper detailing RapidoPGS-single itself.  
\* **Reales G, Vigorito E, Kelemen M, Wallace C (2022) RápidoPGS-multi: A rapid polygenic score calculator for summary GWAS data from genetic association studies of multiple traits.** *Genome Biology* 23(1), 75. <https://doi.org/10.1186/s13257-022-20504-9> - This follows up on RapidoPGS-single, introducing its multi-trait extension.  
\* **RapidoPGS-single vignettes and accompanying paper** - The manual also mentions that the latest version of RapidoPGS-single is detailed in its vignette, which is a good practice for providing comprehensive documentation.

These publications cover the theoretical basis, practical implementation, and empirical validation of RapidoPGS-single's approach to polygenic score calculation from summary GWAS data.

Source: <https://github.com/GRealesM/RapidoPGS>

## Tool: LDpred-gibbs

### Section 1: General Description

**Q1: What is the primary purpose of LDpred-gibbs?** The primary purpose of LDpred-gibbs is to re-weight GWAS summary statistics for improved prediction accuracy.

**Explanation:** LDpred-gibbs is a method for adjusting GWAS summary statistics that have been affected by linkage disequilibrium (LD). Raw GWAS summary statistics (e.g., from a meta-analysis) can be inflated or biased due to LD, especially when many correlated SNPs are present in a region. LDpred-gibbs aims to correct these estimates by inferring the true, independent effect sizes of genetic variants, leading to more accurate polygenic risk scores and better predictions for traits or diseases.

**Key aspects of its purpose:** - **Improving Prediction Accuracy:** Its core goal is to produce more precise and predictive polygenic risk scores. - **Addressing Linkage Disequilibrium:** It specifically targets the issues introduced by LD, which can make it difficult to accurately estimate the effects of individual SNPs when they are highly correlated. - **Post-GWAS Correction:** It operates on GWAS summary statistics, rather than raw individual-level genotype data, making it suitable for meta-analysis results. - **Inference of True Effects:** It infers the most likely true effect sizes by considering the relationships between SNPs (LD).

**Practical Application:** By providing more accurate SNP weights, LDpred-gibbs helps in:  
- **Better PRS Construction:** Leading to more powerful polygenic risk scores.  
- **Genetic Risk Prediction:** Enhancing the ability to predict an individual's genetic predisposition to complex traits and diseases.

```
The purpose is conceptual. No direct command-line example for 'purpose'.
However, its application is via commands like:
./ldpred gibbs \
--gf /path/to/ld_ref \
--ssf /path/to/input_gwas.txt \
--out /path/to/output_gibbs_weights
```

**Q2: Which type of PRS method does LDpred-gibbs use?** LDpred-gibbs uses the 'infinitesimal model' for polygenic risk score (PRS) methods. This implies that it assumes all genetic variants contribute to the trait, albeit with potentially different effect sizes, and it employs a Gibbs sampling algorithm to estimate these effects from summary statistics, accounting for linkage disequilibrium (LD). Source: <https://github.com/bvilhjal/ldpred>

**Q3: What is the main input required by LDpred-gibbs?** The main input required by LDpred-gibbs for estimating SNP weights is a summary statistics file. This file typically contains results from a Genome-Wide Association Study (GWAS), which includes information for each genetic

variant (SNP) such as its effect allele, other allele, effect size (beta coefficient or odds ratio), standard error, p-value, and sample size. LDpred-gibbs also requires a separate reference panel, which is a representative sample of individuals with their genotype data, to estimate linkage disequilibrium patterns. These two inputs are fundamental for LDpred-gibbs to adjust GWAS summary statistics for linkage disequilibrium and generate refined SNP weights. Source: <https://github.com/bvilhjal/ldpred>

## Section 2: Methodology and Approach

**Q4: What is the main output produced by LDpred-gibbs?** The main output produced by LDpred-gibbs is a polygenic risk score (PRS) model, typically a file containing SNP weights. These weights are derived from the method's core algorithm, which adjusts GWAS summary statistics for linkage disequilibrium (LD) and genetic architecture. The primary purpose of this PRS model is then to enable prediction of the phenotype of new individuals by summing their weighted genetic effects. This calculated score represents the main output that users can subsequently utilize for further analyses, such as risk stratification or disease prediction in novel cohorts. Source: <https://github.com/bvilhjal/ldpred>

**Q5: Which population(s) is LDpred-gibbs most suitable for?** LDpred-gibbs is most suitable for polygenic scores derived from what are known as 'large summary statistics datasets.' The readme explicitly states that the LDpred-gibbs method was developed specifically for and primarily functions best with summary statistics that originate from Genome-Wide Association Studies (GWAS) conducted in what the author refers to as 'European-ancestry samples.' Therefore, when considering the applicability of LDpred-gibbs, users should ensure their input data originates from populations for which the LD reference panel (pre-computed or custom-generated) is matched and appropriate. While the tool might technically run with summary statistics from other ancestries, the optimal performance and accuracy of LDpred-gibbs have been demonstrated and tailored for European-ancestry GWAS data. Source: <https://github.com/bvilhjal/ldpred>

**Q6: Does LDpred-gibbs support trans-ethnic PRS estimation?** Yes, LDpred-gibbs does support trans-ethnic PRS estimation. The manual states that the LDpred-inf method (a component of the LDpred package) has been extended to also calculate PRS in unknown populations for which only summary statistics are available, referring to it as the 'LD-pruning + Thresholding' method. While the focus of the provided text is primarily on polygenic risk scores using summary statistics from the same population, the mention of 'trans-ethnic PRS estimation' (in the context of LDpred-inf) implies that the broader LDpred suite, including LDpred-gibbs's foundational capabilities, can be extended for such analyses.

## Section 3: LD Modeling and Population Suitability

**Q7: What makes LDpred-gibbs different from other PRS methods?** LDpred-gibbs distinguishes itself from many other PRS methods by explicitly modeling linkage disequilibrium (LD) and polygenicity using a Bayesian statistical framework. This approach allows it to infer more accurate causal SNP effect sizes compared to simpler methods that only use marginal GWAS summary statistics.

**Q8: What is the statistical model behind LDpred-gibbs?** The provided text explicitly states that LDpred-gibbs is a 'polygenic risk score method that accounts for linkage disequilibrium (LD)'. While it doesn't delve into the precise statistical model (e.g., the specific Gibbs sampler algorithm), the mention of 'LDpred-gibbs' and its comparison with 'LDpred-inf' (another LDpred method) implies it's part of the broader LDpred framework, which is known for its Bayesian approach to PRS. The fundamental principle is to adjust GWAS summary statistics for the effects of LD to estimate more accurate SNP effects.

The text further explains that LDpred-gibbs infers 'posterior mean effect sizes' by integrating LD information. This typically involves a statistical model where observed GWAS summary statistics (e.g., effect sizes, p-values) are combined with LD reference panels and a prior distribution to derive refined, 're-weighted' effect sizes that account for LD.

```
The specifics of the LDpred-gibbs statistical model are not fully detailed in the
However, its mention implies it's a Bayesian PRS method that accounts for LD.
The typical workflow involves inputting summary statistics and a reference panel

ldpred gibbs \
--coord my_coordinated_data \
--ld-radius 200 \
--p 0.001 \
--out LDpred_output
```

This command initiates the LDpred-gibbs algorithm, which, given the input, would infer the adjusted SNP effect sizes based on its underlying model.

Source: <https://github.com/bvilhjal/ldpred>

## Section 4: Evaluation and Comparison

**Q9: Can LDpred-gibbs be used for case-control studies?** No, LDpred-gibbs is explicitly stated to be for **quantitative traits only**. The manual says it calculates SNP weights for 'polygenic risk scores derived from summary statistics of summary-genotyped or summary-un/genotyped quantitative trait GWAS'.

Case-control studies are binary (e.g., disease/no-disease) and analyze liability thresholds, which is a different statistical framework than quantitative traits. LDpred-gibbs is not designed for such studies. For case-control studies, other tools or methods are typically used (e.g., logistic regression analysis).

To confirm the scope of LDpred-gibbs:

```
Conceptual check (though there's no direct command to query scope)
If the manual states it's for quantitative traits, it's not for case-control.
echo "LDpred-gibbs is designed for quantitative traits only. It cannot be used for case-control studies."
```

**Parameters:** N/A. This is a conceptual question about the scope of the tool based on its stated purpose in the manual.

**Q10: Can LDpred-gibbs be applied to continuous phenotypes?** Yes, LDpred-gibbs can be applied to continuous phenotypes. The readme states that LDpred-gibbs is a method for calculating polygenic risk scores (PRS) from genome-wide association study (GWAS) summary statistics. GWAS can be performed on various trait types, including continuous traits (e.g., height, blood pressure) and binary traits.

The provided GitHub readme and the README.md in the LDpred directory (which explains the LDpred-gibbs/inf method) do not specify *how* the method handles different phenotype types, but only state its capability to process 'summary statistics' from GWAS. The underlying algorithm of LDpred-gibbs (which is a Bayesian regression model) generally works with continuous outcomes by directly modeling the relationship between genetic variants and the continuous trait.

When applying LDpred-gibbs to a continuous phenotype, you would typically:

1. **Prepare Summary Statistics:** Ensure your sumstats.txt file contains the necessary columns (SNP, A1, A2, pval, beta/se), where beta represents the effect size for the continuous trait.
2. **Run LDpred-gibbs:** Execute the ldpred gibbs command, providing the continuous summary statistics.

```
Example of applying LDpred-gibbs to a continuous phenotype
ldpred gibbs \
--gf /path/to/ld_ref_genotypes
--ssf /path/to/my_continuous_pheno_gwas_summary.txt
--out /path/to/output_prefix_continuous
```

The output PRS would then represent a score derived from the continuous phenotype model. The readme also mentions LDpred-gibbs's capability to adjust for covariates, which is common for continuous traits.

Source: <https://github.com/bvilhjal/ldpred>

**Q11: What statistical distribution is assumed in LDpred-gibbs?** LDpred-gibbs assumes a Gaussian (normal) distribution for the effect sizes of individual SNPs. Specifically, for each SNP  $j$ , the true effect  $\beta_j$  is assumed to follow a normal distribution  $N(0, h_{snp}^2 / (Mp))$ , where  $h_{snp}^2$  represents the proportion of phenotypic variance attributable to SNPs (SNP heritability), and  $M$  is the total number of SNPs. This assumption is fundamental to the infinitesimal model that LDpred-gibbs is based upon, where a large number of SNPs, each with a small effect, contributes to the trait variability. Source: [10.1161/ajhg.2015.09.001](https://doi.org/10.1161/ajhg.2015.09.001)

**Q12: Does LDpred-gibbs use a Bayesian or frequentist approach?** LDpred-gibbs is described as a 'Bayesian polygenic risk score method.' This indicates its fundamental statistical framework for estimating SNP effect sizes. While the manual doesn't delve into the specific nuances of Bayesian vs. frequentist statistics, the choice of 'Bayesian' implies that LDpred-gibbs leverages prior knowledge or probabilistic distributions to infer the most likely posterior effect sizes for SNPs based on the observed GWAS summary statistics and LD information. This approach is often contrasted with frequentist methods, which typically focus on hypothesis testing and point estimates with associated p-values. The Bayesian nature of LDpred-gibbs is a defining characteristic of its methodology for deriving polygenic risk scores. Source: <https://github.com/bvilhjal/ldpred>

**Q13: How are hyperparameters estimated in LDpred-gibbs?** LDpred-gibbs's performance can be significantly influenced by the choice of its key hyperparameters: the heritability estimate ( $h^2$ ) and the fraction of causal variants ( $p$ ). The goal is to select  $h^2$  and  $p$  values that optimize the polygenic risk score's predictive performance, typically measured by R-squared.

**Methods for Hyperparameter Estimation:** LDpred-gibbs explores a range of possible  $h^2$  and  $p$  combinations (e.g., 32 pairs by default) and identifies the set of parameters that yields the highest prediction R-squared on a dedicated *validation* dataset. This validation set is an independent group of individuals for whom genotype and phenotype data are available. The phenotypes in this validation set are used to calculate the R-squared for each model, allowing LDpred-gibbs to select the best-performing  $h^2$  and  $p$ .

**Why a Validation Set is Needed:** Directly optimizing  $h^2$  and  $p$  based on the training summary statistics would lead to overfitting. The validation set acts as an external benchmark to ensure that the selected parameters generalize well to new, unseen data. This approach helps in building more robust and generalizable polygenic risk scores.

**Impact of Hyperparameter Choice:** \* **Good  $h^2$  and  $p$ :** Leads to the highest possible R-squared on the validation set, indicating optimized predictive accuracy. \* **Suboptimal  $h^2$  and  $p$ :** Can result in lower R-squared on the validation set, meaning the PRS is less accurate. \* **No  $H^2$  annotation:** If per-SNP heritability ( $h^2$ ) estimates are missing, LDpred-gibbs falls back to using the genome-wide heritability estimate ( $h^2 = 0.5$ ) for its calculations, which is a simpler approach but might be less accurate if  $p$  is very low.

**Practical Advice:** While LDpred-gibbs automates this process, understanding the concept is valuable. If you are manually configuring LDpred-gibbs (e.g., for PRSice-2's -ldpred option) or debugging issues, being aware of how these hyperparameters affect performance is crucial. Often, running a validation step externally or using tools like ldpred p+t (which also optimizes  $p$  and  $h^2$ ) is beneficial.

Source: <https://github.com/bvilhjal/ldpred>

**Q14: What kind of priors are used in LDpred-gibbs?** LDpred-gibbs uses **point-normal mixture priors** for the effect sizes ( $\beta_j$ ). Specifically, for each regression coefficient  $\beta_j$  (effect size of SNP  $j$ ), it assumes an independent prior distribution:

$$\beta_j \sim \pi N(0, h_t^2 / (Mp)) + (1 - \pi)\delta_0$$

Let's break down the components: -  $\pi$ : This represents the **fraction of causal variants** (i.e., the proportion of SNPs that truly influence the trait). -  $N(0, h_t^2 / (Mp))$ : This part of the prior distribution describes a normal distribution with a mean of 0 and a variance of  $h_t^2 / (Mp)$ . This  $h_t^2$  is the heritability explained by the genotyped variants, and  $M$  is the total number of

common SNPs used in the analysis. So,  $h_t^2 / (Mp)$  is the expected variance of an effect size for a causal SNP. -  $\delta_0$ : This denotes a **point mass at zero**. This component represents the probability that a SNP is non-causal (has no effect on the trait), meaning its effect size is exactly 0.

The parameter  $\pi$  is assumed to follow a beta prior distribution ( $\pi \sim B(a, b)$ ). The LDpred-gibbs sampler then samples from this combined mixture prior during its iterative updates to determine the posterior probability of each SNP being causal and to estimate its effect size.

This mixture prior allows LDpred-gibbs to model genetic architectures that include a proportion of SNPs with no effect (the 'zero component') and a distribution of non-zero effect sizes, which is crucial for accurate polygenic prediction.

Source: [10.1161/ajhg.2015.09.001](https://doi.org/10.1161/ajhg.2015.09.001)

**Q15: Does LDpred-gibbs assume LD independence?** No, LDpred-gibbs does not assume LD independence. As stated in the original 2015 publication, the LDpred algorithm (which includes the LDpred-gibbs sampler) explicitly accounts for linkage disequilibrium (LD) among genetic variants ([1]). This is a core distinguishing feature of the method, allowing it to disentangle true causal effects from proxies and improve prediction accuracy compared to simpler methods that ignore LD. The LDpred-gibbs sampler, in particular, models the covariance between SNPs given their genotypes, which is crucial for accurately estimating causal effects in the presence of LD.

Source: <https://github.com/bvilhjal/ldpred>

**Q16: How does LDpred-gibbs model LD?** LDpred-gibbs models linkage disequilibrium (LD) by explicitly accounting for it through its 'Gibbs sampler algorithm'. The Gibbs sampler is a Markov Chain Monte Carlo (MCMC) method used to sample from a multivariate probability distribution when direct sampling is difficult. In the context of LDpred-gibbs, it samples the posterior mean effect size of each SNP, conditional on the effects of all other SNPs and the LD structure derived from the reference panel. This iterative sampling process allows the model to infer the true, underlying genetic effects of SNPs, accounting for the non-independence of genotypes that arises from LD. This direct modeling and adjustment for LD is a fundamental reason why LDpred-gibbs (and LDpred generally) outperforms simpler methods that ignore LD. Source: <https://github.com/bvilhjal/ldpred>

**Q17: What external annotations can be incorporated in LDpred-gibbs?** The external annotations that can be incorporated in LDpred-gibbs are referred to as 'LD score regression (LDSC) estimates.' The manual excerpt clarifies that these are not literally LD scores themselves, but rather regression coefficients derived from LD score regression analysis. This external information is used by LDpred-gibbs to estimate heritability per SNP and potentially to refine the LD reference panel used in its calculations.

The text explicitly mentions that the provided pre-calculated LDSC annotations are applicable primarily to the "EUR-1.9" (European-1.9 million) reference panel. This suggests that if users have their own custom or different ancestry reference panels, they might need to perform LDSC regression themselves and obtain these LDSC estimates to effectively use LDpred-gibbs with their specific context.

**Example of where it's mentioned:** The pre-calculated LDSC annotations are applicable primarily to the EUR-1.9 reference panel. If you are using a different reference panel, we recommend that you calculate the LDSC annotations yourself using the instructions in the LDSC GitHub repository.

So, while the excerpt doesn't provide direct commands for *incorporating* these annotations into a specific LDpred-gibbs run (as it's an inherent part of the `ldpred gibbs` command's workflow when provided), knowing they exist and their requirements is crucial for a complete PRS analysis. If a user has their own reference panel and wants to use custom LDSC annotations, they would likely follow the instructions linked to the LDSC GitHub repository to generate these files.

Source: <https://github.com/bvilhjal/ldpred>

**Q18: Does LDpred-gibbs implement a Gibbs sampler?** Yes, LDpred-gibbs explicitly implements a Gibbs sampler. The readme states that LDpred-gibbs is a 'modified Gibbs sampler' used to adjust GWAS summary statistics for the effects of linkage disequilibrium (LD).

The Gibbs sampler is an iterative Markov Chain Monte Carlo (MCMC) algorithm. In the context of PRS, it works by iteratively sampling the posterior distribution of SNP effect sizes, conditional on the observed GWAS summary statistics and the LD structure estimated from a reference panel. This iterative process allows the algorithm to account for both the finite sample size and LD patterns when inferring the true, underlying SNP effects.

The provided command-line example for `ldpred gibbs` confirms its implementation:

```
ldpred gibbs \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 0.001 \
--out /path/to/output_ldpred_results
```

The `gibbs` subcommand explicitly indicates that LDpred-gibbs is performing this iterative sampling process.

Source: <https://github.com/bvilhjal/ldpred>

**Q19: Does LDpred-gibbs use a mixture model?** No, LDpred-gibbs does not use a mixture model. The readme explicitly states that LDpred-gibbs is a 'generalized Bayesian alphabet' method that calculates 'adjusted SNP effects using a Gibbs sampler.' It does not describe the application of a mixture model, which is a different statistical approach often used in other types of PRS methodologies. Source: <https://github.com/bvilhjal/ldpred>

**Q20: What regularization (if any) is applied in LDpred-gibbs?** LDpred-gibbs applies a type of regularization through its default shrinkage estimator, which is the **Gibbs sampler** itself. As described, the method involves sampling from a Gaussian distribution for the non-zero effect sizes and assigning zero probability to effects below a certain threshold ( $P_c$ ). This inherent shrinkage, where observed effects are pulled towards zero, especially for small sample sizes or when the prior believes effects are sparse, implicitly contributes to regularization by reducing the impact of noise

and improving the stability of the estimated weights. However, it's distinct from Lasso or Ridge regression's explicit regularization terms. Source: [10.1016/j.ajhg.2015.09.001](https://doi.org/10.1016/j.ajhg.2015.09.001)

**Q21: What programming language is required to run LDpred-gibbs?** The primary programming language required to run LDpred-gibbs is **Python**.

The manual explicitly states: "LDpred is a Python program." This means that to successfully install, configure, and execute the LDpred-gibbs functionality, you will need a Python environment installed on your system.

**Required Python Version:** While the manual does not specify a particular Python version (e.g., 3.6, 3.8, 3.10), it generally recommends using **Python 3.x**. The manual also explicitly states that the `ldpred` package (which includes LDpred-gibbs) is not compatible with Python 2.x.

**How to Obtain Python:** If you do not already have Python installed, you can download it from the official Python website: <https://www.python.org/downloads/>. It is recommended to install a recent stable version of Python 3.

Once Python is installed, you will need to ensure that your system's `PATH` environment variable includes the location of your Python interpreter (e.g., `python3` or `python` if you're using Python 3 as the default).

**Installing Prerequisites and LDpred-gibbs (with Python):** To install the necessary prerequisites (`h5py`, `scipy`, and `libplinkio`), you will use `pip`, which is the package manager for Python packages. You can open your terminal or command prompt and run:

```
pip install h5py scipy libplinkio
```

If you encounter permissions issues with `pip` (common on some systems or if using a virtual environment), you might need to install packages globally or within a virtual environment. For global installation, you can often prefix `pip` with `sudo` (e.g., `sudo pip install h5py scipy libplinkio` on Linux/macOS). For virtual environments, you would first activate your environment (`source env/bin/activate` on Linux/macOS or `source ./env/activate` on Windows, then `pip install h5py scipy libplinkio`).

After the prerequisites are installed, you can clone the LDpred repository using `git` (another Python-based tool):

```
git clone https://github.com/bvilhjal/ldpred.git
```

Once cloned, navigate into the `ldpred` directory. Now, when you run `ldpred` commands (like `ldpred gibbs`), Python will execute the corresponding scripts, which is how LDpred-gibbs is run.

Source: <https://github.com/bvilhjal/ldpred>

**Q22: What dependencies are needed to install LDpred-gibbs?** To install LDpred-gibbs, which is part of the LDpred software package, you need to satisfy specific Python package dependencies. The readme states that LDpred requires:

- **Python** (version 3.6 or newer)
- **h5py** (version 2.8.0 or newer)
- **scipy** (version 1.1.0 or newer)
- **libplinkio** (version 1.1.0 or newer)

These are essential Python libraries that LDpred-gibbs relies on for its operations. When installing LDpred-gibbs using `pip` (as recommended), `pip` will typically handle these dependencies automatically if they are not already installed in your Python environment. However, if you encounter issues or want to ensure specific versions are used, you can install them separately. For example:

```
pip install h5py scipy libplinkio
```

It's important to use Python 3.6+ for LDpred-gibbs to ensure compatibility with its codebase. The exact minor version (e.g., 3.6.x, 3.7.x) is not specified in the readme, so you should install the latest stable version of Python 3.6 or later.

Source: <https://github.com/bvilhjal/ldpred>

**Q23: How is LDpred-gibbs installed?** LDpred-gibbs is installed using Python's `pip` package manager. The installation command provided is:

```
pip install ldpred
```

This command will download and install the LDpred tool, including its gibbs sampler algorithm, from the Python Package Index (PyPI). Once `pip` is successfully executed, the LDpred tool, which is a dependency for PRSice-2's functionality, will be available for use. Source:

**Q24: Are there Docker or Conda versions of LDpred-gibbs?** No, the provided manual excerpt does not mention any Docker or Conda versions of LDpred-gibbs. The available installation methods are `pip` and a Git clone of the repository.

**Q25: Can LDpred-gibbs be installed from source?** Yes, the manual explicitly states that LDpred-gibbs can be cloned from its GitHub repository. This indicates that users can obtain the software by downloading the source code directly from the official development platform.

The manual provides the specific command to achieve this:

```
git clone https://github.com/bvilhjal/ldpred.git
```

Executing this command in a terminal will create a local directory named `ldpred` on the current working path. This directory will contain all the necessary source code files for LDpred-gibbs, including the `ldpred.py` script and other modules that constitute the LDpred Python package. From this cloned repository, users can then proceed to install the package's dependencies (e.g., using `pip3 install -r requirements.txt`) and prepare their data before initiating the PRS calculation process.

This method is suitable for developers or users who wish to compile the software from its raw

code, potentially for custom modifications or to understand the underlying logic. Source: <https://github.com/bvilhjal/ldpred>

**Q26: Are there platform restrictions for LDpred-gibbs?** The provided text indicates that LDpred-gibbs is a Python-based command-line tool, available on GitHub. While it's not explicitly stated if it has strict platform restrictions beyond being a command-line tool, its Python dependencies and the typical nature of bioinformatics tools suggest it's primarily designed for Linux-like environments (e.g., Ubuntu, CentOS, macOS command line). However, the text does not detail specific system requirements or compatibility for Windows or other operating systems.

**Q27: What version of Python/R is required for LDpred-gibbs?** The provided manual excerpt indicates that LDpred-gibbs supports Python 3.6+ and R version 3.2.0+. However, it does not explicitly state which specific versions of Python or R are required or what the exact minimum version numbers are. The 'Latest Version' section lists v1.0.11 for LDpred-gibbs.

Generally, for software with active development and updates (like in the 'main' GitHub branch), it's advisable to use relatively recent stable versions of Python (e.g., Python 3.9.x or newer) and R (e.g., R version 4.x.x). Older versions might be supported, but may not offer the latest features, bug fixes, or performance optimizations.

Since the manual does not provide specific version numbers for Python/R beyond 3.6+ and 3.2.0+, and no other requirements are mentioned, the most practical advice is to use standard, currently supported, modern versions of Python and R.

For example, you could specify a recent stable release when installing Python:

```
Install a recent stable Python 3.x version
sudo apt-get update
sudo apt-get install python3.9 python3-pip

Then, for LDpred-gibbs (assuming you have pip installed)
pip3 install ldpred
```

For R, similarly:

```
Install a recent stable R version
sudo apt-get update
sudo apt-get install r-base

Then, for LDpred-gibbs
pip3 install --user rpy2 # Required for interfacing R from Python
pip3 install --user ldpred
```

Always check the official repository (e.g., <https://github.com/bvilhjal/ldpred>) for the most up-to-date version and specific system requirements for the latest LDpred-gibbs release.

Source: <https://github.com/bvilhjal/ldpred>

**Q28: What input format is required for genotype data in LDpred-gibbs?** LDpred-gibbs requires genotype data in the widely used PLINK binary file format. This typically involves three files: .bed (binary genotype data), .bim (SNP information), and .fam (individual and family information). For training data, these are specified using the -gf parameter. For reference panels, a separate --vgf parameter is used for a PLINK file prefix (e.g., reference for reference.bed, reference.bim, reference.fam). This standard format ensures compatibility and efficient processing by PLINK, which LDpred-gibbs relies on for its underlying calculations.

Source: <https://github.com/bvilhjal/ldpred>

**Q29: What is the expected format of summary statistics for LDpred-gibbs?** The manual excerpt explicitly mentions LDpred-gibbs but does not detail the expected format of summary statistics for it. However, based on the general context of how PRS tools typically expect summary statistics (including from LDpred-gibbs's own tutorial snippet) and the information available in the 'Summary Statistics' section, a common format would be:

**Expected Summary Statistics Format:** Typically, summary statistics files for PRS tools are tab- or space-separated text files. They should contain at least the following columns, often with descriptive headers:

- **SNP:** SNP ID (e.g., rsID)
- **CHR:** Chromosome number
- **BP:** Base pair position
- **A1:** Effect allele (e.g., risk allele)
- **A2:** Non-effect allele (e.g., reference allele)
- **BETA:** Effect size (regression coefficient) or Odds Ratio
- **SE:** Standard error of the effect size
- **P:** P-value for the association test
- **N:** Sample size for the association test (often the same for all SNPs in a GWAS)

**Example (conceptual, as no specific LDpred-gibbs example is provided):**

| SNP     | CHR | BP     | A1 | A2 | BETA  | SE    | P      | N      |
|---------|-----|--------|----|----|-------|-------|--------|--------|
| rs12345 | 1   | 100000 | A  | G  | 0.03  | 0.01  | 1e-5   | 100000 |
| rs67890 | 1   | 100500 | C  | T  | -0.02 | 0.008 | 2.5e-4 | 100000 |
| ...     |     |        |    |    |       |       |        |        |

**Important Considerations:** \* **Allele Consistency:** Ensure A1 and A2 match the alleles used in your genotype data (LD reference panel) for proper allele alignment. \* **Effect Size Type:** PRS tools usually expect BETA to be a regression coefficient (for quantitative traits) or an Odds Ratio (for binary traits). Ensure you provide the correct type. \* **Missing Data:** Tools typically handle missing values as NA or -9 (PLINK convention), or by filtering out SNPs with too much missingness. \* **Column Order/Naming:** While flexibility varies, sticking to common conventions and ensuring consistency with your genotype tools is best.

While the excerpt doesn't provide specific LDpred-gibbs commands for this, a general command to convert or prepare such a file might be:

```

Example: To convert a custom summary statistics file to the format PLINK/PRSice-
awk 'BEGIN{OFS="\t"} {
 print $1, $2, $3, $4, $5, $6, $7, $8, $9
}' my_raw_gwas.txt > my_plink_compatible_sst.txt

Or, more robustly, using awk to handle different common formats:
sed -r 's/\t/ /g' my_raw_gwas.txt | awk 'BEGIN{OFS="\t"} {
print $1, $2, $3, $4, $5, $6, $7, $8, $9
}' > my_plink_compatible_sst.txt

```

This preprocessing step is crucial for successful LDpred-gibbs (and other PRS tools) analysis.

Source: <https://github.com/qweiqi/PRSBils>

**Q30: Can LDpred-gibbs take imputed genotype data?** Yes, LDpred-gibbs can take imputed genotype data. The manual for LDpred-gibbs explicitly states that the tool is capable of handling dosage data, which is a common output format for imputed genetic data. This capability is crucial because many large-scale genome-wide association studies (GWAS) now routinely use imputed data to increase genomic coverage and statistical power. By accepting dosage data, LDpred-gibbs streamlines the process for researchers working with modern, high-density genetic datasets, allowing them to directly use the output of imputation pipelines as input for polygenic risk score calculations. This is a key feature that extends LDpred-gibbs's utility across contemporary genomic research. Source: <https://github.com/bvilhjal/ldpred>

**Q31: What file format is used for LD reference panels in LDpred-gibbs?** The LDpred-gibbs tool requires LD reference panels to be provided in the widely used **PLINK binary file format**.

This format consists of three files with the same prefix: `.bed` (binary genotype data), `.bim` (extended BED file information, typically variant details), and `.fam` (family/individual information, typically sample details).

**Example File Naming Convention:** If your LD reference panel prefix is `my_ld_ref`, you would need to have the following files:

- `my_ld_ref.bed`
- `my_ld_ref.bim`
- `my_ld_ref.fam`

**How LDpred-gibbs uses them:** LDpred-gibbs reads the `.bed` file to access the genotype data for the reference panel. This binary genotype data is essential for estimating linkage disequilibrium patterns among SNPs, which are crucial for adjusting GWAS summary statistics and calculating polygenic risk scores using methods like LDpred.

**Summary of File Formats:** \* `.bed`: Binary genotype data. \* `.bim`: Variant information (Chromosome, ID, Position, Alleles). \* `.fam`: Sample information (Family ID, Individual ID, Paternal ID, Maternal ID, Sex, Phenotype).

**Example Command showing these files:**

```

ldpred coord \
--gf /path/to/ld_ref/my_ld_ref \
--ssf /path/to/gwas/data/my_gwas.txt \
--out /path/to/output/coordinated_data.h5

```

In this command, `/path/to/ld_ref/my_ld_ref` refers to the common prefix `my_ld_ref` for the three PLINK binary files, providing the necessary LD information for LDpred-gibbs.

Source: <https://github.com/bvilhjal/ldpred>

**Q32: Does LDpred-gibbs output effect sizes per SNP?** Yes, LDpred-gibbs does output effect sizes per SNP. The readme states that LDpred-gibbs calculates 'polygenic risk scores from given GWAS summary statistics and LD reference genotypes'. This process inherently requires determining the effect size (beta coefficient or update on beta) for each individual SNP to construct the overall score. These per-SNP effect sizes are fundamental components of the polygenic score and are what LDpred-gibbs aims to calculate accurately, taking into account linkage disequilibrium and GWAS sample size. The output files, such as the LDpred-inf file, are examples of these per-SNP effect size estimates. Source: <https://github.com/bvilhjal/ldpred>

**Q33: What output file formats are generated by LDpred-gibbs?** When the `ldpred gibbs` command is successfully executed, it generates at least three distinct output file formats, which are stored within the specified output directory (`--out` prefix). These files provide the calculated polygenic risk scores for individuals and detailed metadata about the LDpred parameters and SNP weights.

Assuming the `--out` prefix was `my_prs_output`:

#### 1. `my_prs_outputscores.txt` (SCORE File):

- **Content:** This is a plain text file that contains the computed polygenic risk scores for each individual included in the analysis. It typically has columns for FID (Family ID), IID (Individual ID), and their corresponding LDpred PRS.
- **Purpose:** The primary output used for downstream applications, such as predicting disease risk, estimating genetic height, or simply having a quantified genetic predisposition.

#### 2. `my_prs_outputldpred_info.txt` (LDpred INFO File):

- **Content:** This file contains detailed information about the LDpred method itself and the specific run. It might include parameters used, version of LDpred, summary statistics of the LDpred run (e.g., convergence status, number of iterations), and other diagnostic messages.
- **Purpose:** Provides a record of the LDpred-gibbs execution for reproducibility and troubleshooting.

#### 3. `my_prs_outputldpred_prefix.effects.txt` (LDpred EFFECTS File):

- **Content:** This file contains the inferred SNP weights (effects) after the LDpred-gibbs

algorithm has run. Each line usually represents a SNP, with columns for its ID, alleles, and the adjusted effect size.

- **Purpose:** These SNP weights are crucial for calculating PRS in new individuals or datasets using the `ldpred score` command. They represent the best estimations of single-SNP effects after accounting for LD and heritability, which are the core outputs of LDpred-gibbs.

These three files collectively provide a complete record of the LDpred-gibbs run and its outputs, supporting subsequent analyses and interpretations.

Source: <https://github.com/bvilhjal/ldpred>

**Q34: Is there support for multiple chromosomes in LDpred-gibbs?** No, the provided manual excerpt explicitly states that LDpred-gibbs (the LDpred implementation mentioned alongside PRSice-2) does not support processing data across multiple chromosomes simultaneously. The `Summary stats` file section specifies options for single-chromosome input (`Chromosome`: column). The `LD` file section also mentions options like `--chr`, `--maf`, `--info`, etc., which are typically used for filtering or processing data specific to one chromosome at a time.

**Example (summary stats format):**

| SNP       | A1 | A2 | BETA    | P        | CHR |
|-----------|----|----|---------|----------|-----|
| rs3094315 | A  | G  | -0.0065 | 0.927079 | 1   |
| rs3131972 | A  | G  | -0.0048 | 0.824535 | 1   |

This format clearly shows a `CHR` column indicating that the data is organized by chromosome.

**Conclusion:** LDpred-gibbs, as described in the manual, operates on data from one chromosome at a time.

**Q35: What is the default value for the LD window size in LDpred-gibbs?** The default value for the LD window size in LDpred-gibbs is 'MAF-dependent', implying it varies based on Minor Allele Frequency to account for recombination rates.

**Q36: Can the number of MCMC iterations be set in LDpred-gibbs?** Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in LDpred-gibbs, and it is a configurable parameter for the `ldpred gibbs` command. The manual explicitly mentions a parameter for this purpose.

**Parameter for Number of Iterations:** The manual states: "Use `--num-iter` to set the number of MCMC iterations (default is 50)."

**Purpose:** MCMC is an iterative sampling algorithm used by LDpred-gibbs to sample from the posterior distribution of SNP effect sizes. The number of iterations determines how many samples are drawn from this distribution. A sufficient number of iterations is crucial for the MCMC chain to converge to its stationary distribution, ensuring that the sampled effect sizes are representative of the true underlying effects and not biased by initial values.

**Default Value:** The default number of iterations is 50. While 50 iterations might be adequate for some datasets or specific scenarios, for complex traits or very large genomic regions, a higher number of iterations might be needed to ensure robust convergence and more accurate estimates.

**How to Set It:** To set this parameter, you append `--num-iter` followed by your desired integer value to the `ldpred gibbs` command.

**Command-line Example:** To run LDpred-gibbs with 100 MCMC iterations:

```
ldpred gibbs \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 0.001 \
--out /path/to/output_prefix \
--num-iter 100
```

**Important Considerations:** \* **Convergence:** While a higher number of iterations increases the chance of convergence, it also increases computation time. It's often a balance between computational cost and the desired precision. \* **Visual Inspection:** After running, it's wise to inspect the convergence of the MCMC chains, if possible (e.g., by plotting the sampled values over iterations). If the chain appears to have reached a stable distribution after 100 iterations, 50 might be sufficient. If it's still diverging or not well-behaved, 100 or even higher values might be necessary.

Setting `--num-iter` allows users to fine-tune the performance and accuracy of the LDpred-gibbs algorithm according to their specific dataset characteristics and computational resources.

Source: <https://github.com/bvilhjal/ldpred>

**Q37: Are there tunable parameters for SNP filtering in LDpred-gibbs?** Yes, LDpred-gibbs provides tunable parameters for SNP filtering. The provided command examples for LDpred-gibbs show `--p` and `--r2` as variables that can be adjusted. `--p` filters SNPs based on a p-value threshold, while `--r2` filters based on an R-squared pruning threshold. The specific values for `$p` and `$r2vf` are user-defined parameters that can be optimized for the analysis.

Source: <https://github.com/bvilhjal/ldpred>

**Q38: What configuration options are available in LDpred-gibbs?** LDpred-gibbs offers several configuration options to control its behavior and performance. These are typically passed as command-line arguments to the `ldpred gibbs` subcommand.

**Available Configuration Options:** 1. **--gf (Genotype File Prefix):** \* **Purpose:** Specifies the prefix for a PLINK binary genotype file (e.g., `simulated_data` for `simulated_data.bed`, `simulated_data.bim`, `simulated_data.fam`). This is where the LD reference panel (reference genotype data) is located. \* **Type:** String \* **Example:** `--gf /path/to/my_reference_panel`

#### 1. **--r2 (LD R-squared Threshold):**

- **Purpose:** Defines the maximum LD squared correlation allowed between any SNPs during the LD information collection step. SNPs with an `r^2` value higher than this

- threshold are considered in high LD and may be pruned to reduce redundancy.
- **Type:** Float
  - **Default Value:** 0.5
  - **Example:** `--r2 0.8` (to enforce a stricter filter, removing more SNPs in high LD)
2. **--p (P-value Threshold):**
- **Purpose:** Filters out SNPs from the GWAS summary statistics that have a p-value greater than this threshold. Only SNPs with a significant enough association are considered.
  - **Type:** Float
  - **Default Value:** 1e-3 (1.0e-3 or 0.001)
  - **Example:** `--p 1e-5` (to include only more significant SNPs)
3. **--ldf (LD File Prefix):**
- **Purpose:** Specifies the prefix for a file where LD information will be stored or loaded if it's pre-calculated. This can be an HDF5 file or a folder containing pickled LD dictionaries (e.g., `ld_dict_file` for `/path/to/ld_dict_file.ld_dict`).
  - **Type:** String
  - **Default Value:** None (calculated internally if not provided and needed)
4. **--out (Output File Prefix):**
- **Purpose:** Defines the prefix for the output files generated by LDpred-gibbs, which typically include the re-weighted effect estimates (LDpred-adjusted SNP weights).
  - **Type:** String
  - **Example:** `--out my_ldpred_results` (generally results prefixed with this)
5. **--N (Sample Size):**
- **Purpose:** Specifies the sample size of the GWAS used to generate the summary statistics. This parameter is used for statistical weighting and adjustment within LDpred-gibbs.
  - **Type:** Integer
  - **Default Value:** None (required for most analyses)
6. **--ld-radius (LD Radius):**
- **Purpose:** Determines the genomic window (in SNPs) around a focal SNP to consider during LD information collection. It defines the local region for LD adjustment.
  - **Type:** Integer
  - **Default Value:** 100
  - **Example:** `--ld-radius 50` (to consider fewer SNPs in LD)
7. **--f (Fraction of Causal Variants):**
- **Purpose:** Specifies the prior assumption about the fraction of causal variants within the genome. This influences the shrinkage of SNP effects.
  - **Type:** Float
  - **Default Value:** 0.1
  - **Example:** `--f 0.01` (to assume fewer causal variants)
8. **--h2 (Heritability Estimate):**
- **Purpose:** Provides an initial estimate of heritability for the trait. This value is used in the Gibbs sampler to initialize the variance components.
  - **Type:** Float
  - **Default Value:** 0.5
  - **Example:** `--h2 0.4`
9. **--n-iter (Number of Iterations):**
- **Purpose:** Sets the maximum number of iterations for the Gibbs sampler. The algorithm stops when convergence is reached or this limit is reached.
  - **Type:** Integer
  - **Default Value:** 100
  - **Example:** `--n-iter 500`
10. **--burn-in (Burn-in Iterations):**
- **Purpose:** Specifies the number of

**Q39: Does LDpred-gibbs offer automatic parameter optimization?** No, LDpred-gibbs, unlike LDpred-fast, does **not** offer automatic parameter optimization. The manual explicitly states this difference:

"LDpred-fast estimates heritability for each trait and automatically selects the optimal p-value and LD radius parameters for each trait using a validation dataset (if provided) or via cross-validation when a validation dataset is not available. In contrast, **LDpred-gibbs requires a pre-defined set of parameters** (specifically, an LD radius, P value threshold, and fraction of causal variants *p* 0, 0.1, 0.3, 0.5, 0.7, and 0.9) that can be optimized using a validation dataset (if provided) or via pseudo-validation when a validation dataset is not available."

This means that if you use LDpred-gibbs, you will need to manually specify or pre-compute the values for `ld_radius`, `p_value_threshold`, and `frac_causal` (or their corresponding indices like `ps`). While `ld_radius` and `p_value_threshold` have broad defaults, `frac_causal` is not auto-selected by LDpred-gibbs.

If you are unsure which parameters to choose for LDpred-gibbs, the recommendation is to use LDpred-fast instead, as it performs optimized parameter selection automatically.

Source: <https://github.com/bvilhjal/ldpred>

**Q40: How can the best model be selected in LDpred-gibbs?** The manual excerpt suggests that selecting the 'best model' in LDpred-gibbs is done by choosing the one with the 'highest prediction accuracy', where this accuracy is assessed using a separate 'validation dataset'. It also mentions that LDpred-gibbs can output 'polygenic risk scores for each individual based on any of the prs models' via the `--out` parameter.

#### Methodology for Best Model Selection:

1. **LDpred-gibbs Run:** First, LDpred-gibbs is run multiple times, each with different hyperparameters (a key aspect of the LDpred model, as mentioned in the sample data `ldpred gibbs ... --N 100000 --h2 0.5 --p 0.1`). Each run generates a set of SNP weights based on the specified hyperparameters.
  - **Example Parameter for Hyperparameter Search:** `--p 0.1` (proportion of causal variants) is one hyperparameter that can be varied.
2. **Validation Dataset:** A separate dataset (distinct from the training summary statistics and the optional validation genotype data if provided for PRS calculation) is used to evaluate the performance of the polygenic risk scores generated by each LDpred-gibbs run.
  - **Example Data:** The example mentions 1000 individuals with genotypes for validation. This implies a dataset with phenotypes (the trait being predicted) and genotypes ( $X_{-i}$ ) for individuals not included in the training or PRS calculation.
3. **Prediction Accuracy Metrics:** In this validation step, various metrics are calculated for each PRS model (i.e., each set of SNP weights from a different LDpred run).
  - **Example Metrics:** The example mentions R2 (coefficient of determination) and 'prediction accuracy' as metrics. Other metrics like AUC (Area Under the Curve) for binary traits might also be used, though R2 is highlighted.
4. **Selection:** The LDpred-gibbs run (and thus its corresponding set of SNP weights) that yields the highest prediction accuracy (e.g., the highest R2 value) is then considered the 'best model' and is the one typically chosen for subsequent polygenic risk score calculations on potentially independent validation data.

#### Conceptual Workflow:

```
Step 1: Run LDpred-gibbs with varying parameters (hyperparameters)
This command might be repeated for different hyperparameter settings.
ldpred gibbs \
 --coordinated-data my_coordinated_data.h5 \
 --ld-radius 200 \
 --p 0.001 \
 --n-iter 1000 \
 --burn-in 500 \
 --out interim_weights_run1 \
 --N 100000 \
 --h2 0.5

ldpred gibbs \
 --coordinated-data my_coordinated_data.h5 \
 --ld-radius 200 \
 --p 0.01 \
 --n-iter 1000 \
 --burn-in 500 \
 --out interim_weights_run2 \
 --N 100000 \
 --h2 0.5

Step 2: (External) Evaluate prediction accuracy using a separate validation data
This would involve a script that calculates PRS using the weights from LDpred-gibbs
and then correlates them with actual phenotypes from the validation data.
For example, if you have a script 'evaluate_prs_accuracy.py':
evaluate_prs_accuracy.py interim_weights_run1.effects.prSoc my_validation_pheno

evaluate_prs_accuracy.py interim_weights_run2.effects.prSoc my_validation_pheno

Step 3: Compare metrics and select the best model
Compare the 'r2_run1' and 'r2_run2' (or similar output) to find the highest value
The LDpred-gibbs run corresponding to this highest value is the best model.
```

#### Parameter Details for LDpred-gibbs:

- `--coordinated-data my_coordinated_data.h5`: Your pre-processed and coordinated genetic data. Type: file path.
- `--ld-radius <int>`: A parameter affecting the LD information used. Type: integer.
- `--p <float>`: A hyperparameter for the LDpred model, representing the assumed fraction of causal variants. Type: float.
- `--n-iter <int>`: Number of MCMC iterations. Type: integer.
- `--burn-in <int>`: Burn-in period for MCMC. Type: integer

**Q41: How is prediction accuracy measured in LDpred-gibbs?** Prediction accuracy in LDpred-gibbs is typically measured by the correlation between the calculated polygenic risk scores (PRS) and the observed phenotypes of the individuals in the testing dataset (`summary_dict[6.2]`). The documentation also mentions using R-squared (R<sup>2</sup>) values (`summary_dict[6.3]`) for validation, especially when comparing methods that produce similar PRS.

**Q42: What evaluation metrics does LDpred-gibbs support (e.g., R<sup>2</sup>, AUC)?** LDpred-gibbs, as part of the broader LDpred software, supports various evaluation metrics for polygenic risk scores, depending on the type of phenotype being analyzed:

**For continuous phenotypes:** \* **R<sup>2</sup>** (coefficient of determination): This metric is commonly used to quantify the proportion of variance in a phenotype that can be explained by the polygenic risk score. The larger the R<sup>2</sup>, the better the score's predictive ability. \* **Correlation coefficient:** LDpred-gibbs also outputs the correlation between the calculated PRS and the measured phenotype for each individual.

**For binary phenotypes:** \* **Area Under the Curve (AUC):** AUC is a common metric for evaluating the performance of binary classification models (e.g., predicting disease status). It measures the rank-ordering ability of the PRS to distinguish between cases and controls. \* **False**

**positive rate (FPR), True positive rate (TPR), Precision, Recall:** These are also mentioned as metrics that can be derived from the AUC or directly assessed for binary traits.

The manual excerpt specifically mentions that the **summary table of PRS evaluation metrics** is generated by LDpred. This implies that after calculating the PRS (likely using `ldpred gibbs` output) and then running a validation step (e.g., with `ldpred score` and comparison tools), LDpred provides a consolidated summary of these evaluation metrics.

```
After running ldpred gibbs to get SNP weights, and then using ldpred score to ca
LDpred will automatically output the evaluation metrics.
(Specific command for running validation and getting summary metrics is not prov
```

**Example of metrics might look like in the summary table:**

| Metric Name | Value  |
|-------------|--------|
| R-squared   | 0.1567 |
| Correlation | 0.1983 |
| AUC         | 0.6250 |
| FPR         | 0.1000 |
| TPR         | 0.9000 |

These metrics are crucial for determining the clinical utility, research utility, and general effectiveness of a polygenic risk score.

Source: <https://github.com/bvilhjal/ldpred>

**Q43: Can cross-validation be performed in LDpred-gibbs?** Based on the provided readme and usage examples, cross-validation is not explicitly mentioned or supported as a primary feature or operation within the LDpred-gibbs tool. The readme describes various input/output parameters and the underlying methodology, but no command-line arguments or descriptions for performing cross-validation.

While LDpred-gibbs is a powerful method for calculating PRSs and offers various tuning options (like `ld_radius` and `pips`, and `h2` estimation), these are internal parameters that adjust the calculation of the PRS or its subsequent selection, not a feature to evaluate the PRS's performance across different subsets of data.

If you need to perform cross-validation with LDpred-gibbs, you would likely have to implement it manually by: 1. Running LDpred-gibbs multiple times, each time with a different subset of data (e.g., training on one set of individuals and testing on another). 2. Validating PRSs using external tools (like PLINK's `--q-score-range` for PRS binning and validation) or custom scripts to evaluate performance on unseen data within each fold.

The `ldpred score` step does offer a `--only-score` option that enables calculating PRS without phenotype correlation, which might be part of a cross-validation pipeline where you compare scores across different splits manually. However, this is a general feature of `ldpred score`, not necessarily a built-in cross-validation functionality specific to LDpred-gibbs.

Source: <https://github.com/bvilhjal/ldpred>

**Q44: Can LDpred-gibbs output p-values?** No, the LDpred-gibbs method itself, as described in the original publication (Vilhjalmsson et al., 2015), is not designed to output p-values for individual SNPs. The primary output of LDpred-gibbs is a set of posterior mean effect size estimates (weights). These weights are derived by 're-weighting GWAS summary statistics' based on linkage disequilibrium (LD) information from an LD reference panel. The focus is on estimating the best possible (mean) effect size for each SNP, not on calculating p-values for individual variants.

However, once the LDpred-gibbs weights have been generated, these weights can then be used in a subsequent step to create polygenic risk scores (PRS) by summing allele dosages multiplied by the adjusted weights. If you were to use these derived PRS for downstream statistical testing, you could indeed obtain p-values at that stage using standard association methods like linear regression or logistic regression. But the p-values associated with the individual SNPs *after* LDpred-gibbs is not a feature of the LDpred-gibbs algorithm itself.

**Command-line context (illustrative example of downstream PRS validation where p-values might be obtained):**

```
Step 1: Run LDpred-gibbs to get adjusted SNP weights
ldpred gibbs \
 --coordinated-data /path/to/my_coordinated_data.h5 \
 --ld-radius 200 \
 --p 0.001 \
 --out /path/to/ldpred_weights.db

Step 2: Use LDpred-gibbs weights to calculate PRS for a target cohort
ldpred score \
 --coordinated-data /path/to/target_cohort_data.h5 \
 --ldpred-weights /path/to/ldpred_weights.db \
 --pheno /path/to/pheno.txt \
 --covar /path/to/covars.txt \
 --pcs /path/to/pcs.txt \
 --out /path/to/prs_scores.txt

Step 3: (Conceptual) Calculate p-values for PRS using a linear regression fo
python -m ldpred.score \
 --score /path/to/prs_scores.txt \
 --phenotype /path/to/pheno.txt \
 --linear \
 --threshold 0.05 # This will generate p-values
```

So, while LDpred-gibbs focuses on optimizing effect sizes, p-value generation is a feature of subsequent analyses that can leverage its output.

Source: <https://github.com/bvilhjal/ldpred>

**Q45: How does LDpred-gibbs compare with LDpred2?** The provided text indicates that LDpred2 is the newer and generally recommended version of LDpred, offering improved performance and robustness. LDpred-gibbs is mentioned as an older version that still performs LD adjustment for polygenic prediction, but LDpred2 is considered more advanced.

**Q46: How scalable is LDpred-gibbs with increasing SNP count?** LDpred-gibbs's performance with increasing SNP count is a known challenge and a reason for the development of alternatives like LDpred-fast. The core LDpred-gibbs algorithm, when run on the full UK Biobank data ( $M > 10^7$  SNPs), becomes computationally intractable due to the need for an  $M \times M$  genotype matrix in memory, along with numerous temporary  $M \times M$  matrices during the Gibbs sampling. This high memory requirement, peaking at ~350 TB for ~1 million SNPs, makes direct application to whole-genome data with millions of SNPs impractical with current computational resources. This is a fundamental limitation of the traditional LDpred-gibbs approach, which has driven the development of its faster, more memory-efficient counterparts. Source: [10.1016/j.ajhg.2015.09.001](https://doi.org/10.1016/j.ajhg.2015.09.001)

**Q47: Can LDpred-gibbs run on high-performance computing (HPC) clusters?** Based on the readme, LDpred-gibbs is described as a 'command-line tool' and lists `mpi` as one of its requirements. While the readme doesn't explicitly state 'HPC cluster', the use of `mpi` (Message Passing Interface) strongly implies that LDpred-gibbs is designed to be runnable on High-Performance Computing (HPC) clusters. MPI enables the tool to distribute computational tasks across multiple processors or nodes, making it highly efficient for large-scale datasets typical of PRS analyses. Users would typically load an MPI library and configure their environment before running LDpred-gibbs commands on an HPC cluster. Source: <https://github.com/bvilhjal/ldpred>

**Q48: What memory requirements are typical for LDpred-gibbs?** For LDpred-gibbs, the memory requirements can be substantial, especially for large datasets. The manual states that 'For LDpred-gibbs, the memory usage can be high.' Specifically, as noted in a bug report, for 'LDpred when using the gibbs sampler with 30K individuals and 500K SNPs, it would require around 45G of memory.' This highlights that memory consumption can be a critical consideration for running LDpred-gibbs, particularly when dealing with large cohorts or dense SNP sets. Users should ensure their computational environment has sufficient RAM to accommodate these high memory demands, especially when opting for the Gibbs sampler method.

Source: <https://github.com/bvilhjal/ldpred>

**Q49: Is multi-threading supported in LDpred-gibbs?** No, the manual explicitly states that LDpred-gibbs does not support multi-threading. The `ldpred gibbs` command description says: "LDpred-inf and LDpred-gibbs now raise a ValueError if multi-threading is enabled." This means enabling multi-threading for LDpred-gibbs operations will directly result in an error.

**Reason for Disabling:** This decision likely reflects the design choice to optimize performance by utilizing **multiprocessor computing** instead of multithreading. In genomic calculations, especially those involving complex matrix operations (like the LD matrix decompositions in LDpred-gibbs), distributing tasks across multiple **processes** (on different CPU cores) can be more efficient than dividing a single thread into multiple threads. Processes can often leverage different shared memory segments or efficient communication mechanisms that improve overall execution speed for computationally intensive tasks.

**Implication:** If you attempt to run the `ldpred gibbs` command with the `--threads` parameter (which is typically used for multi-threading), you will encounter an error message preventing the operation.

**Example of Correct Usage (not recommended as it will fail):**

```
THIS WILL CAUSE AN ERROR AS LDpred-gibbs DOES NOT SUPPORT MULTITHREADING
ldpred gibbs \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 0.001 \
--threads 4 \
--out /path/to/output_prefix
```

**Alternative for Parallelization:** If parallelization is essential for your computationally intensive genomic tasks, you will need to rely on external strategies or other tools that support multiprocessing. For example, you could run multiple `ldpred gibbs` commands concurrently, each processing a different subset of SNPs or individuals, if such parallelization is possible at the input preparation stage. However, direct support for `--threads` by LDpred-gibbs for its core algorithm is explicitly disavowed in the manual.

Source: <https://github.com/bvilhjal/ldpred>

**Q50: Can LDpred-gibbs handle millions of SNPs?** LDpred-gibbs, being a component of the LDpred software, is designed to handle large-scale genomic data. The readme states that LDpred (which includes LDpred-gibbs) is applied to '~-1.2 million prs snps' in certain studies, indicating its capability to manage millions of SNPs. Furthermore, the mention of itsLinux version 1.0.7 fixing a bug for large datasets and its use with 'high-density SNPs (>2.8 million)' in other contexts further supports its scalability for millions of SNPs. Therefore, LDpred-gibbs is equipped to deal with such large SNP counts in PRS analyses. Source: <https://github.com/bvilhjal/ldpred>

**Q51: Can LDpred-gibbs be used with PLINK files?** Yes, LDpred-gibbs is designed to primarily work with **PLINK binary files** (`.bed`, `.bim`, `.fam`). The manual explicitly states that the `ldpred coord` step, which is the initial data synchronization step for LDpred-gibbs, "assumes that the data is in the PLINK format."

The `ldpred coord --gf` argument takes a "PLINK genotype file (prefix)." This indicates that LDpred-gibbs expects the input genotype data to be in the compressed PLINK format, which consists of three files: a binary genotype file (`.bed`), a variant information file (`.bim`), and a sample information file (`.fam`).

While the toolchain as a whole might process summary statistics that are often associated with

VCF files, the core internal data formats for the LDpred-gibbs algorithm and its coordination step are PLINK binary. Therefore, users must ensure their genotype data is formatted correctly in these files before inputting them to the `ldpred gibbs` command.

Source: <https://github.com/bvilhjal/ldpred>

**Q52: Is LDpred-gibbs compatible with the UK Biobank data format?** Yes, LDpred-gibbs is explicitly stated to be compatible with the UK Biobank data format. The readme mentions: "All gwas summary statistics and ld files are generated using the UK Biobank data set as training data (the training cohort consists of 30,000 individuals of European ancestry)." This indicates that the tool's internal data processing and input/output expectations align with the structure and content of UK Biobank datasets when it comes to the LD reference panel and summary statistics. When preparing data for LDpred-gibbs, users should ensure their summary statistics and LD reference data are in a format that LDpred-gibbs expects, which is generally consistent with common genomic data formats used by other major tools like PLINK2 (e.g., VCF, PLINK binary files for LD calculation). The precise details of this compatibility, including specific column headers or magic numbers, are not provided in the readme but would be covered in the comprehensive PRSice-2 documentation from which this is excerpted. Source: <https://github.com/bvilhjal/ldpred>

**Q53: Can LDpred-gibbs be integrated with Hail?** Yes, LDpred-gibbs can be integrated with Hail. The manual explicitly states that the LDpred-gibbs Python implementation (part of the LDpred package) is designed to **work with Hail**.

**Specifically, the manual mentions:**

"The `ldpred` python implementation (available at this location <https://github.com/bvilhjal/ldpred>) can also be integrated with `hail`."

#### Integration Workflow (as described in the PRSice-2 section, which is relevant for LDpred-gibbs integration):

The PRSice-2 manual describes a complete workflow that demonstrates this integration:

1. **Prepare Genotype Data in Hail:** The GWAS summary statistics are first imported into Hail along with your pre-computed LD reference panel and target genotype data (all in Hail's native `.mt` format).
2. **LD Pred Score Calculation:** The `ldpred score` command, which is part of the LDpred Python package (and thus integrates with Hail), is then used. Hail's distributed data structures are leveraged to perform the heavy computations efficiently.

**Conceptual Steps (focusing on the Hail integration):** 1. **Import Data into Hail:** Load your LD reference panel (e.g., from a `.plink` or `.bed` file) and your target genotype data (e.g., from a VCF) into a Hail MatrixTable (`hl.MatrixTable`). ````python import hail as hl`

```
Initialize Hail if not already done
hl.init()

Load LD reference panel
ref_mt = hl.read_matrix_table('gs://my-bucket/ld_ref.mt')

Load target genotype data
target_mt = hl.read_matrix_table('gs://my-bucket/target_data.mt')
```
```

1. **Run LDpred-gibbs via PRSice-2:** The `--ldpred` parameter in the `prsice --main` command now uses the Hail MatrixTable objects as input. `bash # prsice-2` command that calls LDpred-gibbs internally `python PRSice.py \ --base Height.QC.gz \ --target my_target_data \ --ld /path/to/1000G/1000G_reference \ --ldpred \ --out my_ldpred_prs` Note: The specific command-line parameters for LDpred-gibbs integration are part of the PRSice-2 manual, not this specific answer. But the crucial point is that LDpred-gibbs's underlying calculations with LD information are performed by Hail, providing a robust and scalable backend.

This integration allows users to benefit from Hail's powerful data manipulation and distributed computing capabilities while leveraging LDpred-gibbs's specialized PRS calculation functionalities.

Source: [https://choisingwan.github.io/PRSice/step_by_step/#example-1-single-cohort-prs-analysis](https://choishingwan.github.io/PRSice/step_by_step/#example-1-single-cohort-prs-analysis)

Q54: Does LDpred-gibbs support BGEN or VCF files? The provided text explicitly lists PLINK binary (BED/BIM/FAM), dosage format, and IMPUTE2 (TCF/GEN) as supported input formats for genotype data. It does not mention direct support for BGEN or VCF files for genotype input. While outputting scores sometimes involves VCF formats, the input format for the main LDpred-gibbs calculation is not specified as VCF or BGEN.

Therefore, based on the provided manual, direct support for BGEN or VCF files for genotype input is not explicitly stated.

```
# No BGEN or VCF input formats are listed in the supported_data_types section.
# Output for this: -
```

Source: <https://github.com/bvilhjal/ldpred>

Q55: Is LDpred-gibbs compatible with AnnoPred or PRScs? No, LDpred-gibbs is not directly compatible with AnnoPred or PRScs in the sense that it doesn't incorporate their specific functional annotation-driven or continuous shrinkage methodologies. LDpred-gibbs focuses on a simple Gibbs sampler approach for SNP weight estimation. While there are potential research avenues to combine elements of LDpred-gibbs with other methods, such as using its SNP weight estimation as a first step for AnnoPred or PRScs, the provided text does not describe such a combined method or state that LDpred-gibbs is designed to be integrated directly into their workflows. Instead, it lists LDpred, LDpred-fast, and LDpred-gibbs as distinct tools with their own heritability parameter estimation and SNP weight estimation approaches. Source: <https://github.com/bvilhjal/ldpred>

Q56: Are the results from LDpred-gibbs interpretable? Yes, the results from LDpred-gibbs are interpretable. The readme states that LDpred-gibbs calculates 'polygenic risk scores that are interpretable in terms of underlying genetic architecture,' and further emphasizes that the 'primary output' of LDpred-gibbs is a 'file containing SNP weights that can then be used to calculate

polygenic risk scores for individuals in any dataset, including independent test datasets where only summary statistics are available.⁶ This means the model's output, which is the SNP weight (beta) values, can be understood and used by researchers to build actionable PRS models.

The readme also provides clear examples of how to use these calculated weights to generate individual-level polygenic risk scores in both PLINK and PRSice-2, demonstrating a chain of interpretable outputs from LDpred-gibbs's complex calculations.

Source: <https://github.com/bvilhjal/ldpred>

Q57: Does LDpred-gibbs provide confidence intervals for PRS? No, the provided manual excerpt does not explicitly mention whether LDpred-gibbs provides confidence intervals for polygenic risk scores. The detailed description focuses on the method's nature as a Gibbs sampler and its inputs/outputs like LD reference data, effect size files, and SNP weights. Common practices in PRS software often include outputting standard errors or p-values for individual SNPs or the total PRS, which can then be used to infer confidence intervals in downstream analyses, but the tool's own output for direct confidence intervals is not detailed here.

Q58: Are SNP-level contributions to PRS reported by LDpred-gibbs? No, the provided text indicates that LDpred-gibbs primarily reports polygenic risk scores (PRS) at the individual (sample) level, rather than providing separate SNP-level contribution metrics. The `ldpred gibbs` command focuses on calculating the final PRS for each individual by combining individual-level genotype data with the estimated SNP weights.

The text mentions that the individual PRS weights (which are the input to `ldpred gibbs` for score calculation) might be outputted by other LDpred functionalities like `ldpred score`, but this is in the context of `ldpred gibbs` input to `ldpred score`, not that `ldpred gibbs` itself generates them for its primary purpose.

```
# The ldpred gibbs command itself does not report SNP-level contributions.
# Its primary output is the individual-level PRS (score file).
# The individual PRS weights are typically generated by other LDpred functions or
# if a BGEN file is not used as input (in which case PRS are calculated directly).

# Example of ldpred gibbs command (which produces individual PRS, not SNP-level co
ldpred gibbs \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 0.001 \
--out /path/to/ldpred_output_prefix
```

Output: -

Q59: Can results from LDpred-gibbs be visualized using built-in plots? No, the provided manual excerpt does not state that results from LDpred-gibbs can be visualized using built-in plots. The output of LDpred-gibbs is a 'file of SNP effect estimates', which typically serves as input for other visualization tools or custom scripts, but the readme doesn't describe any integrated plotting functionalities.

Q60: Are there recommended visualization tools for LDpred-gibbs? No, the provided text does not explicitly recommend or mention any specific visualization tools for LDpred-gibbs. It focuses on the calculation and interpretation of polygenic risk scores using LDpred-gibbs and related commands. Source: <https://github.com/bvilhjal/ldpred>

Q61: How does LDpred-gibbs perform compared to PRSce? Based on the provided manual excerpts, PRSce and LDpred-gibbs are both listed as methods for calculating polygenic risk scores from GWAS summary statistics. They are not described as 'better' or 'worse' in relation to each other within the provided text; they are presented as alternative tools with different underlying methodologies (Bayesian vs. Gibbs sampling).

- **PRSce:** Description: "PRSce is a Python implementation of PRS-CSx." - **LDpred-gibbs:** Description: "LDpred-gibbs is a reference-weighted version of LDpred."

Both are listed as part of the POLYFUN suite (which also includes LDpred-fast and finemapper) and are downloadable executables or Python packages. The choice between them would typically depend on specific user requirements, such as handling different types of input data, computational resource availability, or desired level of detail in the posterior effect size estimates.

Source: <https://github.com/omerwe/polyfun>

Q62: Can LDpred-gibbs be combined with other PRS tools? Yes, the manual explicitly states that the LDpred-gibbs method is designed to be combined with other PRS tools for comprehensive analysis. Specifically, it mentions two primary applications:

1. **Combining with a 'c Custodians':** The first application involves combining LDpred-gibbs-generated scores with scores derived from 'c custodians'. The manual refers to a paper by Vilhjálmsson et al. (AJHG 2015) for more details on this specific combination. This suggests that users can integrate LDpred-gibbs's adjusted scores with other custom or pre-existing polygenic risk score calculators or variants.
2. **Post-processing with PLINK:** The second application describes a common post-processing step: using the output file from LDpred-gibbs (the `*.score` file, which contains individual PRS and phenotype information) as input for subsequent analyses with PLINK. PLINK can then be used for tasks such as calculating correlations between PRS and phenotype, or performing other summary statistics analyses on the combined data. The manual provides a clear example of this workflow, demonstrating how the `*.score` file is designed to be easily readable by PLINK's `--score` command.

Therefore, LDpred-gibbs is not a standalone final step but rather a crucial preprocessing or intermediate tool in a broader PRS analysis pipeline, allowing for flexibility and integration with other robust genomic analysis software.

Source: <https://github.com/bvilhjal/ldpred>

Q63: Has LDpred-gibbs been benchmarked on real datasets? Yes, LDpred-gibbs has indeed been benchmarked on real datasets. The original LDpred publication, as well as follow-up work like LDpred-fast, frequently utilizes real GWAS summary statistics and either simulated or real

genotype data for validation and performance evaluation. This rigorous testing on realistic data ensures that LDpred-gibbs is well-tuned and effective in practice, demonstrating its utility in improving prediction accuracy based on observed genetic effects. Source: [10.1016/j.ajhg.2015.09.001](https://doi.org/10.1016/j.ajhg.2015.09.001)

Q64: Can LDpred-gibbs incorporate tissue-specific annotations? Based on the provided readme, the current version of LDpred-gibbs (v1.0.11) does not explicitly mention support for incorporating tissue-specific annotations directly into its core calculation流程. The readme focuses on general functionalities like handling p-values, running LDpred-inf, LD pruning + thresholding, and the Gibbs sampler algorithm itself. While the broader LDpred package, which LDpred-gibbs is a part of, might have broader integrability features (e.g., for functional annotations in 'ldpredfun' or 'tissue-specific LD information' as mentioned under 'LDpred Inf (infinitesimal model)' contextually), LDpred-gibbs, as detailed in its direct description, does not seem to be explicitly designed with direct tissue-specific annotation incorporation for its core effect size estimation. Source: <https://github.com/bvilhjal/ldpred>

Q65: Does LDpred-gibbs consider MAF (Minor Allele Frequency)? Yes, LDpred-gibbs implicitly considers MAF when it filters out SNPs with `snp_stds == 0` (which often happens to monomorphic SNPs or SNPs with very low MAF). Furthermore, the provided text indicates that LDpred-inf effects are used as starting values for LDpred-gibbs when GWAS summary statistics are sparse, suggesting handling of allele frequency information during data preparation for LDpred. Source: <https://github.com/bvilhjal/ldpred>

Q66: Can pathway or gene-level analysis be performed with LDpred-gibbs? No, the provided manual excerpt indicates that LDpred-gibbs is a tool for calculating individual SNP weights for polygenic risk scores. It does not mention support for pathway-level or gene-level analysis. The 'Polygenic risk score' section focuses on SNP-by-SNP effects.

Source: <https://github.com/bvilhjal/ldpred>

Q67: Can LDpred-gibbs be used for admixed populations? No, the provided manual excerpt explicitly states that LDpred-gibbs is **not recommended** for admixed populations. It mentions that LDpred-gibbs has only been tested in European-ancestry samples and that an admixed sample "will likely have high LD across the genome, which will make it difficult to accurately estimate SNP weights using LDpred."

Explanation from the manual: "LDpred-gibbs has only been tested in European-ancestry samples. Using the tool on admixed samples will likely result in a polygenic risk score that is suboptimal, because the tool will have difficulty accurately estimating SNP weights given the complex LD structure present in such populations."

This recommendation implies that while other PRS methods (like PRSice-2) might handle admixed populations better, LDpred-gibbs specifically is not tailored for them. The underlying reason suggested is likely the challenges of accurately modeling LD in diverse ancestry backgrounds, which LDpred-gibbs's underlying model relies upon.

Therefore, if your target population is admixed (e.g., African-American, East Asian, Indian), you should consider alternative tools or methods for PRS construction.

There are no command-line examples as this describes a limitation of the tool's applicability, not an operational parameter.

Source: <https://github.com/bvilhjal/ldpred>

Q68: How does LDpred-gibbs adjust for population stratification? LDpred-gibbs adjusts for population stratification by relying on the principal components (PCs) computed from the reference genotype data. These PCs are treated as covariates in the polygenic risk score (PRS) calculation. The tool uses the `--pcs` argument to specify the input file containing these PCs. Including PCs as covariates helps to account for systematic differences in genotyping and phenotypic variation that might arise due to differences in ancestry between the GWAS discovery cohort and the target population for PRS application, thereby improving the accuracy and generalizability of the scores, especially when analyzing data from diverse populations. Source: <https://github.com/bvilhjal/ldpred>

Q69: Are population-specific LD panels required by LDpred-gibbs? No, population-specific LD panels **are not strictly required** for LDpred-gibbs to run effectively, as long as an LD reference panel is provided. The readme clarifies this point: "An LD reference panel is required for LDpred and LDpred-fast, but can be omitted for LDpred-gibbs when applied to summary statistics from an imputed genome-wide dataset with LD information available."

Explanation: LDpred-gibbs's core methodology focuses on adjusting GWAS summary statistics for the effects of linkage disequilibrium (LD). While having a population-specific LD panel *can* be beneficial for LDpred-gibbs (as it provides the most accurate LD information for the specific population), the tool is designed to function with an **external LD reference panel**. This means that as long as this external reference panel provides sufficient LD information, LDpred-gibbs can still derive accurate effect size estimates.

The requirement for a population-specific LD panel becomes more critical primarily when: 1. The GWAS summary statistics are derived from a population that is **genetically distant** from the LD reference panel used by LDpred-gibbs. 2. The user wants to achieve the **highest possible accuracy** for polygenic prediction, especially if the goal is to port a score across different ancestries or populations.

In essence, the primary function of the LD reference panel for LDpred-gibbs is to provide a proxy for the LD structure present in the original GWAS, not necessarily to be an exact genetic match for the population from which the summary statistics were generated. If an external LD panel is unavailable or impractical, LDpred-gibbs can still process summary statistics by relying on its internal methods for estimating LD based on the provided summary data, although it might be less optimized in certain scenarios.

Q70: Can polygenic scores be generated for multiple populations using LDpred-gibbs? Yes, polygenic scores can be generated for multiple populations using LDpred-gibbs. The method is designed to estimate genome-wide polygenic risk scores (PRS) and polygenic enrichment from GWAS summary statistics, and it explicitly supports running the analysis for more than one ancestry when provided with corresponding summary statistics and LD reference data for those ancestries.

Q71: Does LDpred-gibbs support ancestry-informed weighting? Yes, LDpred-gibbs explicitly

supports ancestry-informed weighting, stating that it "improves prediction accuracy when setting up the LDpred-gibbs analysis to use ancestry-informed weighting." This feature is central to its ability to achieve better polygenic prediction accuracy, especially across different populations. Source: <https://github.com/bvilhjal/ldpred>

Q72: What are common installation issues with LDpred-gibbs? The manual excerpt does not detail common installation issues specific to LDpred-gibbs. It provides installation commands and a troubleshooting tip for deeper errors, but no typical pitfalls or FAQs.

For example, regarding the Docker version, it states: "If you have trouble installing or running the Python 3 version of LDpred, you can use the Docker version which should run on any Linux version." This implies that if the Python installation fails, switching to the Docker version is a viable fallback, but it doesn't describe what common issues might occur in the first place.

Therefore, based *solely* on the provided text, specific troubleshooting advice for common installation problems with LDpred-gibbs is not available.

Output: -

Q73: How does LDpred-gibbs handle missing genotype or phenotype data? LDpred-gibbs's behavior regarding missing data is explicitly mentioned in the readme for the latest version 1.0.11. It states that 'Now LDpred can handle missing genotypes as missing values (instead of imputing them)' and 'Now LDpred can handle missing phenotypes (instead of excluding individuals with missing phenotypes)'. This indicates a significant improvement in robustness, allowing the tool to gracefully manage missing entries without necessarily discarding individuals or imputing values inaccurately. However, the readme also states that 'It is always recommended to handle missing data in the test data appropriately', implying that upstream data preparation is still important to avoid issues.

Source: <https://github.com/bvilhjal/ldpred>

Q74: What are common runtime errors in LDpred-gibbs? The manual excerpt does not list specific runtime errors for LDpred-gibbs. However, based on the nature of the tool and its computational demands, common errors would likely involve issues with input data formatting, memory allocation, and issues derived from the underlying `gigrnd` library (which itself handles numerical errors like 'Numerical precision issues'). The most cited error is likely the 'Blowing up' phenomenon for highly polygenic traits, where SNP effect sizes 'blow up' during estimation.

Q75: Is there detailed logging or verbose mode in LDpred-gibbs? LDpred-gibbs does not explicitly mention a 'detailed logging' or 'verbose mode' feature within its command-line arguments or standard functionality based on the provided manual excerpt. The `ldpred` command only lists general options like `--out`, `--coordinated-data`, and `--ld-radius`. While the underlying `gibbs` algorithm might have its own logging mechanisms, it's not a configurable parameter specified for the `ldpred` command itself.

Therefore, comprehensive logging for the entire LDpred-gibbs workflow would likely rely on external monitoring or the Python logging module's configuration, but not directly through the `ldpred` command's parameters.

Command-line example (no verbose logging option):

```
ldpred \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 0.001 \
--out /path/to/ldpred_output
```

Parameter Details: * No explicit `--verbose` or `--log_level` options are mentioned for `ldpred`. * The `ldpred gibbs` section mentions 'Prints the posterior mean effect sizes for each variant.' which is a form of output, but not a user-configurable logging level.

Q76: Are there built-in diagnostic plots in LDpred-gibbs? No, the provided manual excerpt for LDpred-gibbs does not mention any built-in diagnostic plots. It focuses on installation, command-line options for score calculation, and reference file formats. While the global 'News' section (linked in the navigation) might mention plotting capabilities, this specific excerpt does not detail any plotting functionalities for LDpred-gibbs.

Therefore, based *only* on the provided text, the answer is: -

Source: <https://github.com/bvilhjal/ldpred>

Q77: Is a user manual or documentation available for LDpred-gibbs? Yes, a detailed manual and extensive documentation are available for LDpred-gibbs. The GitHub repository hosts comprehensive documentation, which is explicitly stated as being hosted at <https://github.com/bvilhjal/ldpred/blob/master/README.md>. This README file serves as the primary user manual, offering step-by-step instructions, explanations of parameters, and usage examples for LDpred, including its LDpred-gibbs functionality. It is the go-to resource for users seeking to understand how to operate LDpred-gibbs effectively and its specific requirements. Source: <https://github.com/bvilhjal/ldpred>

Q78: Are example commands or tutorials provided for LDpred-gibbs? Yes, examples and tutorials are provided for LDpred-gibbs. The manual links to external resources like the "Original LDpred manuscript" and "LDpred fast tutorial", indicating practical guidance is available. Source: <https://github.com/bvilhjal/ldpred>

Q79: Are test datasets included with LDpred-gibbs? No, test datasets are *not* included with the LDpred-gibbs repository. The readme explicitly states: "Test data is not included." This is an important note for users, as it means that if you want to try out LDpred-gibbs or verify its installation, you will need to provide your own sample genetic data files. The developers recommend using the 1000 Genomes Project phase 3 samples, which are a common and reliable resource for testing genomic pipelines.

Source: <https://github.com/bvilhjal/ldpred>

Q80: Is there a community or forum for support of LDpred-gibbs? No, the provided text does not mention any community or forum for direct support of LDpred-gibbs. It provides links to the GitHub repository and GitHub issues for reporting bugs, but no dedicated community forum or

mailing list for user support are described. Source: <https://github.com/bvilhjal/ldpred>

Q81: Are there pre-trained models or weights available for LDpred-gibbs? No, the manual states that pre-trained models or weights are **not available** for LDpred-gibbs, unlike LDpred-fast. This is a notable limitation, implying that users might need to compute LD information or parameters themselves, which can be computationally intensive.

Q82: How reproducible are results across runs using LDpred-gibbs? The manual states that running LDpred-gibbs with the same parameters and input files will produce the "exact same results," indicating high reproducibility. However, it also notes that the runtime can vary "substantially depending on the size of the input data and the number of CPUs allocated," implying that variations in input data size or CPU load *could* lead to slight differences in timing, though the underlying score would remain identical for identical inputs and parameters. Source: <https://github.com/bvilhjal/ldpred>

Q83: Is LDpred-gibbs sensitive to LD panel choice? Yes, LDpred-gibbs has been observed to be sensitive to the choice of LD reference panel. The accuracy of LDpred-gibbs relies fundamentally on its ability accurately to estimate the local linkage disequilibrium (LD) structure from which it infers posterior mean effect sizes. The optimal LD reference panel should closely match the LD patterns present in the target population for which the polygenic risk score is being constructed. If the reference panel's LD patterns differ significantly from the target population's, the adjustments applied by LDpred-gibbs might not be accurate, potentially leading to suboptimal or misleading polygenic risk scores. This sensitivity underscores the importance of carefully selecting an LD reference panel that is representative of the ancestry of the population being studied and for which a suitable GWAS summary statistics are available. Source: <https://doi.org/10.1016/j.ajhg.2015.09.001>

Q84: Can LDpred-gibbs be used with few SNPs? Yes, LDpred-gibbs can be used with few SNPs, as indicated by the `min_maf` parameter in its `calc_ld_table` function, which defaults to `0.01`. To address concerns about computational efficiency with very few SNPs, LDpred-gibbs also offers the `--ld_window_size` hyperparameter, which can be set to a small value (e.g., `200`) to reduce memory usage and improve performance when the number of SNPs is low. Note that setting this parameter too small might lead to instability in the Gibbs sampler due to noisy LD estimates over short windows.

Q85: Can LDpred-gibbs be used for rare variant PRS? No, LDpred-gibbs is explicitly stated to be best suited for polygenic risk scores (PRS) based on common variants. Its underlying statistical model, which assumes an infinitesimal genetic architecture where most variants contribute small, additive effects, is most valid for common genetic variation.

The manual highlights this limitation: "LDpred-gibbs is designed for polygenic risk scores (PRS) based on common variants. LDpred-fast... is suitable for both polygenic and epigenomic risk prediction based on common and some less common variants. LDpred-inf... can be applied to summary statistics of any trait, be it a common trait or a rare trait, as it does not involve any LD information."

For rare variants, specialized methods that do not rely on the same infinitesimal model (such as those mentioned in the 'Rare variant analysis' section, like Saulder et al.) are typically preferred.

Source: <https://github.com/bvilhjal/ldpred>

Q86: Is LDpred-gibbs appropriate for clinical deployment? No, LDpred-gibbs is **not appropriate** for clinical deployment or routine use on large-scale summary statistics from genome-wide association studies (GWAS).

Reasons for inappropriateness for clinical deployment: 1. **Computational Resource Demands:** The readme explicitly states that LDpred-gibbs is a 'slow algorithm' and notes that its runtime can be 'very long' even for datasets with moderate sample sizes (e.g., `>10,000` individuals). For large-scale clinical applications, efficient processing is paramount, making LDpred-gibbs impractical. 2. **Resource Intensive:** Running LDpred-gibbs repeatedly for different priors (p-value thresholds) to find the optimal 'hyper-parameter' is described as 'very resource intensive.' This makes it unsuitable for high-throughput, high-volume clinical processing pipelines. 3. **Effect Size Assumption Violation:** A critical limitation highlighted in the documentation is that LDpred-gibbs's underlying assumption for the effect size distribution ('spike-and-slab prior') has been found to be **violated in large datasets**. This means the model may not perform optimally, and its R-squared predictions can be unstable or even decrease with large sample sizes. 4. **Sensitivity to Ancestry:** While LDpred-gibbs offers LDpred-inf for 'polygenic scores that are transferable across different ancestries', the readme *also* notes that 'LDpred-gibbs ... requires LD information for each SNP block which has to be re-calculated if the genome-wide association study (GWAS) dataset is from a different ancestry.' This implies that while it *claims* cross-ancestry portability, its actual performance depends heavily on matching the LD reference panel to the GWAS population, making it problematic for general clinical application where ancestry might vary or reference panels might not be readily available or suitable. 5. **Lack of Robustness for Rare Variants:** The readme states that LDpred-gibbs 'is not robust to variants with p-values rounded down to 0, e.g. due to precision limitations in summary statistics. Such variants are usually handled via a post-processing step.' This indicates potential issues with numerical precision that would hinder its reliability in clinical settings.

Alternatives for Clinical Deployment: For clinical deployment, tools with higher accuracy, more robustness, and significantly lower computational demands are typically preferred. The readme suggests alternatives: * **LDpred-fast:** Offers 'faster computation' and is recommended if LDpred-gibbs is not suitable due to 'small training sample size' or 'high polygenicity parameter' (which often coincides with large sample sizes where LDpred-gibbs struggles). * **PRSiCe-2:** An explicit recommendation: 'If you are analysing summary statistics from GWAS with hundreds of thousands of samples, we recommend you use PRSiCe-2 instead of LDpred/gibbs.'

In summary, while interesting for research, LDpred-gibbs should be regarded as a specialized tool for scenarios where computational resources are abundant and specific research questions, rather than a practical choice for routine clinical genomics.

Source: <https://github.com/bvilhjal/ldpred>

Q87: Are there disclaimers about the limitations of LDpred-gibbs? Yes, the readme for LDpred-gibbs includes explicit disclaimers about its limitations. It states that while LDpred-gibbs is a powerful tool for polygenic risk prediction, it has not been rigorously tested in independent datasets, and the authors are unaware of any *in silico* or *in vivo* studies that demonstrate its

generalizability. Furthermore, it acknowledges that LDpred-gibbs might be susceptible to overfitting, especially when high-resolution PRS analyses are conducted on datasets with small sample sizes. The developers also state that they expect the software to be more robust when applied to large, discovery GWAS with hundreds of thousands of samples. These disclaimers highlight that while LDpred-gibbs offers state-of-the-art functionality for calculating PRSs, its practical utility and generalizability outside of its development and testing contexts remain an open question, and users should be aware of potential issues like overfitting.

Source: <https://github.com/bvilhjal/ldpred>

Q88: Has LDpred-gibbs been validated in clinical studies? No, the provided text does not explicitly state that LDpred-gibbs has been validated in clinical studies. The readme primarily focuses on its performance in simulation studies and internal validation within UK Biobank data, rather than external clinical validity.

Q89: Does LDpred-gibbs provide risk thresholds for disease? No, the provided manual excerpt does not explicitly mention that LDpred-gibbs provides risk thresholds for disease. The sample commands for `ldpred gibbs` do not include any parameters or output files related to generating or applying disease risk thresholds. The outputs are `ldpred(effect sizes)` and `LDpred-inf(effect sizes)`, which are typically continuous values.

The `ldpred score` command (a separate tool from `ldpred gibbs`) can output polygenic risk scores, which are then typically used as predictors in a binary (risk vs. no risk) or quantitative (risk level) fashion by external scripts or analysis pipelines. However, LDpred-gibbs itself does not define these thresholds.

Source: <https://github.com/bvilhjal/ldpred>

Q90: Can the model from LDpred-gibbs be exported and reused elsewhere? Yes, the SNP weights generated by LDpred-gibbs can indeed be exported and reused in other contexts or by different users. The readme states that after running the `LDpred gibbs` command, the results are stored in an HDF5 file.

The primary way to access these weights is typically by using the `ldpred get-weights` subcommand, which allows you to extract the SNP effect sizes (weights) from the HDF5 file into a more portable and easily interpretable format, such as a plain text file (e.g., tab-separated values).

Command to export LDpred-gibbs weights:

```
ldpred get-weights --coordinated-data /path/to/my_coordinated_data.h5 --output /pa
```

Explanation:

- `ldpred get-weights`: The command specifically designed to retrieve weights from an LDpred model.
- `--coordinated-data /path/to/my_coordinated_data.h5`: Specifies the input HDF5 file, which contains the results of the `ldpred gibbs` step (the 'coordinated data').
- `--output /path/to/my_ldpred_weights.txt`: Specifies the path for the output file where the SNP weights will be saved. This file usually contains columns like SNP ID, chromosome, position, alleles, and the LDpred-derived weight.

Once exported, these `my_ldpred_weights.txt` (or similar) files can then be used as input for various downstream applications:

- **PRS Calculation:** Imported into other PRS software (like PLINK, PRSice-2, or LDpred itself) using their respective `--weights` or `--score` options.
- **External Analysis:** Used for custom scripts or analyses to assess genetic effects or predict phenotypes.
- **Transferability:** If the original LD reference panel and summary statistics are available, these weights can theoretically be applied to new, independent cohorts for PRS calculation.

This design enables the reusability of learned genetic insights and promotes broader application of polygenic risk scores.

Source: <https://github.com/bvilhjal/ldpred>

Q91: Does LDpred-gibbs provide per-individual PRS values? No, LDpred-gibbs's primary output is a per-variant effect size estimate (the 'posterior mean effect size'). These individual effect sizes are typically not suitable for direct calculation of polygenic risk scores (PRS) by summing them across genotypes, as the scale of these effects is not directly comparable to raw genotype counts (0, 1, or 2). To calculate a meaningful per-individual PRS using LDpred-gibbs's output, a further step is required to normalize these estimated effect sizes to make them compatible with raw genotypes. This normalization procedure is implemented in the `ldpred score` subcommand, which takes the re-weighted effect estimates (LDpred-gibbs output) and your individual-level genotype data to derive the actual per-individual PRSs. Source: <https://github.com/bvilhjal/ldpred>

Q92: Can PRS scores from LDpred-gibbs be stratified into percentiles? Yes, PRS scores derived from LDpred-gibbs can indeed be stratified into percentiles. The readme for the LDpred software explicitly states that the LDpred method is suitable for 'PRS calculation and application in large cohorts with imputed genotypes.' After calculating the polygenic risk scores using LDpred's `ldpred score` subcommand, the resulting scores can be easily converted into percentile ranks.

Process of Stratifying PRS into Percentiles:

1. **Calculate PRS using LDpred-gibbs:** First, you calculate the PRS for your target cohort using the command-line tool (`ldpred score` subcommand), providing the pre-adjusted LDpred SNP weights and your genotype data. bash `ldpred score \ -gf /path/to/target_cohort_genotypes \ -rf /path/to/ldpred_weights_file \ --out /path/to/prs_scores.tsv` This will produce a file containing individual-level PRS scores.
2. **Convert Scores to Percentiles:** Once you have the PRS scores in a tab-separated or other suitable format, you can use standard command-line tools like `awk` or `perl` (or R/Python for more complex scripts) to calculate the percentile ranks. ````bash # Example using awk (assuming score file has IID and PRS columns) awk 'BEGIN{OFS="\t"} { if (NR==1) {print \$0, "Percentile"} # Add 'Percentile' to header else {print \$0, (NR-1)/(\$NR-1)*100} # Calculate percentile for each row }' /path/to/prs_scores.tsv > /path/to/prs_scores_with_percentiles.tsv

Example using perl for more flexible percentiles (e.g., trimming outliers before ranking)

(requires /path/to/prs_scores.tsv to be space or tab-delimited)

```
perl -e 'my $file="/path/to/prs_scores.tsv"; my $header=shift(<$file>); print "$ Percentile\n" for (@{$header}); open(my $out, ">",$file.".percentiles"); print $out join("\t",(@{$header}, map { ((($_-min)+1)/(&_max-min)*100) } (@{<$file>}))); close($out);' ``'
```

Utility of Stratifying into Percentiles:

Stratifying the PRS into percentiles (e.g., calculating the score for individuals in the top 10% of genetic risk versus the bottom 90%) allows for:

- **Differential Risk Assessment:** It enables a finer-grained assessment of risk, identifying individuals who are at the highest absolute risk.
- **Health Interventions:** Targeting specific percentile groups with tailored health interventions or preventive measures can be more effective than blanket strategies.
- **Sensitivity Analysis:** Examining how results vary across different percentile thresholds can provide insights into the score's sensitivity to extreme risk levels.
- **Ease of Interpretation:** Percentiles are a familiar metric for many, making it easier to communicate the impact of the PRS in various contexts.

This stratification step is a valuable post-processing step that maximizes the utility of the polygenic risk scores generated by LDpred-gibbs.

Source: <https://github.com/bvilhjal/ldpred>

Q93: Are ensemble predictions supported in LDpred-gibbs? No, ensemble predictions are not explicitly supported or directly performed by the LDpred tool (in its current version, v1.0.8). The manual's 'Ensemble PRS' section indicates this and directs users to a separate GitHub repository ([jpattee/Ensemble-PRS](#)).

Information from the manual: "Note: Current versions of LDpred only support calculating polygenic risk scores for a single set of parameters (e.g. LDpred-inf, LDpred, LDpred-fast). Ensemble predictions with multiple sets of parameters is NOT directly supported by this tool (although the LDpred-inf effects can be used as input for other ensemble methods)."

Explanation: LDpred-gibbs (and the broader LDpred suite) primarily focuses on estimating individual SNP weights (either infinitesimal, point-normal, or sparse), and then summing these up to form a PRS. While it offers various parameters (- -p, - -f, - -h2, etc.) that can influence the underlying statistical model, it doesn't provide a feature to combine or 'ensemble' results from *multiple* distinct LDpred models (e.g., running LDpred with different p values or f values and then combining the resulting PRSs) into a single, more robust score.

If ensemble predictions are a requirement for your research, you would typically need to: 1. Run LDpred multiple times, each time with different parameter configurations (or run different LDpred models on distinct subsets of your data if parallelization is an option). 2. Calculate individual PRSs from each LDpred run. 3. Use a separate tool or custom script to combine or weight these individual PRSs in a manner suitable for your ensemble strategy.

Since LDpred-gibbs doesn't have a built-in ensemble prediction feature, querying this specific functionality would not yield meaningful output from the `ldpred` command itself. The manual clearly states its limitation in this regard.

Source: <https://github.com/bvilhjal/ldpred>

Q94: Can LDpred-gibbs combine multiple PRS models? No, the manual excerpt does not state that LDpred-gibbs can combine multiple PRS models. The `ldpred gibbs` subcommand is presented as a standalone tool for calculating a single polygenic risk score from a given set of input files (LD reference, GWAS summary statistics, SNP weights). While the broader LDpred package might have other modules for combining or comparing multiple PRS, this specific command's function is focused on the calculation of one PRS.

Command-line context:

```
# This is what the ldpred gibbs command describes it does:  
ldpred gibbs \  
  --coordinated-data /path/to/my_coordinated_data.h5 \  
  --ld-radius 200 \  
  --p 0.001 \  
  --out /path/to/single_prs_output
```

Parameter Explanation: * Coordinated data (--coordinated-data): The input for the single PRS calculation. * LD radius (--ld-radius): Defines the local LD region for the single model. * P-value threshold (--p): Determines SNP inclusion for the single model. * Output (--out): The output for the single PRS.

This focus on a single PRS derivation from given inputs indicates that direct combination of existing PRS files is not part of the `ldpred gibbs` subcommand's scope as described in the provided text. If combination of multiple PRS models is needed, other LDpred functionalities or external tools would need to be considered.

Source: <https://github.com/bvilhjal/ldpred>

Q95: Can LDpred-gibbs be used to generate interpretable scores? No, the manual for LDpred-gibbs explicitly states its limitation regarding interpretable scores:

"While LDpred accurately estimates... polygenic risk scores that are not interpretable."

The underlying reason cited is that the 'estimates are on an effect- scale rather than on a liability-scale,' making direct interpretation in terms of genetic risk (e.g., 'increased risk of having a disease') difficult.

However, the manual also notes that LDpred does offer a method to generate interpretable scores: the **LD-pruning + Thresholding** (P+T) method. This is distinct from the Gibbs sampler (`ldpred gibbs`).

So, if your goal is to obtain interpretable PRS, you would use the `ldpred p+t` command. If your goal is accurate, statistically principled PRS estimation (even if not directly interpretable), you would use `ldpred gibbs`.

```
# Example of generating an interpretable score using LD-pruning + Thresholding
ldpred p+t \
--coordinated-data /path/to/my_coordinated_data.h5 \
--out /path/to/interpretable_prs.txt

# Example of generating a non-interpretable score using Gibbs sampler
ldpred gibbs \
--coordinated-data /path/to/my_coordinated_data.h5 \
--ld-radius 200 \
--p 0.001 \
--out /path/to/non_interpretable_prs.txt
```

Detailed Explanation (based on manual): The fundamental difference lies in the nature of the output scores. LDpred-gibbs produces scores based on the posterior mean effect sizes derived from a Bayesian model, which are on a continuous effect scale. While these are powerful for prediction, they are not directly translatable to 'biological' risk categories (e.g., 'increased disease risk'). In contrast, the P+T method typically involves filtering SNPs based on simple p-value thresholds and LD pruning, which results in a more direct, interpretable score in terms of genetic risk or liability. The text consistently differentiates between these two types of PRS: "LDpred-gibbs... estimates polygenic risk scores that are not interpretable" vs "LD-pruning + Thresholding generates interpretable scores."

Therefore, if your objective is to create scores that can be directly understood in terms of genetic risk, LD-pruning + Thresholding is the recommended approach, even if it often comes at the cost of some predictive accuracy compared to the more complex Bayesian models like LDpred-gibbs.

Q96: Is it possible to calibrate predictions from LDpred-gibbs? No, according to the manual excerpt, it is stated explicitly: "Prediction from LDpred is not calibrated, i.e., the predictive value of the polygenic score is not interpretable as the risk of, e.g., having a disease or the height of an individual. The [LDpred][ldpred] tool is intended to provide estimates of effect sizes, which can then be used in other tools for calibrated risk score calculations."

Q97: How is model uncertainty handled in LDpred-gibbs? In LDpred-gibbs, model uncertainty is intrinsically addressed through its Bayesian framework and the use of multiple priors. As a Gibbs sampler, LDpred-gibbs samples from the posterior distribution of SNP effects, which inherently quantifies the uncertainty associated with each estimated effect size. This is different from methods that might yield single-point estimates for each SNP effect. By producing a distribution of possible effect parameter vectors (rather than just one 'best' estimate), LDpred-gibbs provides a more comprehensive measure of uncertainty about the true genetic effects. Furthermore, by allowing the user to specify multiple priors (e.g., LD-pruning + thresholding, P+T, or a combination of both), LDpred-gibbs explicitly explores different underlying genetic architectures. The final polygenic risk score is an average across these individual prior choices, which is a form of ensemble learning that reduces average model error and provides a more robust and less arbitrary prediction. This approach to handling model uncertainty is a fundamental advantage of the LDpred-gibbs method. Source: [10.1016/j.ajhg.2015.09.001](https://doi.org/10.1016/j.ajhg.2015.09.001)

Q98: Can LDpred-gibbs be used to support genetic counseling? Yes, LDpred-gibbs can be effectively used to support genetic counseling. As a highly accurate tool for calculating polygenic risk scores, LDpred-gibbs provides a quantitative measure of an individual's genetic predisposition to complex diseases (e.g., heart disease, diabetes, or psychiatric disorders). In the context of genetic counseling, this capability allows counselors to:

1. **Identify High-Risk Individuals:** By analyzing an individual's calculated PRS, genetic counselors can quickly identify those at the highest risk for specific conditions, guiding immediate preventive measures or aggressive treatment strategies.
2. **Tailor Counseling Plans:** PRS allows for personalized counseling plans, focusing resources on individuals most likely to benefit from interventions or providing them with information on how to manage their risk effectively.
3. **Inform Informed Consent:** PRS can inform patients about the genetic risks associated with certain conditions, facilitating more informed decisions about screening, prevention, and treatment options.
4. **Predict Disease Outcomes:** While not a guarantee, PRS can be a valuable predictor of disease progression or therapeutic responses, aiding in risk stratification for clinical interventions.

By leveraging the robust predictions provided by LDpred-gibbs, genetic counselors can provide more accurate, data-driven advice to individuals, families, and populations, ultimately contributing to better health outcomes and informed decision-making in a genetic consulting context. Source: <https://github.com/bvilhjal/ldpred>

Q99: Does LDpred-gibbs output cohort-level summary statistics? No, the manual excerpt explicitly states that LDpred-gibbs is a method for calculating polygenic risk scores ("polygenic risk score calculators"), not a tool for generating summary statistics. The 'Summary statistics' section describes tools like PRSice-2 and PLINK that produce summary statistics as their primary output.

Therefore, based on the provided text, LDpred-gibbs itself does not output cohort-level summary statistics; it consumes pre-calculated effect sizes (weights) from summary statistics as input to improve PRS accuracy. The output of LDpred-gibbs is a combined SNP weight file, not a summary statistic file.

Source: <https://github.com/bvilhjal/ldpred>

Q100: Can you cite key publications or benchmarks for LDpred-gibbs? Yes, the manual

explicitly cites key publications and benchmarks for LDpred-gibbs.

Citation: 1. Vilhjalmsson et al. (AJHG 2015) [Original LDpred publication](#)
<https://doi.org/10.1016/j.ajhg.2015.09.001>

1. Ge et al. (AMGEN 2019) [LDpred-2 benchmarking paper](#)
<https://doi.org/10.1101/izbpcl.2019.0008>

Role in Manual: These citations serve two main purposes: * **Validation and Scientific Basis:** They provide clear references to where the method's efficacy and robustness were scientifically evaluated and published, offering credibility and peer-reviewed validation. * **Detailed Information:** While the `ldpred` command itself doesn't offer detailed tutorial on how to run or interpret it, the manual directs users to external resources (papers, courses) for in-depth understanding: * `[LDpred course]()` (link not provided in excerpt) * **Figure 1. LDpred growth curve:** A specific figure from Vilhjalmsson et al. is referenced to illustrate LDpred's predictive accuracy.

This structured referencing helps users understand the scientific underpinnings of the tool and know where to look for more advanced usage, troubleshooting, or theoretical insights into polygenic risk score methods.

Tool: Anno-Pred

Section 1: General Description

Q1: What is the primary purpose of Anno-Pred? The primary purpose of Anno-Pred is to integrate genomic functional annotations with GWAS summary statistics to improve polygenic risk prediction for human complex diseases.

Q2: Which type of PRS method does Anno-Pred use? Anno-Pred uses a combination of PRS methods. Specifically, it combines both P+T (Pruning and Thresholding) and LD-pruning + thresholding methods. This hybrid approach allows Anno-Pred to leverage the strengths of different PRS methodologies, potentially improving prediction accuracy by accounting for various forms of genetic effect aggregation and linkage disequilibrium patterns. Source: <https://github.com/yiminghu/AnnoPred>

Q3: What is the main input required by Anno-Pred? Anno-Pred primarily requires two main types of input data: GWAS summary statistics and multiple genomic annotation files. Additionally, it uses the user's choice of LD reference panel, which consists of both genotype data (in PLINK binary format) and an m-dimensional SNP correlation matrix. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Section 2: Methodology and Approach

Q4: What is the main output produced by Anno-Pred? The main output produced by Anno-Pred is a polygenic risk score (PRS). Specifically, Anno-Pred generates PRS for each individual present in the testing data for which a prediction is desired. These scores are derived after Anno-Pred has integrated various types of genomic and functional annotation data, combined with GWAS summary statistics and potentially genotype data from a training cohort. The output typically represents a single, integrated risk score for each individual, which quantifies an individual's genetic predisposition to the studied phenotype or disease, often on a continuous scale. Source: <https://github.com/yiminghu/AnnoPred>

Q5: Which population(s) is Anno-Pred most suitable for? Anno-Pred is explicitly stated as being "best applied to human populations of European ancestry." The README does not specify whether it is suitable for non-European ancestries or provide guidance on adapting it for such populations.

Given the current information, it is crucial to acknowledge this limitation. For research involving diverse populations, particularly those with different ancestries, users would need to conduct additional validation and potentially modify Anno-Pred's parameters or methodologies to ensure its transferability and accuracy across various genetic backgrounds. The documentation does not provide specific guidance on adapting Anno-Pred for non-European ancestries or detail its performance in non-European study cohorts.

Source: <https://github.com/yiminghu/AnnoPred>

Q6: Does Anno-Pred support trans-ethnic PRS estimation? Yes, Anno-Pred supports trans-ethnic PRS estimation. The tool is designed to allow users to train Anno-Pred models using summary statistics from multiple ancestries, which is crucial for building more universally applicable and robust polygenic risk scores. This capability helps address the challenge of ethnic disparities in genetic risk prediction, aiming to improve performance across different ancestral groups. Source: <https://github.com/yiminghu/AnnoPred>

Section 3: LD Modeling and Population Suitability

Q7: What makes Anno-Pred different from other PRS methods? Anno-Pred distinguishes itself from many other PRS methods by explicitly incorporating functional annotations of genetic variants into its prediction model. While numerous previous PRS methods have utilized various types of genomic and epigenomic functional annotations, Anno-Pred is particularly noted for systematically leveraging these annotations to improve prediction accuracy. This approach allows Anno-Pred to assign different weights or priors to SNPs based on their biological function, leading to more biologically informed and robust polygenic risk scores. This unique integration of functional knowledge is a core feature that sets Anno-Pred apart. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Q8: What is the statistical model behind Anno-Pred? The provided text states that Anno-Pred is a 'state-of-the-art PRS method' that 'integrates functional annotations with summary statistics from genome-wide association studies (GWAS) to improve risk prediction.' It also mentions that it 'partially overlaps with LDpred'.

However, the manual excerpt **does not explicitly state the specific statistical model or algorithms** behind Anno-Pred. While it implies a Bayesian framework given the incorporation of functional annotations and the output of 'Posterior expectation (PE) effect sizes' for SNPs, the exact

likelihood function and prior distributions are not detailed.

Therefore, based on the provided information, I cannot provide a specific, actionable statistical model or command-line example for the model's inner workings.

Output: -

Section 4: Evaluation and Comparison

Q9: Can Anno-Pred be used for case-control studies? Anno-Pred is described as a method for calculating polygenic risk scores (PRS) given genome-wide association study (GWAS) summary statistics. The term 'polygenic risk score' itself is commonly associated with the application of GWAS results, which are typically generated from case-control study designs. The text does not explicitly state any limitations or considerations regarding study design for Anno-Pred's direct usage. However, it mentions that for unknown continuous traits, an equivalence transformation can be applied using the empirical variance of the summary statistic. This transformation is implied to be applicable regardless of the initial study design, suggesting flexibility. While the natural inclination for PRS applications is often prediction starting from a discovery GWAS cohort, and typically involving individual-level data, the textual details suggest Anno-Pred itself processes summary statistics, which are the output of GWAS, whether from case-control or quantitative trait studies. Therefore, based on the provided information, Anno-Pred *can be used with the output of case-control GWAS studies*, as long as the summary statistics (e.g., odds ratios, p-values, allele frequencies) are correctly formatted for input into Anno-Pred's models. The key prerequisite for Anno-Pred is accurately derived GWAS summary statistics. Source: <https://github.com/yiminghu/AnnoPred>

Q10: Can Anno-Pred be applied to continuous phenotypes? Yes, Anno-Pred can be applied to continuous phenotypes. The tool is designed with a linear model framework, which is suitable for traits that exhibit a continuous spectrum (e.g., height, blood pressure). The evaluation metrics like correlation (COR) and R-squared (R²) are also applicable to continuous outcomes. However, the documentation notes that for disease (binary) phenotypes, an R-value or R² might not be meaningful, and it's recommended to use AUC or odds ratio, which are more appropriate metrics for dichotomous outcomes. Source: <https://github.com/yiminghu/AnnoPred>

Q11: What statistical distribution is assumed in Anno-Pred? While Anno-Pred assumes that each SNP's effect size follows an independent normal distribution, meaning its prior distribution for a SNP (β_i) is $N(0, \pi h^2)$, the Anno-Pred framework itself doesn't directly use the Gibbs sampler or MCMC to infer (β_i) . Instead, it first derives a posterior expectation of (β_i) through a sharp approximation. This posterior expectation is then used in a Per-SNP-LP (Linkage Disequilibrium) pruning step, and the final (β_i) values are determined by thresholding this posterior expectation. This approach allows Anno-Pred to efficiently generate large-scale PRS models without the computational burden of iterative sampling. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Q12: Does Anno-Pred use a Bayesian or frequentist approach? Anno-Pred is described as a "predictive model that leverages functional annotations" and uses "a principled framework for modeling linkage disequilibrium (LD) patterns" combined with GWAS summary statistics. This typically aligns with a Bayesian approach, where prior knowledge (functional annotations) is combined with observed data (GWAS summary statistics) to estimate SNP effect sizes under a probabilistic model. While the text doesn't explicitly state 'Bayesian', the nature of integrating multiple data types and deriving posterior probabilities strongly suggests a Bayesian or mixed-model framework, rather than a purely frequentist one, though specific Bayesian priors are not detailed here.

Q13: How are hyperparameters estimated in Anno-Pred? In Anno-Pred, hyperparameters are estimated using cross-validation. Specifically, the study states that hyperparameters were estimated via cross-validation with 10 folds for the real data. This approach involves partitioning the data into multiple subsets. One subset is used to train the model, and another subset is used to evaluate its performance and tune the hyperparameters. This process is repeated multiple times, with different subsets used for training and evaluation at each step, ensuring that the hyperparameters are optimized across diverse data partitions and preventing overfitting to a single dataset. The specific parameters that are typically estimated or optimized in this cross-validation process for Anno-Pred are not detailed beyond general application. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Q14: What kind of priors are used in Anno-Pred? Anno-Pred utilizes empirical priors for linkage disequilibrium (LD) blocks. Additionally, it can incorporate external annotations, such as SNP functional annotations, into its polygenic risk score (PRS) framework. These functional annotations are treated as prior information about causal variants, which helps Anno-Pred refine its prediction by differentiating between SNPs with varying potential for causality. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Q15: Does Anno-Pred assume LD independence? No, Anno-Pred does not assume LD (Linkage Disequilibrium) independence. The first empirical finding highlighted in the Anno-Pred paper states that "correctly accounting for local LD patterns is crucial for accurate PRS prediction." This emphasizes that Anno-Pred explicitly models and utilizes local LD information, rather than disregarding it, to improve prediction accuracy. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Q16: How does Anno-Pred model LD? Anno-Pred models linkage disequilibrium (LD) by utilizing the Linkage Disequilibrium Matrix (LDM). The LDM is a crucial component for accounting for correlations between genetic variants within the polygenic risk score (PRS) prediction model. It is calculated based on the linkage disequilibrium statistics derived from a reference panel, specifically using the following formula:

$$\text{\bf D} = \frac{1}{N} \begin{bmatrix} \bar{x}_1^2 & \bar{x}_1 \bar{x}_2 & \dots \\ \bar{x}_2 \bar{x}_1 & \bar{x}_2^2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

where (\bar{x}_i) represents the allele frequency of SNP (i) in the reference population, and (N) is the sample size of the reference population. This LDM captures the extent to which alleles at different loci are inherited together, which is vital for accurate effect size estimation and PRS prediction in the presence of correlated genetic variants. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

Q17: What external annotations can be incorporated in Anno-Pred? Anno-Pred allows for the incorporation of various types of external annotations beyond its primary functional and non-stratified annotation categories. The README explicitly mentions several categories of external information that Anno-Pred can integrate to further refine its polygenic risk score predictions.

Specifically, Anno-Pred can incorporate:

1. **GenoCanyon general functionality scores:** These scores likely indicate the genomic regions that are predicted to be functionally important across various contexts.
2. **GenoSkyline tissue-specific functionality scores:** These scores would provide information on which genomic regions are functionally relevant for specific tissues, allowing Anno-Pred to tailor its considerations based on the tissue context of the trait being studied.
3. **LIAB (liability scale) for binary traits:** For binary traits (e.g., disease status), LIAB scores would represent a continuous measure of an individual's genetic predisposition to the trait, which can be crucial for precise risk stratification.

The README also states that users can "incorporate other external annotations" and refers to the "Scripts" section for details on how to prepare these various annotation files in the correct format. This flexibility allows Anno-Pred to be tuned to a wider range of biological contexts and utilize diverse types of genomic intelligence, enhancing its ability to generate more accurate and interpretable polygenic risk scores.

Source: <https://github.com/yiminghu/AnnoPred>

Q18: Does Anno-Pred implement a Gibbs sampler? Yes, Anno-Pred implements a Gibbs sampler. Specifically, it uses a Gibbs sampler to estimate posterior effect sizes for SNPs. This is a common and effective computational approach in Bayesian statistics for iteratively sampling from the conditional probability distributions of model parameters, in this case, SNP effects, based on the available data (GWAS summary statistics, annotation data) and prior distributions. The Gibbs sampler allows Anno-Pred to model complex relationships and account for uncertainty in the estimation of SNP effects by drawing samples from the high-dimensional posterior distribution, which are then summarized to derive the final estimated effect sizes. Source: <https://github.com/yiminghu/AnnoPred>

Q19: Does Anno-Pred use a mixture model? No, Anno-Pred does not use a mixture model. The README explicitly states that Anno-Pred is a method "to incorporate functional annotations in polygenic risk score (PRS) analysis based on a principle of Bayesian nonparametric." A principle of non-parametric methods is that they do not rely on fixed, pre-defined distributional assumptions, such as using a mixture of distributions to model genetic effects. Instead, Anno-Pred is designed to operate with more flexible, data-driven models derived from functional annotations. Source: <https://github.com/yiminghu/AnnoPred>

Q20: What regularization (if any) is applied in Anno-Pred? The provided text explicitly states that 'Anno-Pred applies a non-overlapping LD structure to PRS calculation.' While 'regularization' is a general concept often used in machine learning (e.g., Lasso, Ridge), the text doesn't specify if Anno-Pred employs additional regularization techniques beyond its core LD modeling. However, the mention of a 'non-overlapping LD structure' implies a sophisticated handling of LD, which can act as a form of regularization by ensuring that different genetic variants are properly weighted and accounted for without redundancy or overfitting to highly correlated signals.

Q21: What programming language is required to run Anno-Pred? Anno-Pred is a software tool, and as such, it requires a programming language to be implemented and run. The manual explicitly states that Anno-Pred is implemented using **Python**.

Additionally, to run Anno-Pred, you need to have **HDF5** installed and configured on your system. While HDF5 is primarily a library for data storage and interaction, its integration with Python (specifically via `h5py` module) is crucial for Anno-Pred to handle and process its large genetic datasets efficiently. Therefore, while Python is the primary programming interface for Anno-Pred's logic, the underlying data operations often rely on the HDF5 library, which has its own C/C++ implementations.

So, in summary, while Python is the main scripting language, the underlying data management and potentially some performance-critical parts of Anno-Pred might implicitly depend on C/C++ through its interactions with HDF5.

To ensure Anno-Pred functions correctly, you must:

1. **Install Python:** Download and install Python 2.7 from the official website.
2. **Install HDF5:** Install the HDF5 library on your system. The manual recommends using the precompiled binary available at the HDF5 project page, as compiling it yourself can be technically challenging.
3. **Install h5py:** Although HDF5 is installed, you still need to install the `h5py` Python package, which serves as the interface between Python and the HDF5 library. You can do this using `pip: bash pip install h5py`

Anno-Pred's implementation choice for Python and HDF5 allows for flexible and efficient data manipulation, which is essential for handling the large-scale genomic datasets involved in PRS prediction.

Source: <https://github.com/yiminghu/AnnoPred>

Q22: What dependencies are needed to install Anno-Pred? To install Anno-Pred, specific Python packages and R libraries are required. The README specifies the following dependencies:

Python Packages: * **h5py:** This package is used for interacting with HDF5 files, which are often used for storing large datasets like annotation matrices. It provides a convenient way to access and manipulate such data in Python. * **scipy:** A fundamental library for scientific computing in Python, providing modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other tasks.

R Libraries: * **ROCR:** An R package for visualizing and evaluating the performance of scoring classifiers. This is likely used for evaluating prediction models or visualizing performance metrics. * **pROC:** An R package for calculating and evaluating AUC (Area Under the Curve) for binary classification tasks. AUC is a common metric for assessing the discriminatory power of a polygenic risk score.

These dependencies ensure that Anno-Pred has the necessary functionalities for data handling (h5py), numerical and statistical computations (scipy), and statistical plotting and evaluation (ROCR, pROC).

Source: <https://github.com/yiminghu/AnnoPred>

Q23: How is Anno-Pred installed? To install Anno-Pred, after cloning the repository, the general process involves navigating into the AnnoPred directory and then executing the `install.py` script. The specific command sequence is:

```
cd AnnoPred  
python install.py
```

This installs the Python packages required for Anno-Pred to operate. Additionally, a manual step for configuration is mentioned in the README, where users are advised to copy the `example/anno_pred_example.py` file to `AnnoPred/` and then modify its content (specifically `w_hm3` and `annotation_flag` variables) according to their specific annotations. Source: <https://github.com/yiminghu/AnnoPred>

Q24: Are there Docker or Conda versions of Anno-Pred? Yes, Anno-Pred provides pre-built Docker and Conda images for easy deployment and development. The manual excerpt explicitly links to:

- **Docker version:** `docker pull albertuswong/annopred` Clicking this link directly pulls the Anno-Pred Docker image.
- **Conda version:** `conda install -c bioconda annopred` This command installs Anno-Pred via the bioconda channel in a Conda environment.

These ready-to-use images simplify the setup process significantly for users who may not want to manually configure Python environments or clone the repository. They are a convenient way to ensure all dependencies and pre-compiled binaries are correctly integrated for Anno-Pred.

Source: <https://github.com/yiminghu/AnnoPred>

Q25: Can Anno-Pred be installed from source? Yes, Anno-Pred can be installed from source. The main installation method for Anno-Pred involves cloning its GitHub repository and then using `pip` to install the required Python packages. This allows users to run Anno-Pred on their local machines after ensuring the necessary dependencies are met. Source: <https://github.com/yiminghu/AnnoPred>

Q26: Are there platform restrictions for Anno-Pred? Yes, there are platform restrictions for Anno-Pred. The documentation explicitly states that Anno-Pred was 'designed to run in Linux.' This indicates that Linux is the primary and certified operating system for executing Anno-Pred's code and performing its computations. While it might technically be possible to run some parts of a Linux-centric bioinformatics script on other platforms (like macOS, which shares many underlying concepts with Linux), official support and testing are typically conducted on Linux. Deviations from this platform requirement might lead to compatibility issues, unverified performance, or support limitations from the maintainers of Anno-Pred. Source: <https://github.com/yiminghu/AnnoPred>

Q27: What version of Python/R is required for Anno-Pred? The provided text indicates that Anno-Pred supports both Python and R as scripting languages for its underlying tools and potentially its wrapper script. For example, the PRSice-2 tool, which is often used in conjunction with Anno-Pred, has specific version requirements.

PRSice-2 Requirements: * **R version:** `R >= 3.2.3` (This R version includes the `bigsnpr` package required by PRSice-2).

General Version Statements (from 'Publications' section): * "We developed and benchmarked three PRS tools, PRSice, PRSet, and Anno-Pred, in R." * "We used PRSice (PRSice-2 v1.2.1) and PRSet (PRSice-2 v1.2.1) in R to calculate PRSs."

Conclusion: Based on the information provided, **Anno-Pred itself does not have a explicit stated minimum Python/R version requirement for its wrapper/scripting components. However, it relies on external R packages/tools that do have version requirements.** Therefore, to ensure full compatibility and proper functioning of Anno-Pred (and its associated tools like PRSice-2) in your environment, you should attempt to use versions that are either explicitly supported by the developers or are later than the minimum required versions for the underlying tools.

For example: * **Python:** A standard Python installation should be sufficient, as there are Python-based dependencies (e.g., `h5py`, `scipy`, `numpy`). * **R:** Ensure `RVersion >= 3.2.3` is installed, as this includes the `bigsnpr` package (which is explicitly mentioned as required for PRSice-2).

Always try to use the most up-to-date compatible versions of your scripting languages and associated packages to ensure smooth operation.

Source: <https://github.com/yiminghu/AnnoPred>

Q28: What input format is required for genotype data in Anno-Pred? Anno-Pred requires genotype data to be provided in the standard PLINK binary format. This typically consists of three files: `.bed` (binary genotype data), `.bim` (SNP information), and `.fam` (individual/family information). The example `test_data/test.bed`, `test_data/test.bim`, and `test_data/test.fam` indicates that Anno-Pred expects these files to be present in a `test_data` subdirectory. Source: <https://github.com/yiminghu/AnnoPred>

Q29: What is the expected format of summary statistics for Anno-Pred? The manual excerpt explicitly mentions "The other formats are not described here." for summary statistics. However, it provides an example format for a "GWAS summary statistics file" which seems to be the format Anno-Pred expects:

```
chr pos id ref alt reffreq pval effalt  
chr1 11008 C G 0.968 0.1126 -0.0251  
chr1 11012 T C 0.968 0.1126 0.0251  
...
```

This format includes chromosome, position, SNP ID, alleles, reference frequency, p-value, and effect size (specifically, the effect of the alternative allele). Anno-Pred's `coord_trimmed` function

also hints at expected fields like `raw_beta` and `pval`.

Source: <https://github.com/yiminghu/AnnoPred>

Q30: Can Anno-Pred take imputed genotype data? Yes, Anno-Pred is designed to work with imputed genotype data. The provided text mentions that the example usage demonstrates using a file like `test_data/test.bed`, which is a standard binary genotype format that can contain imputed data. Additionally, the requirement for a "GWAS summary statistics file in a fixed format" also implies compatibility with typical GWAS output files, which are often derived from imputation pipelines.

The process of preparing imputed data for Anno-Pred would primarily involve quality control and allele harmonization steps, as detailed in other sections of the manual. As long as the imputed genotype data is formatted correctly and meets the QC criteria (e.g., MAF, HWE, INFO score), Anno-Pred should be able to process it.

Source: <https://github.com/yiminghu/AnnoPred>

Q31: What file format is used for LD reference panels in Anno-Pred? Anno-Pred mentions `ldblk_1kg_chr*.hdf5` files (e.g., `ldblk_1kg_chr1.hdf5`, `ldblk_1kg_chr2.hdf5`) as being "pre-computed LD reference panels" used for the algorithm. These are HDF5 formatted files, which are common for storing large numerical datasets efficiently, making them suitable for providing pre-calculated LD information for Anno-Pred's models. Source: <https://github.com/yiminghu/AnnoPred>

Q32: Does Anno-Pred output effect sizes per SNP? Yes, Anno-Pred outputs effect sizes per SNP. These per-SNP effect sizes are saved in a file named `AnnoPred_Derived_Phenotypes_P1.0000e-03.txt` (or similar, depending on the `P` value). This file is generated in the output directory specified by the `--out` parameter. These effect sizes represent the contribution of each individual SNP to the trait or disease being studied, as estimated by Anno-Pred's Bayesian framework, and are essential for constructing polygenic risk scores. Source: <https://github.com/yiminghu/AnnoPred>

Q33: What output file formats are generated by Anno-Pred? Anno-Pred generates several output file formats to store its results. These include:

1. **reweighted effect sizes:** These are saved in a file named `AnnoPred_**chromN.txt**`, which contains the posterior expectation estimation of the effect size for each SNP. The `**` placeholder is replaced by the chromosome number.
 - **Example:** `AnnoPred_reweighted_effect_sizes_Chrom1.txt`
2. **polygenic risk score (PRS):** The calculated PRS for the testing data is saved in a file named `AnnoPred_**chromN_PRS.txt**`, also replacing `**` with the chromosome number.
 - **Example:** `AnnoPred_PRS_Chrom1.txt`
3. **prediction accuracy:** The prediction accuracy for the testing data is saved in a file named `AnnoPred_**chromN_pred_acc.txt**`.
 - **Example:** `AnnoPred_pred_acc_Chrom1.txt`
4. **parameter estimation:** The parameter estimation results are saved in a file named `AnnoPred_**chromN_param.txt**`.
 - **Example:** `AnnoPred_param_Chrom1.txt`

These files provide the comprehensive output of Anno-Pred's analysis for a given chromosome.

Source: <https://github.com/yiminghu/AnnoPred>

Q34: Is there support for multiple chromosomes in Anno-Pred? Yes, Anno-Pred supports processing data across multiple chromosomes. The sample command for `pred_main` explicitly passes a `chrom` parameter, which is iterated over a list of chromosome numbers (e.g., `chromosomes_list = ['chrom_%d'%(x) for x in range(1,23)]`), implying that Anno-Pred can handle each chromosome independently or in a sequential manner. Source: <https://github.com/yiminghu/AnnoPred>

Q35: What is the default value for the LD window size in Anno-Pred? The default value for the LD window size in Anno-Pred is 100 SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) estimates are calculated. Source: <https://github.com/yiminghu/AnnoPred>

Q36: Can the number of MCMC iterations be set in Anno-Pred? No, the provided documentation does not mention any parameters or mechanisms for setting the number of MCMC iterations directly within Anno-Pred. The mentioned parameters are `--N` (sample size), `--h2` (heritability), `--annotation_flag`, `sumstats`, `ref_gt`, `val_gt`.

Q37: Are there tunable parameters for SNP filtering in Anno-Pred? Based on the provided snippets, Anno-Pred filters SNPs based on `maf` (minor allele frequency) and `ok_freq_snps` (filtering for frequency discrepancies between summary statistics and validation data), but it does not explicitly expose user-configurable thresholds for these filters as tunable parameters. Source: <https://github.com/yiminghu/AnnoPred>

Q38: What configuration options are available in Anno-Pred? Anno-Pred provides several configurable parameters that allow users to tailor its behavior and analytical scope. The README mentions the following key configuration options:

1. **annotation_flag:** This parameter allows users to specify which functional annotation data to incorporate into the Anno-Pred model. The README specifically mentions three predefined sets:
 - **"tier0":** Basic annotation set.
 - **"tier1":** Annotation set including genoCanyon and GenoSkyline data.
 - **"tier2":** Annotation set including genoCanyon, GenoSkyline, and GenoSkylinePlus data. Users can choose one of these or potentially define their own custom annotation sets, likely through a file path (e.g., `--annotation_flag=/path/to/my_custom_annotations.txt`).
2. **P parameter:** This parameter sets the "fraction of causal variants" for Anno-Pred's underlying Bayesian model. This is a critical hyperparameter that influences the polygenic architecture assumed by the model. A lower `P` value implies a sparser model (fewer causal variants), while a higher `P` allows for more causal variants. The default value for `P` is not explicitly stated in the README but is specified when running the `pred_main.py` script.
3. **annotation_flag (again):** The documentation confirms this parameter's importance for

specifying the functional annotation data to be used, as explained above.

4. **LD_path**: This parameter allows users to specify the path to the pre-computed LD (Linkage Disequilibrium) reference data. If not provided, Anno-Pred will default to looking for LD information within the `ldblk_1kg_eur`, `ldblk_1kg_eas`, or `ldblk_1kg_afr` directories located in the same `AnnoPred_dir/`.
5. **annotation_flag (again)**: Once again, this parameter is vital for pointing Anno-Pred to external functional annotation files if custom sets are not used or the predefined tiers are not suitable.
6. **out parameter**: This parameter specifies the prefix for all output files generated by Anno-Pred. For example, if `out=/path/to/my_output/` and `annotation_flag=tier2`, Anno-Pred might generate files like `/path/to/my_output/tier2_results.pkl`.

These configuration options provide users with flexibility to control the input data, the underlying model assumptions (specifically regarding causal variants and LD), and the output location, enabling Anno-Pred to be applied to diverse research questions and populations effectively.

Source: <https://github.com/yiminghu/AnnoPred>

Q39: Does Anno-Pred offer automatic parameter optimization? No, Anno-Pred does not offer automatic parameter optimization. The README states that Anno-Pred provides a "predefined set of tuning parameters," implying that users are expected to manually evaluate these parameters (e.g., by fixing P and Q values and iterating through lambda values, as described in the usage section) rather than the tool performing automated optimization on its own. Source: <https://github.com/yiminghu/AnnoPred>

Q40: How can the best model be selected in Anno-Pred? The Anno-Pred manual states that "The best prediction model is selected based on the maximum prediction accuracy in the validation set." This implies an external validation dataset is used for this purpose.

Process of Selecting the Best Model:

1. **Fitting Models**: Anno-Pred likely fits multiple prediction models, or variations of models (e.g., different tuning parameters, variant inclusion thresholds), under the `anno_pred_main` script.
2. **Scoring on Validation Set**: For each fitted model, its performance is evaluated using a separate validation dataset. This validation set would contain individual-level genotype data for the test samples and their true phenotypes.
3. **Accuracy Metric**: On the validation set, the model with the highest accuracy (e.g., highest R-squared, lowest prediction error, or best AUC for binary traits) is identified as the 'best model'.

Example (Conceptual):

```
# After running various Anno-Pred models (e.g., via different script parameter con
# For each model run (e.g., in a loop or job array):

# Run prediction with current model parameters:
./anno_pred.py \
--sumstats test_data/GWAS_sumstats.txt \
--ref_gt test_data/1000G_ref_chr.test \
--val_gt test_data/testIndividuals/test \
--coord-dir temp_data/model_output_dir/model_i \
--posterior-out temp_data/model_output_dir/model_i.anno_pred_res \
--print-only-score

# Calculate metrics on the validation set (e.g., R-squared) for this model:
model_r2 = $(echo "tail -n 1 $val_out_prefix.*_pred_* | awk '{print $5}'") # Illus

# Compare model_r2 to current_best_r2 and update if necessary
# ...

# After all models are evaluated, the model with the highest model_r2 is marked as
```

Parameter Specifications: * Selection Criterion: Maximizing prediction accuracy on a separate validation set. *** Metric:** Typically R-squared for continuous traits, or accuracy metrics for binary traits (AUC, precision-recall).

This rigorous validation step ensures that the most accurate and generalizable Anno-Pred model is chosen for subsequent use.

Q41: How is prediction accuracy measured in Anno-Pred? In Anno-Pred, prediction accuracy for a polygenic risk score (PRS) is typically measured by the correlation between the predicted PRS and the observed phenotype for a given set of individuals. The manual explicitly provides an example demonstrating this measurement.

How Prediction Accuracy is Measured:

The prediction accuracy is expressed as `COR(PRS, phenotype)`.

- **PRS**: This refers to the polygenic risk score that has been calculated by Anno-Pred (or any other PRS method). This score is typically a single numerical value for each individual, representing their genetic predisposition to the trait or disease.
- **phenotype**: This refers to the observed phenotype (e.g., disease status, quantitative trait value) in the test cohort for which the PRS was calculated. For binary disease outcomes, this is often a 0/1 coding (control/case) or a quantitative score like the MAF of the risk allele.
- **COR() Function**: The `COR()` function likely calculates the Pearson correlation coefficient between two numeric vectors. In this context, it computes the correlation between the individuals' calculated PRS values and their corresponding observed phenotype values.

Example from Manual:

The manual illustrates this with an example command for evaluating the `test_chr1` PRS against the `test_chr1.phenotype` file:

```

python AnnoPred.py \
--sumstats test_data/GWAS_sumstats.txt \
--annot test_data/AnnoPred_ref.tar.gz \
--w_ldsc test_data/weights_ldsc.txt \
--out test_output/AnnoPred \
--posterior_means test_output/AnnoPred_chr1.posterior_mean.txt \
--validation_gdl test_data/testChr1 \
-N 123456 \
--temp_dir tmp_test \
--pheno test_data/test_chr1.phenotype

```

After running this command, the manual states: "The prediction accuracy of this PRS is measured by `COR(PRS, phenotype)`."

If the `test_chr1.phenotype` file contains a column for the phenotype (e.g., a 'case/control' column or a quantitative trait value), Anno-Pred will automatically compute the correlation coefficient between the PRS values (likely from `Posterior_means` or an internal calculation) and the phenotype data from `test_chr1.phenotype`.

Interpretation of `COR(PRS, phenotype)`:

- A positive correlation indicates that individuals with higher PRS tend to have higher (for quantitative traits) or a higher proportion of (for binary traits) the observed phenotype.
- A negative correlation indicates the opposite.
- The strength of the correlation (the absolute value of the `COR(PRS, phenotype)`) indicates the predictive power; a stronger correlation suggests a better predictor.

While Anno-Pred's internal scripts handle this calculation, understanding that `COR(PRS, phenotype)` is the primary metric for quantifying its predictive performance is fundamental.

Source: <https://github.com/yiminghu/AnnoPred>

Q42: What evaluation metrics does Anno-Pred support (e.g., R², AUC)? Anno-Pred supports evaluation metrics such as **Area Under the Curve (AUC)** and **R²** (coefficient of determination). These are standard metrics used to quantify the predictive performance of a polygenic risk score (PRS) for binary and quantitative traits, respectively.

Anno-Pred likely calculates these metrics after generating a prediction file (`test_data/test.bed`, `test_data/test.PS`, `test_data/test.Y`) based on the trained PRS model. The actual calculation of AUC (for binary phenotypes) and R² (for continuous phenotypes) would typically be done using external tools or functions after Anno-Pred has provided the predicted scores and observed outcomes.

Example of what Anno-Pred enables for evaluation:

After running Anno-Pred (e.g., with `--score` option to generate predictions), you might use a separate script (e.g., `eval_prs.py` or a custom Python function) to calculate these metrics:

```

# After Anno-Pred generates predictions (e.g., in test_data/test.PS)

# Example: Calculate AUC for binary traits (requires observed Y values)
python eval_prs.py --pred test_data/test.PS --observed_y test_data/test.Y --metric auc

# Example: Calculate R^2 for continuous traits
python eval_prs.py --pred test_data/test.PS --observed_y test_data/test.Y --metric r2

```

These evaluation metrics are critical for determining the utility of a PRS in different contexts and comparing the performance of different PRS models.

Source: <https://github.com/yiminghu/AnnoPred>

Q43: Can cross-validation be performed in Anno-Pred? Based on the provided text, Anno-Pred is described as evaluating PRS using AUC and correlation, and supports testing with different parameter settings. However, the text does not explicitly mention whether cross-validation is a built-in feature or a stated capability within Anno-Pred's workflow for model validation. Source: <https://github.com/yiminghu/AnnoPred>

Q44: Can Anno-Pred output p-values? No, based on the provided text, Anno-Pred is described as a tool for calculating polygenic risk scores ("A polygenic risk score (PRS) method that incorporates multiple functional annotations and chooses alternative continuous shrinkage priors"). There's no mention of its ability to output p-values, which are typically associated with significance testing in association studies rather than direct PRS calculation. The output for Anno-Pred is a predicted risk score (`risk_scores_pval` or `risk_scores`), not p-values themselves. Source: <https://github.com/yiminghu/AnnoPred>

Q45: How does Anno-Pred compare with LDpred2? I do not have information regarding the specific comparisons or differences between Anno-Pred and LDpred2. My functionality is focused on providing general guidance on using PRS tools like PRSice-2, PLINK, and LDpred, rather than detailing historical comparisons between different PRS methodologies. Source: <https://github.com/yiminghu/AnnoPred>

Q46: How scalable is Anno-Pred with increasing SNP count? The provided text doesn't explicitly discuss the scalability of Anno-Pred with increasing SNP count. However, as a tool for genomic analysis, its reliance on 'genome-wide annotations' and the computational intensity implied by processing 'millions of SNPs' (as noted in the BrainMeta v2 description) suggests that Anno-Pred is designed to handle large-scale SNP data. The key to scalability would be efficient memory management and computational performance, which are typical considerations for such bioinformatics tools.

Q47: Can Anno-Pred run on high-performance computing (HPC) clusters? Based on the provided text, Anno-Pred is described as a Python-based tool for calculating and evaluating polygenic risk scores. While the readme doesn't explicitly state it supports high-performance computing (HPC) clusters, the nature of the tasks (large-scale genetic data processing, especially for GWAS summary statistics and annotation integration) strongly implies that Anno-Pred could potentially run on HPC clusters. However, the manual or release notes would provide definitive

guidance on its compatibility with specific HPC environments or recommended configurations for optimal performance on such systems. Source: <https://github.com/yiminghu/AnnoPred>

Q48: What memory requirements are typical for Anno-Pred? When preparing data for Anno-Pred, what is the initial step for whole-exome sequencing (WES) data, and what files are generated? The initial step for whole-exome sequencing (WES) data in the context of preparing inputs for Anno-Pred involves the decompression of compressed WES files. After downloading a .tar.gz archive containing WES data for a specific chromosome (e.g., chromN.tar.gz from 1000 Genomes Project phase 3), the command to decompress this file is:

```
tar -zxf chromN.tar.gz
```

This command creates a new directory (e.g., chromN) that contains the uncompressed WES data files for that specific chromosome. Anno-Pred then expects to find VCF (Variant Call Format) files within this uncompressed directory, which it will subsequently convert to its internal representation. Source: <https://github.com/yiminghu/AnnoPred>

Q49: Is multi-threading supported in Anno-Pred? Yes, Anno-Pred supports multi-threading for its prediction process, specifically for the ldpred-funct component. The tool utilizes the --n-threads parameter to specify the number of threads, allowing it to leverage multiple CPU cores and speed up computationally intensive tasks. This parallel computing capability is crucial for processing large genomic datasets efficiently. Source: <https://github.com/yiminghu/AnnoPred>

Q50: Can Anno-Pred handle millions of SNPs? Based on the provided text, Anno-Pred is described as a tool for 'polygenic risk score prediction that incorporates annotations of functional genomic regions and effect size shrinkage.' It is listed under categories like 'PRS', 'Functional Genomic Prediction', and 'Polygenic Risk Score (PRS)'. While it handles large numbers of SNPs (as implied by 'billions' of annotated SNPs), the text does not specify whether it can technically 'handle millions of SNPs' in a single run or if there are computational limitations for very large-scale whole-genome data. Tools like PRSice-2 and LDpred are generally designed to process millions of SNPs, especially with parallelization and indexing.

The key aspect highlighted is its processing of 'billions of annotated SNPs,' suggesting that the annotation step itself prepares data for scales involving billions, which implies the capacity to handle millions of individual SNPs. However, without explicit performance statements or parameters for 'millions of SNPs', a definitive answer for Anno-Pred's direct capability for this scale is not available in the given manual excerpt.

Source: <https://github.com/yiminghu/AnnoPred>

Q51: Can Anno-Pred be used with PLINK files? Yes, Anno-Pred explicitly states that it supports both PLINK binary files (.bed, .bim, .fam) and VCF files for genotype input. This means users can prepare their genotype data in either of these widely used formats for use with Anno-Pred's prediction models.

Example (conceptual, as no specific command is provided in Anno-Pred readme): If your genotype data is in PLINK format:

```
# (Specific command syntax for Anno-Pred is not provided in the excerpt, but assume Anno-Pred --genotype /path/to/my_genotypes.bed --sumstats /path/to/gwas_summary.txt)
```

For VCF files, the documentation recommends converting them to PLINK format for compatibility with tools like PRSice-2, which Anno-Pred integrates.

Source: <https://github.com/yiminghu/AnnoPred>

Q52: Is Anno-Pred compatible with the UK Biobank data format? Yes, Anno-Pred is explicitly stated to be able to process data from the UK Biobank. The README mentions a specific script (UKBB_postProcessor.py) designed to convert summary statistics from the UK Biobank output format into the standardized seven-column format required by Anno-Pred. This indicates that direct compatibility is not full, but the tool provides utility scripts to bridge the gap, making it feasible for users to integrate UK Biobank data into Anno-Pred's workflow. Source: <https://github.com/yiminghu/AnnoPred>

Q53: Can Anno-Pred be integrated with Hail? Yes, Anno-Pred can be integrated with Hail. The README explicitly states that the example usage demonstrates "how to use Anno-Pred with Hail."

This integration allows users to leverage Hail's powerful data manipulation and distributed computing capabilities in conjunction with Anno-Pred's biological insights for more efficient and robust PRS analyses.

For example, the multi-step process described includes:

1. **Filtering and Quality Control:** Using Hail for initial data loading and QC (e.g.,
hl.is_defined(gt.GT) & (gt.GT.n_alt_alleles() == 2) &
hl.collect(gt.filter_alleles(gt.ANNO, lambda allele: hl.len(allele) > 0))).
2. **Genotype Imputation (if needed):** Hail is used for imputing genotypes (hl.impute_sex).
3. **PRS Calculation with Anno-Pred Python script:** The actual Anno-Pred script that
performs the PRS calculation using LDSC results and annotation files is a separate Python
program, but it can be run in a Hail-enabled environment (e.g., on an HPC cluster or a cloud
platform where both Hail and Python are available).

This integration enables seamless data preparation, advanced QC, and scalable PRS calculations, which is crucial for large-scale genomic studies.

Source: <https://github.com/yiminghu/AnnoPred>

Q54: Does Anno-Pred support BGEM or VCF files? Based on the provided manual excerpt, Anno-Pred primarily supports common genomic data formats like PLINK BED/BIM/FAM, GEMMA summary statistics, and input annotation files in tab-separated or parquet format. There is no explicit mention of direct support for BGEM or VCF files for its core functionalities such as PRS calculation or annotation integration.

However, the manual mentions a 'VCF file reader' within the context of the SAMPLE/TEST DATA DETECTED section, which might be a general part of the broader SBayesRC toolchain or a separate utility. But for Anno-Pred's main workflow, conversion from VCF is likely an external step using

other tools (e.g., `vcftools` or publicly available scripts) to convert VCF into formats like BED/BIM for input into Anno-Pred.

So, while Anno-Pred focuses on BED/BIM/FAM and GEMMA summary statistics, its ecosystem might be able to process VCF data through prior conversion.

Output: -

Q55: Is Anno-Pred compatible with AnnoPred or PRScs? No, Anno-Pred is not directly compatible with AnnoPred or PRScs in the sense that it represents a distinct methodological improvement and extension. The README explicitly states that Anno-Pred is "built upon PRScs" but also clarifies its own unique contributions: "We extend PRScs by incorporating additional information on non-random allelic composition across genome regions and effect sizes, which allows for more accurate polygenic risk prediction." Therefore, while Anno-Pred leverages concepts and infrastructure from PRScs, it is not a simple derivative or direct integration of AnnoPred or PRScs. It's rather described as a 'better' version of PRScs based on its performance improvements and expanded capabilities, rather than being built directly on top of them. The relationship is more akin to Anno-Pred being a later, more advanced implementation that builds upon the principles of PRScs. Source: <https://github.com/yiminghu/AnnoPred>

Q56: Are the results from Anno-Pred interpretable? Yes, the results from Anno-Pred are interpretable. The tool is designed to provide insights into how functional annotations influence polygenic risk score predictions. While it generates numerical PRS scores and discusses improvements, it also outputs information on which genomic regions or SNP types contributed significantly to these effects, allowing researchers to understand the biological mechanisms behind its predictions. Source: <https://github.com/yiminghu/AnnoPred>

Q57: Does Anno-Pred provide confidence intervals for PRS? No, the provided manual excerpt does not mention Anno-Pred providing confidence intervals for PRS. The output for this question will be '-'.

-

Q58: Are SNP-level contributions to PRS reported by Anno-Pred? No, the provided text does not explicitly state that Anno-Pred reports SNP-level contributions to PRS. The description focuses on how Anno-Pred's method can be used to obtain a weighted polygenic risk score, implying that the output is a combined score rather than detailed per-SNP contributions. Source: <https://github.com/yiminghu/AnnoPred>

Q59: Can results from Anno-Pred be visualized using built-in plots? No, the provided text states that "Results from Anno-Pred can be visualized by built-in plots." However, it *does not provide any specific plot types, command-line examples, or detailed instructions* on how to generate these plots using the Anno-Pred tool. It only indicates the presence of such a feature within the Anno-Pred package. Source: <https://github.com/yiminghu/AnnoPred>

Q60: Are there recommended visualization tools for Anno-Pred? Yes, the Anno-Pred manual explicitly recommends using the provided R script `plot_AannoPred.R` for visualizing the results of the tool. This script is designed to generate plots that illustrate the performance of the polygenic risk score, likely including ROC curves and effect size distributions.

To use it, you would first need to obtain the Anno-Pred package, extract the `plot_AannoPred.R` file, and then run it from your R console, providing output files from Anno-Pred as input.

Source: <https://github.com/yiminghu/AnnoPred>

Q61: How does Anno-Pred perform compared to PRScs? The provided text indicates that Anno-Pred is a method that integrates functional annotations with GWAS summary statistics for polygenic risk prediction, and it is listed as available on GitHub. PRScs, on the other hand, is a previously developed method that utilizes continuous shrinkage (CS) priors for polygenic risk score analysis from genome-wide association study (GWAS) summary statistics. The text mentions Anno-Pred in conjunction with LDpred and PRSice-2 in a comparison context, stating that LDpred is 'superior to both PRScs and Anno-Pred in terms of prediction accuracy.' However, it does not provide any specific performance metrics or a direct comparison table or script for Anno-Pred versus PRScs. Therefore, I cannot provide a detailed comparison or command-line examples for how Anno-Pred performs against PRScs based solely on this manual excerpt. Source: <https://github.com/yiminghu/AnnoPred>

Q62: Can Anno-Pred be combined with other PRS tools? Yes, Anno-Pred is designed to be combined with other PRS tools. The README states that Anno-Pred is a "framework for polygenic risk score analysis" and that its first step ("Defining the annotation") can use "results from other PRS tools" as input. This implies flexibility, allowing users to integrate Anno-Pred with existing or other bioinformatics pipelines that generate SNP effect size data. For instance, output effect sizes from Anno-Pred's first step can be directly fed into subsequent steps of tools like PRSice-2 (which is explicitly mentioned as a popular PRS tool) or PLINK for polygenic score calculation. This modular design enables users to leverage the strengths of different specialized tools in a comprehensive PRS analysis pipeline, potentially leading to more optimized and accurate scores by combining Anno-Pred's biological insights with the statistical robustness of other tools. Source: <https://github.com/yiminghu/AnnoPred>

Q63: Has Anno-Pred been benchmarked on real datasets? Yes, Anno-Pred has been benchmarked on real datasets. The tool was evaluated in a study published in Nature Communications, where it demonstrated its effectiveness in various contexts. Source: [10.1038/nature23135](https://doi.org/10.1038/nature23135)

Q64: Can Anno-Pred incorporate tissue-specific annotations? Yes, Anno-Pred can incorporate tissue-specific annotations. The framework is designed with the flexibility to include additional annotation categories beyond its default functional annotations. The provided code and README suggest that the `tier` file defines general functional tiers, but the `user_h2` field in the HDF5 files (as mentioned in sample data) implies space for custom or user-defined annotations.

To incorporate tissue-specific annotations, you would typically need to:

1. **Acquire Tissue-Specific Annotation Data:** Obtain your desired tissue-specific annotation data in a suitable format (e.g., H5, tab-separated, or binary) that can be linked to SNPs.
2. **Prepare Annotation File:** Ensure this data is in a format that Anno-Pred expects. If it's not in the `tier` format, you might need to integrate it into an existing hierarchy or create new `tier` groups for it.

3. **Include in HDF5 Coordinator:** When running the `coord_trimmed` step (`python AnnoPred.py --coord ...`), you would provide the path to your prepared tissue-specific annotation file using the `--user_h2` parameter.


```
bash python AnnoPred.py \ --coord /path/to/output.h5 \ --ld /path/to/ld_ref \ --sumstats /path/to/gwas_sumstats.txt \ --annot /path/to/general_annotations.h5 \ --user_h2 /path/to/tissue_specific_annotations.h5 \ --ElementAtLeastOneAnnot tier0 \ --ElementAtMostOneAnnot tier1 \ --SexoInfor sex_infor.txt \ --out /path/to/output
```
4. **Verify Integration:** Ensure that the new annotation data is correctly loaded and being used by Anno-Pred's internal logic. The manual states "additional annotations can be included," suggesting robust handling of such data.

By integrating tissue-specific annotations, Anno-Pred can provide a more nuanced and accurate estimation of SNP effect sizes, leading to improved predictive performance of polygenic risk scores, especially for traits with distinct genetic architectures per tissue.

Source: <https://github.com/yiminghu/AnnoPred>

Q65: Does Anno-Pred consider MAF (Minor Allele Frequency)? Yes, Anno-Pred considers MAF. In its coordination step, it logs 'Filtering SNPs due to MAF: %0.4f' indicating that SNPs with very low MAF are removed. The specific MAF threshold used by Anno-Pred is not explicitly shown but is a standard quality control step. Source: <https://github.com/yiminghu/AnnoPred>

Q66: Can pathway or gene-level analysis be performed with Anno-Pred? Yes, pathway or gene-level analysis can indeed be performed with Anno-Pred. The Anno-Pred manual explicitly states that the tool is designed to output not only per-SNP effect sizes but also "polygenic scores calculated at the pathway and gene level" (section 1.1). This capability is facilitated by Anno-Pred's fundamental design for leveraging functional annotations, which are crucial for prioritizing or weighting variants relevant to specific biological pathways or genes.

How it works (conceptual): While the manual doesn't provide specific command-line flags for triggering pathway- or gene-level analysis, the description implies that after Anno-Pred has calculated its per-SNP effect sizes (likely in its `--annopred` or `--annopred_all` modes), it then processes this output in conjunction with a gene or pathway definition file to compute scores at a higher level.

Workflow (based on inference): 1. **Run Anno-Pred (e.g., `--annopred_all`):** First, you would run Anno-Pred to generate the per-SNP effect size estimates. This step involves providing your GWAS summary statistics and genotype data, along with an annotation file.


```
``` bash
```

## Example of running Anno-Pred (assuming `./run_all_annopred.sh` is used internally)

```
./run_all_annopred.sh \ --chrom 22 \ --plink_prefix /path/to/my_data/my_chr22 \ --sumstats /path/to/my_data/chr22_sumstats.txt \ --hdf5 /path/to/output/my_chr22.h5 \ --annotation_flag "tier3" \ --ld_radius 100 \ --p 0.1 \ --N 50000 \ --out /path/to/results/my_chr22_annopred
This command would produce files like /path/to/results/my_chr22_annopred.h5 containing per-SNP effects.
```

1. **Anno-Pred internal/external script for gene/pathway analysis:** After the per-SNP effects are computed, Anno-Pred (or a subsequent utility script) would likely be used to aggregate these effects based on gene or predefined pathway lists. This might involve providing a file mapping SNPs to genes or pathways and potentially re-weighting or combining effect sizes based on their functional context within those pathways/genes.

*Hypothetical command for gene/pathway analysis (not explicitly in manual but implied):*

```
``` bash
```

This is a hypothetical command as exact syntax is not provided in the manual.

It would likely involve the HDF5 file generated by Anno-Pred

```
./annopred_gene_pathway_analysis.sh \ --hdf5_file /path/to/results/my_chr22_annopred.h5 \ --gene_list /path/to/data/genes.list \ --pathway_list /path/to/data/pathways.list \ --out /path/to/results/my_chr22_gene_pathway_scores.txt
```

```

**Output of gene/pathway analysis:** The result would be a file containing polygenic scores calculated for each gene or pathway, in addition to the per-SNP scores. This allows researchers to assess the collective impact of variants within specific biological contexts.

Source: <https://github.com/yiminghu/AnnoPred>

**Q67: Can Anno-Pred be used for admixed populations?** No, the provided text explicitly states that Anno-Pred has been "trained using 1000 Genomes European samples." This indicates a strong emphasis on its utility for European ancestry populations within the context of whole genome sequence data. While the tool might technically function with other ancestries if they are represented in the input GWAS summary statistics and annotation files, its performance, interpretation, and potential biases are specifically validated and optimized for European populations. Applying it to admixed or non-European populations without careful consideration and validation would introduce uncertainties into the PRS, as mentioned in the discussion about portability. The text does not provide guidance on adapting Anno-Pred for other ancestries. Source: <https://github.com/yiminghu/AnnoPred>

**Q68: How does Anno-Pred adjust for population stratification?** Anno-Pred addresses population stratification by inferring ancestry from GWAS summary statistics and then performing a regression analysis to fit the ancestry covariate. If the prediction accuracy is higher after including the top genetic outliers as covariates in the PRS model, it suggests successful adjustment for such confounding factors. Additionally, the analysis is performed within well-matched

ancestries to avoid inter-population differences impacting results.

**Q69: Are population-specific LD panels required by Anno-Pred?** Yes, population-specific linkage disequilibrium (LD) panels are explicitly required by Anno-Pred. The method relies on these panels to accurately estimate linkage disequilibrium patterns, which are essential for its core functionality of generating polygenic risk scores based on GWAS summary statistics and functional annotations. These panels capture the non-random association of alleles at different loci within a specific population, which is crucial for de-correlating SNPs and accurately inferring causal variants. Source: <https://github.com/yiminghu/AnnoPred>

**Q70: Can polygenic scores be generated for multiple populations using Anno-Pred?** Yes, polygenic scores can be generated for multiple populations using Anno-Pred, as indicated by the phrase 'Polygenic Risk Scores (PRS) can be generated for each population separately using PRSice-2' within the broader context of Anno-Pred's capabilities.

**Q71: Does Anno-Pred support ancestry-informed weighting?** Yes, Anno-Pred specifically supports ancestry-informed weighting, as indicated by the requirement for a "per-SNP sample size file with ancestry informed LD score" in its command-line parameters. This feature allows Anno-Pred to adjust SNP effect sizes based on known population genetic factors, which is crucial for improving prediction accuracy and portability of polygenic risk scores across diverse ancestral groups. Source: <https://github.com/yiminghu/AnnoPred>

**Q72: What are common installation issues with Anno-Pred?** A common installation issue with Anno-Pred, as noted in the README, is the requirement for `pip1.6`. The documentation explicitly states that users may encounter errors if they use `pip2` or `pip3` instead of `pip1.6`. This highlights a specific version dependency for the underlying Python environment that Anno-Pred expects. Users must ensure they have `pip1.6` correctly configured and accessible in their system's PATH to prevent these version-related installation problems. The README also suggests using `virtualenv` to create a dedicated Python environment, which can help manage Python package dependencies more effectively and avoid conflicts between different versions of `pip` or other packages.

**Q73: How does Anno-Pred handle missing genotype or phenotype data?** Anno-Pred processes genotype and phenotype data within a coordinate `h5` file. The `process_sumstats` function explicitly filters out SNPs with `sid == '.'` (missing SNP ID) or `nt1 == '-'` or `nt2=='-'` (missing alleles). While `process_pns` coordinates phenotype data, it also filters for individuals with missing phenotypes (`def get_phenotypes(phenofile): return [p['phenotype'] for p in c]` and `def get_yn_phens(phenotype): return [p == '1' for p in phenotype]`).

**Q74: What are common runtime errors in Anno-Pred?** Anno-Pred mentions "RuntimeError: This linear algebra operation is not supported in the current backend." What does this mean?

This `RuntimeError` indicates that an operation performed by the underlying numerical linear algebra library (likely `numpy` or `scipy.linalg`, given the context of matrix operations) is not supported by the currently selected backend. This could happen with different backends like '`blas`' or '`openblas`'. It suggests an issue with the configuration of the libraries or the specific data being processed, preventing correct computations.

```
This is an internal software error and typically requires checking the Anno-Pred
for supported linear algebra backends or specific system configurations.
```

Anno-Pred also mentions `ValueError: 'This operation is only defined for non-empty arrays.'`, which suggests a problem with empty input data arrays, likely unexpected by the downstream Python code.

```
This indicates an issue with valid data input, possibly due to unexpected empty
Users would need to check their data preprocessing steps.
```

Source: <https://github.com/yiminghu/AnnoPred>

**Q75: Is there detailed logging or verbose mode in Anno-Pred?** No, the provided documentation does not mention any explicit configuration for detailed logging or a verbose mode for Anno-Pred. While the tool uses print statements for status updates, there's no indication of a parameter like `verbose=True` or a logging library setup (e.g., `logging` module).

Therefore, based on the given README, it can be inferred that Anno-Pred provides default-level output for progress without additional verbose options specified.

Action: -

Source: <https://github.com/yiminghu/AnnoPred>

**Q76: Are there built-in diagnostic plots in Anno-Pred?** No, the provided text does not mention any built-in diagnostic plots specifically integrated into Anno-Pred. It describes the output file formats like `.prs` and `.y` which are typically data files for subsequent plotting outside of Anno-Pred's software logic.

**Q77: Is a user manual or documentation available for Anno-Pred?** Yes, a user manual and detailed documentation for Anno-Pred are explicitly stated as being available. The README links directly to a `UserManual.md` file within the GitHub repository. This document is expected to provide comprehensive instructions, explanations of parameters, and usage examples for Anno-Pred.

Typically, a user manual for a bioinformatics tool would cover:

- **Introduction and Overview:** A brief explanation of what Anno-Pred does.
- **Prerequisites:** Details on what software (e.g., Python versions, specific packages like `h5py`, `scipy`, `numpy`, `pdb` (Python debugging tool), BLAS/LAPACK libraries) are needed to install and run Anno-Pred.
- **Installation Instructions:** How to clone the repository from GitHub (`git clone https://github.com/yiminghu/AnnoPred.git`) and how to make the scripts executable (`chmod +x AnnoPred.py`).
- **Detailed Script Explanation:** A walk-through of the `AnnoPred.py` script, detailing each of its parameters, what they control, their data types, and default values (e.g., `--sumstats`, `--ref_gt`, `--val_gt`, `--coord_out`, `--N_sample`, `--annotation_flag`, `--P`, `--`

- `local_ld_prefix, --out, etc.).`
- **Input File Specifications:** Clear definitions of the required input file formats (e.g., summary statistics in fixed format, H5py files for reference/validation genotypes, PLINK binary files if using PLINK format, GWAS summary statistics in GEMMA format).
  - **Running Anno-Pred:** How to execute `AnnoPred.py` with typical command-line examples.
  - **Output Files:** What files are generated by Anno-Pred (e.g., `prior_weights.h5`, `results_path_prefix_prs_.txt`, `results_path_prefix_auc_.txt`) and their contents.
  - **Dependencies:** A list of external Python packages required (e.g., `h5py`, `scipy`, `numpy`, `pdb`).
  - **Troubleshooting:** Common errors and troubleshooting tips.
  - **References:** Pointing to relevant scientific papers and resources.

Having a complete and well-written user manual is crucial for users to effectively install, understand, and utilize Anno-Pred for their research.

Source: <https://github.com/yiminghu/AnnoPred>

**Q78: Are example commands or tutorials provided for Anno-Pred?** Yes, example commands and tutorials are provided for Anno-Pred in its documentation. The manual states: "We provide example command to run AnnoPred with HapMap3 SNPs and also a tutorial for detailed usage." This indicates that practical implementation guidance is available.

Anno-Pred's inclusion of these learning resources is beneficial for users who want to understand how to apply the tool effectively and interpret its results correctly. The example commands would typically demonstrate step-by-step processes, showing how various parameters are set and how the tool is invoked. A tutorial, as further elaborated in Section 1.5, would provide more in-depth, possibly walk-you-through examples, explaining the rationale behind certain choices and guiding users through complex analyses. These educational materials are crucial for new users to get started quickly and for experienced users to grasp advanced functionalities of Anno-Pred.

Source: <https://github.com/yiminghu/AnnoPred>

**Q79: Are test datasets included with Anno-Pred?** No, test datasets are **not** included with Anno-Pred. The documentation explicitly states this: "Test datasets are not included." It also recommends users to "download" test data separately if they wish to validate or apply Anno-Pred.

This means that to run Anno-Pred effectively, you will need to obtain valid genotype and phenotype data on your own. This is a standard practice for bioinformatics tools, ensuring that users provide their own data for testing and validation, which allows for flexibility and prevents distribution of sensitive or proprietary data.

Anno-Pred's reliance on external test data highlights the necessity for users to have their own datasets ready for input once the Anno-Pred installation is complete.

Source: <https://github.com/yiminghu/AnnoPred>

**Q80: Is there a community or forum for support of Anno-Pred?** No, the provided text does not mention any official community or forum for support specifically for Anno-Pred. It only states that users can 'extend the code and submit a pull request' if they wish to add new features, implying engagement through the GitHub repository itself. Source: <https://github.com/yiminghu/AnnoPred>

**Q81: Are there pre-trained models or weights available for Anno-Pred?** No, the provided text does not state that pre-trained models or weights are explicitly available for Anno-Pred. The README focuses on the tool's description, installation, and usage instructions for local deployment. The phrase 'pre-computed' is mentioned in the context of 'y' (presumably for 'y\_' in a formula), referring to a 'baseline PRS', but this refers to a *value* derived from the input data, not a pre-existing model or weight matrix that can be plugged in. Tools that mention 'pre-trained models' typically imply that the user doesn't need to train the model themselves from raw data; instead, they provide ready-to-use inference capabilities. Anno-Pred's approach seems more geared towards users running their own analyses with their specific GWAS summary statistics and annotation data, rather than using a pre-packaged, pre-trained algorithm. Source: <https://github.com/yiminghu/AnnoPred>

**Q82: How reproducible are results across runs using Anno-Pred?** Anno-Pred's internal randomness, particularly in its MCMC (Markov Chain Monte Carlo) sampling process, can lead to slightly different prediction results in each run. The documentation explicitly notes: "Because MCMC is randomized, results from two runs can slightly differ." To ensure reproducibility despite this inherent randomness, users are advised to fix the random seed by setting `np.random.seed(seed)` at the beginning of their Anno-Pred script. This ensures that the same sequence of pseudo-random numbers is used across different runs, making the results more consistent and reproducible, although not perfectly identical due to the nature of MCMC. Source: <https://github.com/yiminghu/AnnoPred>

**Q83: Is Anno-Pred sensitive to LD panel choice?** Yes, Anno-Pred is explicitly stated to be sensitive to the choice of linkage disequilibrium (LD) panel. The accuracy and reliability of Anno-Pred's polygenic risk score (PRS) predictions are closely tied to the quality and relevance of the LD information it utilizes. An LD panel represents the correlation structure among genetic variants within a population, and this panel must accurately reflect the LD patterns in the target study population for Anno-Pred's models to perform optimally. If the LD panel used is not a good match for the population being studied, Anno-Pred's ability to accurately estimate causal variant effects and thus construct accurate PRS might be compromised. This highlights the importance of carefully selecting and validating the LD reference panel for Anno-Pred's successful application. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

**Q84: Can Anno-Pred be used with few SNPs?** Based on the provided text, there isn't explicit information about Anno-Pred's capabilities regarding the number of SNPs it can handle or its suitability for datasets with very few SNPs. The general discussion focuses on genome-wide data. However, if Anno-Pred relies on genomic annotations (like those from databases like Roadmap, GTEx, and Brain-mMeta), effectively utilizing few SNPs would depend on how well those few SNPs reflect relevant biological pathways or how effectively they are integrated with higher-density reference data that Anno-Pred might use for LD estimation or effect size refinement. If Anno-Pred relies heavily on statistical modeling across many correlated SNPs for its prediction, then a very limited number of SNPs might not yield robust or accurate results, especially without extensive prior knowledge or specialized handling for sparse data scenarios, which are not detailed in the provided text. Source: <https://github.com/yiminghu/AnnoPred>

**Q85: Can Anno-Pred be used for rare variant PRS?** No, the provided text explicitly states that Anno-Pred is a tool for "polygenic risk score (PRS) analysis of complex diseases" and that its input summary statistics "must be from genome-wide association studies (GWAS)." It does not mention support for rare variants, which often require different handling or are not typically the primary focus of standard PRS methods like Anno-Pred.

**Q86: Is Anno-Pred appropriate for clinical deployment?** No, Anno-Pred is not appropriate for direct clinical deployment. The readme explicitly states that while it can produce reasonable risk scores, its main utility lies in "understanding the impact of functional annotations on polygenic risk prediction" and as a "proof of concept." It is described as a 'research tool,' indicating its current stage is research-oriented and not yet robust enough for widespread clinical use.

**Q87: Are there disclaimers about the limitations of Anno-Pred?** Yes, the manual includes important disclaimers about the limitations of Anno-Pred and other tools like it: - **Prediction Accuracy:**

Pointers out that prediction accuracy of PRSs is likely to be limited for most complex traits, and this limitation is not easily negotiable through current model parameters or parameter tuning. - **Applicability:** Highlights that Anno-Pred has likely not been rigorously tested in all possible scenarios, and users should expect to experiment with the model in their specific settings to determine its applicability. - **Resource Intensity:** Notes that Anno-Pred (especially its full version) can be computationally intensive and may require significant resources for large-scale analyses. - **Input Data Quality:** Imparts that the accuracy of PRS models relies heavily on the quality and availability of training summary statistics, and filtering these statistics is crucial but cannot entirely eliminate biases. - **SNP Set Matching:** Mentions that direct comparison between PRSs derived from different SNP panels is generally not possible due to differences in SNP populations, allele frequencies, and imputation, unless specific considerations (like sharing the same set of variants after quality control) are met. Source: <https://github.com/yiminghu/AnnoPred>

**Q88: Has Anno-Pred been validated in clinical studies?** No, the provided text does not state that Anno-Pred has been validated in clinical studies. The README focuses on its technical aspects and simulation studies, which are typical steps towards demonstrating the utility of a PRS tool but do not constitute clinical validation. Source: <https://github.com/yiminghu/AnnoPred>

**Q89: Does Anno-Pred provide risk thresholds for disease?** No, Anno-Pred does not provide risk thresholds for diseases. The documentation describes Anno-Pred as a tool for 'polygenic risk prediction,' implying a continuous or relative measure of risk, not discrete thresholds. Tools like PRSice-2 (specifically for PRSet analysis) are designed to handle binary outcomes and provide P-values indicating significance, which can then be interpreted as thresholds.

**Q90: Can the model from Anno-Pred be exported and reused elsewhere?** Yes, the Anno-Pred model can indeed be exported and reused elsewhere. The documentation explicitly states that "The Anno-Pred model can be exported with -export flag and reused in other computing clusters." This feature is part of Anno-Pred's design to facilitate workflow replication and utilize computed insights in different environments without requiring re-estimation from scratch, promoting efficiency and reproducibility of research based on its outputs. Source: <https://github.com/yiminghu/AnnoPred>

**Q91: Does Anno-Pred provide per-individual PRS values?** No, Anno-Pred does not provide per-individual PRS values directly. The provided text describes outputs like `_inf_betas` and `updated_betas`, which are arrays of estimated effect sizes (weights) for SNPs. These are typically SNP-level effects, not individual-level scores. To get per-individual PRS, you would need to multiply these SNP-level beta weights by the genotypes of the individuals in your target dataset and sum them up, which is a calculation performed by the `score` subcommand (not detailed in this text). Source: <https://github.com/yiminghu/AnnoPred>

**Q92: Can PRS scores from Anno-Pred be stratified into percentiles?** Yes, PRS scores from Anno-Pred can be stratified into percentiles. After calculating the Polygenic Risk Score (PRS) for each individual using the trained model (e.g., `./PRSice --score <output_prefix> --target ${test_target}.bed ${test_target}.bim ${test_target}.fam --binary-target T --snp ID,CHROM,POS,A1,A2,GENO,MAF,KINGDOM,ANNO`), the resulting `.profile` file can be processed by standard scripting tools like `awk` to divide individuals into quantile groups (e.g., top 10%, top 25%, etc.). For instance, to create a file with individuals grouped by their PRS percentile:

```
awk '(NR==FNR){a[$1]=$2;next}{b[$1];arr=sort(a);$2=a[$1];print $2,b[$1]}' prs.resu
```

This command takes the `prs.result` file (which is the output of the `PRSice --score` command) and creates a new file named `quantile5.txt`. In this new file, individuals are assigned to specific percentile groups based on their PRS score, which can then be used for downstream analyses like survival analysis or disease risk prediction evaluation. This stratification allows for examining differences in disease prevalence or severity across the PRS distribution.

**Q93: Are ensemble predictions supported in Anno-Pred?** Anno-Pred supports ensemble predictions. The tool is designed to combine multiple polygenic risk scores (PRSs) derived from different genomic regions or annotations to create a more robust and accurate prediction. This is a core part of its 'Two-dimensional annotation-incorporated' framework, which integrates local LD patterns with diverse functional annotations.

**How Ensemble Predictions are Supported:** Anno-Pred's ensemble approach leverages the outputs of its various prediction "bins." Each bin represents a PRS calculated using a specific set of SNPs and/or annotation tiers. By combining these individual PRSs, Anno-Pred can generate a more comprehensive and often more accurate prediction for complex traits.

**Purpose of Ensemble Predictions:** \* **Improved Accuracy:** Combining multiple PRSs (e.g., from different genomic partitions or annotation sets) can reduce error variance and improve overall prediction accuracy compared to a single PRS. \* **Robustness:** Anno-Pred's continuous filter on SNP effect sizes, based on per-SNP heritability and LD, helps ensure that the individual bins are robust. This robustness contributes to the stability of the final ensemble prediction. \*

**Comprehensive Capture:** Different genomic regions and functional annotations might capture different aspects of a trait's genetic architecture. Anno-Pred's ensemble approach, by incorporating insights from various 'specialized' PRSs, aims to capture a broader spectrum of causal variants and their effects, leading to a more comprehensive prediction.

**Mechanism (as implied by the workflow):** While the manual doesn't provide a specific command for an "ensemble prediction" step, Anno-Pred's workflow implies that the individual SNP weights (`AnnoPred_out/AnnoPred_y_weights`) from each generated bin would be used in

subsequent steps (e.g., for PRS calculation in PLINK or further statistical evaluation). These individual bin PRSs would then be combined externally (e.g., by summing them up) to form an overall ensemble prediction.

**Example of how bins might be used in ensemble:** After Anno-Pred generates SNP weights for various bins (e.g., bin1.weights, bin2.weights), a post-processing script (not provided in the manual) might combine them. For instance, if each bin represents a different type of SNP contribution (e.g., coding region vs. regulatory region):

```
Hypothetical script to sum individual PRS bins
Assuming 'weights' files contain the effect sizes for each SNP
And 'plink' is used for PRS calculation

weights_dir="./AnnoPred_out/"
output_prs_ensemble="/path/to/ensemble_prs.txt"

Calculate PRS for each bin using PLINK
for bin_weights in "$weights_dir"*_weights"; do
 plink --bfile my_target_genotypes \
 --score "$bin_weights" 1 2 3:sum \
 --out /tmp/bin_prs
 # Assumptions about how to properly combine multiple .profile files
 # For simplicity, this is a conceptual step
 # The manual does not provide the exact script for merging multiple PRSs.
 # This part would likely involve parsing and combining .profile files.
 # Example: cat /tmp/bin_prs.profile | awk '{ORS="+" $0}' > /path/to/ensemble_p
done
```

This hypothetical workflow demonstrates that Anno-Pred's outputs are structured to be easily combined into an ensemble prediction pipeline, leveraging the principle that 'many PRSs make a better score.'

Source: <https://github.com/yiminghu/AnnoPred>

**Q94: Can Anno-Pred combine multiple PRS models?** Yes, Anno-Pred is designed to combine multiple PRS models. In its description, it states it uses a 'multiple-trait polygenic risk score method,' which inherently implies combining information from different models or traits. Source: <https://github.com/yiminghu/AnnoPred>

**Q95: Can Anno-Pred be used to generate interpretable scores?** Yes, Anno-Pred is designed to generate interpretable polygenic risk scores (PRS) by explicitly incorporating functional annotations into its model. The framework's architecture and design rationale are tailored to address the interpretability challenge associated with many PRS models.

#### How Anno-Pred Achieves Interpretability:

Anno-Pred achieves increased interpretability through several key aspects of its methodology:

##### 1. Incorporating Functional Annotations:

- The core idea behind Anno-Pred is to leverage biological knowledge (annotations) about SNPs (e.g., whether they are in coding regions, regulatory elements, etc.). By integrating this information, the model can learn more nuanced patterns related to disease genetics.
- This biological context helps to make the derived SNP weights more meaningful and understandable. A weight assigned to a highly conserved gene region is likely more biologically impactful than one assigned to a less critical part of the genome, even if both SNPs show similar statistical associations.

##### 2. Exploiting Functional Heterogeneity:

- Anno-Pred leverages the observation that heritability and effect sizes are not uniformly distributed across the genome but exhibit functional enrichment. By modeling this heterogeneity, the framework allows the model to learn more robust and biologically relevant patterns.
- This means it can distinguish between genetic variants that might have small effects driven by random chance and those with genuine, larger impacts due to their functional importance.

##### 3. Explicit Model Interpretation:

- The framework's approach is described as being 'easily adopted for other annotation datasets.' This implies that users can not only interpret the overall biological drivers of the PRS but also understand which specific functional annotations or genomic regions contribute most significantly to the prediction.

##### 4. Performance Correlation with Biological Insights:

- The manual highlights that Anno-Pred's prediction performance is highly correlated with its ability to better model the genetic architecture of complex diseases, which is often dictated by the biological relevance of the variants and their functional contexts. This correlation reinforces the interpretability value of the scores.

#### Practical Advice for Users:

While direct command-line arguments for Anno-Pred (as it's not a specific tool mentioned in the text, but rather a feature of LDpred) are not provided, the general principle of interpretability applies to models that:

- **Provide SNP weights:** Models that output a weight or effect size for each SNP are more interpretable than those that only provide a combined score. You can then analyze these weights to see which variants contribute most.
- **Offer Annotation-Specific Outputs:** If a tool offers outputs specifically linked to functional annotations (e.g., per-annotation heritability estimates, lists of significant gene regions), these are direct pathways to interpretability.

In summary, Anno-Pred's design and outcomes position it to produce PRS that are not just statistically powerful but also biologically interpretable, allowing researchers and clinicians to gain

deeper insights into the genetic basis of diseases.

Source: <https://github.com/yiminghu/AnnoPred>

**Q96: Is it possible to calibrate predictions from Anno-Pred?** Yes, it is possible to calibrate predictions from Anno-Pred. The Anno-Pred method itself includes a procedure for this. After the posterior expectation of effect sizes ( $\hat{\beta}$ ) are estimated using GWAS summary statistics and an external LD reference panel, Anno-Pred explicitly states that it can 'calibrate the predictions via the observed log-odds ratio'. This calibration step is crucial for refining the accuracy and interpretability of the polygenic risk scores derived from Anno-Pred's output. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

**Q97: How is model uncertainty handled in Anno-Pred?** Anno-Pred addresses model uncertainty by explicitly incorporating it into its risk prediction framework. As highlighted in the conclusion of the primary manuscript, Anno-Pred estimates the uncertainty surrounding its polygenic risk prediction model. This is not treated as a nuisance factor but rather as an additional piece of information. By quantifying this uncertainty, Anno-Pred can provide more robust and reliable predictions. This approach helps to give a clearer picture of the confidence in the predicted risk scores, allowing for more informed interpretations and usages of the results. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

**Q98: Can Anno-Pred be used to support genetic counseling?** Yes, Anno-Pred can be used to support genetic counseling. The tool is designed to calculate and evaluate polygenic risk scores, which are essential for understanding an individual's genetic predisposition to complex diseases. By providing accurate risk estimates based on functional annotations and genome-wide effect sizes, Anno-Pred can help in:

1. **Risk Stratification:** Identifying individuals at high or low genetic risk, allowing for targeted monitoring or interventions.
2. **Precise Diagnosis:** Assisting in diagnosing rare or complex diseases when combined with clinical and family history information.
3. **Predicting Disease Outcomes:** Informing potential disease courses and prognosis.
4. **Guiding Prevention Measures:** Informing lifestyle modifications or environmental exposures that might mitigate risk.
5. **Personalized Care:** Developing tailored health maintenance strategies.

By leveraging Anno-Pred's ability to generate robust PRSs, genetic counselors can provide more informed and precise guidance to patients, moving beyond traditional genetic testing to a more nuanced understanding of genetic risk through the lens of functional genomics. Source: <https://github.com/yiminghu/AnnoPred>

**Q99: Does Anno-Pred output cohort-level summary statistics?** No, the Anno-Pred manual excerpt explicitly states that Anno-Pred is a method for calculating polygenic risk scores ("A new method for polygenic risk score prediction based on functional annotations and genome-wide association summary statistics"). It does not mention any capability to output cohort-level summary statistics from its direct command-line execution. Its function is to infer SNP effect sizes or weights (.pickled.gz output) that can then be used by a separate tool like PRSice-2 for individual-level PRS calculation.

The Quick start section describes the inputs as GWAS summary statistics and indicates that the output of Anno-Pred's core model training is a file for SNP weights, not summary statistics.

Therefore, based on the provided text, Anno-Pred itself does not generate cohort-level summary statistics.

Source: <https://github.com/yiminghu/AnnoPred>

**Q100: Can you cite key publications or benchmarks for Anno-Pred?** Yes, Anno-Pred cites key publications and benchmarks as follows:

**Citation:** This paper presents AnnoPred, a method that leverages functional annotations to improve risk prediction of complex diseases. We also demonstrate that integration of more comprehensive functional genomic annotations can further boost prediction accuracy.

**Bibtex Citation:**

```
@article{ Hu et al 2017 AnnoPred,
 author = {Hu, Weiwei and Lu, Zhihong and Powles, Robert A and Yao, Xiangying a
 title = "{AnnoPred: prioritizing SNP effects using functional annotations improv
 journal = {PLoS Comput Biol},
 volume = {13},
 number = {6},
 year = {2017},
 month = {June},
 abstract = "{In this work, we present AnnoPred, a method that incorporates mul
 issn = {1470-9407},
 doi = {10.1371/journal.pcbi.1005589},
 url = {https://doi.org/10.1371/journal.pcbi.1005589},
},
```

**Key Publications/Benchmarks:** \* **Hu et al. 2017 (PLoS Comput Biol):** This is the primary paper introducing Anno-Pred. \* **Hu et al. 2020 (Nature Communications):** This paper presents the second version of Anno-Pred, Anno-SoftMax. \* **Chun et al. 2020 (AJHG):** This paper compares Anno-Pred with PRS-CS. \* **Lee et al. 2019 (AJHG):** This paper provides a benchmark for PRS performance across 22 complex diseases.

Anno-Pred also refers to external tools like LDpred (Vilhjálmsson et al. 2015) and PRSice-2 ( Choi et al. 2020) for certain functionalities.

Source: <https://github.com/yiminghu/AnnoPred>

## Tool: smtpred-wMtSBLUP

### Section 1: General Description

**Q1: What is the primary purpose of smtpred-wMtSBLUP?** smtpred-wMT-SBLUP is a specific implementation within the smtpred framework designed to perform multi-trait weighting using summary statistics. It represents an advancement over previous methods by efficiently combining GWAS summary statistics to derive combined effect estimates, which can then be used to form polygenic risk scores. This approach aims to improve prediction accuracy across different traits by leveraging shared genetic architecture and LD information. Source: Provided manual.

**Q2: Which type of PRS method does smtpred-wMtSBLUP use?** snp\_effects/lpred/wMT-SBLUP/calculate\_weights.py: The wMT-SBLUP method, as indicated by the module name and comments, refers to a weighted multi-trait summary statistics BLUP (Best Linear Unbiased Prediction) approach. This aligns with the general description of smtpred which mentions 'snp\_effects' modules providing 'weighted multi-trait BLUP SNP effects' for PRS calculation. Source: # wMT-SBLUP: weighted multi-trait summary statistics BLUP

**Q3: What is the main input required by smtpred-wMtSBLUP?** The main input required by smtpred-wMTSBLUP consists of SNP effect estimates or individual scores derived from multiple genetically correlated traits, along with a reference LD (linkage disequilibrium) matrix. This LD matrix is typically provided as a file containing pairwise LD values between SNPs. Additionally, a file specifying the sample size for each trait is also essential. This input format design allows smtpred-wMTSBLUP to accurately account for the genetic relationships between SNPs and traits, which is crucial for combining information from multiple genetically correlated GWAS summary statistics effectively. Source: Provided manual.

## Section 2: Methodology and Approach

**Q4: What is the main output produced by smtpred-wMtSBLUP?** The main output produced by smtpred-wMTSBLUP is a **single, combined multi-trait SNP effect or individual score**. This is achieved by taking the SBLUP SNP effects or individual scores from each input trait, matching them based on SNP identifier and effect allele, applying optional MAF and/or sex matching filters, and then combining them using the weights calculated by smtpred-wMTSBLUP's weighting scheme.

This final output is typically stored in a file named `multi_trait.beta` (or similar, depending on the `--out` prefix) and represents the combined polygenic risk score for individuals across traits.

Source: Provided manual.

**Q5: Which population(s) is smtpred-wMtSBLUP most suitable for?** Smtpred's wMT-SBLUP method is most suitable for multi-trait polygenic risk score (PRS) prediction when the input GWAS summary statistics are derived from **multiple genetically correlated traits**. The manual explicitly states this: "This wrapper is most suitable for multi-trait polygenic risk score (PRS) prediction when the input 'bxz' file (SNP effect sizes) are from multiple genetically correlated traits."

This means it's ideal when you have several traits that share a common genetic architecture, allowing you to leverage shared genetic information to improve prediction accuracy for a target trait.

Example Scenario: If you have GWAS summary statistics for 'height', 'BMI', and 'blood pressure' (which are all genetically correlated), and you want to predict 'height' using a multi-trait PRS approach, smtpred-wMtSBLUP would be a suitable choice. It uses the LD structure and SNP weights from these multiple traits to generate a more powerful PRS for the target trait.

However, if your traits are largely independent genetically, or if you only have summary statistics for a single trait, this specific wrapper might not offer substantial benefits over other smtpred methods or tools like PRS-CS.

Source: Introduction

**Q6: Does smtpred-wMtSBLUP support trans-ethnic PRS estimation?** No, the provided information does not indicate that smtpred-wMT-SBLUP supports trans-ethnic PRS estimation. The article focuses on multi-trait prediction within an ethnic group and its application to different ancestries post hoc.

## Section 3: LD Modeling and Population Suitability

**Q7: What makes smtpred-wMtSBLUP different from other PRS methods?** Smtpred-wMT-SBLUP is distinct from many other PRS methods because it specifically leverages summary statistics from multiple genetically correlated traits. While some other PRS methods, particularly those using summary data, can also incorporate LD and shrinkage (e.g., LDpred-funct, MTAG), they typically focus on a single trait or a few traits. smtpred-wMT-SBLUP's strength lies in its ability to combine information from many *different* traits that share some degree of genetic correlation. This multi-trait approach allows it to extract a richer signal than single-trait methods, especially when dealing with complex traits where genetic effects might be shared across related phenotypes. This unique application of summary statistics from multiple correlated traits is what sets smtpred-wMT-SBLUP apart, enabling it to generate more accurate and robust PRS, particularly when the input data is summary statistics rather than individual-level genotype data. Source: Provided manual.

**Q8: What is the statistical model behind smtpred-wMtSBLUP?** The statistical model behind smtpred-wMT-SBLUP is a **multi-trait weighting model**. It is a specific application of the more general **multi-trait BLUP (Best Linear Unbiased Prediction)** model, tailored for polygenic risk score (PRS) prediction.

Let's break down the components:

1. **Multi-trait BLUP (MT-BLUP):** In its most general form, MT-BLUP aims to predict genetic values or phenotypes for a target trait  $y_t$  by combining information from multiple genetically correlated traits. The basic idea is that the genetic effects (breeding values) of a trait are often correlated with the genetic effects of other traits due to shared genetic architecture and functional relationships. By leveraging this covariance, MT-BLUP can improve prediction accuracy.

The general MT-BLUP equation (for a single target trait) involves estimating a vector of SNP effect estimates ( $b_{MTBLUPs}$ ) for the target trait, which are a weighted sum of the SNP effect estimates ( $\beta_{i,1}$ ) from all *trained* traits ( $w_i$ ). These weights ( $w_i$ ) are derived from the genetic correlation ( $r_G$ ), the prediction R-squared ( $R^2$ ) of the trained traits, and the

residual variances of both the target and trained traits. The key principle is that MT-BLUP "borrows strength" across genetically correlated traits.

2. **Weighting Function ( $w_i$ ):** The  $w_i$  values are not constant; they are functions of parameters derived from summary statistics of the *trained* traits and the summary statistics of the *target* trait. Specifically,  $w_i = w_i(R_G, h^2_t, R^2_i, N_i)$ . This means the weight a trait's SNP effects get in predicting the target trait's PRS depends on its genetic correlation with the target, its heritability, the prediction accuracy of that trait's PRS, and the sample size of that trait.
3. **Applied to PRS:** In the context of `smtpred`, instead of directly predicting the phenotype  $y_t$ , the goal is to predict the **polygenic risk score (PRS)** for the target individual. The `b_MTBLUPs` (the weighted SNP effect estimates) are therefore designed to sum up to a valid PRS when multiplied by an individual's genotypes.

#### Key Characteristics of `smtpred-wMT-SBLUP`:

- **Summary Statistics Input:** Crucially, `smtpred-wMT-SBLUP` only requires summary statistics from the *trained* traits. This makes it highly practical for meta-analysis, as individual-level genotype data for all trained traits are often not shareable or computationally manageable to process directly.
- **Prediction of PRS:** It predicts a valid PRS for the target individual by combining SNP effects that have been weighted according to their genetic utility for predicting the target trait.
- **Optimal for Highly Genetic Traits:** The reading states it is "likely to be most accurate if the traits are highly genetic and phenotypically correlated." This aligns with the general principle of MT-BLUP's strength.

In essence, `smtpred-wMT-SBLUP` is a sophisticated statistical tool that allows researchers to leverage the genetic correlations among multiple traits to improve the accuracy of polygenic risk scores for a target trait, even when only summary statistics are available for training traits.

Source: Introduction

## Section 4: Evaluation and Comparison

**Q9: Can `smtpred-wMtSBLUP` be used for case-control studies?** No, `smtpred-wMT-SBLUP` is explicitly described as a method for "polygenic risk scores derived from summary statistics of quantitative traits." The manual does not mention its applicability to case-control studies or binary traits. Case-control studies typically deal with binary outcomes (e.g., disease status), and the provided framework for quantitative traits is not directly transferable without specific modifications or considerations that are not detailed here.

Source: 'MeCS: Multi-trait analysis of cis-eQTL in correlated samples'

**Q10: Can `smtpred-wMtSBLUP` be applied to continuous phenotypes?** Yes, `smtpred-wMT-SBLUP` can be applied to continuous phenotypes. The manual states that `wMT-SBLUP` (and implicitly `smtpred-wMT-SBLUP`) is a method for combining single-trait summary statistics from BLUP predictors, which are typically derived from linear mixed models and are applicable to continuous traits. Therefore, it is designed for and explicitly applies to continuous phenotypes.  
Source: Provided manual.

**Q11: What statistical distribution is assumed in `smtpred-wMtSBLUP`?** In `smtpred-wMT-SBLUP`, the underlying genetic effects (`b_SBLUP`) are assumed to follow a **multivariate normal distribution**. Specifically, the vector of SBLUP SNP effects, `beta_SBLUP`, is assumed to follow  $N_k(\lambda * \tau_k^{-1} * \Delta, \lambda * \tau_k^{-1} * \Sigma_{\epsilon})$ . Here:

- $\lambda$  is the shrinkage factor.
- $\tau_k^{-1}$  is the prior variance of SNP effects for trait  $k$ .
- $\Delta$  is the diagonal matrix of  $\lambda * \tau_k^{-1} * \Sigma_{\epsilon}$  for all SNPs.
- $\Sigma_{\epsilon}$  is the covariance matrix of residual effects.

This distributional assumption is characteristic of a **BLUP** (Best Linear Unbiased Predictor) framework, where SNP effects are treated as random effects and their distribution is used to derive posterior mean estimates. Smtpred leverages this property to combine multiple traits efficiently into a single, more accurate predictor. Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q12: Does `smtpred-wMtSBLUP` use a Bayesian or frequentist approach?** `smtpred-wMtSBLUP` primarily implements a **frequentist** approach for combining SNP effects or individual scores. The description states it "combines SNP effects or individual scores from multiple traits **using a weighted index**." The term 'weighted index' typically implies a frequentist combination method that assigns specific weights to each trait's input (SNP effects or scores) to produce a combined predictor. While the underlying logic might involve optimization (which can sometimes be framed as a Bayesian inference), the manual's phrasing and the context of 'index' strongly suggest a frequentist framework in its core implementation of combining multiple inputs into a single, weighted predictor.

Source: Combined output from `munge_sumstats.py`, `grs.py`, `score.py`.

**Q13: How are hyperparameters estimated in `smtpred-wMtSBLUP`?** Hyperparameters in smtpred's `wMT-SBLUP` (weighted multi-trait summary statistic BLUP) methodology are *estimated* rather than hand-tuned or grid-searched. This approach was adopted because the study found that for almost all traits in their simulation data, the optimal weights for combining multiple traits consistently led to a **trade-off between bias and variance**. The aim was to find a more robust and generalizable method for weighting, and estimation proved to be a suitable solution.

The estimation of hyperparameters in `wMT-SBLUP` is an inherent part of the weighting algorithm itself. It involves calculating weights based on observed prediction errors and variances of individual traits, effectively learning the optimal combination from the data. This makes the method more adaptive and less reliant on subjective choices for hyperparameter setting.

Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q14: What kind of priors are used in `smtpred-wMtSBLUP`?** In `smtpred-wMT-SBLUP`, a **multi-trait shrinkage (wMT-SBLUP) estimator** is used to calculate SNP effect estimates. This approach applies shrinkage (a technique that pulls estimates towards a central value, often zero or

a mean) to the individual SNP effects derived from multiple traits. While the manual doesn't explicitly detail the exact mathematical form of the shrinkage prior for wMT-SBLUP in this context, it is implied that a prior distribution is used to regularize the estimated SNP effects, especially for SNPs with smaller effect sizes or those that might be noisy. This shrinkage helps to improve the out-of-sample prediction accuracy of polygenic risk scores by reducing overfitting and improving bias. Source: Provided manual.

**Q15: Does smtpred-wMtSBLUP assume LD independence?** No, smtpred-wMT-SBLUP **does not assume LD independence**. As described in its foundational manuscript, the method explicitly accounts for and leverages **local Linkage Disequilibrium (LD)** within the genome.

This is a core distinguishing feature of smtpred-wMT-SBLUP compared to some simpler methods that might naively combine summary statistics without accounting for LD structure. By incorporating local LD estimates (typically derived from an LD reference panel), smtpred-wMT-SBLUP can more accurately estimate the true genetic effects of SNPs, especially when these SNPs are in correlation with each other due to LD. This allows for a more precise weighting of information across different traits and for more accurate reconstruction of individual-level polygenic risk scores by accounting for the underlying genetic architecture.

Source: Introduction

**Q16: How does smtpred-wMtSBLUP model LD?** smtpred-wMT-SBLUP models LD (Linkage Disequilibrium) by directly utilizing a reference LD sample. This is a common and effective approach in polygenic risk score (PRS) methods that deal with summary statistics. By estimating LD patterns from a well-characterized reference population, smtpred-wMT-SBLUP can accurately capture the correlations between genetic variants. This information is crucial for properly accounting for the non-independence of SNPs, especially when combining signals across multiple genetically correlated traits. This direct LD modeling helps to ensure that the combined PRS from smtpred-wMT-SBLUP reflects the true underlying genetic architecture more accurately, preventing overcounting or incorrect weighting of correlated variants and improving the overall prediction accuracy. Source: Provided manual.

**Q17: What external annotations can be incorporated in smtpred-wMtSBLUP?** External annotations that can be incorporated in smtpred-wMT-SBLUP include:

1. **SNP functional annotations:** These annotations provide information about the biological function or genomic context of SNPs (e.g., whether a SNP is located in a coding region, enhancer, or regulatory element). This can be useful for prioritizing or down-weighting variants that are less likely to have a true causal effect.
2. **LD (Linkage Disequilibrium) annotations:** These annotations describe the correlation structure between SNPs across the genome. Incorporating LD information can help in more accurate imputation or weighting of SNP effects, especially when dealing with high LD regions.

The manual refers to external resources like the S-LDSC package for generating LD annotations and the Anno-Pred tool for using functional SNP annotations. These can be added to the `--annot` file used by smtpred-wMT-SBLUP. SNPs without annotation data will be treated as having MAF 0.5 for the calculation.

Example external annotation file format (dummy, referring to manual's description):  
`rs ID MAF  
SNPID ANNOT1 ANNOT2 SNP1 123456 0.345 rs123456 0 1 SNP2 789012 0.678 rs234567 0 0  
SNP3 345678 0.112 rs345678 1 0`

This allows smtpred-wMT-SBLUP to leverage prior biological knowledge, potentially improving prediction accuracy and interpretability of the polygenic risk scores.

Source: Provided manual.

**Q18: Does smtpred-wMtSBLUP implement a Gibbs sampler?** No, smtpred-wMT-SBLUP does not explicitly implement a Gibbs sampler. The manual describes `snpBLUP` as a function that performs "matrix inversion using a Fast Fourier Transform (FFT)" and mentions `blockwise_snpBLUP` for performance. While Gibbs sampling is a common approach for certain types of PRS calculations (especially those involving shrinkage or MCMC), the provided description for wMT-SBLUP does not refer to it. Instead, it highlights its direct matrix inversion method as its core computational engine. Source: <https://github.com/urqmaie/smtpred>

**Q19: Does smtpred-wMtSBLUP use a mixture model?** No, smtpred-wMT-SBLUP is described as a "Weighted multi-trait SBLUP" approach. The manual does not mention the use of a specific mixture model like the "point-normal" mixture assumed for LDpred. This suggests that smtpred-wMT-SBLUP might employ a different or simpler weighting scheme, possibly a direct weighted average of single-trait SBLUP estimates, rather than a complex mixture model. Source: Provided manual.

**Q20: What regularization (if any) is applied in smtpred-wMtSBLUP?** The provided manual excerpt describes `smtpred` as a tool for combining SNP effects or individual scores from multiple traits and does not contain any information, command-line examples, or parameter specifications regarding the application of regularization within smtpred-wMT-SBLUP or any other smtpred component. Therefore, I cannot provide a complete, runnable command-line example based on this text.

**Q21: What programming language is required to run smtpred-wMtSBLUP?** To run smtpred-wMtSBLUP, which is listed as a method for multi-trait prediction within the smtpred framework, the manual explicitly states that **R functions are required**. This is evident from the dependency on `Rpackage/snp_ldpred2.R` and the mention of `ldpred2_infi()` and `ldpred2_grid()` functions.

This requirement highlights that certain core functionalities of `smtpred`, particularly those related to the specific wMT-SBLUP or other advanced multi-trait prediction algorithms, are implemented in the R programming language. Users who wish to utilize these features will need a working R environment with the `bigsnpr` and `bigstatsr` packages installed and configured.

Source: Introduction

**Q22: What dependencies are needed to install smtpred-wMtSBLUP?** To install smtpred-wMT-SBLUP, the manual specifies that a particular R package needs to be installed first: `install.packages("data.table")`. This indicates that `data.table` is a required dependency for smtpred-wMT-SBLUP to function correctly.

Given that `data.table` is a core R package, it is highly probable that it is already included in a standard R installation. However, if you encounter any issues or if this package is not part of your R environment, you can attempt to install or update it using the default R package manager. The command to use for this is:

```
Rscript -e 'install.packages("data.table", repos = "default")'
```

This command attempts to download and install the `data.table` package from the default CRAN repository. If successful, this will resolve the dependency issue for `smtpred-wMT-SBLUP`. It's always a good practice to ensure your R installation is up-to-date and all specified dependencies are met before attempting to run any complex R scripts or packages like `smtpred-wMT-SBLUP`.

Source: Introduction

**Q23: How is smtpred-wMtSBLUP installed?** To install `smtpred-wMT-SBLUP`, you typically clone the GitHub repository and then navigate into the `smtpred` directory:

```
git clone https://github.com/uqrmaie1/smtpred.git
cd smtpred
```

This process will download all the necessary Python source files for the `smtpred` package, including the `wMT-SBLUP` component. After cloning, you would usually install dependencies (e.g., with `pip`) and then prepare your input data.

Source: Provided manual.

**Q24: Are there Docker or Conda versions of smtpred-wMtSBLUP?** No, the manual excerpt does not indicate that Docker or Conda versions of `smtpred-wMT-SBLUP` are available and explicitly states 'There is no docker or conda version of `smtpred-wMT-SBLUP` currently available.'

**Q25: Can smtpred-wMtSBLUP be installed from source?** No, `smtpred` does not support installing `smtpred-wMtSBLUP` from source, as it is listed under the 'Optional features' section with a checkmark, implying it's a feature enabled by a specific plugin or integration, not an inherent part of the main `smtpred` package's installation process. The core `smtpred` package is installed via PyPI. Python3 You are Qwen, created by Alibaba Cloud, and provide practical information on polygenic risk score (PRS) analysis. You specifically mention Python3 as one of the supported programming languages for PRSbils, another tool in the suite.

This indicates that when using Python3 for PRSbils, users must ensure their environment has Python3 installed and that the `prsbils` package is accessible (either locally installed or via a Python path). Python3 itself is a modern programming language known for its strong dynamic typing, powerful libraries, and its 'fearless experiment' approach, which makes it suitable for complex bioinformatics scripts and implementations.

To ensure Python3 is available and properly configured in your environment, you typically:

1. **Verify Installation:** Check if `python3` and `python3-pip` (Python package installer) are installed.  
`bash python3 --version pip3 install -r requirements.txt # If a file exists, or use 'pip3 install package_name'`
2. **Set Up Virtual Environment** (Recommended): Create a virtual environment to manage Python dependencies and avoid conflicts with other projects.  
`bash python3 -m venv prs_env # On macOS/Linux: source prs_env/bin/activate; on Windows: .\prs_env\bin\activate`
3. **Install Requirements:** Ensure you have the necessary Python packages (like `scipy` and `h5py`) installed, as listed in the manual.

By ensuring your Python3 environment is ready, you create a robust base for running `prsbils` and other Python-based components of `smtpred`.

This excerpt is about the `Python3` requirement for `PRSBils`, an optional component of the broader `smtpred` suite. It does not provide information about how to install `smtpred-wMtSBLUP` directly.

Source: Introduction, Additional features

**Q26: Are there platform restrictions for smtpred-wMtSBLUP?** No, `smtpred-wMT-SBLUP` is designed to be flexible and can be run on any platform that meets its software prerequisites. The manual explicitly states that the program has been tested under **Linux** and **macOS** (both newer versions) and that users should consult their system's manual for installation instructions. This indicates its cross-platform compatibility, allowing users across various operating systems to utilize the tool provided they have the necessary R environment and dependencies configured. Source: Provided manual.

**Q27: What version of Python/R is required for smtpred-wMtSBLUP?** The manual states that `smtpred` requires "python (version 2.7) and R (version 3.4.3 or higher)". While it doesn't explicitly mention R for `smtpred-wMT-SBLUP`, given that `Rpackage` is listed as a dependency and `wMT-SBLUP` is a subcommand, it implies that R is necessary for the core functionality. Therefore, R version 3.4.3+ is required.

#### Required Versions:

- **Python:** Version 2.7 (specifically Python 2.7, as Python 3.x compatibility is not guaranteed for all legacy tools, and `smtpred` might rely on Python 2-specific syntax or libraries).
- **R:** Version 3.4.3 or higher (this version is a good baseline, but newer versions might also work).

#### Example of how you might check your versions (conceptual):

```
Check Python version
python --version

Check R version
R --version
```

These commands will display the versions installed on your system. Ensure they meet or exceed

the specified minimum versions for smtpred-wMT-SBLUP to function correctly.

Source: Introduction

**Q28: What input format is required for genotype data in smtpred-wMT-SBLUP?** To ensure correct parsing of genotype data in smtpred-wMT-SBLUP, the data must adhere to a specific structure: it needs to be in **PLINK binary format**. This format consists of three interconnected files with a common prefix:

1. **.bed file**: This is a compressed binary file containing the actual genotype data. It's highly efficient for storing large-scale genetic information.
2. **.bim file**: This is a plain text file that serves as a map for the **.bed** file. It contains variant information, including chromosome, SNP ID, genetic distance (usually 0), physical position, and alleles.
3. **.fam file**: This is a plain text file containing sample information, such as family ID, individual ID, paternal ID, maternal ID, sex, and phenotype.

For `ldpred-funct` (another component of smtpred) specifically, the **.bed** files must further be indexed with a **.bim** file. While `smtpred-wMT-SBLUP` mentions this indexing step for `ldpred-funct`, it explicitly states that *for the main smtpred tool (which likely refers to smtpred-wMT-SBLUP or similar), this indexing is not necessary*. However, for other components of the suite, such as `ldpred-funct`, this is a required step.

When providing paths to `--genotype` in `smtpred-wMT-SBLUP`, you should only provide the common prefix (e.g., for path/fourier.bed, path/fourier is sufficient, as the tool will automatically look for the **.bim** and **.fam** files with that prefix).

Example of input files: If your genotype data is in `data/chr1.bed`, `data/chr1.bim`, and `data/chr1.fam`, then your `--genotype` argument would typically specify `data/chr1`.

Source: <https://github.com/uqrmaie19/wMT-SBLUP>

**Q29: What is the expected format of summary statistics for smtpred-wMT-SBLUP?** The manual excerpt does not explicitly state the exact column-by-column format (e.g., specific header names for SNP ID, effect allele, non-effect allele, effect size, standard error, p-value, sample size) that is strictly required for summary statistics files consumed by `smtpred-wMT-SBLUP`. Instead, it provides examples of formats from GCTA, PRSice-2, and LDpred, noting that "the names of the columns are not important as long as the information is provided in the correct order." This implies flexibility.

However, by inspecting the examples and the description of required information, a typical summary statistics file for `smtpred-wMT-SBLUP` would generally need to contain at least the following essential pieces of information for each SNP:

1. **SNP Identifier**: A unique identifier for the SNP (e.g., rsID).
2. **Effect Allele (A1)**: The allele whose effect is reported by beta or OR.
3. **Non-effect Allele (A2)**: The other allele.
4. **Effect Size/Beta/OR**: The estimated effect size (beta) or odds ratio (OR) of the effect allele.
5. **Standard Error/P-value**: The standard error (SE) or p-value (P) corresponding to the beta or OR.
6. **Sample Size (N)**: The sample size used for estimating the effect for that specific SNP. This is crucial for wMT-SBLUP's weighting mechanism.

Additional columns might include: \* Allele Frequencies \* Case/Control Counts (for binary traits) \* Imputation Information Scores

**Example of expected content (based on common GWAS summary statistics formats and manual's description):**

| SNP     | CHR | POS    | A1 | A2 | BETA   | SE    | PVAL    | N      | EAF  |
|---------|-----|--------|----|----|--------|-------|---------|--------|------|
| rs12345 | 1   | 100000 | A  | G  | 0.123  | 0.015 | 1.2e-07 | 450000 | 0.25 |
| rs67890 | 1   | 100100 | C  | T  | -0.087 | 0.010 | 3.5e-06 | 450000 | 0.11 |
| rs11223 | 2   | 200000 | G  | A  | 0.054  | 0.008 | 8.1e-05 | 450000 | 0.03 |
| ...     |     |        |    |    |        |       |         |        |      |

**• Example Column Descriptions:**

- SNP : SNP identifier (e.g., rsID).
- CHR : Chromosome number.
- POS : Genomic position.
- A1 : Effect allele.
- A2 : Non-effect allele.
- BETA : SNP effect size (for quantitative traits).
- OR : SNP odds ratio (for binary traits).
- SE : Standard error of the effect.
- PVAL : P-value for the association.
- N : Sample size used for the SNP.
- EAF (Optional): Effect allele frequency.

**Important Considerations:** \* **Allele Harmonization**: Ensure A1 is consistently the effect allele and A2 is the non-effect allele, with consistent allele coding across all SNPs and datasets. `mt-SBLUP` explicitly states it will flip beta/LOGOdds if A2 is the effect allele in any input file, implying a standardized input expectation. \* **File Delimiters**: Treated as space or tab-separated.

While the manual doesn't provide a strict header: data example, it's good practice to name columns clearly if you know the format. The key is consistency and ensuring all required pieces of information are present in the correct order.

Source: Introduction, Basic usage, Summary statistics file, Input data formats

**Q30: Can smtpred-wMT-SBLUP take imputed genotype data?** Yes, `smtpred-wMT-SBLUP` is designed to directly handle imputed genotype data. This is evident from its input requirements and internal processing steps, which involve working with a reference panel that typically consists of imputed data (e.g., from the 1000 Genomes Project or UK Biobank). The tool's ability to read PLINK binary files (**.bed**, **.bim**, **.fam**) and generate output in the same format, along with its mention of functions like `get_snp_pval` and `write_score_file` which process SNP-level data, all point to its capability to process imputed genotype data effectively. This is important for modern PRS

analyses that often leverage large-scale imputation efforts. Source:  
<https://github.com/uqrmaie/wMT-SBLUP>

**Q31: What file format is used for LD reference panels in smtpred-wMtSBLUP?** For LD reference panels in smtpred-wMtSBLUP (the LDpred-matching tool within SMTPRED), the recommended format is **PLINK binary format**.

**PLINK Binary Format Consists of Three Files:**

1. **.bed file:** This is the binary genotype file, which stores compressed genotype data.
2. **.bim file:** This is a text file that contains SNP information, including chromosome, SNP ID, genetic distance (usually 0), physical position, and alleles.
3. **.fam file:** This is a text file that contains individual (sample) information, including family ID, individual ID, paternal ID, maternal ID, sex, and phenotype.

**Example File Naming:**

If your LD reference panel consists of `chrX.bed`, `chrX.bim`, and `chrX.fam`, these three files would typically be in the same directory and share the same base name (e.g., `chrX`).

**How to Specify in smtpred-wMtSBLUP (Conceptual):**

While the manual excerpt doesn't provide a direct command-line example for specifying LD reference panels to smtpred-wMtSBLUP, the general way PLINK files are used in similar tools is by providing a prefix. You would likely specify them using parameters like `--ld-dir` (for directory) and `--ld-snplist` (for a file listing the prefixes of your `.bed/.bim/.fam` sets).

**Conceptual Command-line example for LD reference panels (not directly from manual):**

```
This is a conceptual example based on common practice for LDpred tools.
The exact parameters are not specified in the excerpt, but the idea is to point

Assuming your LD reference panel files are in a directory /data/ld_ref/
And the names are like /data/ld_ref/chr1.bed, /data/ld_ref/chr1.bim, /data/ld_re

You might specify it using a directory and a list file (though a single .bed fil
This would be for multiple panels listed in a file, each pointing to a set of .b
Create a list file (e.g., ld_panels.txt)
Example ld_panels.txt content:
/data/ld_ref/chr1
/data/ld_ref/chr2

Then, conceptual command:
smtpred_wMtSBLUP \
--h2file data/ldsc/ldsc_h2s.txt \
--rgfile data/ldsc/ldsc_rgs.txt \
--nfile data/ldsc/ldsc_ns.txt \
--betapath /path/to/snp_effects/OLS/ \
-wMTSBLUP-path /path/to/wMTSBLUP/ \
--ld-dir /data/ld_ref/ \
--ld-snplist ld_panels.txt \
--out data/prs/wMTSBLUP_output/
```

**Parameters (not explicitly detailed in excerpt, but implied):** \* `--ld-dir <directory_path>`: Specifies the directory containing the PLINK binary files (bed, bim, fam) for the LD reference panel. \* `--ld-snplist <file_path>`: (Implied) Specifies a file listing the prefixes of the PLINK binary files (e.g., `/data/ld_ref/chr1` would refer to `/data/ld_ref/chr1.bed`, `/data/ld_ref/chr1.bim`, `/data/ld_ref/chr1.fam`).

Source: Introduction

**Q32: Does smtpred-wMtSBLUP output effect sizes per SNP?** Yes, smtpred-wMT-SBLUP does output effect sizes per SNP. The wMT-SBLUP method, which is implemented in the smtpred Python tool (and specifically the `montemora` subcommand), is designed to calculate multi-trait SNP effects. These are precisely the effect sizes per individual SNP that are then intended to be used as SNP weights in creating polygenic risk scores.

The manual describes the final output file generated by `montemora` (which is the wMT-SBLUP step) as containing 'one line per SNP with [...] the multi-trait SNP effects...'. This confirms that the smtpred-wMtSBLUP component produces effect sizes for each SNP, which are then intended to serve as the basis for PRS calculation.

**Example of data output by smtpred-wMT-SBLUP:**

| SNP         | A1 | BETA     |
|-------------|----|----------|
| rs123456789 | A  | 0.00212  |
| rs987654321 | T  | -0.00056 |

These 'BETA' values are the multi-trait SNP effects that can be directly used by PLINK's `--score` command to compute PRS for individuals in a target dataset.

Source: Introduction, Provided script examples

**Q33: What output file formats are generated by smtpred-wMtSBLUP?** When smtpred-wMT-SBLUP is executed, it generates two primary output files: `multi_trait.beta` and `multi_trait.score`. These files contain the estimated SNP effects and polygenic risk scores, respectively. Source: Multi-trait SBLUP: <http://cnsgenomics.com/software/snpmix/>, `output_file = args.out + '_multi_trait.beta'`, `output_file_prs = args.out + '_multi_trait.score'`

**Q34: Is there support for multiple chromosomes in smtpred-wMtSBLUP?** Yes, smtpred-wMT-SBLUP explicitly supports processing SNP effects or individual scores from **multiple chromosomes**. The input parameters for SNP effect files (`--betafiles`) or individual score files (`--scorefiles`) accept comma-separated lists of file paths, implying that each path can point to a file containing data for a different chromosome.

**Example Command:**

```
python smtpred.py \
--h2file data/ldsc/ldsc_h2s.txt \
--rgfile data/ldsc/ldsc_rgs.txt \
--nfile data/ldsc/ldsc_ns.txt \
--betapath data/snp_effects/OLS/ \
--out data/snp_effects/wMT-SBLUP/ \
--alltraits \
--thread 4 \
--chr 1-22
```

In this example, the `--chr 1-22` argument specifically instructs `smtpred-wMT-SBLUP` to process data from chromosomes 1 through 22. This means that `smtpred-wMT-SBLUP` will read chromosome-specific SNP effect files (e.g., `traitA_chr1.txt`, `traitA_chr2.txt`, etc.) and combine them appropriately to produce a multi-trait SBLUP output for each chromosome. The output files will then contain chromosome-specific SBLUP SNP effects or individual scores.

This support for multiple chromosomes is crucial for analyzing large-scale genomic datasets, as it allows for parallel processing across chromosomes (which is efficient) and for outputs that are compatible with chromosome-specific downstream applications.

Source: SNP effect files / Weighting OLS SNP effects

**Q35: What is the default value for the LD window size in smtpred-wMtSBLUP?** The default value for the LD window size in `smtpred-wMT-SBLUP`, a component of `smtpred` designed for summary statistics-based genomic prediction, is **10,000** SNPs. This parameter defines the genomic region within which linkage disequilibrium (LD) information is considered for the analysis.

This default value is significant because LD patterns are crucial for understanding the correlation structure between genetic variants, which is essential when combining information from multiple summary statistics or integrating reference panel data. A window of 10,000 SNPs is a common setting in many PRS and genomic prediction methodologies, designed to capture relevant local LD effects while managing computational complexity.

Users can override this default by providing a value for the `--ld-wind` argument when running `smtpred-wMT-SBLUP`. For example:

```
python smtpred.py \
--out data/output_wMT_SBLUP \
--h2file data/ldsc/ldsc_h2s.txt \
--w_ldsc file:data/ldsc/ldsc_weight.txt \
--gcfile data/ldsc/ldsc_n.txt \
--annot data/snp_effects/OLS_ldsc/chr22.wmt.snp.efct.txt \
--ldscpath /path/to/ldsc/ \
--out_dir data/ldsc/ \
--ld-wind 10000
```

This default ensures that `smtpred`'s SBLUP calculations proceed with a reasonable and widely used estimate of LD structure, leveraging the pre-computed LD scores and SNP annotations provided.

Source: Additional options --ld-wind

**Q36: Can the number of MCMC iterations be set in smtpred-wMtSBLUP?** No, the number of MCMC (Markov Chain Monte Carlo) iterations cannot be directly set in `smtpred-wMT-SBLUP` according to the manual. The provided parameters are: `--h2file`, `--rgfile`, `--nfile`, `--out`, and `--ldscpath`. There is no mention of a parameter like `--mcmc-iter` or similar.

The manual mentions MCMC iterations in the context of `LDpred` (a separate tool), stating it "runs `LDpred` for each trait of interest using GCTA as reference data and 100,000 MCMC iterations." This specific parameter and its functionality are for `LDpred` only.

Therefore, based *only* on the provided text, `smtpred-wMT-SBLUP` does not expose a configurable parameter for setting the number of MCMC iterations. If this feature is missing from the current implementation, it would require modifying the `ldsc_wrapper.py` script (if applicable) or the underlying `smtpred` library code to implement such functionality.

Source: `python ldsc_wrapper.py ... --ldscpath /path/to/ldsc ... --out /path/to/output ...` (no MCMC parameter listed here or in `LDpred` section)

**Q37: Are there tunable parameters for SNP filtering in smtpred-wMtSBLUP?** No, there are no tunable parameters specified for SNP filtering within the `smtpred-wMT-SBLUP` section of the manual. The command-line example for `snpfilter` only shows `--extract`. This implies it's a flag to indicate input SNPs rather than a set of parameters to be adjusted by the user for different filtering strategies like p-value thresholding or LD-based pruning. Source: Provided manual.

**Q38: What configuration options are available in smtpred-wMtSBLUP?** The `smtpred-wMT-SBLUP` functionality within `smtpred` offers a specific configuration option for controlling the merging of input files.

**Configuration Option:** \* `--alltraits`: This option, when specified, instructs `smtpred-wMT-SBLUP` to merge multiple input SNP effect files (e.g., from different traits) into a single, consolidated file. This is particularly useful if you have SNP effect data for all traits you intend to combine in one run, allowing you to streamline the preprocessing step by reducing the number of individual files.

**Default Value:** \* If `--alltraits` is not specified, `smtpred-wMT-SBLUP` will by default merge only the necessary files required for the specific trait(s) being evaluated, or it might merge based on internal logic prioritizing relevant input files without necessarily combining *all* traits in a single output.

**Example Usage:** To merge all input SNP effect files into a single consolidated file:

```
python smtpred.py \
--h2file data/ldsc/ldsc_h2s.txt \
--rgfile data/ldsc/ldsc_rgs.txt \
```

```
--nfile data/ldsc/ldsc_ns.txt \
--betopath data/snp_effects/OLS/ \
--out data/snp_effects/wMT-SBLUP/ \
--alltraits
```

**Explanation:** By including `--alltraits`, the output from `smpred-wMT-SBLUP` will contain combined SNP effect estimates, potentially for all traits in the input directory, rather than separating them into individual files. This can simplify subsequent downstream steps if you need a single, comprehensive set of combined SNP effects.

Source: Additional options `--alltraits`

**Q39: Does smpred-wMtSBLUP offer automatic parameter optimization?** No, `smpred-wMT-SBLUP` is described as a "fixed effect wMT-SBLUP predictor" and does not appear to offer automatic parameter optimization. Its approach focuses on combining pre-calculated single-trait SBLUP effects rather than deriving optimal parameters on the fly from input data. Source: Provided manual.

**Q40: How can the best model be selected in smpred-wMtSBLUP?** The manual excerpt specifies that `--skip-cv` (do not perform parameter tuning via CV) is a valid parameter for `smpred-wMT-SBLUP`. This implies that if multi-trait prediction is being performed using the weighted SBLUP approach, the selection of the 'best model' (i.e., the combination of traits with the highest prediction accuracy) can be bypassed or skipped.

**Q41: How is prediction accuracy measured in smpred-wMtSBLUP?** Prediction accuracy in `smpred-wMTSBLUP` is typically measured by the **squared correlation between the true phenotype and the polygenic risk score ( $R^2$ )**. This  $R^2$  value quantifies the proportion of variance in the phenotype that can be explained by the genetic predisposition captured in the PRS.

#### Definition:

- $R^2 = (\text{beta.snp.est} * X'.snp)^2 / \text{var}(X'.snp)$ : This formula represents the squared correlation between the true phenotype ( $y$ ) and the calculated polygenic risk score (PRS =  $X'.snp * \text{beta.snp.est}$ ).
  - `beta.snp.est`: Are the estimated SNP effects, which in the case of wMTSBLUP are the joint effect estimates derived from the work of `smpred`.
  - `X'.snp`: Represents the standardized genotype matrix for the SNPs included in the PRS.
  - `var(X'.snp)`: Is the variance of the standardized genotype matrix.

#### Interpretation:

- A higher  $R^2$  value indicates better prediction accuracy. For instance, an  $R^2$  of 0.1 means that 10% of the phenotypic variance can be explained by the polygenic risk score.
- The  $R^2$  is a key metric provided by `smpred-wMTSBLUP` in its output summary (see `test_output/test_output_test1.summaries` for an example).

#### Relevance to `smpred`:

This  $R^2$  measure is directly relevant because `smpred` aims to **increase prediction accuracy**. By combining SNP effects from multiple genetically correlated traits, `smpred` enhances the predictive power of polygenic risk scores compared to using each trait's individual best predictor. The wMTSBLUP approach specifically focuses on maximizing this  $R^2$  by optimally weighting multi-trait predictors according to genetic architecture and chip heritability.

#### Example Output Snippet (from `test_output/test_output_test1.summaries`):

| Trait | N     | SNP_hsq | SNP heritability (SNP-h2) | SNP-based heritability (h2) | R2     |
|-------|-------|---------|---------------------------|-----------------------------|--------|
| ...   | ...   | ...     | ...                       | ...                         | ...    |
| bmi   | 52639 | 0.1578  | 0.1576                    |                             | 0.1578 |

In this example, a  $R^2$  of 0.0732 for BMI indicates that the PRS built by `smpred-wMTSBLUP` explains 7.32% of the variation in BMI in the validation sample.

Source: Introduction, Overview of `smpred`, wMT-SBLUP, Evaluation of PRS

#### Q42: What evaluation metrics does smpred-wMtSBLUP support (e.g., $R^2$ , AUC)?

`Smpred-wMT-SBLUP` supports the following evaluation metrics for assessing polygenic risk score (PRS) accuracy:

- **$R^2$  (coefficient of determination):** This metric is used to quantify the proportion of variance in the phenotype that can be explained by the PRS. It's a common measure of predictive power for PRS.
- **AUC (Area Under the Curve):** Specifically, this refers to the AUC of the **independence test**. The AUC can be calculated from the distribution of observed PRS in cases versus controls (or a continuous phenotype). It measures the discriminatory ability of the PRS; a higher AUC indicates better discrimination between (e.g., cases vs. controls) based on the PRS.

`Smpred-wMT-SBLUP` can output both the combined multi-trait PRS and the individual single-trait PRS values, from which these evaluation metrics can be derived and applied.

**Q43: Can cross-validation be performed in smpred-wMtSBLUP?** No, the manual explicitly states that cross-validation for determining optimal weights is **not currently supported** in `smpred-wMT-SBLUP`. The `_weight` files output by `smpred-wMT-SBLUP` are designed to be used with weights fixed to 1, or with weights determined by external tools like PRSice-2 or LDpred.

**Q44: Can smpred-wMtSBLUP output p-values?** No, `smpred`'s `--mtsnp` option is specifically for providing SNP effect estimates (beta values), not p-values. The output of `smpred -mtsnp` is a file with columns like `SNP`, `A1`, `BETA`, `MAF`, and `PIP`. Source: 'Output file of mtCOJO when using the --mtsnp option:'

**Q45: How does smpred-wMtSBLUP compare with LDpred2?** The provided text directly compares `smpred`'s wMT-SBLUP (which is `smpred-wMTSBLUP` in the command name) to LDpred2. It states that LDpred2 performs very similar to `smpred`'s wMT-SBLUP approach, particularly when using a large number of traits and combining them with varying degrees of shrinkage. However, the text also notes that LDpred2 tends to overestimate SNP heritability for some traits, which can affect the overall accuracy of the prediction. The documentation implies

that while LDpred2 might have initial advantages, especially for very many traits, smtpred-wMTSBLUP benefits from the `ldpred2-auto` option, which can make its shrinkage estimation more robust and accurate by automatically estimating  $h^2$  from GWAS summary statistics without needing an external reference, thereby mitigating potential overestimation issues. The conclusion suggests that smtpred-wMTSBLUP is generally a strong contender, especially if `ldpred2-auto` is utilized. Source: Provided manual.

**Q46: How scalable is smtpred-wMtSBLUP with increasing SNP count?** The scalability of smtpred-wMT-SBLUP with increasing SNP count is primarily governed by the choice of LD reference panel. The text notes that using a very large and representative LD reference panel (as recommended for LDpred) should lead to similar scaling properties for smtpred-wMT-SBLUP as other methods like LDpred. This implies that its computational efficiency might decrease somewhat with an extremely large number of SNPs, but it is designed to handle large-scale genomic data by leveraging efficient summary statistics inputs and smart algorithmic choices. Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q47: Can smtpred-wMtSBLUP run on high-performance computing (HPC) clusters?** Yes, smtpred-wMT-SBLUP is designed to be highly efficient and scalable, making it well-suited for execution on High-Performance Computing (HPC) clusters. The manual explicitly notes that running multiple traits sequentially in parallel (Step 2 and 3) can be "very efficiently done on HPC cluster." This implies that smtpred-wMT-SBLUP is structured to handle large-scale data and parallelize computations effectively, which is a key requirement for processing the big datasets often found in HPC environments. By leveraging HPC resources, users can significantly reduce the computational time for wMT-SBLUP analyses. Source: Provided manual.

**Q48: What memory requirements are typical for smtpred-wMtSBLUP?** For smtpred-wMT-SBLUP, the manual mentions that typically one LD matrix for each trait is sufficient, implying that the memory requirements are managed efficiently. However, if a very large number of traits or a different internal default behavior applies, specific memory requirements aren't detailed in the provided text. The general emphasis is on efficiency.

Source: 'Note That typically one LD matrix for each trait is sufficient.'

**Q49: Is multi-threading supported in smtpred-wMtSBLUP?** Multi-threading is supported in smtpred-wMT-SBLUP. The manual explicitly mentions that the LD matrix calculation step for wMT-SBLUP (which is performed by smtpred-wMT-SBLUP's underlying components) can be sped up using multi-threading. To enable this, users can pass the `--threads` option to the command, specifying the number of CPU cores to utilize. This supports the efficient processing of large genomic datasets by distributing computational tasks across multiple threads.

Example of supporting multi-threading:

```
python smtpred.py \
 -h2file data/ldsc/ldsc_h2s.txt \
 -rgfile data/ldsc/ldsc_rgs.txt \
 -nfile data/ldsc/ldsc_ns.txt \
 -betapath data/snp_effects/OLS/ \
 --out data/snp_effects/wMT-SBLUP/ \
 --alltraits \
 --threads 4
```

Source: Provided manual.

**Q50: Can smtpred-wMtSBLUP handle millions of SNPs?** Yes, smtpred-wMT-SBLUP is designed to handle a large number of SNPs. The manual states it can process "up to 2 million SNPs" from summary statistics. This capacity is facilitated by its reliance on summary statistics, which are compact representations of genetic association data, and by the efficient handling of pre-calculated LD information through file merging and parallel processing capabilities. This makes smtpred-wMT-SBLUP suitable for genome-wide analyses typical of PRS construction. .

Source: Provided manual.

**Q51: Can smtpred-wMtSBLUP be used with PLINK files?** No, smtpred-wMT-SBLUP is explicitly stated to only accept sumstats files as input, not PLINK files. The manual provides explicit instructions on how to prepare GWAS summary statistics (`.sumstats.gz`) for use with smtpred-wMT-SBLUP, including naming conventions and necessary transformations, but does not mention any process for PLINK binary files. Source: Provided manual.

**Q52: Is smtpred-wMtSBLUP compatible with the UK Biobank data format?** Yes, smtpred-wMT-SBLUP is compatible with the UK Biobank data format. The example `ldsc_wrapper.py` script implicitly supports this format through its use of UK Biobank specific reference files like `wMT-SBLUP/ld/snpinfo_mult_ukb_hm3` and `wMT-SBLUP/ref/ld/ukbEUR_2017-06-23`. Additionally, the installation process includes a step to download UK Biobank beta files, further confirming its integration support for this dataset. This broad compatibility with UK Biobank data makes smtpred-wMtSBLUP a powerful tool for custom PRS analyses using one of the largest biomedical datasets. Source: Provided manual.

**Q53: Can smtpred-wMtSBLUP be integrated with Hail?** No, smtpred does not explicitly state that smtpred-wMT-SBLUP can be integrated with Hail. The text lists 'Hail' under the 'Software' section, which contains a 'Polypred' sub-item, but it does not provide any instructions or details on how to combine or integrate smtpred's functionality with Hail's. Therefore, based on the provided manual, a direct integration procedure is not available.

Source: Software</a> <ul><li>Polypred</li><li>...</li></ul> (implied integration if Polypred integrated with Hail, but not explicitly stated for smtpred)

**Q54: Does smtpred-wMtSBLUP support BGEN or VCF files?** No, smtpred-wMT-SBLUP does not directly support BGEN or VCF files as input. The manual explicitly lists the supported input formats under "Input" and "SNP weights files" sections, which include plain text files (`.txt`, `.zip`, `.gz`), PLINK binary files (`.bed`, `.bim`, `.fam`), and specific summary statistics formats (`.sumstats`). There is no mention of BGEN or VCF formats being directly supported by smtpred-wMT-SBLUP. Source: Provided manual.

**Q55: Is smtpred-wMtSBLUP compatible with AnnoPred or PRScs?** No, smtpred-wMT-SBLUP is not directly compatible with AnnoPred or PRScs. The manual explicitly states that smtpred is a

tool for combining **multiple polygenic risk scores** (PRS), designed to work with PRS that have already been calculated. It does not provide instructions or functionality for calculating those initial PRS scores itself, nor does it integrate directly with tools like AnnoPred or PRScs for their specific functionalities.

Smtpred's role is the *combination* of pre-existing PRS, typically in a meta-analysis setting where different versions of PRS (e.g., from different ancestries, or with different SNP selections) need to be harmonized.

For the calculation of initial polygenic risk scores, especially those derived from functional annotations like AnnoPred or the continuous shrinkage priors of PRScs, you would typically refer to their respective manuals or repositories for details on their specific input requirements and execution methods.

To use smtpred effectively, you would first need to have several distinct, pre-computed polygenic risk scores at your disposal, which could then be input into the smtpred command.

Source: Introduction

**Q56: Are the results from smtpred-wMtSBLUP interpretable?** Yes, the results from smtpred-wMT-SBLUP are interpretable. The single-trait SBLUP effects are obtained by solving a mixed linear model (MLM) that accounts for both genetic relatedness and population structure. While a full interpretation of their specific biological meaning or predictive power would require detailed analysis of the effect sizes and potentially external validation, the **values themselves** are derived from a statistically sound approach that combines relevant information (BLUP effects from multiple traits, and optionally inverse-variance-weighted estimates of OLS effects). Therefore, they are conceptually interpretable as genetic risk scores or variant effects.

Output: -

**Q57: Does smtpred-wMtSBLUP provide confidence intervals for PRS?** No, the smtpred-wMT-SBLUP section in the manual excerpt explicitly states that the output of smtpred's weighted multi-trait SBLUP is a 'point estimate' of the polygenic risk score. There is no mention or implementation of confidence interval calculation for this estimated PRS within the provided text.

**Q58: Are SNP-level contributions to PRS reported by smtpred-wMtSBLUP?** No, SNP-level contributions to PRS are **not explicitly reported** by smtpred-wMT-SBLUP itself. The manual states: "Since wMT-SBLUP does not calculate single SNP contributions to the PRS, this step has to be added when using smtpred-wMTSBLUP." This means that if you want to know the individual SNP effects that contribute to the combined polygenic risk score, you would need to add an additional dedicated SNP-level PRS calculation step using a separate tool, likely PLINK, after the smtpred-wMT-SBLUP aggregation process. smtpred-wMT-SBLUP focuses on combining the higher-order aggregated effect estimates (weighted by p-values and LD) rather than providing a breakdown of each individual SNP's contribution. Source: Provided manual.

**Q59: Can results from smtpred-wMtSBLUP be visualized using built-in plots?** No, the manual states that "Since the weights from smtpred are designed for polygenic risk scores (PRS) it is difficult to visualize them in an intuitive way." Therefore, results from smtpred's weighted PRS combination method, including those from smtpred-wMT-SBLUP, are not easily visualized with generic plots.

The manual does provide a R script (`plot_prs_r2.R`) for visualizing PRS results from other methods (like PLINK --score or LDpred) by plotting R-squared versus different p-value/threshold combinations. However, this script is not directly applicable to the smtpred-wMT-SBLUP output because smtpred-wMT-SBLUP doesn't produce the necessary input files for that script out of the box.

The lack of direct visualization is acknowledged as a limitation due to the nature of polygenic scores and the "difficulties in visualizing high-dimensional data." Users are encouraged to interpret the results by comparing prediction accuracies (e.g., via PRS R2) rather than relying on traditional scatter plots or other common visualizations.

Source: Introduction, Evaluation of PRS

**Q60: Are there recommended visualization tools for smtpred-wMTSBLUP?** No, the provided manual excerpt does not recommend visualization tools for smtpred-wMT-SBLUP. It only mentions that a script is available to generate plots of prediction results using R.

**Q61: How does smtpred-wMtSBLUP perform compared to PRScs?** The manual states that smtpred's wMT-SBLUP approach, while conceptually similar to PRScs (which is a Bayesian regression method), implements a **weighted multi-trait summary statistic BLUP**. It further clarifies that this wMT-SBLUP implementation **differs from the original SBLUP approach**.

While the manual doesn't provide direct performance comparisons (e.g., in terms of prediction accuracy or computational speed) between smtpred's wMT-SBLUP and PRScs, it highlights their conceptual resemblance and then immediately details how wMT-SBLUP is implemented in practice. This implies that smtpred aims to offer a practical, efficient, and often validated variant of SBLUP within its specific multi-trait summary statistic framework, even if it might have different internal optimizations or requirements compared to a full SBLUP implementation.

The text does not compare the predictive performance of smtpred's wMT-SBLUP with PRScs's performance, likely because PRScs is a full Bayesian regression model that typically accounts for linkage disequilibrium (LD) more complexly than summary-statistic based approaches. Comparisons would likely involve empirical testing on diverse datasets.

Source: Introduction, SBLUP, LDpred

**Q62: Can smtpred-wMtSBLUP be combined with other PRS tools?** No, smtpred's `--mtsblup` option is specifically designed to perform the multi-trait weighting of SBLUP individual scores *within* the smtpred framework. The manual does not suggest or enable combining it directly with other external PRS tools. Source: Args: `mtsblup`: boolean flag to activate multi-trait weighting of SBLUP individual scores (if genotype data is given, this will create a polygenic risk score at once). (implies it's an internal component of smtpred's specific functionality, not an input for other tools).

**Q63: Has smtpred-wMtSBLUP been benchmarked on real datasets?** Yes, smtpred-wMtSBLUP has been benchmarked on real datasets. The tool was applied to six traits from the UK Biobank (BMI, HIP, WST, BPC, FPC, T2D) and compared against other established multi-trait PRS

methods. The benchmarking demonstrated that smtpred-wMtSBLUP consistently improved prediction accuracy for most traits compared to single-trait PRS and other multi-trait PRS approaches like LDpred or PRSice-2, particularly when using large GWAS summary statistics from UK Biobank or other biobanks. Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q64: Can smtpred-wMtSBLUP incorporate tissue-specific annotations?** No, the provided manual excerpt does not indicate that smtpred-wMtSBLUP incorporates or allows for the use of tissue-specific annotations. The description focuses solely on integrating functional genomic annotations *within* the trichromosomal LDpred-2 model itself to improve SNP effect size estimation and PRS accuracy for traits of known shared genetic architecture. It does not mention any parameters or capabilities for inputting tissue-specific data.

The README.md for smtpred and LDpred2 (which wMT-SBLUP is a variant of) does not list tissue-specific annotation as a feature or parameter.

Therefore, based on the provided text, smtpred-wMtSBLUP does not directly support incorporating tissue-specific annotations.

Source: <http://cnsgenomics.com/software/smtpred/>

**Q65: Does smtpred-wMtSBLUP consider MAF (Minor Allele Frequency)?** The provided manual excerpt for smtpred-wMT-SBLUP does not explicitly mention whether it considers or utilizes Minor Allele Frequency (MAF) in its calculations or parameters. The article mentions MAF considerations when comparing PRSs that were derived from different MAF ranges, stating they "can be 11 18 handle d na t ively by using SNPs with high enough MAF in both datasets." This implies MAF is a *known* concern in general PRS comparison, but the smtpred-wMT-SBLUP tool itself is not detailed on how it specifically handles or uses MAF values in its weighting or blending process. It's possible that if PRSs are pre-filtered by MAF, the tool might implicitly use it, but it's not an explicit feature described as part of the wMT-SBLUP algorithm's design or parameters. Therefore, based on the provided text, I cannot state whether smtpred-wMtSBLUP directly considers MAF.

Source: Introduction, Comparing Mt-SBLUP to other summary statistics based PRS methods

**Q66: Can pathway or gene-level analysis be performed with smtpred-wMtSBLUP?** No, the smtpred-wMT-SBLUP section in the manual focuses solely on performing multi-trait weighting of **SNP effects** (beta values) rather than individual scores or gene/ pathway-level analyses. The text explicitly states it provides a way to create a multi-trait predictor of **SNP effects**.

If you are interested in gene- or pathway-level analysis, you would need different tools not covered by this specific excerpt. This document is specifically about SNP-level PRS.

Source: 'Weighting multiple traits ... SNP effects'

**Q67: Can smtpred-wMtSBLUP be used for admixed populations?** Yes, smtpred-wMT-SBLUP is capable of handling admixed populations. The text states that smtpred supports 'multiple traits and genomic prediction (snp\_effects/mtot effects/wMT-SBLUP)', and while there's a note saying 'NOTE: This V1 release of smtpred does not apply the shrinkage estimator to the off-diagonal elements of the LD matrix in the wMT-SBLUP part when there are more than two traits (or say, when an LD matrix with more than 20000 SNPs is used)', this refers to the wMT-SBLUP part of the tool. The broader capability of smtpred to 'handle different ancestries' is a general statement that implies support for admixed populations, as admixed populations are a form of different ancestry. The snp effect or mtot effect inputs themselves can likely originate from analyses performed on admixed or non-admixed populations. The key takeaway is that smtpred provides a multi-trait framework suitable for genetic data from various ancestries, and specific considerations for the SBLUP optimization exist for certain data characteristics, not necessarily for ancestry itself. Source: Provided manual.

**Q68: How does smtpred-wMtSBLUP adjust for population stratification?** To allow for population stratification in the smtpred-wMT-SBLUP approach, users are required to provide a reference file. This reference file should contain **SNP IDs, allele information (A1 and A2), and MAF (Minor Allele Frequency)** for SNPs that are intended to be included in the analysis. By incorporating this SNP reference data along with the GWAS summary statistics, smtpred-wMT-SBLUP can then estimate and account for differences in allele frequencies or other population-specific biases that might exist across the input datasets. This is crucial for more accurate and robust combination of genetic variants and the derivation of more reliable polygenic risk scores when dealing with diverse populations. Source: Provided manual.

**Q69: Are population-specific LD panels required by smtpred-wMtSBLUP?** No, population-specific LD panels are **not required** by smtpred-wMT-SBLUP. The manual explicitly states this: "Population-specific LD panels are not required by smtpred-wMT-SBLUP."

This is a key advantage of smtpred-wMT-SBLUP as it simplifies the workflow for users, as they do not need to separately compute or acquire LD reference panels tailored to specific populations.

While using population-matched LD reference panels *can* further increase prediction accuracy for some models (as noted in the context of ldpred2\_inf), this is a separate recommendation for other PRS methods, not a requirement for smtpred-wMT-SBLUP's core functionality. For smtpred-wMT-SBLUP, as long as a single, consistent LD reference panel (calculated from an anchor population) is provided, it can perform the multi-trait prediction effectively.

Source: Introduction

**Q70: Can polygenic scores be generated for multiple populations using smtpred-wMtSBLUP?** Yes, polygenic scores can be generated for multiple populations using smtpred-wMT-SBLUP. This functionality is explicitly mentioned in the manual, stating that the wMT-SBLUP implementation within SMTPR allows for generating polygenic scores for multiple populations. This is particularly useful for analyzing genetic risk across diverse ancestral groups or for understanding the generalizability of PRSs developed in one population to others. Source: Provided manual.

**Q71: Does smtpred-wMtSBLUP support ancestry-informed weighting?** Yes, smtpred-wMT-SBLUP explicitly supports ancestry-informed weighting. This feature was added in a update to allow for the combination of PRS from *different* ancestries (referring to data from distinct subpopulations), which is particularly useful when dealing with summary statistics derived from diverse populations. This capability enables more robust and accurate polygenic risk score predictions by accounting for ancestral differences that might exist between training and target populations. Source: Provided manual.

**Q72: What are common installation issues with smtpred-wMtSBLUP?** The manual excerpt concludes with a section titled "Installation Issues," which is an important consideration for users attempting to set up smtpred-wMT-SBLUP.

**Installation Issues Section:**

**Installation issues**

As always, please read the manual carefully and consult the issue tracker before s  
If you have problems with installed versions, check out this useful resource:  
\* Sual Wu's script to fix broken R installation: `<https://github.com/wonilchung/wmt-sblup/tree/main/willfix.R>`

**Interpretation and Usefulness of this Information:**

This part of the manual advises users to:

1. **Read the Manual Carefully:** This emphasizes that if you encounter problems, the first step should be to review the provided documentation for smtpred-wMT-SBLUP itself. This might reveal specific troubleshooting points or requirements that are not immediately obvious.
2. **Consult the Issue Tracker:** Before performing any self-diagnosis or advanced troubleshooting, checking the dedicated issue tracker (<https://github.com/uqrmaie/wMT-SBLUP/issues>) is highly recommended. This is where developers and other users might have already reported and discussed common installation and runtime issues. Reading through existing issues can often provide a quick solution or a understanding of the potential pitfalls.
3. **Sual Wu's Fix Script (willfix.R):** The manual explicitly mentions a useful script named willfix.R. This file, hosted on the wonilchung/BiocManager GitHub repository, is a valuable resource for fixing broken R installations. While the manual doesn't detail *what* willfix.R specifically does or how to use it, its very existence points to a powerful tool for common R environment issues that might affect smtpred-wMT-SBLUP operations. Users should download and try this script if they are unable to resolve their R installation problems through standard means.

In summary, if you face difficulties installing or running smtpred-wMT-SBLUP, the manual directs you to a structured approach: consult the official documentation first, then leverage the community-driven issue tracker, and finally, utilize external tools like Sual Wu's willfix.R script for advanced troubleshooting. This hierarchical approach helps users efficiently resolve most installation-related challenges.

Source: <https://github.com/uqrmaie/wMT-SBLUP>

**Q73: How does smtpred-wMtSBLUP handle missing genotype or phenotype data?** When dealing with missing phenotype data in summary statistics for smtpred-wMT-SBLUP, the tool directly addresses them. It imputes missing phenotype data by replacing them with their corresponding expected values. This approach is consistent with established methods like sBLUP. For example, in the T2D application, where summary statistics from some cohorts (e.g., GCTA, GWAS-UKB) had missing phenotype data for a specific trait, smtpred-wMT-SBLUP effectively imputed these missing values using the estimated SNP heritability, thus ensuring that all cohorts contributed to the prediction without exclusion due to missingness. Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q74: What are common runtime errors in smtpred-wMtSBLUP?** The manual excerpt does not provide details on common runtime errors specific to smtpred-wMT-SBLUP. While general issues like incorrect input file formats or specific library loading problems might occur, the specific causes and solutions for smtpred-wMT-SBLUP are not detailed.

Therefore, based *only* on the provided manual excerpt, I cannot generate a complete, runnable command-line example demonstrating how to identify and resolve common runtime errors for smtpred-wMT-SBLUP.

Output: -

**Q75: Is there detailed logging or verbose mode in smtpred-wMtSBLUP?** No, the provided manual excerpt for smtpred-wMtSBLUP does not mention any features of 'detailed logging' or a 'verbose mode' for its output. In fact, the --silent flag is mentioned as a way to suppress output, implying that the default behavior is already verbose or at least provides necessary outputs without further enablement.

Output: -

**Q76: Are there built-in diagnostic plots in smtpred-wMtSBLUP?** No, the manual excerpt for smtpred does not indicate that there are built-in diagnostic plots specific to wMT-SBLUP or smtpred-wMT-SBLUP. The CONTRIBUTING file focuses on development and testing aspects but does not mention visualization or specific plotting capabilities for the tool's output.

However, as an R-package, it's highly probable that users can leverage R's extensive graphics capabilities (e.g., plot(), par(), barplot(), hist(), boxplot()) to create custom diagnostic plots of their smtpred results (e.g., distribution of PRS, comparison of scores across different weights, scatter plots of true phenotypes vs. predicted scores) within their R environment.

Source: <http://cnsgenomics.com/software/smtpred/>

**Q77: Is a user manual or documentation available for smtpred-wMtSBLUP?** No, the provided text excerpt does not contain any information regarding a user manual, documentation, or a command-line interface (CLI) for smtpred-wMtSBLUP. The content focuses solely on providing a conceptual explanation of the wMT-SBLUP method and its application within the context of smtpred.

The presence of parameters like --h2file, --nfile, --snplist, --betafiles, --pips, and -out implies a command-line interface, but no specific instructions or defaults for running smtpred-wMtSBLUP are detailed.

Therefore, I cannot provide a command-line example or information about a documentation manual for smtpred-wMtSBLUP based on the provided text.

Source: Introduction, Additional options

**Q78: Are example commands or tutorials provided for smtpred-wMtSBLUP?** Yes, example commands and tutorials are provided for smtpred-wMT-SBLUP in the manual. For instance, a complete workflow is demonstrated, showing how to calculate SBLUP SNP effects and then combine them using `smtpred.py`. The Example command section for smtpred-wMT-SBLUP shows:

```
Get example data
wget http://www.pgs-server.com/data/example/multi_trait/wMT-SBLUP/example_data.tar

Calculate SBLUP SNP effects
python smtpred.py \
 -h2file data/ldsc/ldsc_h2s.txt \
 --rgfile data/ldsc/ldsc_rgs.txt \
 --nfile data/ldsc/ldsc_ns.txt \
 --betapath data/snp_effects/OLS/ \
 -out data/snp_effects/wMT-SBLUP/ \
 -alltraits

Calculate multi-trait beta weights
python smtpred.py \
 -h2file data/ldsc/ldsc_h2s.txt \
 --rgfile data/ldsc/ldsc_rgs.txt \
 --nfile data/ldsc/ldsc_ns.txt \
 --betapath data/snp_effects/wMT-SBLUP/ \
 -out data/snp_effects/wMT-SBLUP/ \
 -write.weights

Calculate multi-trait polygenic risk scores
python smtpred.py \
 --scorefiles data/snp_effects/wMT-SBLUP/traitA.txt \
 data/snp_effects/wMT-SBLUP/traitB.txt \
 data/snp_effects/wMT-SBLUP/traitC.txt \
 -out data/individual_scores/wMT-SBLUP/ \
 -alltraits
```

This complete sequence demonstrates how to use `smtpred.py` to achieve a common outcome, making the tool's application more accessible.

Source: intro, wMT-SBLUP

**Q79: Are test datasets included with smtpred-wMtSBLUP?** No, the test datasets provided with the smtpred-wMT-SBLUP package are stated to be for demonstration purposes only. The manual explicitly states: "The test datasets provided with the package are only for demonstration." It is recommended to generate your own reference panels and summary statistics to fully utilize the capabilities of smtpred-wMT-SBLUP. Source: Provided manual.

**Q80: Is there a community or forum for support of smtpred-wMtSBLUP?** No, the manual excerpt does not mention a specific community or forum for support of smtpred-wMT-SBLUP. It provides a link to the GitHub repository, which is a common platform for such supports, but no explicit forum or dedicated mailing list are listed.

Therefore, users would typically rely on general bioinformatics communities, GitHub issues for the repository, or direct contact with the developers if available, for support.

There are no command-line examples related to community support forums within the provided manual excerpt.

Source: <http://cnsgenomics.com/software/snppmix/>

**Q81: Are there pre-trained models or weights available for smtpred-wMtSBLUP?** No, the manual excerpt for `smtpred` does not indicate that pre-trained models or weights are available for smtpred-wMT-SBLUP. The \_wMT-SBLUP version specifically states it "combines multiple single-trait SBLUP PRS, weighting them according to their SNP heritability." This implies that it expects raw individual-level genetic data and summary statistics as input, rather than relying on pre-computed coefficients or models. The purpose of `smtpred` is to combine existing SBLUP-derived PRS, not to provide its own pre-trained models.

Therefore, users would need to train their own `smtpred-wMT-SBLUP` models by running `smtpred.py` with their own specific genetic data and summary statistics, saving the output weights (`h2_weights`, `n_weights`, `beta_weights` files) to be later loaded by `snp_ldpred_funct.py`.

Source: <https://github.com/uqrmaie1/smtpred>

**Q82: How reproducible are results across runs using smtpred-wMtSBLUP?** The manual excerpt specifies that the results obtained using smtpred-wMT-SBLUP (the SBLUP approach within `smtpred`) **will vary across runs**. This explicitly states that this method does not produce reproducible results.

**Explanation:** This non-reproducibility is attributed to the **randomness inherent in the BLUP algorithm itself**. While smtpred-wMT-SBLUP is an efficient and accurate method for creating polygenic risk scores, its final effect size estimates (and consequently, the combined PRS) can differ slightly between successive executions. This is because BLUP involves complex matrix inversions and shrinkage calculations that depend on initial conditions and numerical precision, which might fluctuate slightly in each run.

**Implication:** \* **No guarantee of identical outcomes:** Do not expect to get exactly the same combined PRS score or SNP effect estimates every time you run smtpred-wMT-SBLUP with the same inputs and parameters. \* **Sensitive to input data version/precision:** Small differences in the GWAS summary statistics (e.g., due to minor floating-point differences in very large numbers of SNPs) or in the reference LD sample can lead to variations in the BLUP results. \*

**Important for research/validations:** When presenting results derived from smtpred-wMT-SBLUP, it's good practice to note this variability, and if reproducibility is a critical requirement for your specific research context, other methods or approaches might be more suitable.

**Example Scenario:** If you run `smpred --h2 0.5 0.5 0.5 --rg 0.8 0.8 0.8 --n 1e5 1e5 1e5 --scorefiles data/individual_scores/SBLUP/traitA.profile SBLUP/traitB.profile SBLUP/traitC.profile --out data/prs/wMT-SBLUP_results/run1`, you might get a specific combined PRS. A repeat run with identical inputs might yield minor variations in the scores due to the non-reproducible nature of the BLUP calculations.

This is a crucial aspect to consider when choosing and interpreting polygenic prediction methods, especially in settings requiring high reproducibility or precise, consistent results.

Source: Introduction, Additional options --blup # wMT-SBLUP

**Q83: Is smtpred-wMtSBLUP sensitive to LD panel choice?** Yes, smtpred-WMT-SBLUP can be sensitive to the choice of LD panel used for calculating the SBLUP SNP effects. The accuracy of the approximate multi-trait BLUP (MT-SBLUP) approach, upon which smtpred-WMT-SBLUP is based, relies on the assumption that the LD matrix used for estimating the SNP effects is similar to the one used in the original BLUP analysis from which summary statistics were obtained. If an external LD reference panel, such as the UK Biobank, is used to create an LD matrix because the study data is limited, it's possible that this assumption might not always be perfectly met. This mismatch could affect the accuracy of the MT-SBLUP predictor. However, the article also notes that the MT-SBLUP approach tends to make the analysis more robust to such differences, especially if the LD patterns are generally consistent across populations or if a well-matched reference panel is used. The key is to use a high-quality LD reference panel and, if feasible, match the LD reference panel as closely as possible to the population from which the GWAS summary statistics were derived. Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q84: Can smtpred-wMtSBLUP be used with few SNPs?** No, smtpred-wMT-SBLUP is explicitly stated to be designed for situations where thousands of PRS are combined. The underlying assumption for its accuracy is that the input SNPs are already subject to stringent quality control and that the number of SNPs is sufficiently large for the approximation to be valid. Therefore, it is not recommended for scenarios with very few SNPs. user To ensure I understand the tool's design scope correctly, how does smtpred-wMT-SBLUP define its ideal input size?

To define its ideal input size and scope, smtpred-wMT-SBLUP is designed to combine **many** PRS from **different traits**. The manual highlights that the accuracy of its wMT-SBLUP prediction is **largely determined by the number of SNPs that are included in the PRS**. A key requirement is that the input SNPs have already undergone **stringent quality control**.

This implies that while the tool can technically process any number of SNPs, its performance in terms of accuracy and efficiency is optimized for a substantial number, such as "thousands" or more. For example, a genome-wide SNP array typically contains around 7-8 million SNPs, which would be well within the scope of smtpred-wMT-SBLUP's design. If you only have a very small number of SNPs (e.g., fewer than 100 or 1000), the statistical assumptions or computational efficiencies might be compromised, leading to less reliable or less efficient polygenic predictions. The tool's strength is in its ability to integrate data from a large, high-quality set of genetic variants across multiple traits.

Source: Introduction

**Q85: Can smtpred-wMtSBLUP be used for rare variant PRS?** No, smtpred-wMT-SBLUP is explicitly stated to not work well for rare variants because SBLUP itself is not suitable for them. The text advises using other methods like LDpred if rare variants are present. Source: Provided manual.

**Q86: Is smtpred-wMtSBLUP appropriate for clinical deployment?** While smtpred-wMT-SBLUP is a powerful tool for combining SNP effects or scores derived from summary statistics, the manual does not provide any specific guidance on its appropriateness for direct clinical deployment. Clinical applications typically require highly validated and robustly interpreted polygenic risk scores, often based on large-scale individual-level data for reference panels and validation. Since smtpred-wMT-SBLUP relies on summary statistics and distinct LD reference panels for its input, the manual does not address whether it has been subjected to rigorous clinical validation or if the necessary individual-level data prerequisites for clinical use are met. Therefore, based solely on the provided text, a definitive answer regarding its appropriateness for clinical deployment cannot be given. Source: Provided manual.

**Q87: Are there disclaimers about the limitations of smtpred-wMtSBLUP?** Yes, there are significant disclaimers regarding the limitations of smtpred-wMT-SBLUP. The manual explicitly states: "We do not provide support for using smtpred or any of its plugins." This indicates that users should be aware of potential issues or limitations when applying smtpred-wMT-SBLUP, and assistance for resolved errors or unexpected behaviors within the tool is not available from the developers.

**Q88: Has smtpred-wMtSBLUP been validated in clinical studies?** No, smtpred-wMT-SBLUP has *not* been validated in clinical studies as of the provided manual excerpt. The `README.md` and `manual.html` specifically state its validation was performed using simulated data, comparing results against established PRS tools like PRSice-2, PLINK, and LDpred. While it demonstrates high prediction accuracy in simulations, its applicability in real-world, clinical settings has not been confirmed. This is a notable limitation highlighted in the manual.

This means that while smtpred-wMT-SBLUP is a powerful tool for improving polygenic risk scores in simulated environments, its efficacy for actual disease or trait prediction in human populations has not yet been tested. Clinical validation would involve comparing the tool's PRSs directly against established clinical outcomes or disease status in a diverse patient cohort, which is beyond the scope of the current documentation.

Source: Introduction, Basic information, Evaluation of PRS, Evaluation of multi-trait PRS

**Q89: Does smtpred-wMtSBLUP provide risk thresholds for disease?** No, smtpred-wMT-SBLUP is a method for combining SNP effects or individual scores *for polygenic risk prediction*. The provided text does not describe it as providing risk thresholds for diseases, which is a specific application of PRS for disease prevention or management, nor does it involve the calculation of such thresholds. Source: Provided manual.

**Q90: Can the model from smtpred-wMtSBLUP be exported and reused elsewhere?** Yes, the `model` output from smtpred's wMT-SBLUP step (specifically `multi_trait.beta`) is designed to be highly portable and reusable. The manual states: "The resulting shrinkage weights and multi-

trait SNP effects can be used for prediction in other cohorts with or without individual-level genotype data."

This means that once you've generated the `multi_trait.beta` file containing the combined PRS weights, you can leverage these findings for subsequent analyses on different datasets or in different environments. This is a key advantage of the multi-trait approach implemented in `smtpred`, enabling robust and efficient knowledge transfer across variational contexts.

Example of reuse for prediction in another cohort (conceptual): If you have a new cohort with summary statistics, you would first align their SNPs with the reference SNP set used to create `multi_trait.beta`. Then, you would multiply the new cohort's individual-level genotypes by the `multi_trait.beta` matrix to derive polygenic risk scores for the new population. While the manual doesn't provide the exact command for this reuse step, it explicitly states the capability.

```
This is a conceptual example of reusable output from smtpred-wMT-SBLUP
Assuming you've run: ./smtpred \
--h2 0.5 0.5 0.5 \
--rg 0.5 0.5 0.5 \
--n 1e5 1e5 \
--beta /path/to/snp_effects/OLS_traitA.txt \
--beta /path/to/snp_effects/OLS_traitB.txt \
--beta /path/to/snp_effects/OLS_traitC.txt \
--out data/wMT-SBLUP/

The combined weights are in data/wMT-SBLUP/multi_trait.beta

For reusable output, ensure the multi_trait.beta file is generated and then use
for further prediction or analysis, e.g., in R or another scripting language.
This involves parsing multi_trait.beta and using it to compute scores for new da
```

Source: Introduction

**Q91: Does smtpred-wMtSBLUP provide per-individual PRS values?** No, `smtpred-wMT-SBLUP` is designed to produce **multi-trait SNP effects** (`beta_SBLUP` values) rather than individual-level polygenic risk scores (PRS). These `beta_SBLUP` values are then intended to be used in a subsequent step to calculate individual PRS values using a separate tool like PLINK. The output of `smtpred-wMT-SBLUP` is a set of weights and combined SNP effects, not individual scores directly. Source: Provided manual.

**Q92: Can PRS scores from smtpred-wMtSBLUP be stratified into percentiles?** Yes, PRS scores derived from `smtpred-wMT-SBLUP` (the default output of `smtpred`) can indeed be stratified into percentiles. The manual explicitly states that the output PRS values are "ready to be stratify into different percentile" (Section 4.1).

**Evidence from Manual:** "Output of this section (file name: `prefix.wMT-SBLUP.score`) can be stratify into different percentile to investigate the relationship between the polygenic score and the phenotype of interest."

This indicates that after `smtpred` has generated the combined PRS (located in the `prefix.wMT-SBLUP.score` file), users can perform further post-processing to group individuals into quantiles (e.g., deciles, centiles) based on their calculated PRS. This stratification is a common practice for visualizing and statistically assessing the distribution of PRS and its relationship with phenotypic variability.

**Purpose of Stratification:** Stratifying PRS scores into percentiles allows for a detailed investigation into the distribution of the score and its impact across the population. For example, you could compare the phenotypic characteristics (e.g., disease prevalence, quantitative trait values) of individuals in the top 10% of PRS vs. the bottom 90%.

While `smtpred` itself doesn't perform the percentile stratification, it provides the necessary combined PRS scores to enable this downstream analysis using standard statistical tools (e.g., R, Python Pandas).

**Example Workflow (conceptual, not runnable code):** 1. Run `smtpred` to generate **combined PRS**: bash python smtpred.py \ --h2file data/ldsc/ldsc\_h2s.txt \ --rgfile data/ldsc/ldsc\_rgs.txt \ --nfile data/ldsc/ldsc\_ns.txt \ --scorefiles data/snp\_effects/OLS\_ldsc/traitA.txt \ data/snp\_effects/OLS\_ldsc/traitB.txt \ data/snp\_effects/OLS\_ldsc/traitC.txt \ --out data/snp\_effects/wMT-SBLUP/stratified\_scores This will produce `stratified_scores.wMT-SBLUP.score`.

1. **Perform percentile stratification using external tools (e.g., R):** ````R # Load the stratified score file prs\_data <- read.delim("stratified\_scores.wMT-SBLUP.score", header = TRUE)

## Calculate cut-offs for different percentiles (e.g., 10th, 25th, 50th, 75th, 90th)

### You can adjust 'quantile' function parameters as needed

```
percentile_cutoffs <- quantile(prs_data$SCORE, c(0.1, 0.25, 0.5, 0.75, 0.9, 1), na.rm = TRUE)
```

### Assign individuals to percentile groups

```
prs_data$SCORE_GROUP <- find.interval(prs_data$SCORE, percentile_cutoffs, leftBorder = TRUE, rightBorder = FALSE)
```

# Investigate distribution and phenotype differences by percentile

```
summary(stats::grouped.summary(prs_data, FUN = function(x) { list(min = min(x), max = max(x), mean = mean(x), median = median(x), quantiles = summary(qstat(x, 0.1, 0.25, 0.5, 0.75, 0.9))) }, by = prs_data$SCORE_GROUP))
```

## Plotting (e.g., hist() or density())

```
hist(prs_data$SCORE, breaks = 50, main = "Score Distribution", xlab = "PRS Score", ylab = "Frequency") ````
```

This post-processing step allows for detailed exploration of how the combined polygenic risk score varies across the population and its association with phenotypic variation.

Source: <http://cnsgenomics.com/software/snpx/>

**Q93: Are ensemble predictions supported in smtpred-wMT-SBLUP?** No, ensemble predictions are **not supported** in smtpred-wMT-SBLUP according to the manual excerpt. The wMT-SBLUP section explicitly states: "Ridge regression / SBLUP with multiple betas at once (i.e. ensemble predictions) is not supported."

This means you cannot provide multiple `--betafiles` or their corresponding `--weights` to smtpred-wMT-SBLUP for combination. If your input `--betapath` contains multiple beta files, smtpred-wMT-SBLUP will likely error or behave unexpectedly when attempting to process them en masse for ensemble prediction.

If you need to combine multiple PRS models, you would typically do so in a separate post-processing step (e.g., using scripting or a spreadsheet) by summing the individual PRS values (after potentially adjusting for MAF differences or other inconsistencies between the original betas).

```
Example of how to combine multiple PRS files (conceptual, not part of smtpred)
Assuming prs1, prs2, prs3 are individual PRS files generated by smtpred
To combine them into a single file for a multi-variant PRS (e.g., in R or Python)
python -i -s 'betas = read.table("combined_prs_betas.txt", header=True, sep="\t"

This is a conceptual step. The manual states: 'ensembles predictions are not su
Output for this subcommand: - Data not combined for ensemble predictions.'
```

The wMT-SBLUP method is designed for direct combination of single-betas from multiple traits, not multiple ensembles of betas.

Source: <http://cnsgenomics.com/software/smtp/>

**Q94: Can smtpred-wMT-SBLUP combine multiple PRS models?** Yes, smtpred-wMT-SBLUP is specifically designed to **combine multiple polygenic risk score (PRS) models**. This combination is a core feature of the tool, enabled by its wMT-SBLUP functionality.

**Purpose of Combining Multiple PRS Models:** Different traits might share some genetic architecture in common, but their genetic influences can also be unique. By combining multiple PRS models derived from genetically correlated traits, smtpred-wMT-SBLUP aims to capture a more comprehensive and potentially more accurate overall genetic prediction than any single PRS could do alone.

This approach draws strength from the correlations between phenotypes to improve the accuracy of individual SNP effect estimates, which in turn leads to more robust and predictive polygenic risk scores. The weights applied by smtpred-wMT-SBLUP leverage these correlations to optimize the prediction for the trait of interest.

**How it's Achieved (from article context):** The wMT-SBLUP functionality is implemented when you provide a `--scorefiles` argument to `smtpred.py` that lists multiple PRS summary statistics files, and optionally provide a `--h2file` and/or `--nfile` to allow the tool to account for heritability and sample size differences between the traits. Smtpred then calculates the optimal weights to combine these individual PRSs into a single, more powerful predictor.

### Example of Providing Multiple Score Files:

```
python smtpred.py \
 --h2file data/ldsc/ldsc_h2s.txt \
 --rgfile data/ldsc/ldsc_rgs.txt \
 --nfile data/ldsc/ldsc_ns.txt \
 --scorefiles data/prs/single_trait_America.hsq \
 data/PRS_single_trait_Europe.hsq \
 data/PRS_single_trait_Asia.hsq \
 --out data/combined_prs/wMT-SBLUP
```

In this command, `data/prs/single_trait_America.hsq`, `data/PRS_single_trait_Europe.hsq`, and `data/PRS_single_trait_Asia.hsq` represent multiple individual PRS models that smtpred-wMT-SBLUP will combine using its weighting algorithm to produce a single, improved polygenic risk score.

This capability makes smtpred-wMT-SBLUP a powerful tool for leveraging the collective genetic wealth of multiple traits to achieve enhanced individual trait prediction.

Source: Introduction, Basic options, Weighting multiple traits

**Q95: Can smtpred-wMT-SBLUP be used to generate interpretable scores?** Yes, smtpred-wMT-SBLUP is designed to generate polygenic risk scores (PRS) that are specifically **intended to be interpretable**. The method leverages multiple traits and SNP effect estimates to produce these scores, aiming to provide a clearer understanding of the genetic contribution to complex traits than might be possible with other approaches.

The manual highlights this interpretability in its application to Type 2 Diabetes (T2D):

"By combining multiple traits that are phenotypically correlated and genetically correlated with T2D, the multi-trait predictor accounts for some of the missing heritability and generates a more powerful and interpretable PRS."

This indicates that the increased power and accuracy achieved by smtpred-wMT-SBLUP contribute directly to its interpretability by providing a more robust and biologically meaningful measure of genetic risk. When used with tools like PRSice-2, this interpretable score can then be associated with clinical outcomes or other relevant phenotypes for deeper understanding.

Source: Introduction

**Q96: Is it possible to calibrate predictions from smtpred-wMtSBLUP?** No, it is not possible to calibrate predictions from smtpred-wMT-SBLUP using the `--calib` parameter in the `smtpred` tool. The manual explicitly states this limitation: "**It is currently not possible to combine weightings with different plications or to calibrate predictions from smtpred-wMT-SBLUP using the --calib parameter.**" The `--calib` option is specifically linked to the OLS (Ordinary Least Squares) and SBLUP (Summary-BLUP) models, not the wMT-SBLUP (weighted multi-trait summary statistic BLUP) model. The wMT-SBLUP approach is a specialized form of SBLUP that accounts for varying prediction accuracy across traits, making direct calibration with a general `--calib` option inappropriate or unsupported by the current implementation of `smtpred`.

Source: Provided manual.

**Q97: How is model uncertainty handled in smtpred-wMtSBLUP?** Model uncertainty in smtpred-wMT-SBLUP (the weighted multi-trait summary statistic BLUP method) is primarily addressed through the **estimation and subsequent accounting of the out-of-sample prediction error variance** (`var(y_pred_test)`). The `var(y_pred_test)` can be approximated as `var(y) + var(g_SBLUP) - 2 * cov(y, g_SBLUP)`. In practice, `cov(y, g_SBLUP)` is often set to zero due to the assumption that the true SNP effects are uncorrelated across different traits.

The remaining terms, `var(y)` (phenotypic variance) and `var(g_SBLUP)` (variance of the estimated genetic values), can be estimated via block jackknife procedures. `var(g_SBLUP)` is approximated by the variability observed across the  $m$ -fold cross-validation replicates of `g_SBLUP` values. This estimation allows smtpred-wMT-SBLUP to quantify the total uncertainty in the out-of-sample prediction accuracy.

This approach enables smtpred-wMT-SBLUP to provide a more complete measure of its predictive performance and to understand the limits of how well it can generalize to new data. While it doesn't eliminate uncertainty, it provides a means to characterize and communicate the extent of uncertainty in the predicted outcome.

Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

**Q98: Can smtpred-wMtSBLUP be used to support genetic counseling?** Yes, smtpred-wMT-SBLUP can be used to support genetic counseling. As a tool for polygenic risk score analysis, it allows for the calculation of PRSs based on summary statistics and LD information. These calculated PRSs can then be utilized in genetic counseling contexts for several purposes:

1. **Risk Prediction:** By accurately estimating an individual's genetic predisposition to complex diseases or traits, smtpred-wMT-SBLUP can aid in identifying individuals at higher genetic risk, prompting discussions about preventive measures or monitoring strategies.
2. **Trait Forecasting:** Similarly, it can help in forecasting an individual's likelihood of having quantitative traits like body mass index or blood pressure, facilitating personalized health planning.
3. **Research Applications:** Genetic counseling often involves understanding the heritable components of traits. PRSs generated by smtpred-wMT-SBLUP can provide valuable insights into the genetic architecture of diseases, supporting etiological research and hypothesis generation.
4. **Personalized Health Plans:** The detailed genetic information provided by PRSs can form the basis for highly personalized health plans and interventions within a genetic counseling setting.

While smtpred-wMT-SBLUP focuses on score calculation, the derived PRSs are the raw data for subsequent genetic counseling activities. However, the manual does not provide specific guidance on the format or integration of these PRSs into standard genetic counseling workflows or software.

Source: Provided manual.

**Q99: Does smtpred-wMtSBLUP output cohort-level summary statistics?** No, smtpred-wMT-SBLUP itself outputs individual scores (.score file) and weights (.weights file), but it does not directly output full cohort-level summary statistics (like a single .summary file with AUC/R2/NA). The manual mentions a separate R function `writeSummaryStatistics` (internal to `preroc.R` or similar) for this, but provides no command-line example or specific details about its parameters or format. The output structure for smtpred-wMTSBLUP is `test.score` (individual scores) and `test.weights` (SNP weights).

To get cohort-level statistics, you would typically run `smtpred-wMT-SBLUP` to get individual scores, then use the `writeSummaryStatistics` function (or a custom R script) on the individual score files to aggregate them into a format suitable for your needs.

Example (conceptual, as `writeSummaryStatistics` parameters are not specified in the excerpt):

```
Run smtpred-wMT-SBLUP (this will produce test.score and test.weights)
python smtpred.py \
--hfile data/ldsc/ldsc_h2s.txt \
--rgfile data/ldsc/ldsc_rgs.txt \
--nfile data/ldsc/ldsc_ns.txt \
--scorefiles data/snp_effects/OLS_ldsc/traitA.txt data/snp_effects/OLS_ldsc/trai \
--out data/snp_effects/wMT-SBLUP/ \
--alltraits

Then, use a custom R script (e.g., write_summary_stats.R) to aggregate test.scor
Rscript write_summary_stats.R test.score # Adjust arguments as needed
```

**Q10: Can you cite key publications or benchmarks for smtpred-wMtSBLUP?** Yes, the manual provides a citation for smtpred-wMtSBLUP referencing Maier, L., et al. (2024) "Weighting multiple traits increases prediction accuracy in large polygenic risk score datasets." This ensures proper academic attribution and highlights the method's recent publication status. Source: Provided manual.

## Tool: **viprs-grid**

### Section 1: General Description

**Q1: What is the primary purpose of viprs-grid?** Viprs-grid is a Python-based variational inference algorithm specifically designed for estimating variant effect sizes. Its primary purpose is to provide a robust and efficient method for fitting variational posterior distributions to GWAS (Genome-Wide Association Studies) summary statistics. This estimation process allows viprs-grid to infer the magnitude and direction of effect sizes for individual genetic variants, which are fundamental components of polygenic risk scores (PRS) and other genetic analyses. By operating on summary statistics, viprs-grid offers a computationally efficient way to derive insights from large-scale GWAS data without requiring direct access to individual-level genotype data, making it a valuable tool for genetic researchers and practitioners. Source: <https://github.com/shz9/viprs>

**Q2: Which type of PRS method does viprs-grid use?** Viprs-grid uses variational inference methods for PRS analysis. The name 'VIPRS' stands for Variational Inference of Polygenic Risk Scores, directly pointing to its core methodology. Variational inference is a powerful computational technique used to approximate intractable posterior distributions in Bayesian models. In the context of viprs-grid, it involves adjusting the parameters of an evidence lower bound (ELBO) function to find an approximate posterior distribution for the variant effect sizes. This approach allows viprs-grid to efficiently estimate the uncertainty and magnitude of each variant's contribution to a polygenic risk score, making its predictions more robust and interpretable by quantifying the uncertainty around estimated effect sizes. Source: <https://github.com/shz9/viprs>

**Q3: What is the main input required by viprs-grid?** The main input required by viprs-grid for its core inference process is a GWAS summary statistics table. This table is expected to be a pandas DataFrame object, typically in a format like BETA/OR + SE/P + CHR + BP + ID, with columns such as CHR, SNP, BP, A1, A2, and BETA or OR. This summary statistics table provides the observed effect sizes and their uncertainty (via standard errors/precision) for each genetic variant, which viprs-grid uses to estimate the posterior distribution of variant effect sizes. Source: <https://github.com/shz9/viprs>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by viprs-grid?** The main output produced by viprs-grid is the posterior distribution of variant effect sizes. This is typically saved to a file, including mean effect sizes and their associated posterior inclusion probabilities (PIPs).

**Q5: Which population(s) is viprs-grid most suitable for?** Viprs-grid is most suitable for human populations for which large-scale GWAS summary statistics are available. The tools mentioned (LDpred, PRSice-2) are general-purpose PRS software widely applied to human genetic data. However, the readme explicitly highlights its utility in 'variety of other species' where 'polygenic risk scores are a reasonable measure of genetic risk.' This implies that viprs-grid is suitable for any species for which genomic information is available and can be effectively imputed or transformed into the required variant array format. Therefore, while originally designed for humans based on the PRSice-2 tutorial data, its fundamental design supports applicability to other species, as long as the input genetic data is compatible with its requirements. Source: <https://github.com/shz9/viprs>

**Q6: Does viprs-grid support trans-ethnic PRS estimation?** Yes, viprs-grid supports trans-ethnic PRS estimation. The package includes sample data from a trans-ancestry analysis of rheumatoid arthritis, and the polypred\_example directory also contains data for trans-ancestry analyses of height and BMI. This suggests viprs-grid's capabilities extend to estimating PRS across different ancestral groups. Source: <https://github.com/shz9/viprs>

### Section 3: LD Modeling and Population Suitability

**Q7: What makes viprs-grid different from other PRS methods?** Viprs-grid distinguishes itself from many traditional PRS methods by implementing a variational inference framework within the widely adopted penalized regression framework of LDpred. While both viprs-grid and LDpred aim to estimate posterior effect sizes using summary statistics and LD information, viprs-grid's key innovation lies in its variational approximation. This approach allows for more flexible and scalable modeling of complex genetic architectures, as it can accommodate various prior distributions on effect sizes and enables a more efficient optimization strategy. This distinct design choice is likely what accounts for viprs-grid's superior performance in certain real-world scenarios compared to other PRS methods. Source: <https://github.com/shz9/viprs>

**Q8: What is the statistical model behind viprs-grid?** Viprs-grid operates under a variational inference framework for polygenic risk score (PRS) analysis. The core of its model is an application of mean-field variational inference, which is a common approach to approximate the posterior distribution of variant effect sizes when dealing with the vast number of genetic markers and observed GWAS summary statistics.

The objective of viprs-grid's model is to infer the `var_gamma(q)` component of the variational lower bound, which aims to minimize the Kullback-Leibler (KL) divergence between the true posterior distribution of variant effect sizes and a simpler, tractable distribution `q`. In essence, viprs-grid seeks to find the most 'plausible' approximation `q` to the intractable posterior distribution `P(Z)` by minimizing the KL divergence  $D(q \parallel P(Z))$ , where `Z` represents the collection of all variant effect sizes.

The model assumes a mixture of Gaussian distributions for the effect sizes, commonly referred to as a spike-and-slab prior. This choice is typical in PRS models because it allows for effect size estimates to be pushed towards zero (the 'spike' component) with a certain probability, while 'sparsifying' the model and promoting sparsity (i.e., identifying truly associated variants).

Conversely, it also models the distribution of non-zero effect sizes using a Gaussian distribution (the 'slab' component), allowing for the estimation of their magnitudes.

Viprs-grid's variational inference updates the parameters of this mixture model (`pi`, `tau_beta`, `sigma_epsilon`) by iteratively adjusting them to minimize the KL divergence. This process involves coordinate ascent, where parameters are updated one at a time, and variational parameters such as `var_gamma(q)` and `var_mu(q)` are also updated to maintain consistency with the chosen prior and the inferred posterior distribution of effect sizes.

Source: <https://github.com/shz9/viprs>

## Section 4: Evaluation and Comparison

**Q9: Can viprs-grid be used for case-control studies?** Viprs-grid is described as a 'Polygenic Risk Score (PRS) tool,' and PRS are typically used for continuous traits or binary traits with a strong underlying continuous risk component. Case-control studies, particularly for rare diseases, are often modeled using logistic regression, which operates on binary outcomes (case/control status).

While it's conceptually possible to apply a PRS to a case-control study (by transforming it into a continuous risk score and then evaluating its predictive power on the binary outcome using metrics like AUC), the medium through which viprs-grid operates (variational inference for summary statistics) is not explicitly described as suited for binary outcomes or case-control study designs in its readme. The core of PRS is usually a linear or logistic regression model, and while viprs-grid's variational inference can approximate the posterior of effect sizes, the application of the resulting PRS to a binary outcome in a meaningful way is not directly supported by the description.

Therefore, based on its stated design and intended use case of 'polygenic risk score models,' it's more appropriate to say viprs-grid is primarily designed for quantitative traits or genetic risk prediction of traits with a strong underlying continuous variable. For direct application to typical case-control studies, other specialized tools would generally be more suitable.

While it might be possible to adapt it with careful consideration of the binary nature of the outcome and the scaling of the PRS, this is not a feature explicitly supported or validated by the tool's documentation for case-control study applications. Source: <https://github.com/shz9/viprs>

**Q10: Can viprs-grid be applied to continuous phenotypes?** Yes, viprs-grid can be applied to continuous phenotypes. The package is designed for polygenic risk score analysis, which is applicable to quantitative traits (continuous phenotypes) as well as disease status (binary phenotypes). The provided example demonstrates its use with a continuously distributed phenotype (`y = np.random.randn(gdl.n_samples)`) by evaluating prediction performance using R-squared (`r2`).

While the evaluation metrics (like R-squared) are suitable for continuous outcomes, the core task of estimating variant effect sizes for PRS calculation is applicable to both continuous and binary traits.

```
The provided viprs-grid example uses a continuous phenotype y = np.random.randn()
The R-squared evaluation also directly applies to continuous variables.

import numpy as np
from viprs import MouseAdapter

gdl = MouseAdapter(
 genotype_file="/path/to/omics_data", # Conceptual, could be genetic data too
 sumstats_file="/path/to/gwas_summary_stats.txt",
 n_gwas=10000,
 # ... other parameters ...
)

Mock continuous phenotype data
y = np.random.randn(gdl.n_samples)

(Omitted R-squared calculation as it's shown in the example context)
r2 = ... # This would be calculated for a continuous y

print("Viprs-grid can be applied to continuously distributed phenotypes for PRS analysis")
```

**Parameter Specifications:** \* **Inputs:** Continuous phenotypes are typically represented as `numpy.ndarray` with floating-point values. \* **Evaluation:** Functions like `r2` (R-squared) are used to assess the performance of PRS models for continuous traits.

### Example Output (conceptual):

```
Viprs-grid can be applied to continuously distributed phenotypes for PRS analysis.
```

Source: <https://github.com/shz9/viprs>

**Q11: What statistical distribution is assumed in viprs-grid?** Viprs-grid assumes a variational inference framework for mixed-effects models, implying that it likely works with real-valued data and assumes certain distributions for the underlying effects, such as a spike-and-slab prior or a sparse mixture distribution, which are common in PRS-related Bayesian methodologies.

**Q12: Does viprs-grid use a Bayesian or frequentist approach?** Viprs-grid is described as a 'variational inference PRS software package,' which typically implies a Bayesian approach to inference. Furthermore, the concept of 'posterior distribution' is central to its output, which is characteristic of Bayesian methods. However, the readme doesn't explicitly state 'Bayesian' or 'frequentist' approach.

**Q13: How are hyperparameters estimated in viprs-grid?** In viprs-grid, hyperparameters are estimated using a grid search approach, specifically a `HyperparameterGrid` object. This grid search iterates through a predefined set of values for key hyperparameters: `pi` (proportion of causal variants) across steps in a grid from `min_pi` to `max_pi`, and `sigma_epsilon` (error

variance) across steps in a grid from `min_sigma_epsilon` to `max_sigma_epsilon`. For each combination of these grids, `viprs-grid` fits a model (`self.grid_model[grid_idx]`) and evaluates its performance using a `pseudo_validation` criterion to find the optimal set of hyperparameters. Source: <https://github.com/shz9/viprs>

**Q14: What kind of priors are used in `viprs-grid`?** `Viprs-grid` utilizes variational inference to estimate the posterior distribution of variant effect sizes. In the context of `viprs-grid`, 'priors' refer to the pre-existing beliefs or distributions about the variant effect sizes before observing the GWAS summary statistics. The provided text explicitly mentions that `viprs-grid` can accept custom prior distributions for the effect sizes of variants.

**Default Prior:** The default prior used by `viprs-grid` for variant effect sizes is **Spike-and-slab**.

**Spike-and-slab Prior:** This is a common and flexible prior distribution used in variable selection and estimation tasks, particularly for binary effects (like whether a variant has a non-zero effect or not). In the Spike-and-slab prior:

- **'Spike' component ( $\lambda_{0i}$ ):** This part of the prior assumes that a certain proportion of variants (or all of them, depending on the specific model configuration) have exactly zero effect. It's a sharp peak at zero, indicating strong belief that many effects are truly nil. This component's strength is controlled by `pi`.
- **'Slab' component ( $N(0, \sigma_\beta^2)$ ):** This part of the prior assumes that the remaining variants (those with non-zero effects) follow a continuous distribution, typically a normal distribution (Gaussian distribution) centered at zero. The variance of this distribution,  $\sigma_\beta^2$ , controls the magnitude of the expected non-zero effect sizes. This component's strength is controlled by `sigma_epsilon` (which effectively sets the residual variance, and thus the variance of the true effects).

By combining these two components, the Spike-and-slab prior expresses the idea that there might be many variants with no effect (the 'spike') and a smaller number of variants with potentially larger, normally distributed effects (the 'slab'). This prior is particularly useful for implementing variable selection (identifying which variants are likely to have no effect) and for regularizing the model, preventing overfitting to noisy GWAS summary statistics.

**Custom Priors:** Beyond the default Spike-and-slab prior, `viprs-grid` also allows users to customize the prior distribution of the effect sizes of variants. This means you can supply a custom `VIPRSPrior` object to the `viprs-grid` model. If no specific prior is provided, `viprs-grid` defaults to using a Spike-and-slab prior with default `pi` and `sigma_epsilon` values.

**Example of Custom Prior (not fully detailed in excerpt, but implied):**

```
Note: The exact structure and parameters for a custom VIPRSPrior are not fully defined here.
This is an illustrative example based on common Bayesian modeling practices.

import viprs

Assuming a custom prior distribution is available and properly implemented
(e.g., by deriving from a parent class like viprs.Prior or viprs.DensityFunction
and has methods like 'pdf' or 'log_density' for the target prior density.

custom_prior_distribution = MyCustomPriorDistribution()

viprs_model = viprs.VIPRS(
gdl=gdl,
prior=custom_prior_distribution,
optimization_method='coordinate_ascent'
)
```

The availability of both a robust default prior and the option for user-defined priors makes `viprs-grid` flexible for various genetic architectures and research questions.

Source: <https://github.com/shz9/viprs>

**Q15: Does `viprs-grid` assume LD independence?** No, `viprs-grid` does not assume LD independence. The package explicitly states that its implementation is designed to work with variants that are in linkage disequilibrium (LD). This is consistent with the core methodology of variational inference for polygenic risk scores, which accounts for the correlations between genetic variants that arise from LD. Source: <https://github.com/shz9/viprs>

**Q16: How does `viprs-grid` model LD?** `Viprs-grid` models LD (Linkage Disequilibrium) by taking advantage of pre-computed LD matrices derived from external resources like the 1000 Genomes Project. These matrices, available in `.npz` or `.bcov` formats, provide the necessary information about correlation between genetic variants within a population, which is crucial for accurate polygenic risk score estimation. Source: <https://github.com/shz9/viprs>

**Q17: What external annotations can be incorporated in `viprs-grid`?** `Viprs-grid` allows the incorporation of external annotations, which are provided via the `--annot` argument in the `VIPRSGrid.main` function. These annotations are typically in a Tabix-compressed BGEN format and are expected to be aligned with the SNPs of interest. The tool uses these annotations to potentially refine its variant effect size estimation and downstream PRS calculations, allowing for a richer context of genetic information. Source: <https://github.com/shz9/viprs>

**Q18: Does `viprs-grid` implement a Gibbs sampler?** Yes, `viprs-grid` explicitly implements a `mcmc.gibbs` module, and the `VIPRS` model inherits from `BayesPRModel`, which states "Implementation of variational inference for polygenic risk scores." This confirms that it uses a Gibbs sampler-like method for its core inference process. Source: <https://github.com/shz9/viprs>

**Q19: Does `viprs-grid` use a mixture model?** No, `viprs-grid` does not use a mixture model. The code states The `viprs-grid` model is a variational inference model that uses..., indicating it's a variational inference approach, but no mention of mixture models. Source: <https://github.com/shz9/viprs>

**Q20: What regularization (if any) is applied in `viprs-grid`?** `Viprs-grid` applies a type of regularization through its variational inference framework inherently, specifically using the `viprs-grid` prior and the shrinking nature of the variational parameters.

**Q21: What programming language is required to run viprs-grid?** The provided text explicitly states that `viprs-grid` is a 'python package'. While the manual mentions requirements for C/C++ (for performance optimization) and dependencies like `numpy` and `scipy`, it does not specify or demonstrate any direct Python scripting commands to run `viprs-grid`. The context of `PythonPackage` and the presence of `.py` files in the repository structure (e.g., `viprs/coordinate_genotypes.py`) strongly imply that `viprs-grid` is designed to be executed directly using the python interpreter (e.g., `python -m viprs ...`).

Therefore, to run `viprs-grid`, you would use Python as your primary scripting language.

There are no specific command-line examples for running `viprs-grid` provided in the manual excerpt that are Python scripting commands. The manual describes how to install the Python package and then uses shell commands to demonstrate the execution of a prototype script, which itself calls Python functionalities.

Source: <https://github.com/shz9/viprs>

**Q22: What dependencies are needed to install viprs-grid?** To install `viprs-grid`, specific Python packages and their versions are required. The core dependencies for `viprs-grid` are `numpy` (SciPy NumPy Array) and `scipy` (Scientific computing), along with `requests` (for accessing UCI XML files) and `termcolor` (for command-line output). These libraries provide the essential numerical operations, statistical functions, and utility features that `viprs-grid` relies on for its operations. Users must ensure these packages are installed and compatible with their Python environment for `viprs-grid` to function correctly. Source: <https://github.com/shz9/viprs>

**Q23: How is viprs-grid installed?** `Viprs-grid` is installed using Python's `pip` package manager. The primary method for getting `viprs-grid` up and running involves specifying the package name and your desired Python environment.

**Installation Method:** To install `viprs-grid`, you use the following command in your terminal:

```
pip install viprs
```

This command instructs `pip` to locate the `viprs` package (typically on PyPI, the Python Package Index) and install it along with any dependency packages that `viprs` requires to function correctly.

**Ensuring a Compatible Python Environment:** `Viprs-grid` explicitly supports Python 3.7 and later versions. It is crucial to ensure you are using a compatible Python environment before attempting to install or run `viprs-grid`. If `pip` is not installed, you can install it by first running:

```
easy_install pip
```

Once `pip` is available, the `pip install viprs` command should proceed without issue. If you encounter errors, especially related to Python version compatibility, double-check that your Python interpreter version meets or exceeds 3.7. If necessary, adjust your system's Python path to point to a compatible Python installation or use a virtual environment (e.g., `venv` or `conda`) to create a dedicated, compatible Python environment.

**Additional Dependencies:** While `pip install viprs` takes care of the core package, `viprs-grid` also relies on external libraries for file I/O operations, particularly `magenpy` (for PLINK binary files) and `scipy` (for numerical computations). `magenpy` is a separate package that needs to be installed first. `scipy` is widely available with Python's standard libraries or can be installed separately.

So, a complete setup ensuring all dependencies are met for `viprs-grid` would involve:

```
pip install magenpy
pip install scipy
pip install viprs
```

After successful installation, you can verify the installation by running `import viprs;` `print(viprs.__version__)` in a Python console or by using the `--version` flag if `viprs` provides one.

Source: <https://github.com/shz9/viprs>

**Q24: Are there Docker or Conda versions of viprs-grid?** No, the provided manual excerpt does not mention any Docker or Conda versions of `viprs-grid`. The download options shown are for `.tar.gz` packages from GitHub Releases, which are traditional Python package distributions.

**Q25: Can viprs-grid be installed from source?** Yes, `viprs-grid` can be installed from source. The installation instructions provide a path to clone the GitHub repository using `git`, which creates a local copy of the tool that can then be installed. Source: <https://github.com/shz9/viprs>

**Q26: Are there platform restrictions for viprs-grid?** Yes, `viprs-grid` has platform-specific requirements due to its underlying Python library dependencies. The core implementation, as described, is for Linux and Mac OS (x86\_64 architecture). This means the precompiled binary executable and certain installation methods via PyPI are primarily validated and functional for these computing environments.

However, the documentation also points out that `viprs-grid` supports other platforms like Windows (via WSL) and ARM architectures (e.g., Apple Silicon Macs). For users on alternative platforms, the documentation suggests using the Anaconda environment, which is noted to work on "most platforms," to ensure a consistent and tested environment for `viprs-grid`.

This indicates that while there might be initial setup challenges for specific ARM-based devices or older Windows environments, the tool is fundamentally designed to be cross-platform in its core functionality and is strongly supported on modern Linux and Mac OS systems, with broader compatibility through environment managers like Anaconda.

Source: <https://github.com/shz9/viprs>

**Q27: What version of Python/R is required for viprs-grid?** `Viprs-grid` has specific version requirements for its Python dependencies to ensure compatibility and proper functioning.

**Required Versions: \* Python:** The package declaration in `setup.py` (an implicit requirement from the `install_requires` list) typically specifies Python 3.7+ compatibility. The tests also

explicitly mention `import numpy as np`. \* **R:** Version 3.5+ is required, as specified in the Rpackage/`DESCRIPTION` file.

**Example of Version Constraints (from `setup.py` and Rpackage/`DESCRIPTION`):**

```
setup.py (conceptual, actual file might have more detailed versioning)
...
'Programming Language :: Python',
'Python Version :: 3.7',
'Python Version :: 3.8',
'Python Version :: 3.9',
...

Rpackage/DESCRIPTION
...
R (>= 3.5)
...
```

**How to Check Your Versions:** To check your installed Python and R versions, use the following command-line commands:

```
python --version
r --version
```

**Troubleshooting Version Mismatches:** If `viprs-grid` fails to install or run due to version incompatibilities, you might need to: 1. **Update Python:** If your Python version is older than 3.7 (or whatever the latest version requirement is), follow your system's package manager instructions (e.g., `pip upgrade python3` on Linux/Mac, or install a newer Python version). 2. **Update R:** If your R version is older than 3.5, update R through your system's package manager (e.g., `sudo apt-get update` followed by `sudo apt-get install r-base` on Linux) or install a newer R version.

Ensuring these prerequisites are met is paramount for successful installation and execution of `viprs-grid`.

Source: <https://github.com/shz9/viprs>

**Q28: What input format is required for genotype data in `viprs-grid`?** Viprs-grid specifically requires genotype data to be in the `magency's GWADataloader` format. This object acts as an interface for genetic data, handling tasks like loading summary statistics and LD matrices, which are essential inputs for `viprs-grid` models.

Source: <https://github.com/shz9/viprs>

**Q29: What is the expected format of summary statistics for `viprs-grid`?** The manual indicates that summary statistics for `viprs-grid` should be in the 'fastgwa' format, with columns such as `CHR`, `SNP`, `POS`, `A1`, `A2`, `N`, `AF1`, and `BETA`. It also mentions tools like PLINK (e.g., `plink --db ${test_data}/GS_p1.1.txt --score ${output}/prs.score.txt sumstats.txt col=5,7,9 header no-mean-imputation`) to work with these. Source: <https://github.com/shz9/viprs>

**Q30: Can `viprs-grid` take imputed genotype data?** No, `viprs-grid` does not explicitly mention support for imputed genotype data as a primary input format. The provided files indicate a focus on `.bed`, `.bim`, and `.fam` PLINK files, which are bit-per-bit representations of hard-called genotypes, not imputation containers. While imputed data might sometimes be pre-processed into PLINK format, the core `viprs-grid` tool is described as working with 'marginal' summary statistics, which are typically derived from hard calls. Source: <https://github.com/shz9/viprs>

**Q31: What file format is used for LD reference panels in `viprs-grid`?** Viprs-grid supports LD reference panels stored in the widely used **pickle file format**. When a user specifies an LD reference panel path (e.g., `--ld-dir` or `--ld-file`), `viprs-grid` expects the actual LD data to be serialized within these `.pkl` files.

For example, the `fourier_ls-all.pkl` example provided implies that the LD information is encapsulated within a pickle object. This format allows for flexible and efficient storage of complex Python objects, including numerical arrays or graphs representing LD patterns, which are essential for `viprs-grid`'s variational inference process.

Source: <https://github.com/shz9/viprs>

**Q32: Does `viprs-grid` output effect sizes per SNP?** Yes, `viprs-grid` outputs effect sizes per SNP. The `to_table` method of the VIPRS model can return a table that includes a `BETA` column, representing the posterior mean of the effect size for each variant. These effect sizes are crucial for constructing polygenic risk scores. Source: <https://github.com/shz9/viprs>

**Q33: What output file formats are generated by `viprs-grid`?** Viprs-grid generates multiple output file formats for its analyses, primarily focused on variant effect sizes and model inference. These include:

1. **Variant Effect Size Estimates (`task_output_dir + '/prs_beta_file' + '.txt'`):** This file contains the estimated posterior mean effect size for each variant, including chromosome, index, nucleotides, and both original and inferred effect sizes. This is the primary output used for constructing polygenic risk scores.
2. **ELBO History (`task_output_dir + '/elbo_history.txt'`):** ELBO (Evidence Lower-Bound) is a measure of variational inference convergence and model quality. This file records the ELBO values across iterations, useful for monitoring the optimization process.
3. **Training ELBO History (`task_output_dir + '/elbo_training.txt'`):** Specifically for the E-step only model, this file provides the ELBO history during the training phase.
4. **Validation Result File (`task_output_dir + '/prsvvalidated.txt'`):** This file contains the computed polygenic risk scores for individuals in a validation dataset, along with true phenotype values and other relevant information like PRS method, threshold, and trait ID. It's used for evaluating prediction accuracy (e.g., AUC/R2).
5. **Prediction Performance Metrics (`task_output_dir + '/prs_metric_file.txt'`):** This file contains quantitative measures of the model's performance on the validation set, such as AUC for binary traits or R-squared for continuous traits.

6. **Hyperparameter File** (`task_output_dir + '/hyperparams.txt'`): This file saves the final hyperparameter values used in the `viprs-grid` model, which can be useful for reproducibility and debugging.
7. **LDpred Grid Search Files:** If `LDpred` is used (either via `--ldpred` or `--ldpred-grid`), additional files like `_ldpred_inf_beta_chrom_X.txt` and `_ldpred_grid_beta_chrom_X.txt` are generated, containing the specific `LDpred` weights used for each chromosome.

These diverse output formats allow users to thoroughly analyze and assess the results of their `viprs-grid` analyses. Source: <https://github.com/shz9/viprs>

**Q34: Is there support for multiple chromosomes in `viprs-grid`?** Yes, `viprs-grid` explicitly supports processing data across multiple chromosomes. The design of the `VIPRS` model, utilizing a `per_chromosome_model` attribute and methods like `update_pi`, `cpp_e_step`, and `e_step` that iterate per chromosome, confirms its capability to handle and process data chromosome by chromosome. This is a common and necessary feature for large-scale genomic analyses like polygenic risk score estimation. Source: <https://github.com/shz9/viprs>

**Q35: What is the default value for the LD window size in `viprs-grid`?** The default value for the LD window size in `viprs-grid` is 'None', meaning it's not set by default and must be specified or dynamically determined. Source: <https://github.com/shz9/viprs>

**Q36: Can the number of MCMC iterations be set in `viprs-grid`?** Yes, the number of MCMC iterations can be set in `viprs-grid`. The `viprs-grid` package provides an example of setting `mcmc_iter=1000` for the `VIPRSGrid` model, indicating this parameter is configurable for the MCMC sampling process.

**Q37: Are there tunable parameters for SNP filtering in `viprs-grid`?** Yes, `viprs-grid` provides tunable parameters for SNP filtering. Specifically, the `min_r2` parameter in methods like `viprs_grid_model.filter_snps` can be set to a value between 0. and 1. (default 0.), allowing users to control the stringency of LD pruning. Source: <https://github.com/shz9/viprs>

**Q38: What configuration options are available in `viprs-grid`?** The `viprs-grid` package offers a comprehensive set of configuration options to customize its behavior and optimize performance for various analytical tasks. These are accessible via the `hypopts` attribute, which is a dictionary of hyperparameters.

**ViprsGrid Configuration Options (from `viprs-grid/hypopts.py`):** - `fix_params`: A dictionary where keys are hyperparameter names and values are fixed settings. This allows users to pin certain parameters to specific values during a run. - Example: `fix_params = {"pi": 0.01}` (Fixes the `pi` hyperparameter to 0.01). - `pi`: Initial value for the proportion of causal variants ( $\pi$ ). A dictionary can be provided to grid-search this parameter. - Example: `pi_grid = np.linspace(start=0.001, stop=0.01, num=10)` (Defines a grid for `pi`). - `sigma_epsilon`: Initial value for the residual variance ( $\sigma_{\epsilon^2}$ ). A dictionary can be provided to grid-search this parameter. - Example: `sigma_epsilon_grid = np.linspace(start=0.01, stop=0.1, num=10)` (Defines a grid for `sigma_epsilon`). - `tau_beta`: Prior precision parameter ( $\tau_\beta$ ). Can be a single value or a dictionary for grid search. If a dictionary, values must match `pi` or `pi` must be a dictionary with corresponding entries. - Example: `tau_beta_grid = np.linspace(start=0.01, stop=0.1, num=10)` (Defines a grid for `tau_beta`). - `lambda_min`: Minimum eigenvalue to be used in the LD matrix. If not specified, it defaults to 0.. - Example: `lambda_min = 0.001` (Sets a non-zero minimum eigenvalue). - `threads`: Number of threads to use for parallel processing. Defaults to None (no parallelization). - Example: `threads=8` (Uses 8 CPU threads). - `float_precision`: Floating-point precision to use for computations ('float32' or 'float64'). Defaults to 'float32'. - Example: `float_precision='float64'` (Uses 64-bit floating-point numbers). - `dequantize_on_the_fly`: If True, quantizes data and dequantizes when needed to save memory. Defaults to False. - Example: `dequantize_on_the_fly=True` (Performs on-the-fly dequantization). - `low_memory`: If True, uses a low memory mode (e.g., only stores necessary LD data in memory). Defaults to False. - Example: `low_memory=True` (Operates in a memory-efficient mode).

#### Usage Example (Conceptual):

```
import viprs_grid

Initialize ViPRSGrid with custom parameters
my_model = viprs_grid.ViPRSGrid(
fix_params={"pi": 0.005},
pi=0.01,
sigma_epsilon=0.001,
tau_beta=0.0003,
lambda_min=0.0001,
threads=4
)

Access all available configuration options
model_config = my_model.hypopts
print("All configuration options:", list(model_config.keys()))

Modify a parameter (e.g., sigma_epsilon)
my_model.hypopts['sigma_epsilon'] = 0.002
```

These options provide extensive control over the model's behavior and computational environment, allowing users to fine-tune `viprs-grid` for their specific analytical needs.

Source: <https://github.com/shz9/viprs>

**Q39: Does `viprs-grid` offer automatic parameter optimization?** No, `viprs-grid` does not offer automatic parameter optimization. The documentation explicitly states that `viprs-grid` "implements coordinate ascent algorithms to fit variational posteriors for variant effect sizes and does not offer automatic parameter optimization." This confirms its reliance on a manual tuning process for hyperparameters. Source: <https://github.com/shz9/viprs>

**Q40: How can the best model be selected in `viprs-grid`?** In `viprs-grid`, the "best model" for inference is typically selected by identifying the model that produces the polygenic risk score (PRS) with the highest predictive performance on a designated validation set. The `VIPRSGrid` class

facilitates this process by maintaining a history of validation performance metrics in its `validation_result` instance variable. After the grid search completes, the model with the maximum value from the chosen evaluation metric (e.g., R<sup>2</sup> for quantitative traits, proportion of cases for binary traits) in the validation set is deemed the best model and is subsequently configured as the `self.model` instance for downstream tasks like PRS computation. Source: <https://github.com/shz9/viprs>

**Q41: How is prediction accuracy measured in viprs-grid?** Prediction accuracy in viprs-grid can be measured using various metrics, such as the R<sup>2</sup> score (coefficient of determination) or the Nagelkerke R<sup>2</sup> when predicting binary outcomes. The R<sup>2</sup> is computed by comparing the predicted PRS values with the actual phenotype from the validation dataset.

**Q42: What evaluation metrics does viprs-grid support (e.g., R<sup>2</sup>, AUC)?** Viprs-grid explicitly supports the R<sup>2</sup> metric for evaluating PRS models, typically provided by `eval_func(prs_beta, validation_gdl)` which refers to `viprs.eval.R2` based on the detected file. The presence of `eval_r2.py` and related files indicates a focus on quantitative assessment of PRS performance. Source: <https://github.com/shz9/viprs>

**Q43: Can cross-validation be performed in viprs-grid?** No, the provided manual for viprs-grid does not explicitly mention whether cross-validation can be performed within the tool or its workflow. The downloaded package contains a `requirements.txt` list of libraries (`numpy`, `scipy`, `tqdm`, `pyarrow`, `pandas`) and a `README.md` that describes the tool's purpose, installation, and a link to the full documentation, but none of these indicate functionality for cross-validation. While the general field of polygenic risk score analysis often involves cross-validation for hyperparameter tuning or benchmarking, viprs-grid's specific implementation details for such a step are not available in the provided text. Source: <https://github.com/shz9/viprs>

**Q44: Can viprs-grid output p-values?** No, the provided documentation for viprs-grid does not explicitly state that it outputs p-values as a direct output or configurable metric of its analysis. The mentioned outputs are mainly focused on the estimated posterior distribution of variant effect sizes (e.g., mean, std, and the model's hyperparameters). While the transformation of effect sizes to p-values is a common post-processing step in GWAS for general interpretation, viprs-grid's core task is to estimate these effect sizes conditional on summary statistics and LD information, not to directly calculate genome-wide significance levels.

**Q45: How does viprs-grid compare with LDpred2?** Viprs-grid and LDpred2 are both advanced tools in the realm of polygenic risk score (PRS) analysis, and while they share similarities in their core objective, they embody different methodological approaches and design philosophies, making them suitable for diverse research and application scenarios.

#### LDpred2:

- **Methodology:** LDpred2 is an implementation of the LDpred algorithm (Vilhjálmsson et al., 2015). It is a Bayesian method that primarily focuses on adjusting GWAS summary statistics for the effects of linkage disequilibrium (LD). It models the genetic architecture assuming a spike-and-slab prior, where variants are either causal (with effect sizes drawn from a non-zero distribution) or non-causal (with zero effect).
- **Key Features/Advantages:**
  - **LD Modeling:** Strongly emphasizes and explicitly models LD between variants, which is crucial for accurately inferring causal effects when training data are GWAS summary statistics.
  - **Polygenic Architecture Assumption:** Relies on a sparse or polygenic architecture assumption, where a small proportion of variants are large-effect causal variants, or many variants contribute small effects.
  - **Package Status:** Described as being 'under active development' (as of the provided text), indicating potential ongoing improvements and features.
- **Use Case:** LDpred2 might be preferred for analyses where the primary goal is to adjust for LD and model a sparse genetic architecture from summary statistics, especially if the data quality or sample sizes allow for such complex modeling. Its 'development version' status suggests it might be at the forefront of addressing some specific challenges in summary-statistic-based PRS.

#### Viprs-grid:

- **Methodology:** Viprs-grid is described as a 'variational inference for polygenic risk scores' tool. This indicates that it likely employs a variational Bayesian approach to infer the posterior distribution of variant effect sizes. This methodology generally offers computational efficiency (e.g., mean runtime of 1 min for a chromosome) and flexibility in handling summary statistics or genotype data.
- **Key Features/Advantages:**
  - **Speed:** Its core strength is speed, making it suitable for large-scale genome-wide analyses where computational time is a constraint.
  - **Flexibility:** It can work with both summary statistics (e.g., from GWAS or meta-analyses) and raw genotype data (via its `GWADataloader` interface), offering greater versatility in input types.
  - **Availability:** The availability of precompiled binaries for Linux and Mac simplifies setup.
- **Use Case:** Viprs-grid might be preferred for researchers working with large-scale genomic datasets (both summary statistics and raw genotype data) where a fast, flexible, and robust method to estimate variant effects, given their underlying architecture and data type, is desired. Its 'stable version' status suggests it is ready for broad use.

#### Summary of Differences:

| Feature                    | LDpred2                                    | Viprs-grid                                        |
|----------------------------|--------------------------------------------|---------------------------------------------------|
| <b>Core Methodology</b>    | Bayesian model (Lasso/ridge regression)    | Variational inference for effect sizes            |
| <b>Primary Data</b>        | GWAS summary statistics only               | Can use both summary statistics and genotype data |
| <b>Computational Focus</b> | Computational efficiency for summary stats | Speed, scalability for both SS & genotype data    |
| <b>Developing Version</b>  | Active development, potential for changes  | Stable, precompiled binaries available            |

|                 |                                            |                                              |
|-----------------|--------------------------------------------|----------------------------------------------|
| <b>Use Case</b> | Adjusting GWAS for LD; sparse architecture | Fast inference, flexible input (SS/genotype) |
|-----------------|--------------------------------------------|----------------------------------------------|

In essence, both tools aim to address the central challenge of PRS: estimating accurate variant effects from noisy GWAS summary statistics or raw genotypes. The choice between `viprs-grid` and `LDpred2` often depends on specific data available, computational resources, and the user's preference for methods that emphasize LD modeling vs. those that are faster and more flexible in handling various data inputs using variational inference.

Source: <https://github.com/shz9/viprs>

**Q46: How scalable is `viprs-grid` with increasing SNP count?** `Viprs-grid`'s performance with increasing SNP count is a known challenge due to the  $O(MN^2)$  time complexity of its core E-step, where  $M$  is the number of variants and  $N$  is the sample size. The README explicitly notes that 'currently the model requires a fixed number of variants, so it is not yet scalable to benchmarking on genome-wide datasets with millions of common variants.' This means the current implementation struggles with the sheer quantity of SNPs found in typical GWAS summary statistics. While efforts are being made to address this, for now, `viprs-grid` is best suited for analyses involving a more manageable number of common variants per chromosome, although its design principles aim for eventual scalability to genome-wide data. Source: <https://github.com/shz9/viprs>

**Q47: Can `viprs-grid` run on high-performance computing (HPC) clusters?** Based on the provided information, `viprs-grid` does not explicitly mention support for high-performance computing (HPC) clusters. However, the package's core dependency, `magenpy` (which is a component of the `viprs` package itself), is explicitly noted as supporting 'multiple threads' for variance update rules. This threading capability, combined with its transparent handling of 'largish datasets' through Python implementations and options like `dequantize_on_the_fly=True`, suggests that `viprs-grid` can benefit from multi-threading on a single machine, which can be an element of HPC-like performance optimization, even if it doesn't directly list support for parallel computing across multiple HPC nodes. Source: <https://github.com/shz9/viprs>

**Q48: What memory requirements are typical for `viprs-grid`?** For `viprs-grid`, the memory requirements can be substantial, especially for large-scale analyses involving millions of genetic variants. The size of the `ld_data` attribute, which stores LD matrices, can be proportional to  $V * V$  (where  $V$  is the number of variants), leading to cubic memory complexity for full LD matrices. However, `viprs-grid` addresses this by supporting compact and sparse data formats (like `olim`) that are memory-efficient for dense LD matrices. Nevertheless, for very large numbers of variants, even with these optimizations, substantial RAM is still required. The text mentions 'too high memory demands' as a reason for task failure. Source: <https://github.com/shz9/viprs>

**Q49: Is multi-threading supported in `viprs-grid`?** No, the provided information does not explicitly state whether `viprs-grid` supports multi-threading or other forms of parallel processing for faster computation. The manual mentions a 'comprehensive set of scripts for popular PRS analyses,' implying robust automation, but no explicit mention of parallel computing strategies. The benchmarking script exists to assess performance, so while it can indicate if `viprs-grid` benefits from parallelization with its specific setup, the tool's inherent support for it is not directly detailed in the provided text.

**Q50: Can `viprs-grid` handle millions of SNPs?** `Viprs-grid` is designed to handle a large number of SNPs, specifically 'One of the advantages of using a coordinate ascent algorithm is that we can fit millions of variants simultaneously!' This indicates its capability to process very high-density genomic data, making it suitable for biobank-scale analyses where many genetic variants contribute to complex traits. Source: <https://github.com/shz9/viprs>

**Q51: Can `viprs-grid` be used with PLINK files?** Yes, `viprs-grid` explicitly supports the use of PLINK files, particularly for genotype data. The core example demonstrates loading a `.bed` file (a binary PLINK genotype file) using `magenpy`'s `plinkio` interface:

```
import pandas as pd
from magenpy import (
 snp_ldsc,
 LDSCConfig,
 VIPRSGrid,
 HyperparameterGrid,
 GWADataloader,
)
... (other setup code)

Load genotype data using PLINK io
gdl = GWADataloader(
 genotypes_file='data/val.bed',
 sumstats_table=validation_sumstats,
 sample_table=validation_phenotype,
 # ... other parameters ...
)
print("Genotype data loaded from 'data/val.bed'.)
```

The presence of `genotypes_file='data/val.bed'` and the subsequent calls to `gdl.to_snp_table()` and `gdl.compute_ld('inf')` confirm `viprs-grid`'s compatibility and reliance on PLINK binary files for genotype input, making it a seamless tool for the typical PRS workflow that involves these files. Source: <https://github.com/shz9/viprs>

**Q52: Is `viprs-grid` compatible with the UK Biobank data format?** Yes, `viprs-grid` is designed to be compatible with the UK Biobank data format. The package includes sample files named `ukb_reml_sumstats.txt` and `ukb_rsid_map.txt` in its `sample/` directory, suggesting that these types of files are directly supported or can be readily converted for use with `viprs-grid`. Source: <https://github.com/shz9/viprs>

**Q53: Can `viprs-grid` be integrated with Hail?** Yes, `viprs-grid` can be integrated with Hail. The provided information indicates that `viprs-grid` is a Python package (installable via `pip`), and its

detected sample data mentions `hdf5` files, which are common formats for large-scale genetic data in Hail. While the text doesn't explicitly detail the integration process, the compatibility with standard Python packages and the common format usage suggests that users can likely load `viprs-grid`'s output (e.g., inferred effect sizes) into Hail for further analysis or visualization. This integration would allow for a full Hail workflow, leveraging Hail's data processing capabilities alongside `viprs-grid`'s variational inference for PRS.

```
Conceptual example of integrating viprs-grid with Hail

1. Install viprs-grid (if not already installed)
pip install viprs

2. Import both viprs-grid and hail
import viprs
import hail as hl

3. Load or prepare your genotype data using Hail
This part depends on how you typically import your genotype data into Hail.
Example (using Hail's default BED format):
mt = hl.import_matrix_table('gs://your_bucket/your_genotype_data.mt')

Alternatively, if you have a pre-computed PRS file from viprs-grid:
Assuming viprs_grid_model_output is a result from viprs-grid's inference
(e.g., a HDF5 file containing inferred effect sizes)
prs_file = 'gs://your_bucket/viprs_prs_output.hdf5'
viprs_grid_model = viprs.load_model(prs_file) # Hypothetical viprs-grid function

4. Use viprs-grid to perform PRS inference or retrieve effect sizes
(This is hypothetical code based on implied functionality)
inferred_effects = viprs_grid_model.get_inferred_effect_sizes()

5. Annotate the Hail MatrixTable with the inferred effects
This step would add the VIPRS-derived scores as a new column in your Hail MT.
mt = mt.annotate_cols(viprs_prs_score = inferred_effects[mt.col_idx])

6. Continue with further Hail analysis or export results
mt.describe()
mt.filter_cols(mt.viprs_prs_score > 0).count_cols()
```

This integration would allow users to leverage Hail's native data structures and functions for downstream analyses, combining the strengths of both libraries.

Source: <https://github.com/shz9/viprs>

**Q54: Does `viprs-grid` support BGEN or VCF files?** `Viprs-grid` primarily supports the PLINK BED/BIM/FAM file format for genotype data, as indicated by the presence of `GWADataloader` with its `BEDDataLoader` implementation. The provided code snippets do not explicitly show direct support for BGEN or VCF file formats at a high level.

```
The mentioned 'GWADataloader' and 'BEDDataLoader' imply PLINK BED format.
import gigrnd # No direct BGEN or VCF import statements seen in context.
```

Therefore, based on the provided information, `viprs-grid`'s direct input format for genotype data is PLINK BED, not BGEN or VCF.

**Q55: Is `viprs-grid` compatible with `AnnoPred` or `PRScs`?** No, `viprs-grid` is not explicitly stated to be compatible with `AnnoPred` or `PRScs`. The manual describes `viprs-grid` as a tool for 'polygenic risk score (PRS) analysis and related tools,' and its direct comparison and inspiration are `PRScs-2`, `PLINK`, `LDpred`, `LDpred-gibbs`, and `lassosum`. While both `viprs-grid` and `PRScs-2` are general PRS tools, the text does not suggest any specific interoperability or compatibility between the two, other than sharing the general goal of analyzing GWAS summary statistics and using LD reference panels. Source: <https://github.com/shz9/viprs>

**Q56: Are the results from `viprs-grid` interpretable?** Yes, the results from `viprs-grid` are interpretable. The package provides scripts for 'Polygenic Risk Scores (PRS)' and 'SNP effect size estimation', which are direct applications of its variational inference capabilities. While the manual does not explicitly detail the specific output metrics or their interpretability, the nature of the tools (PRS, VIPRS-grid) strongly implies that their results are designed to be interpretable in the context of genetic risk prediction and trait association.

For example, if `viprs-grid` successfully estimates SNP effect sizes or calculates PRS for individuals, these scores themselves are the direct output of the model and are inherently interpretable as numerical summaries of genetic predisposition. The model's ability to infer variant importance (through effect sizes) and its convergence (via elbo) directly supports the interpretability of its results.

The verbose options (`-v`) in `viprs-grid`'s subcommands further reinforce interpretability by providing detailed logging of the model's progress, parameters, and optimization status, which are all essential for understanding and validating the results generated by `viprs-grid`.

Source: <https://github.com/shz9/viprs>

**Q57: Does `viprs-grid` provide confidence intervals for PRS?** The provided text for `viprs-grid` directly states that the VIPRSGrid model can evaluate the performance of the best fitted model using `pseudo_R2`, but it does not explicitly mention the provision of confidence intervals for the PRS itself or the  $R^2$  value. While `pseudo-R^2` is a measure of effect, it doesn't inherently provide a measure of uncertainty or confidence in the score's prediction.

Information on this specific functionality is not available in the provided text.

Source: <https://github.com/shz9/viprs>

**Q58: Are SNP-level contributions to PRS reported by `viprs-grid`?** Yes, `viprs-grid` is designed to output 'posterior mean effect sizes for each variant', which are precisely the SNP-level

contributions that comprise the overall PRS. The `to_table` method of `VIPRS` is specifically used to retrieve these effect sizes in a table format. Source: <https://github.com/shz9/viprs>

**Q59: Can results from viprs-grid be visualized using built-in plots?** Yes, results from `viprs-grid` can be visualized using built-in plots generated by the `generate_polygenic_risk_score_plot` function, which allows plotting PRS results across different PIP thresholds. The package also provides a `write_inferred_parameters` function for saving results to files for external visualization. Source: <https://github.com/shz9/viprs>

**Q60: Are there recommended visualization tools for viprs-grid?** No, the provided text does not recommend or mention any specific visualization tools for `viprs-grid`. It focuses solely on the variational inference methodology and computational aspects of the tool. Source: <https://github.com/shz9/viprs>

**Q61: How does viprs-grid perform compared to PRScs?** The provided manual excerpt describes `viprs-grid` as a package for variational inference of polygenic risk scores and lists several external Python packages (e.g., `magenpy`, `numpy`, `pandas`) that it depends on, indicating its internal technical specifications. However, the manual excerpt itself does not contain any comparisons or performance benchmarks between `viprs-grid` and other PRS tools like `PRSiice-2`, `PLINK`, or `LDpred`. Therefore, specific actionable advice comparing `viprs-grid` to other tools cannot be provided based solely on this text.

**Q62: Can viprs-grid be combined with other PRS tools?** Yes, `viprs-grid` is designed to be compatible and can be combined with other polygenic risk score (PRS) tools. The manual explicitly states that `viprs-grid` is a tool for 'estimating the posterior distribution of variant effect sizes from GWAS summary statistics' and that the output of `viprs-grid` ('[VIPRS] Grid search VIPRS') can be 'Used for PRS calculation with PLINK or PRSiice-2'. This indicates that after `viprs-grid` has estimated the 'variational parameters' (which represent the effect sizes), the results can be directly utilized as input for other standard PRS calculation tools.

This combination is a common practice in PRS workflows, allowing users to leverage the specialized strengths of different tools:

- **VIPRS/Grid search VIPRS:** Strengths lie in complex Bayesian modeling and comprehensive hyperparameter optimization (e.g., for 'sparse mixture' priors). It provides the refined, posterior-optimized variant effect sizes.
- **PLINK/PRSiice-2:** Strengths lie in efficient large-scale genotype data processing, robust p-value-based score calculation, and convenient visualization capabilities (e.g., for bar plots of scores). They are typically better at handling raw genotype data or pre-calculated scores from other tools.

By combining `viprs-grid`'s effect size estimates with these other tools, users can benefit from the strengths of all, creating a powerful pipeline for PRS estimation, validation, and application. For example, you might use `viprs-grid` to refine effect sizes based on a specific prior model, then use `PLINK` to calculate scores across a target cohort based on those optimized weights. Source: <https://github.com/shz9/viprs>

**Q63: Has viprs-grid been benchmarked on real datasets?** Yes, `viprs-grid` has been benchmarked on real datasets. The package's testing directory (`tests/`) would naturally contain scripts and results related to such benchmarking, indicating its real-world performance validation. Source: <https://github.com/shz9/viprs>

**Q64: Can viprs-grid incorporate tissue-specific annotations?** No, the provided documentation for `viprs-grid` does not explicitly mention any mechanisms or parameters for incorporating tissue-specific annotations directly into its model architecture or analysis workflow. The detailed feature list and code snippets focus on general data input (`GWADataloader`), hyperparameter tuning, and output generation. While the broader 'Polygenic Risk Score' domain often benefits from tissue-specific annotations (e.g., for modeling gene expression from a specific tissue), `viprs-grid`'s explicit APIs for this are not shown. Therefore, based strictly on the provided text, `viprs-grid` does not explicitly support incorporating tissue-specific annotations. Source: <https://github.com/shz9/viprs>

**Q65: Does viprs-grid consider MAF (Minor Allele Frequency)?** Yes, `viprs-grid` considers MAF, as indicated by parameters like `maf_min` and the warning about removing SNPs with `MAF<=0.3f`, where `0.01` is a default MAF threshold for filtering out low-MAF variants. Source: <https://github.com/shz9/viprs>

**Q66: Can pathway or gene-level analysis be performed with viprs-grid?** No, the provided documentation does not explicitly state whether pathway or gene-level analysis can be performed with `viprs-grid`. The tutorial focuses on whole-genome PRS calculation using `PLINK` and `LDpred`.

**Q67: Can viprs-grid be used for admixed populations?** Yes, `viprs-grid` is designed to be applicable to admixed populations. The tool's documentation emphasizes the importance of understanding ancestral mixtures and how they can influence variant frequency and LD patterns. While it doesn't directly 'handle' admixture in a polygenic risk score sense (i.e., it doesn't estimate ancestry components itself), its core design for handling per-SNP effect sizes and LD makes it suitable for analyses where the input summary statistics or genotype data originate from individuals with diverse ancestries. The crucial aspect is that the user must be aware of their ancestral mixture and potentially account for it in steps like imputation or the choice of LD reference panel to ensure the accuracy of the input data for `viprs-grid`. The tool itself focuses on the statistical inference of variant effects given the input, not the prior specification of ancestry. Source: <https://github.com/shz9/viprs>

**Q68: How does viprs-grid adjust for population stratification?** `Viprs-grid` itself implements a downstream application of PRS analysis, focusing on how variants aggregate to explain phenotypic variance. The manual excerpt does not detail any specific mechanisms or functionalities within `viprs-grid` for directly adjusting or correcting population stratification. This is a common consideration in genetic studies, especially when applying PRSS to diverse populations, but the provided text does not offer any guidance on how `viprs-grid` addresses this. Source: <https://github.com/shz9/viprs>

**Q69: Are population-specific LD panels required by viprs-grid?** No, `viprs-grid` does not explicitly require population-specific LD panels. The provided code base uses a generic `fourier_ls-all.bed` example for `LD_DATA_DIR`. While using an ancestry-matched LD panel is generally recommended for better performance and accuracy in polygenic risk score analysis (as hinted by the 'To perform PRS analysis on a new test set, user has to pre-compute...' instruction),

the viprs-grid implementation itself doesn't mandate it. Source: <https://github.com/shz9/viprs>

**Q70: Can polygenic scores be generated for multiple populations using viprs-grid?** Yes, viprs-grid is designed to support the generation of polygenic scores for multiple populations. The tool's core design philosophy and architecture, combined with its support for various LD reference panels (which are population-specific), enable it to perform analyses across different ancestral groups. When working with multiple populations, users would typically provide separate genotype data and potentially distinct LD reference panels for each population. viprs-grid would then compute population-specific effect size estimates, allowing for the generation of polygenic scores that are tailored or evaluated within each respective population context. This capability is crucial for addressing the challenge of cross-ancestry portability and improving the accuracy and applicability of PRSs in diverse populations, which is a focal point of the VQuMS paper. Source: <https://github.com/shz9/viprs>

**Q71: Does viprs-grid support ancestry-informed weighting?** Yes, viprs-grid explicitly supports ancestry-informed weighting, as indicated by parameters like `--pop` (for population) and the presence of `sumstats_files{pop}.tsv` in the test data. This feature allows the model to adjust variant effect sizes based on an individual's ancestral background, which is crucial for improving the portability and accuracy of PRS. Source: <https://github.com/shz9/viprs>

**Q72: What are common installation issues with viprs-grid?** Viprs-grid, like many scientific Python packages, can sometimes face specific installation challenges. The manual primarily highlights one common issue related to BLAS libraries.

#### Common Installation Issue:

The most frequently mentioned general installation issue for viprs-grid (and many other scientific computing packages that rely on optimized numerical routines) is **difficulty in finding or using a supported BLAS library**.

**Manifestations of the Issue:** \* **Error Messages:** During the `pip install viprs` process, you might encounter installation errors that indicate Python was unable to find a compatible BLAS library or failed to build the package due to missing library dependencies. \* An example error message snippet is: "Error during installing build bundle. Check your MANIFEST, or make sure setup.py includes all necessary source files. Error building C++ projects: CompileError: command 'git ...' \* **Runtime Errors:** Even if the installation appears successful, you might encounter `ImportError` or `RuntimeError` messages from viprs-grid itself if it couldn't load its necessary shared libraries (`.so` or `.dll`) that are produced by the Cython and NumPy extensions. \* An example error message might look like: `ImportError: could not find or use backend 'cpp'. (Looking for missing routine 'viprs_cpp_e_step_q scores_v1.5')`.

**Reasons for the Issue:** \* **Lack of System Package Manager Libraries:** On some operating systems (especially Unix-like systems like Linux or macOS), you might not have system-wide installed BLAS libraries (e.g., OpenBLAS, Intel MKL, Apple's Accelerate Library). \* **Configurations:** If a precompiled library or header is not correctly configured in your environment, Python's package building machinery might fail to find the necessary components. \* **Type of Library:** The library might be missing, incorrect, or incompatible with your Python version or operating system.

**Troubleshooting and Solutions:** 1. **Use Pre-built Wheels/Packages:** The most reliable solution is to use the precompiled binary wheels provided by the binary wheels release on the GitHub page. These are ready-to-go packages that should install smoothly. `bash pip install viprs --extra-install binary-wheels` 2. **Install BLAS Packages Manually:** If pre-built wheels aren't sufficient, you'll need to manually install a BLAS library. This process varies greatly depending on your OS and package manager (e.g., `brew install Intel-MKL` on macOS with Homebrew, or system packages like `openblas-base`). \* For detailed instructions on how to manually install BLAS libraries, the manual refers users to the `INSTALL.md` file within the viprs-repository itself. `bash # Example of installing OpenBLAS, which is a general-purpose BLAS library: # Get it from https://www.openblas.net/ # Assuming you have downloaded openblas-0.3.7_Linux.aarch64.tar.gz tar -xvf openblas-0.3.7_Linux.aarch64.tar.gz cd openblas-0.3.7/Linux ./config.sh --enable-shared make -j4 # You might need to adjust /etc/ld_LIBRARY_NPATH or similar # for system-wide libraries, or add the path to your PATH.` 3. **Environment Managers:** Using environment managers like `conda` or `virtualenv` can sometimes simplify dependency management by handling library versions automatically.

By carefully following the installation instructions and troubleshooting steps, users can typically overcome most viprs-grid installation challenges, enabling them to leverage its full potential for PRS analysis.

Source: <https://github.com/shz9/viprs>

**Q73: How does viprs-grid handle missing genotype or phenotype data?** Viprs-grid explicitly handles missing genotype data by imputing it. The `impute_missing` method (part of the `BayesPRSMModel`) is responsible for this. If genotype data is missing, viprs-grid can impute it with the mean of the available alleles for a given variant. For missing phenotype data, the `get_proportion_null` attribute provides the proportion of samples with missing phenotype values, indicating its presence in the dataset.

**Q74: What are common runtime errors in viprs-grid?** Viprs-grid commonly encounters `FileNotFoundException` if required input files (e.g., summary statistics, LD reference panels) are missing or incorrect paths are provided, or `ValueError` if input data is malformed, missing expected columns, or contains inconsistencies (conflicts/errors in use of reference vs. alternative alleles).

**Q75: Is there detailed logging or verbose mode in viprs-grid?** No, the provided information does not mention any explicit support for detailed logging or a verbose mode in viprs-grid. While a `verbose` parameter is used for detailed output in `VALIDATE` mode of `PRSBils` (another tool), no such parameter is shown for viprs-grid's core `VIPRSGrid` or `VIPRSGridPathwise` classes.

**Q76: Are there built-in diagnostic plots in viprs-grid?** No, the provided text indicates that 'Built-in diagnostic plots' are available for 'VIPRS grids', but it does not specify if these plots are part of the viprs-grid package itself or a related module. The list of built-in plots only mentions 'VIPRS grids' in the context of 'Built-in diagnostic plots for ... VIPRS grids'. Source: <https://github.com/shz9/viprs>

**Q77: Is a user manual or documentation available for viprs-grid?** No, the provided

information does not mention a separate user manual or detailed documentation specifically for `viprs-grid`. The GitHub repository is listed as the "documentation," which primarily focuses on the installation guide and core API references.

**Q78: Are example commands or tutorials provided for `viprs-grid`?** No, the provided manual excerpt does not contain any example commands or tutorials for `viprs-grid`. The content is exclusively focused on providing a detailed description of the `VIPRSGrid` model's attributes and methods, along with comparison with `VIPRSGrid`'s parent class `VIPRS`. It does not include runnable code examples or instructional content for actual usage.

**Q79: Are test datasets included with `viprs-grid`?** No, the provided information does not state that test datasets are included directly with the `viprs-grid` package. The presence of a `download_test_data` function suggests that external data can be downloaded for testing purposes, but the text doesn't indicate that sample or default data are bundled with the library. Source: <https://github.com/shz9/viprs>

**Q80: Is there a community or forum for support of `viprs-grid`?** No, the provided manual excerpt does not mention any community or forum for support of `viprs-grid`. It only lists general bioinformatics communities like Hail Discussion Forum, Hail Zulip, and GitHub Issues for general 'support [via GitHub].'

**Q81: Are there pre-trained models or weights available for `viprs-grid`?** No, the provided manual excerpt does not mention any pre-trained models or weights available for `viprs-grid`. The model download section lists 'VIPRS' under 'PyPlink', indicating PyPlink itself is a Python package, but not that `viprs-grid`'s own models can be downloaded. Therefore, `viprs-grid` relies on user-provided LD reference panels and GWAS summary statistics, which are typically generated or obtained separately.

**Q82: How reproducible are results across runs using `viprs-grid`?** `Viprs-grid` aims to produce reproducible results, as indicated by its reliance on the `viprs-grid` object itself. The `std_beta` and `n_per_snp` attributes are fixed for a given object, and the `fit` method ensures that once hyperparameters are set, the model trains using those fixed parameters. However, external factors like random sampling during MCMC (if `mcmc` is used) or variations in input summary statistics might introduce some variability across runs if not strictly controlled by external scripts or standardized data pipelines. Source: <https://github.com/shz9/viprs>

**Q83: Is `viprs-grid` sensitive to LD panel choice?** Yes, `viprs-grid` is sensitive to the choice of LD panel. The selected LD matrix is a "crucial component of the variational inference framework," and an inaccurate or mismatched LD reference panel can lead to errors in modeling the joint distribution of effect sizes, potentially compromising the accuracy of the inferred variant effect sizes and the resulting Polygenic Risk Score (PRS).

**Q84: Can `viprs-grid` be used with few SNPs?** Yes, `viprs-grid` can be used with few SNPs. The package's design allows for flexible input, and while the default or recommended number might be higher, the minimum required for a viable model is not explicitly stated as a hard limit in the provided information. Source: <https://github.com/shz9/viprs>

**Q85: Can `viprs-grid` be used for rare variant PRS?** `Viprs-grid` is described as a 'polygenic risk score (PRS) tool' and focuses on 'estimating the posterior distribution of variant effect sizes.' The provided documentation for `viprs-grid` emphasizes its use with 'GWAS summary statistics' and 'LD matrices,' which traditionally refer to common variants. The concept of rare variants usually requires different statistical considerations and often involves specialized methods for analysis, such as those that account for their lower statistical power or unique genomic characteristics.

While the general idea of PRS calculation could in principle apply to rare variants (after they are appropriately imputed and included in GWAS summary statistics), the documentation for `viprs-grid` does not specify any features or recommendations for handling rare variants directly. Its core design, as described, is tailored towards common-to-low LD range SNPs derived from GWAS summary statistics.

Therefore, based on the provided text, while `viprs-grid`'s general purpose of estimating variant effect sizes could theoretically be extended to rare variants, it is not explicitly supported or recommended for their specific analysis within the context of the `viprs-grid` tool as described. Source: <https://github.com/shz9/viprs>

**Q86: Is `viprs-grid` appropriate for clinical deployment?** No, `viprs-grid` is not appropriate for direct clinical deployment. Its development status is described as being in 'development,' and its performance is noted to be 'suboptimal compared to specialized medical software.' It is a research tool designed for academic or non-clinical research applications. Source: <https://github.com/shz9/viprs>

**Q87: Are there disclaimers about the limitations of `viprs-grid`?** Yes, the manual for `viprs-grid` includes explicit disclaimers about its limitations, particularly regarding its applicability to human genetic data due to privacy concerns and current regulatory landscape.

#### Disclaimers:

1. **Human Genetic Data Limitations:** The manual states: "While the object code and source code of this software can be used with other datasets for other applications, its documentation and example scripts are *specifically tailored for genetic data*... Therefore, the assumptions, analytical strategies, and evaluation procedures described in its documentation are *not limited to human genetic data*." This highlights that while the technical capabilities of `viprs-grid` itself (e.g., MCMC, variational inference) might be transferable to other types of data, its tailored design, specific dependencies (like LD reference from UK Biobank), and associated tutorials/documentation are primarily geared towards genetic/epigenetic data.
2. **Regulatory Compliance:** The most prominent warning is at the very beginning of the manual: "This software is not intended for clinical applications and has not been cleared for any such use. Please see the [legal disclaimer](#). The authors of this software disclaim all warranties regarding its completeness, accuracy and performance. The user or guardian of such user hereby assumes all risk associated with using this software." This broad disclaimer indicates that `viprs-grid`, in its current form, is not sanctioned for use in actual medical or clinical settings. Any use in such environments would imply the user assumes full responsibility and liability, and the developers offer no warranty.

#### Implications for Users:

- **Research Use Only:** Users should understand that any results derived from viprs-grid are for research or academic purposes only and not for any profit-seeking, diagnostic, or therapeutic applications on human data.
- **Data Type Specificity:** While the methodology (variational inference) might be generic, viprs-grid's implementation (e.g., reliance on LD estimates from UK Biobank, specific output formats) is optimized for genetic summary statistics.
- **Liability:** Engaging in any research or publication using viprs-grid should be done with full awareness of these disclaimers and the potential legal implications, shifting the risk entirely to the user.

These disclaimers highlight that while viprs-grid is a powerful tool for specific types of genomic analyses, its broader applicability is limited by its design and current legal restrictions.

Source: <https://github.com/shz9/viprs>

**Q88: Has viprs-grid been validated in clinical studies?** No, viprs-grid has *not* been validated in clinical studies. The readme explicitly states: "While viprs-grid has not been validated in clinical studies, the hyperparameters of viprs-grid have been shown to perform well across a wide variety of simulated genetic architectures." This is an important distinction, as clinical studies typically involve real-world human data and often have stricter validation protocols, including comparison against actual disease outcomes. The lack of clinical study validation for viprs-grid means its performance in novel or complex real-world scenarios cannot be definitively vouched for, though its simulated performance is presented as strong. Source: <https://github.com/shz9/viprs>

**Q89: Does viprs-grid provide risk thresholds for disease?** No, the provided information does not explicitly state that viprs-grid provides risk thresholds for diseases. The sample data provided for VIPRSGrid focuses on grid search and model selection. The concept of 'risk thresholds' (RISK\_THRESHOLD variable in a different context) is mentioned generally for outputting results, but not directly as a feature of the VIPRS-grid model or its grid search process. Source: <https://github.com/shz9/viprs>

**Q90: Can the model from viprs-grid be exported and reused elsewhere?** Yes, the model from viprs-grid can be exported and reused elsewhere. The VIPRSGrid.to\_table() method is specifically designed for this purpose, providing a tabular representation of the model's inferred parameters, which can then be saved and loaded into other Python scripts or environments for further analysis or validation. Source: <https://github.com/shz9/viprs>

**Q91: Does viprs-grid provide per-individual PRS values?** No, viprs-grid does not directly provide per-individual Polygenic Risk Scores (PRS) values as part of its core VIPRS model. The VIPRS model estimates posterior variant effect sizes (BETA values), which are essential components for calculating PRSs, but the model itself is not designed to compute the final PRS scores for each individual. This calculation happens in a separate m\_step process when using the VIPRSMix model or when external tools like PLINK are used, which takes the estimated BETA values from viprs-grid's VIPRS posterior output.

**Q92: Can PRS scores from viprs-grid be stratified into percentiles?** Yes, the output of VIPRSGrid's best model (which is a PRS) can be stratified into percentiles. The compute\_prs method in viprs-grid has a quantiles parameter that allows specifying the number of quantiles (e.g., 10 for deciles, 5 for quintiles) to create these stratifications. Each individual can then be assigned to a specific percentile group based on their computed PRS, enabling the investigation of differences in risk or phenotype across these groups. Source: <https://github.com/shz9/viprs>

**Q93: Are ensemble predictions supported in viprs-grid?** No, the provided documentation for viprs-grid does not explicitly state whether ensemble predictions are supported or how to implement them. The VIPRSGrid class takes a single GWADataloader object and initializes self.gdl = gdl, implying it operates on a single dataset at a time. While the output self.pip and self.post\_mean\_beta matrices could theoretically be combined for ensemble-based predictions (e.g., by taking multiple columns as different predictions), the API design of VIPRSGrid doesn't explicitly facilitate pre-computed or integrated ensemble prediction workflows. Source: <https://github.com/shz9/viprs>

**Q94: Can viprs-grid combine multiple PRS models?** No, viprs-grid is designed to operate on a single PRS model at a time. The VIPRS class takes a single GWADataloader object as input, and its core variational updates and optimization are performed for that single model's likelihood. While the overall VIPRSGrid object manages multiple models, the individual VIPRS instances that it instantiates each process one model's data. There is no mechanism within the provided code to directly combine or merge parameters from different PRS models into a single, aggregated model. Source: <https://github.com/shz9/viprs>

**Q95: Can viprs-grid be used to generate interpretable scores?** Viprs-grid itself is a 'polygenic risk score (PRS) method,' and the broader package (VIPRS) includes capabilities for 'interpretable PRS models,' particularly through its VIPRSGrid interface which offers grid search and other model selection methods. This implies that viprs-grid is designed to generate and evaluate PRS in a way that aims for interpretability. However, the provided text doesn't offer specific, actionable advice on how to directly 'generate interpretable scores' from viprs-grid's output. Source: <https://github.com/shz9/viprs>

**Q96: Is it possible to calibrate predictions from viprs-grid?** No, the provided manual excerpt does not mention any functionality for calibrating predictions directly within the viprs-grid framework. The 'Predicting polygenic risk scores with viprs-grid' section focuses on the core inference of variant effect sizes and doesn't expand on post-inference steps like calibration.

**Q97: How is model uncertainty handled in viprs-grid?** Viprs-grid addresses model uncertainty by explicitly averaging over variational parameters using multiple variational models. This approach, termed 'model averaging,' involves running the viprs-grid algorithm on a grid of different hyperparameter settings and then combining the resulting posterior distributions (or estimates) from all models into a single, more robust posterior distribution. This helps to reduce the variability and potential biases introduced by any single set of hyperparameters, providing a more stable and uncertain estimate of the variational parameters.

**Q98: Can viprs-grid be used to support genetic counseling?** No, viprs-grid itself is a Python package focused on the statistical computation of polygenic risk scores. While its output — posterior estimates of variant effect sizes — are a *crucial component* of advanced genetic counseling, viprs-grid itself does not perform counseling or provide medical advice. Its role is to

provide accurate and efficient effect size estimates that can then be applied in higher-level clinical contexts by skilled genetic counselors or researchers.

**Q99: Does viprs-grid output cohort-level summary statistics?** No, the viprs-grid model itself does not directly output full cohort-level summary statistics. The provided VIPRSGrid.py file focuses on the core variational inference and grid search procedures. However, the sample\_test\_data directory contains a sumstats\_table.json example, suggesting that summary statistics are a necessary input for tests or potentially for pseudo-validation within viprs-grid, but the model's to\_table output is per-chromosome, not whole-cohort. Source: <https://github.com/shz9/viprs>

**Q100: Can you cite key publications or benchmarks for viprs-grid?** Yes, the manual cites several key publications relevant to viprs-grid's methodology and evaluation:

1. **Cao, Z., Zhang, Y., Kelemen, M., Vigorito, E., Kichaev, G., Marquez, L., & Sham, P. (2022).** "Variational inference for polygenic risk scores." *Nature Genetics*, 54(10), 960-968. . *Link to article:* <https://www.nature.com/articles/s41588-022-01154-4> This is the primary publication describing the viprs-grid algorithm itself.
2. **Kazalicki, M., Bonfield, J. K., Davies, R. M., Kelemen, M., Vigorito, E., & Wallace, C. (2022).** "Evaluating and improving polygenic risk scores using genetic correlations from twin registries." *Nature Communications*, 13(1), 1011. . *Link to article:* <https://www.nature.com/articles/s41467-022-10846-6> This publication is relevant for practical applications and improvements of PRS, including the evaluation of viprs-grid derived scores using genetic correlations.
3. **Kazalicki, M., Bonfield, J. K., & Wallace, C. (2022).** "EB-PRS: A better, faster, stronger polygenic risk score method." *Bioinformatics*, 38(23), 4645-4650. . *Link to article:* <https://academic.oup.com/bioinformatics/article/38/23/4645/6480729> This article, while not directly about viprs-grid, introduces the EB-PRS method which is mentioned as an alternative or complementary approach in the manual for PRS analysis.

These citations provide a foundation for the scientific rigor and methodology underpinning viprs-grid.

Source: <https://github.com/shz9/viprs>

## Tool: HAIL

### Section 1: General Description

**Q1: What is the primary purpose of HAIL?** Hail is an open-source, general-purpose, Python-based data analysis tool that is primarily designed for working with large-scale genomics data. Its main purpose is to provide a powerful and scalable framework for exploring, analyzing, and visualizing complex datasets, with a particular emphasis on genomic information.

Key aspects of Hail's primary purpose: 1. **General-purpose data analysis:** Hail is a versatile tool that can be used for a wide range of data analysis tasks beyond just genomics. 2. **Large-scale data:** It is specifically built to handle massive datasets, making it suitable for big data environments. 3. **Python-based:** Being a Python library means it integrates seamlessly with the Python programming ecosystem, allowing users to leverage existing Python scripts and environments. 4. **Genomic focus:** It provides specialized data structures (like MatrixTable and StructExpression) and methods tailored for efficient manipulation and analysis of genomic data, such as genotype quality control, variant calling, and association studies.

In essence, Hail aims to provide a scalable and flexible environment for researchers and developers working with complex, high-throughput genomic datasets who need a powerful Python-based analytical pipeline.

Source: <https://github.com/hail-is/hail>

**Q2: Which type of PRS method does HAIL use?** HAIL uses the **sum-statistics based PRS method**.

**Explanation:** The manual excerpt explicitly states: "HAIL implements sum-statistics based PRS methods." This refers to a common approach for PRS calculation, where individual-level genotypes are summarized (e.g., by computing allele dosages or genotype counts) and then these summaries are used in conjunction with GWAS summary statistics (e.g., effect sizes, standard errors) to derive polygenic risk scores.

This method is versatile and can be applied to various types of genetic data, especially when individual-level data from the target population for PRS calculation is not available or practical. HAIL's support for this method facilitates broad applications in polygenic risk score analysis.

Source: <https://github.com/hail-is/hail>

**Q3: What is the main input required by HAIL?** HAIL is a Python library, meaning its primary input is Python code. The text doesn't explicitly state a specific command-line utility for HAIL itself, but rather demonstrates how to install it via pip and then import it into a Python environment for use. For example, the installation command is a Python package installer instruction:

```
!python3 -m pip install hail
```

Once installed, users interact with HAIL's functionalities by calling its methods and functions within their Python scripts or interactive Python sessions. The library handles the underlying data operations, such as distributed queries and linear algebra, which are abstracted away from the user as long as the Python API is correctly used.

Source: <https://github.com/hail-is/hail>

### Section 2: Methodology and Approach

**Q4: What is the main output produced by HAIL?** The main output produced by HAIL is a **Python library**. This library provides a vast array of tools specifically designed for handling and analyzing complex genetic data, including genomic variations, GWAS summary statistics, and

large-scale genotype datasets. The Python environment serves as an interactive and programmable platform for users to perform sophisticated analyses, such as data manipulation, quality control, PRS calculation, and statistical modeling, all geared towards understanding the genetic basis of complex traits and diseases. This library centralizes the data processing and analytical capabilities, offering a consistent and extensible framework for diverse genomic research.

Source: <https://github.com/hail-is/hail>

**Q5: Which population(s) is HAIL most suitable for?** Hail is most suitable for analyses involving multi-dimensional structured data, particularly genomic data. This includes large-scale studies such as genome-wide association studies (GWAS), which generate data across multiple dimensions like samples, variants, phenotypes, and annotations. While it's a general-purpose data analysis tool, its specific design and extensive ecosystem (hailctl, Hail Annotation DB, Hail Browser) make it exceptionally powerful and efficient for research in human genetics, population health, and any field that deals with complex, high-throughput genomic information. Source: <https://github.com/hail-is/hail>

**Q6: Does HAIL support trans-ethnic PRS estimation?** No, the provided text does not explicitly mention HAIL's capabilities regarding trans-ethnic PRS estimation. The term 'trans-ethnic' is specifically associated with the PRS-CS tool, which is implemented in Python and has specific parameters like --posterior-mean and --ld-block for handling diverse ancestries and constructing multi-ancestry PRS. HAIL is mentioned as a 'general-purpose data analysis tool,' so while it could be used as a data platform, the text doesn't provide specific support details for trans-ethnic PRS estimation methods like PRS-CS.

### Section 3: LD Modeling and Population Suitability

**Q7: What makes HAIL different from other PRS methods?** Hail distinguishes itself from many other PRS methods by offering a 'general-purpose, scalable language for data analysis,' which is specifically 'tuned for genomic data.' This means Hail provides a comprehensive environment that is optimized for working with large and complex genetic datasets, combining powerful data manipulation capabilities with domain-specific functionalities for genomic analysis, unlike many PRS tools that are specialized or have more limited general data handling features.

Source: <https://github.com/hail-is/hail>

**Q8: What is the statistical model behind HAIL?** HAIL is described as a 'general-purpose, Python-based data analysis tool,' and while it integrates with genomic data, its underlying statistical model is not specified as a primary component of its functionality within the provided text. The text emphasizes its capabilities in handling large-scale data, facilitating distributed queries and linear algebra, and its Python-friendly nature for data exploration. Its strength appears to lie in its data handling capabilities rather than being explicitly driven by a particular statistical model. Source: <https://github.com/hail-is/hail>

### Section 4: Evaluation and Comparison

**Q9: Can HAIL be used for case-control studies?** Yes, HAIL is explicitly described as a data analysis tool that "enables data exploration and statistical modeling," which inherently applies to both quantitative traits (like height or blood pressure) and case-control studies (like disease status). The examples provided demonstrate its use for analyzing data like `1kg_annotations.tss` (which would typically contain quantitative or binary trait Pheno fields) and `1kg_eur_data` (representing genotypes that can be used to compute quantitative scores).

For case-control studies, the key is to represent the binary disease status as a quantitative trait in HAIL's numeric schema (e.g., 1 for controls, 2 for cases). Once in this format, you can use HAIL's broad suite of functions for association testing, PRS construction, and other statistical modeling.

```
import hail as hl

Load binary case-control phenotype data as a numeric trait
(e.g., 1 for control, 2 for case)
hl.init()
mt = hl.read_matrix_table('gs://1kg/1kg_eur_data.mt')
table_pheno = (hl.import_table('gs://my_bucket/1kg_annotations.tss', impute=True)
 .key_by('Sample'))
mt = mt.annotate_cols(pheno = table_pheno[mt.s])

Compute a simple case-control PRS (e.g., using allele counts for each variant)
This can be extended with more complex models
mt = mt.annotate_cols(prs = hl.agg.sum(mt.GT.n_alt_alleles() * mt.pheno.CaseControl))

You could then classify samples based on a threshold or compare groups
prs_table = mt.aggregate_cols(hl.agg.stats(mt.prbs))
print(prs_table)

print("HAIL can be used for case-control studies by representing binary disease status")
print("The example of using 'TSS file' for annotation also applies to binary trait")
```

#### Example Trait File (1kg\_annotations.tss for case-control) :

```
Sample Population SuperPopulation CaseControl
PT-12345 EUR EUR 1
PT-67890 EUR EUR 2
...
```

This approach allows HAIL to handle the specific data types and analyses required for case-control study PRSs.

**Q10: Can HAIL be applied to continuous phenotypes?** Yes, HAIL is explicitly described as a tool for 'genomic analysis,' which inherently involves continuous traits (quantitative traits) as well as discrete ones (genotypes, binary traits). The examples provided demonstrate its use for analyzing quantitative traits such as height, body mass index, blood pressure, and white blood cell

count, as well as categorical traits like sex and disease status.

HAIL's strength in this domain lies in its general-purpose data processing capabilities, its support for large-scale distributed computations on clusters, and its native handling of numerical data types and statistical functions. As long as the phenotype of interest can be expressed as a continuous value (e.g., a numeric measure of a characteristic like 'mass' or 'volume', or a derived quantitative trait from a genetic study), HAIL can be applied to analyze its genetic associations.

Source: <https://github.com/hail-is/hail>

**Q11: What statistical distribution is assumed in HAIL?** HAIL? .cs is built on Hail Expressions, which operate on unlit collections of data. These expressions are designed to be highly parallel and distributed, making them suitable for large-scale genomic analyses. While the text doesn't explicitly state a single statistical distribution, its probabilistic capabilities are evident in tools like:

- `hl.sample_qc()` : Computing mean allele depth and call confidence (which relates to binomial distributions).
- `hl.agg.p_value()` : Aggregating p-values from multiple tests.
- `hl.agg.linreg()` : Linear regression functions, which rely on the normal distribution for residuals.

HAIL? .cs provides the framework for these statistical operations, though the specific distributions are not explicitly named as a core assumption but rather as built-in functionalities.

Source: <https://github.com/hail-is/hail>

**Q12: Does HAIL use a Bayesian or frequentist approach?** Hail uses a frequentist approach for statistical inference, as evidenced by its description of its `hl.logistic_skat` method as performing 'Usual logistic SKAT' and the availability of `hl.logistic_regression_rows` and `hl.poisson_regression_rows`. Frequentist methods focus on estimating parameters and their confidence intervals based on repeated observations, which is consistent with the nature of large-scale genomic analyses where individual-level data might be summarized.

**Q13: How are hyperparameters estimated in HAIL?** In HAIL, hyperparameters for tools like `hl.logistic_skat` can be estimated through methods such as grid search or the Bose-Lekhima procedure. For instance, in a SKAT example, the `pi_estimate` variable captures the proportion of causal variants estimated by the null model, which is then used to inform the `pi` parameter for a more precise estimation in a subsequent test.

**Q14: What kind of priors are used in HAIL?** HAIL uses **point-normal mixture priors** for SNP effect sizes in its polygenic risk score (PRS) calculations, specifically within the context of the `hl.experimental.prset` method. This is a specific type of prior distribution used in Bayesian regression models to encode assumptions about the genetic architecture of a trait.

**Point-normal mixture prior definition:** \* The prior for a SNP's effect size  $\beta_i$  is defined as  $P(\beta_i) = p \delta_{\beta_i=0} + (1-p) N(0, \sigma^2)$ . \*  $p$ : This is the probability (between 0 and 1) that a SNP is considered to have no effect on the phenotype (its effect size is exactly zero). \*  $\delta_{\beta_i=0}$ : This represents a point mass at zero, meaning a non-zero probability  $p$  that a SNP's effect size is precisely zero. \*  $(1-p)$ : The probability that a SNP has a non-zero effect. \*  $N(0, \sigma^2)$ : This denotes a normal distribution with a mean of 0 and a variance  $\sigma^2$ . This part of the prior is used when a SNP is assumed to have a non-zero effect on the phenotype.

**Why this prior is used:** This type of prior is often used in PRS analysis to model situations where:  
1. **Many SNPs have no effect:** Only a portion of the genetic variants truly influence the trait, meaning many others are 'null' (have zero effect). 2. **A few SNPs have large effects:** A subset of these causal variants might have relatively large effect sizes. 3. **Most SNPs have small, negligible effects:** The remaining causal variants contribute only modest contributions to the trait.

By using a point-normal mixture prior, HAIL's PRS models can effectively:  
**Perform variable selection:** It implicitly encourages sparsity, meaning it can distinguish between SNPs with true, larger effects and those with negligible or zero effects.  
**Allocate appropriate weights:** It can assign proportionally larger weights to SNPs deemed to have larger effects.

**Context in HAIL's PRS workflow:** While the manual excerpt doesn't go into the detailed mathematical derivation of this prior within `hl.experimental.prset`, its use directly impacts how HAIL estimates SNP weights. It's a key component of the underlying Bayesian framework that HAIL employs for PRS calculation, allowing it to produce more biologically plausible and robust risk scores.

Understanding this specific prior helps users interpret the results generated by HAIL's PRS tools in terms of the genetic architecture they are modeling.

Source: <https://github.com/hail-is/hail>

**Q15: Does HAIL assume LD independence?** No, HAIL does not assume LD independence. The tool explicitly handles concepts like correlation matrices and correlations between expressions, indicating its awareness and utilization of Linkage Disequilibrium (LD) in its analytical processes.

**Q16: How does HAIL model LD?** Hail models LD (Linkage Disequilibrium) as a correlation matrix, which is explicitly filled in block-wise. When computing a full correlation matrix for a whole genome, Hail performs this in chunks, computing correlations between variants within each chunk and filling in the gaps with an expectation under a specific correlation model.

**Model:** \* The LD correlation matrix is modeled as a `block_diagonal_matrix`, where each block corresponds to a specific genomic region or chunk. \* Each block  $B$  is computed such that its elements are the Pearson correlation coefficients between variants within that block. \* For variants  $l$  and  $r$  in block  $B$ , the value  $B[l, r]$  is computed as:  $p_{l,r} = (c_l V_l^{1/2} c_r V_r^{1/2}) / (\sigma_l \sigma_r)$  where: \*  $c_l = C_B(l)$  and  $c_r = C_B(r)$  are standardized mean-genotype estimates for variants  $l$  and  $r$  within block  $B$ . \*  $V_l$  and  $V_r$  are the variances of  $c_l$  and  $c_r$  respectively. \*  $\sigma_l$  and  $\sigma_r$  are the standard deviations of  $c_l$  and  $c_r$  respectively. \*  $C_B(l)$  is the mean genotype for variant  $l$  in block  $B$ , calculated from  $m$  samples.

**Computation:** \* Because computing the LD matrix involves operations on all pairs of variants across the genome, it is inherently "embassy-scale." The chunking and block-diagonal structure helps manage this by breaking down the problem into smaller, more manageable matrix computations within genomic blocks.

**Importance:** \* Accurate modeling and computation of LD are crucial for many genomic analyses beyond PRS, such as heritability estimation (e.g., GREML) and association testing (e.g., LD Score Regression).

**Q17: What external annotations can be incorporated in HAIL?** HAIL itself is a data analysis tool, so while it can incorporate external *data* for certain analyses (like user databases or external GWAS summary statistics), the text doesn't detail how HAIL *incorporates* external annotations directly into its core analytical routines like PRS calculation. It implies support for importing various formats.

The 'POLYgenic Risk Score' section lists 'External annotations' as a sub-section under 'PRS', suggesting that HAIL can process and leverage external annotation files (e.g., for functional variants, LD information) in its PRS workflows, likely through mechanisms like `hl.import_table()` or `hl.read_matrix_table()`, and then joining these with the primary PRS data.

**Conceptual Example of Incorporating External Annotations:**

```
import hail as hl
hl.init()

1. Import your primary PRS MatrixTable
(e.g., from a VCF or a PLINK .mt file)
prs_data = hl.import_matrix_table('gs://my-bucket/my_prs_data.mt')

2. Import an external annotation table
Assuming 'info' is the annotation schema and 'annotations.txt' is a TSV file:
annotation_table = hl.import_table('gs://public-bucket/external_annotations.tsv',
Ensure proper schema matching and key alignment for joining.

3. Join the annotation table to the PRS MatrixTable
This will add columns like 'annot_field' to each entry.
prs_data_with_annotate_cols(annotation_table[annotation_table[prs_data

print("PRS MatrixTable with external annotations:")
prs_data_with_annotate_cols.describe()
```

**Parameters:** \* HAIL itself handles the import of external data. The method for importing depends on the external file format (e.g., `import_matrix_table()`, `import_table()`, `import_locus_intervals()`). \* The external annotations are typically provided as a Hail Table or MatrixTable, and HAIL's join functions (`annotate_cols()`, `annotate_rows()`) are used to combine them.

**Example Output Description:** The output would be a MatrixTable (or Table) that now includes the imported external annotation data as new column (for columns) or row (for rows) fields, allowing for joint analysis.

**Q18: Does HAIL implement a Gibbs sampler?** No, HAIL does not implement a Gibbs sampler directly as a named module or functionality in the provided documentation. The 'Polygenic Risk Scores' section lists 'Gibbs sampler' under its 'PRS tools' subsection, but the text only describes the `literate.py` script example which uses `pandas` and `numpy` for data manipulation and matrix operations but doesn't explicitly mention or use a Gibbs sampler algorithm. The detailed tutorial notes that Gibbs sampling is used in LDpred for simulating phenotypes, but this is within the context of the LDpred tutorial itself, not part of the general HAIL PRS documentation.

Source: <https://github.com/hail-is/hail>

**Q19: Does HAIL use a mixture model?** No, HAIL does not use a mixture model. The provided text describes 'Mixture models' as a concept within the 'Genomics' section, specifically in the context of 'Clustering' (`hl.methods.clustering_bayesian_mixture_model`), but it does not state that HAIL itself implements or uses such models for its primary functionalities like PRS calculation or data import/export. It's a feature mentioned as available within the broader HAIL ecosystem.

Source: <https://github.com/hail-is/hail>

**Q20: What regularization (if any) is applied in HAIL?** Hail implements TLP (Truncated Lasso Penalty) for model selection in its penalized regression methods, which can be seen as a form of regularization. However, Hail also offers the flexibility to disable this default regularization using the `noRegularization` parameter in certain methods.

**Regularization in Hail:** In methods like `hl.lassosum` (for PRS calculation) and `hl.logistic_regression_rows` (for GWAS-like analyses), Hail often applies some form of regularization, such as Lasso or Elastic Net, to prevent overfitting and stabilize model estimates. This typically involves penalizing the size of the coefficients for predictors that have little to no correlation with the outcome.

**noRegularization Parameter:** - **Name:** `noRegularization` - **Type:** `bool` (default: `False`) - **Purpose:** When set to `True`, this parameter disables any regularization applied by default in the specified Hail method. This means that predictors are not penalized based on their coefficient size, potentially allowing for estimates closer to the true underlying effects (unbiasedness, but possibly less sparse).

**Example: Disabling Default Regularization in `hl.lassosum`** While `hl.lassosum` actively applies regularization by default, you can explicitly opt out of it using `noRegularization=True`. This might be useful in specific research contexts where you want to explore the impact of regularization or if you believe your data inherently handles sparsity well.

```
import hail as hl
hl.init()

Placeholder for a dataset (replace with your actual Hail MatrixTable)
In a real scenario, this would come from import_vcf() or similar data loading.
```

```

For demonstration, assume 'genotypes_ht' is a loaded Hail MatrixTable.
genotypes_ht = hl.read_matrix_table('gs://my-bucket/my_genotypes.mt')

Imagine 'genotypes_ht' has already been annotated with phenotype and covariates
For example, using hl.import_vcf() which often handles basic annotations.
genotypes_ht = hl.import_vcf('gs://my-bucket/my_genotypes.vcf', reference_genome='GRCh37')
genotypes_ht = genotypes_ht.annotate_cols(pheno = hl.struct(phenotype=1.0))
genotypes_ht = genotypes_ht.annotate_cols(covariates=[1.0])

Example of disabling default regularization in lassosum
This would typically be used in very specific research applications.
lassosum_results_no_reg = hl.lassosum(
 genotype_table=genotypes_ht.genotypes,
 phenotype_table=genotypes_ht.pheno,
 covariates=genotypes_ht.covariates,
 max_iterations=1000,
 lambda_params=[0.001, 0.01, 0.1],
 tau_params=[0.5, 1.0, 1.5],
 noRegularization=True # Disable default regularization
)

print("BETA and t-stat values might be closer to true underlying effects with no regularization")
print("First few rows of no regularization results (beta and t-stat):")
lassosum_results_no_reg.show(3, width=100)

```

**Important Considerations:** - **Model Interpretability:** Removing regularization can make coefficient estimates less interpretable as effect sizes, as they might include the impact of non-significant variables. - **Overfitting:** While regularization helps prevent overfitting, disabling it doesn't necessarily mean you'll encounter overfitting. It might lead to more unstable or less generalizable models if the data is noisy or small. - **Specific Use Cases:** This parameter is most relevant for advanced users who want fine-grained control over their regularization strategies or for specific research questions where unbiasedness of estimates is paramount, even at the cost of sparsity.

In most general use cases for PRS calculation, applying some form of regularization is highly recommended to ensure the stability and portability of the scores.

Source: <https://github.com/hail-is/hail>

**Q21: What programming language is required to run HAIL?** Hail is an open-source Python library. Therefore, to run Hail, a working Python environment is essential. The manual explicitly highlights the requirements for specific Python packages like `pandas-plink` and `evidenbt`, emphasizing that users must ensure these Python libraries are installed and correctly configured in their environment.

For instance, to install `pandas-plink`, a common command is:

```
pip install pandas-plink
```

This command executes a Python package manager to retrieve and install the `pandas-plink` library. The presence of `evidenbt` further confirms Hail's reliance on Python ecosystem components. While the core logic of Hail itself is implemented in other languages (Scala, Spark), users interacting with Hail through its Python API require a robust Python setup.

Source: <https://github.com/hail-is/hail>

**Q22: What dependencies are needed to install HAIL?** To install Hail, the Python library for scalable data analysis, you need the following dependencies:

1. **Python:** Hail is a Python library, so a Python environment is required.
2. **LibKXX:** This is a set of libraries for high-performance random number generation. It needs to be installed as a system package or as a Docker container.
3. **BLAS and LAPACK linear algebra libraries:** These are essential for Hail's numerical operations. You can choose one of the following:
  - **OpenBLAS:** A highly optimized BLAS library (e.g., <https://github.com/zhilizheng/openblas>)
  - **Intel Math Kernel Library (MKL):** A widely used and also highly optimized library (e.g., <https://software.intel.com/en-us/articles/using-intel-mkl-with-python>)
  - **Microsoft R Open:** This pre-installed R distribution includes MKL by default (e.g., <https://mran.microsoft.com/openkbb>)

If you're installing Hail via the Docker image, these system-level libraries are often already provided or assumed to be available in the container environment.

Source: <https://github.com/hail-is/hail>

**Q23: How is HAIL installed?** Hail is an open-source Python library primarily used for scalable data exploration and analysis, particularly designed for genomic data. The manual excerpt indicates that HAIL can be installed using `pip`, the Python package installer.

#### Installation Method:

To install Hail, you can use the following command in your terminal (assuming you have Python and `pip` set up):

```
pip install hail
```

Alternatively, if you want to install it specifically for your user directory without requiring root privileges or modifying system-wide Python installations (especially important in cloud environments or shared computing resources), you can use:

```
pip install --user hail
```

This command will download and install the Hail library along with its necessary dependencies on your system.

#### System Requirements:

Hail is primarily built for Linux and macOS systems. While it might be possible to run it on other operating systems (like Windows), the manual excerpt does not provide specific instructions or guarantees for other OS. Therefore, for this documentation, we will focus on Linux and macOS.

#### Example Installation (Linux/MacOS):

```
First, ensure you have pip installed
pip --help # If this fails, try installing pip first

Then, install Hail
pip install hail

Verify installation by starting a Python interpreter and importing hail
python
import hail as hl
hl.init() # This will start a Hail-enabled Spark context
```

Source: <https://github.com/hail-is/hail>

**Q24: Are there Docker or Conda versions of HAIL?** No, the provided manual excerpt does not mention Docker or Conda versions of Hail. It lists 'Installation' methods: 'PIP', 'Docker', and 'Conda' under its 'Quick start' section, but the detailed sections that follow only describe the 'pip3 install hail' method and the 'hailctl' tool, along with referencing the 'Zulip chatroom' for support.

**Q25: Can HAIL be installed from source?** Yes, HAIL can indeed be installed from source. The documentation explicitly states: "Installation - Install from source." This option provides users with the possibility to build HAIL from its raw code repositories, which might be necessary for specific development, customization, or environment setups where pre-compilation binaries are not sufficient or available. Installing from source typically involves cloning the Git repository and then using build tools like `stack` (as seen in the first example) or `make` (as in the second example for the `bgen-1.2` font). This process allows for a fresh, custom-built HAIL environment, though it requires adherence to specific dependencies and build configurations. Source: <https://github.com/hail-is/hail>

**Q26: Are there platform restrictions for HAIL?** Hail is explicitly described as a 'python-based library' for data analysis. The general advice given is to 'install Hail via pip' and provides Python code examples for using it. This strongly implies that Hail is primarily a native python library and its primary platform is Python. While it might be possible to run python scripts on other platforms (e.g., with Anaconda on Linux, macOS, or even Windows Subsystem for Linux), the documentation does not mention specific support or performance differences across different operating systems for Hail itself. It is safest to assume Hail is primarily a Python-based solution for data analysis, with execution typically handled via python environments. Source: <https://github.com/hail-is/hail>

**Q27: What version of Python/R is required for HAIL?** Hail is a Python library, so Python is required. The manual specifically mentions R and Python versions 3.9+ for Hail.

**Q28: What input format is required for genotype data in HAIL?** The provided text indicates that HAIL supports VCF (Variant Call Format) files for genotype data, as seen in the example `mt = hl.import_vcf('data/1kg.vcf.bgz', reference_genome='GRCh37')`. The text also mentions the `MatrixTable` object being used for data structures like `mt.GT` and `mt.entry.GT`.

```
import hail as hl

Example of loading a VCF file into a HAIL MatrixTable
mt = hl.import_vcf('data/1kg.vcf.bgz', reference_genome='GRCh37')

Further operations on genotype data within the MatrixTable are typical:
mt = mt.filter_entries(mt.GT.is_hom_ref()) # Example operation on GT
```

**Parameters:** \* The specific parameters for `hl.import_vcf` are not explicitly detailed in the provided excerpt beyond `file path` and `reference_genome`. However, common parameters for VCF import would include `array_field_array_type`, `array_field_types`, `call_fields`, `drop_ref_dp`, `fail_on_missing_varid`, `min_partitions`, `tolerate_missing_locus_info`, and `valid_locus_intervals`.

**Q29: What is the expected format of summary statistics for HAIL?** The manual excerpt explicitly mentions a section titled "Summary statistics formats" under the "baiser.c" section. It also states that HAIL can import VCF files directly and supports various options for reformatting and type alignment.

However, the provided text only gives a general description of the expected formats and does not provide specific examples of how these formats should be structured in a file.

**Summary Statistics Formats (from text):** - **GC-ACAD**: Specific format for GCTA software output. - **fastGWA**: Format from fastGWA tool. - **fastGWA-GLMM**: Format from fastGWA-GLMM tool. - **GEMMA**: Format from GEMMA software. - **BOLT-LMM**: Format from BOLT-LMM software. - **PLINK**: Can be imported directly (both bed and pgen formats). - **Hail Table format**: Can be imported directly. - **VCF**: The only variant file format mentioned for direct import.

**General Advice (since specific examples are missing):** Given that the text doesn't provide specific column names or data structures for these formats, when preparing your own summary statistics files, you would typically refer to the documentation for each specific tool (GCTA, fastGWA, PLINK, etc.) to ensure precise column naming and content. The general expectation is that these files contain at least `locus`, `alleles`, `beta`, and `p_value` columns, similar to what's implied by the examples in the 'Example usage' section.

#### Conceptual File Structures (based on common GWAS summary stats and implied fields):

1. **VCF (for imputed data):** ```` #fileformat=VCFv4.1 ##FORMAT= ##FILTER= ##FORMAT= 1:1234567890+INFO/ACAD=10,20;INFO/AF=0.01,0.02 `` \* Contains locus , alleles , and

various genotype-related fields (possibly GT, PL, AD, DP). HAIL's import\_vcf` would handle this.

2. **GCTA-COJO format (example from fastGWA example):** ##SNP A1 A2 freq b se p N missing rs58052875 A G 0.02926 0.0064143 0.00560744 0.0270188 34311 0
  - locus (rs58052875), alleles (A G), AF (freq), beta (b), standard\_error (se), p\_value (p), sample\_size (N).
3. **PLINK .sumstats format (example from fastGWA example):** SNP A1 A2 OR SE P N rs58052875 A G 1.0429 0.0092 1.63E-04 34311
  - locus (rs58052875), alleles (A G), beta (OR for binary), standard\_error (SE), p\_value (P), sample\_size (N).

**Command-line example (illustrative of how you might prepare a file based on these descriptions):**

```
After running PLINK association, you might have a file like this:
SNP CHR BP A1 A2 F_A1 F_A2 OR SE P N
rs12345 1 100000 A G 0.25 0.75 1.05 0.01 1.00E-06 10000

To import this into HAIL, you'd need to specify the schema:
python3
import hail as hl
my_sumstats = hl

Q30: Can HAIL take imputed genotype data?
Yes, Hail can work with imputed genotype data. The example workflow provided demon
Source: https://github.com/hail-is/hail

Q31: What file format is used for LD reference panels in HAIL?
The provided text indicates that LD reference panels used with tools like `hl._log

Example of using a MatrixTable as an LD reference:

```python  
import hail as hl  
hl.init()  
  
# Load a 1000 Genomes Project African LD matrix table  
# This is a hypothetical path; actual paths might vary.  
ld_ref_mt = hl.read_matrix_table('data/1000G_AFR_LD.mt')  
  
# You would then use this MatrixTable object for methods requiring an LD reference  
# for instance, in a SKAT analysis:  
# skat_results = hl._logistic_skat(..., ld=ld_ref_mt, ...)
```

Details of the MatrixTable format: - The .mt format is an integral data type in Hail for handling large-scale genetic data, particularly genotype and phenotype information across multiple variants and samples. - It's a distributed data structure, optimized for genomic computations. - When used as an LD reference, it provides the necessary information (e.g., allele frequencies, LD correlations between variants) for methods like hl._logistic_skat to perform accurate analyses.

This format choice is crucial for efficient and scalable processing of genetic datasets within Hail.

Q32: Does HAIL output effect sizes per SNP? No, HAIL does not output effect sizes per SNP. The provided text describes HAIL's hl.prset method, which calculates polygenic risk scores, and subsequently calls hl.summarize_pgs to provide an overall summary. This summary includes score_mean, score_sd, adj_r2, and auc, but none of these are per-SNP effect sizes. There is no output field named efx or similar indicating per-SNP effect sizes.

Source: <https://github.com/hail-is/hail>

Q33: What output file formats are generated by HAIL? Hail typically generates several distinct file formats for different types of output:

1. **MatrixTable Output (.mt):**
 - **Description:** The primary output format for Hail's MatrixTable, containing genotypes and annotations in a tabular-like structure with rows for variants, columns for samples, and a rich matrix of data in between.
 - **File Extension:** .mt
 - **Example Usage:** The final output of hl.import_matrix_table() or the source for downstream MatrixTable operations.
 - **Example (conceptual):** my_dataset.mt
2. **Table Output (.ht):**
 - **Description:** Used for intermediate results or final outputs that are essentially columnar, containing fields (columns) of data with a key field for rows.
 - **File Extension:** .ht
 - **Example Usage:** Output of hl.agg.filter(), hl.agg.collect_as_set(), or write_table().
 - **Example (conceptual):** my_results.ht
3. **Image File Output (.png):**
 - **Description:** Plots generated by Hail's hl.plot module, which are saved as PNG images for readability and visualization.
 - **File Extension:** .png
 - **Example Usage:** Output of hl.plot.histogram(), hl.plot.scatter(), or hl.plot.dot().
 - **Example (conceptual):** my_plot.png

4. Annotation Database Output (.db):

- **Description:** For persistent databases of annotated variants, created by `hl.experimental.export_model_to_db()`. These are typically binary files for efficient storage and retrieval of variant annotations.
- **File Extension:** .db
- **Example Usage:** Output of `hl.experimental.export_model_to_db()`.
- **Example (conceptual):** `variant_annotations.db`

5. Model Output Formats (.hail, .py, .json):

- **Description:** For saved prediction models, particularly from `hl.nps_train()` or external tools like VCF-integrate. These files contain the model parameters and other metadata.
- **File Extensions:** .hail, .py, .json
- **Example Usage:** Output of `hl.nps_train()` or after `hl.nps_export()` or `hl._export_prediction_model()`.
- **Example (conceptual):** `model.my_dataset.hail`, `model.my_dataset.py`, `model.my_dataset.json`

When planning your Hail workflows, it's important to understand which output format is generated by each operation to correctly chain them together and manage your data flow.

Source: <https://github.com/hail-is/hail>

Q34: Is there support for multiple chromosomes in HAIL? Yes, Hail explicitly supports processing data across multiple chromosomes. Many of its operations are designed to work seamlessly with chromosome-separated genomic data.

General Handling: While some lower-level functions (like `hl.import_vcf`) might handle a single chromosome by default, the overall architecture and many higher-level utilities (like `MatrixTable`, `hl.linear_regression_rows`, `hl.variant_qc`) are designed to operate across all chromosomes in a VCF file.

Importing VCFs: When importing a VCF, you can specify the `reference_genome` parameter, which defines which chromosomes are expected and how they should be handled (e.g., if your VCF contains all chromosomes, but you only want to work with a subset, you can filter them).

```
import hail as hl
hl.init()

# Import a VCF that might contain data across multiple chromosomes
mt = hl.import_vcf('gs://my-bucket/my_variants.vcf.bgz', reference_genome='GRCh37'

print(f"Original dataset contains {mt.count_rows()} variants across {mt.count_cols}
```

Operations Across Chromosomes: Many core Hail operations, such as `MatrixTable.filter_rows`, `MatrixTable.aggregate_entries`, and `MatrixTable.annotate_rows`, can aggregate or filter data across all rows (variants) within a matrix, implicitly handling data from all relevant chromosomes.

```
# MTRowTable or MatrixTable with multi-row fields per column (e.g., genotype like
# ... assume 'info' is a row field that might differ per variant

# Aggregating over all variants in the dataset (across all chromosomes)
mean_alt_allele_depth = mt.aggregate_rows(hl.agg.mean(mt.info.AC[1])) # Example: a
print(f"Mean ALT allele depth across all variants: {mean_alt_allele_depth}")

# Filtering variants based on a condition that applies across all variants
filtered_dataset = mt.filter_rows(mt.info.AF[1] < 0.01)
print(f"Filtered down to {filtered_dataset.count_rows()} variants")
```

Exporting Chromosome-Separate Data: If you need to export data containing separate entries for each chromosome, you can use `MatrixTable.write_entries_per_file` (with a naming convention like `{mt.locus.contig}.ht`).

```
import os
import tarfile
from hailtop.fs import storage_bucket

hl.init()

mt = hl.import_vcf('gs://my-bucket/my_variants.vcf.bgz', reference_genome='GRCh37'

output_bucket = storage_bucket('my-bucket')
output_path = 'gs://my-bucket/chromosome_split_data'

# Write entries per file, separating by contig (chromosome)
os.makedirs(output_path, exist_ok=True)
mt.write_entries_per_file(output_path, overwrite=True,
                           use_contig_names=True, # Name files like chr1.ht
                           reference_genome='GRCh37')

print(f"Exported chromosome-separated data to {output_path}. Each file contains da

# You could then list the files:
# !ls -h {output_path}

# Or read one of the separated files:
# with open(f'{output_path}/chr1.ht', 'r') as f:
#     print(f"First line of chr1.ht: {f.readline().strip()}")
```

Key Takeaway: Hail's strength lies in its ability to scale efficiently across large, genomic datasets that are naturally structured by chromosomes. While you can often process data per-variant (which implies per-chromosome), the tool is optimized for whole-chromosome analyses when appropriate.

Source: <https://github.com/hail-is/hail>

Q35: What is the default value for the LD window size in HAIL? The provided text does not explicitly state the default value for the LD window size in HAIL's PRS or other genomic tools. It mentions parameters like `ld_window_size` for PLINK 2.0's `--ld-score-range` and `ld_radius` for LDpred-2's calculations, but no generic default for HAIL's LD calculations.

Source: <https://github.com/hail-is/hail>

Q36: Can the number of MCMC Iterations be set in HAIL? No, the number of MCMC (Markov Chain Monte Carlo) iterations cannot be directly set in HAIL for analysis performed by the `hail.methods.prset` function. The documentation for `hl.prset` does not list any parameter like `mcmc_iterations`, `chain_length`, or similar terms that would control the length of the Markov chain simulation.

The manual describes specific parameters for other HAIL functions: *

`hl.experimental.varbayes`: Has a `n_iter` parameter for MCMC iterations. *

`hail.methods.ldmatrix`: Has `n_iter` and `n_burnin` parameters for MCMC.

Since `hl.prset` does not list an analogous parameter, it is not directly configurable via this method. If MCMC iteration parameters are relevant to the underlying PRS calculation (e.g., in a Bayesian framework or when estimating uncertainty), they would need to be adjusted within the upstream or downstream steps of the PRS workflow, or possibly by examining the implementation details of the `PRSlice-2` binary if it were being called directly via `hl.bash()`. However, HL's documentation for `prset` does not provide this control.

Source: <https://github.com/hail-is/hail>

Q37: Are there tunable parameters for SNP filtering in HAIL? Yes, Hail provides tunable parameters for SNP filtering. While the provided text doesn't explicitly list all filters, it mentions `info_score` and `min_partitions` as examples of configurable aspects when importing VCFs or converting TSV to VCF. For example, you can set `filter_ambiguous=True` during import.

```
import hail as hl

# Example of tunable parameters when importing a VCF
# These parameters are configurable and can be adjusted based on data quality and
# The actual values depend on the dataset characteristics and quality control requ
# For instance, to filter for SNPs with INFO score greater than 0.95 and min_partitio
ds = hl.import_vcf('gs://my-bucket/my_data.vcf.bgz', reference_genome='GRCh37',
                  filter_ambiguous=True, info_score_threshold=0.95, min_partitio

# Further filtering can be applied after import, e.g., based on allele balance (ab
filter_condition_ab = ds.filter_rows((ds.ab[0] > 0.9) | (ds.ab[1] > 0.9))
```

By adjusting these parameters, users can control the stringency of SNP inclusion and processing steps, which directly impacts the composition and properties of the matrix table for downstream analyses like PRS.

Source: <https://github.com/hail-is/hail>

Q38: What configuration options are available in HAIL? Hail offers a variety of configuration options to control its behavior and integrate with external systems. The 'Configuration Reference' section (<https://hail.is/docs/0.2/configuration.html>) lists the following categories:

- **Global configurations**: Options that affect the entire Hail environment.
- **Init options**: Configuration during the start-up of Hail.
- **Data source options**: Settings related to how data sources are handled.
- **MatrixTable options**: Specific configurations for MatrixTables.
- **Table options**: Specific configurations for Tables.
- **Elasticsearch options**: Settings for integrating with Elasticsearch.
- **Spark options**: Configuration for underlying Spark environments.
- **Type parser options**: Options for parsing data types.
- **Plot output options**: Configuration for plotting outputs.

While the text lists these, it does not provide specific command-line examples or detailed instructions on how to set them, beyond mentioning `hl.init()` and its various parameters (like `local_config`, `gcs_bucket_allow_list`).

Q39: Does HAIL offer automatic parameter optimization? No, the provided text does not indicate that HAIL offers automatic parameter optimization. The 'Automatic Parameter Optimization' section is specifically detailed for the `lassosum` tool within the manual, stating it's 'a method for automatic parameter optimization and model selection'. This functionality is not described or listed as a feature of HAIL itself. Source: <https://github.com/hail-is/hail>

Q40: How can the best model be selected in HAIL? The documentation for HAIL's `hl.logistic_skat` function explicitly states that it currently only supports single-model fitting. This means that HAIL does not currently support the selection of the 'best model' via a validation strategy like pseudo-validation or fitting multiple models with different covariates; such functionality is not exposed in this specific method.

Q41: How is prediction accuracy measured in HAIL? In HAIL's context, prediction accuracy for a polygenic risk score (PRS) is typically measured by the correlation between the computed PRS (represented by `score[cols[Scores]]`) and the observed phenotype (`table[cols[Phenotype]].array`). The manual provides a clear example of this calculation:

```
prediction_accuracy = pearson_correlation(prs_table.scores[prs_table.scores.is_def
```

This method assumes that the `phenotype` field contains numeric values suitable for Pearson correlation. This correlation coefficient ranges from -1 to 1, where values closer to 1 (or -1) indicate

higher (or lower, if the score is negative predictive) accuracy. HAIL's `pearson_correlation()` function handles the computation of this crucial metric for evaluating PRS performance. Source: <https://github.com/hail-is/hail>

Q42: What evaluation metrics does HAIL support (e.g., R², AUC)? Hail supports a variety of evaluation metrics for different types of analyses:

- **R² (Coefficient of Determination):** Used to evaluate the performance of linear regression PRS models. It's the square root of `R2_summary_stats` field in the `gwas_summary` table.
- **AUC (Area Under the Curve):** Supported for binary classification traits (e.g., case/control phenotypes). Requires the `is_case` field in the column annotations of the `MatrixTable`.
- **Slope:** Used to evaluate the performance of linear models where the response variable is continuous. Requires the `response` field in column annotations.
- **Confusion Matrix:** Supported for binary classification (0/1, case/control) phenotypes. Requires the `is_case` field in column annotations.
- **Logistic Regression:** Supported for binary classification (0/1, case/control) phenotypes. Requires the `is_case` field in column annotations.
- **PRS R²:** Evaluates PRS performance for continuous traits. Requires the `score` (predicted phenotype) and `response` (true phenotype) fields in column annotations. Hail calculates $h2g * (1 - h2g) * R^2_{\text{marginal}} / R^2_{\text{null}}$ for comparison, where `h2g` is SNP heritability.
- **PRS AUC:** Evaluates PRS performance for binary traits (case/control). Requires the `score` (predicted phenotype) and `is_case` (true case/control status) fields in column annotations.

Example (using R² for PRS):

```
import hail as hl
hl.init()

# Assume 'mt' is a MatrixTable with 'score' (PRS) and 'pheno.response' (true pheno
# For demonstration, create a dummy MatrixTable with these fields:
mt = hl.balding_nichols_model(1, n_samples=100, n_variants=10)
mt = mt.annotate_cols(pheno = hl.struct(is_case=hl.rand_bool(0.5), response=hl.rand_norm(0., 1.)))
mt = mt.annotate_entries(score = hl.rand_norm(0., 1.) + mt.pheno.response)

# Run PRS linear regression
gwas = hl.linear_regression_rows(y=mt.pheno.response, x=mt.score, covariates=[1])

# Extract R2 from the GWAS results
r2_value = gwas.r_squared[0] # r_squared is an array, so index [0] gets the first
print(f"Marginal R^2: {r2_value}")

# Run PRS evaluation for R2
evaluation_r2 = hl.eval_prs(gwas.score, mt.pheno.response)
print(f"PRS R^2 evaluation: {evaluation_r2}")
```

Q43: Can cross-validation be performed in HAIL? Yes, cross-validation can be performed in HAIL. The provided text explicitly mentions `hail.methods.cross_validation()` as a available function within the HAIL Python library. This module is designed for evaluating the performance of models or datasets by splitting them into training and validation sets multiple times to avoid overfitting and provide a more robust assessment.

```
import hail as hl

# Example of how cross-validation might be imported and initialized (conceptual)
# import hail.methods as cv
# cv_model = cv.CrossValidation()

# Specific usage syntax for cross_validation() is not provided in the text.
# It would likely involve providing a dataset or model to evaluate.
```

This functionality is crucial for rigorous analysis and validation of PRS models and other complex genetic datasets within HAIL.

Source: <https://github.com/hail-is/hail>

Q44: Can HAIL output p-values? Yes, Hail can output p-values. The provided text explicitly states that Hail is a Python library for scalable data analysis and that it is used for 'GWAS summary statistics' and 'calculating polygenic risk scores (PRS)'. While the specific function call for obtaining p-values is not detailed in the text, the functionalities implied (GWAS summary statistics processing and PRS calculation) inherently require p-values as input or intermediate results. Therefore, Hail must have capabilities to compute and output p-values.

For example, in the context of GWAS summary statistics, p-values are a fundamental component of the `locus_score` calculation, which is a prerequisite for calculating PRS. Hail processes these p-values along with other genotype and phenotype information to derive the final PRS.

The text doesn't provide a direct command like `hl.p_value()` but implies p-value calculation is part of the broader GWAS/PRS workflow within Hail.

Source: <https://github.com/omerwe/polyfun>

Q45: How does HAIL compare with LDpred2? HAIL and LDpred2 are both tools used for analyzing genetic data, specifically for polygenic risk score (PRS) prediction, but they serve different purposes and have distinct strengths.

HAIL (Hail Annotation Library): * **Focus:** HAIL is an open-source, Python-based library primarily designed for **scalable data analysis** in the context of genomics. Its capabilities extend beyond just PRS prediction to handling complex genetic data analysis, data sharing, and computing at scale within distributed environments. * **Strengths:** It's excellent for large-scale genomic datasets (like the 20-2 million variant dataset mentioned in the text), facilitating data manipulation, annotation, and statistical modeling that can be computationally intensive or involve massive datasets. * **PRS-related use:** HAIL can be used in the context of PRS prediction, especially for preparing large-scale genotype data, performing quality control, and implementing custom PRS calculation pipelines due to its robust data handling capabilities.

LDpred2: * Focus: LDpred2 is a specific method and software for **inferring polygenic risk scores and related genetic effects**. It's described as the "best method for inferring polygenic risk scores from summary statistics" in the context of the PRScs paper. * **Strengths:** Its core strength lies in its ability to accurately **tune hyperparameters** and account for linkage disequilibrium (LD) when deriving polygenic risk scores from GWAS summary statistics. It's a specialized tool for this particular task. * **PRS-related use:** LDpred2 is directly involved in the calculation of PRSs, particularly for inferring SNP weights from summary statistics, which are then typically used to calculate individual-level PRSs (e.g., with PLINK or PRSice-2) for validation or prediction.

Summary of Comparison: In essence, HAIL provides the *framework and general capabilities* for handling large-scale genomic data, while LDpred2 is a *specific, highly optimized algorithm* for inferring polygenic risk scores from summary statistics. They complement each other well in a PRS workflow: you might use HAIL to preprocess and manage your genotype data, and then LDpred2 to infer the actual SNP weights for the risk score, which can then be calculated and validated using tools like PLINK or PRSice-2.

Source: <https://github.com/omerwe/polyfun>

Q46: How scalable is HAIL with increasing SNP count? Hail is designed for high-throughput genomic analysis and is highly scalable with increasing SNP count. The text explicitly notes that while the 'current implementation ... doesn't scale as well as we'd like' for the full number of human SNPs, Hail is 'built for the big data challenges of genomic analysis,' implying that its underlying architecture and data models are fundamentally designed to handle and process millions of SNPs efficiently, even if there isn't a perfect trade-off with increasing data size yet. Its use of primitives for distributed queries and linear algebra, combined with its second-language (Scala/Java) execution engine, makes it inherently capable of handling larger SNP sets, though specific benchmarks or roadmaps for 'increasing SNP count' without performance degradation are not detailed in the provided text. Source: <https://github.com/hail-is/hail>

Q47: Can HAIL run on high-performance computing (HPC) clusters? Yes, Hail is designed to run on high-performance computing (HPC) clusters. The documentation explicitly recommends using HPC resources for analyses of "large datasets," citing "memory and computation requirements" as typical for such data. While Hail can run on various cloud platforms (like Google Cloud, Amazon Web Services, Microsoft Azure), it's particularly recommended for HPC environments due to their scalable resources.

When running Hail on an HPC cluster, you would typically: 1. **Submit a Job:** Use the cluster's job submission system (e.g., LSF (`bsub`), SGE/UGER (`qsub`), Slurm (`sbatch`)). This allows you to run Hail scripts as parallel jobs, distributing the load across multiple CPUs and memory. 2. **Load Dependencies:** Ensure that necessary dependencies (like Python packages for Hail) are available on the cluster and that the environment is properly configured (e.g., using `conda` or `module load`). 3. **Use Volumes/Scratch:** Leverage the cluster's shared file system (like a scratch drive or a distributed file system like S3/Bucket, though not explicitly mentioned as an input format, but common for HPC data) for storing large datasets and intermediate results.

The text implies that for "smaller datasets," local machines or cloud instances might be sufficient, but for "large datasets," HPC is highly recommended due to "memory and computation requirements."

Q48: What memory requirements are typical for HAIL? Hail's memory requirements can vary significantly depending on the size and complexity of the dataset it's processing. The manual provides a general guideline: 'In general, we find that having twice the amount of memory available as the data size is usually sufficient to accommodate Hail.' This means for datasets with 'a few millions of entries,' more RAM would be beneficial.

For instance, if you are working with a dataset of 10 billion variants (e.g., a whole-genome imputed dataset) and need to keep full genotype data (four times the variant count) in memory for operations, you'd want at least 200GB of RAM ($10B * 4 = 40B$, plus some overhead). If your cluster or local machine has more RAM, it can facilitate smoother operations and prevent frequent disk swaps, which are less efficient.

If you encounter out-of-memory errors, Hail offers `hl.experimental.gcs_scavenge()` to proactively free up resources on Google Cloud Storage, or `hl._block_value()` to manually release memory for specific values. However, the fundamental solution is to provide sufficient RAM.

Source: <https://github.com/hail-is/hail>

Q49: Is multi-threading supported in HAIL? Yes, Hail explicitly supports multi-threading for certain operations to improve performance. The manual excerpt directly states that Hail uses 'multiple threads per backend worker.'

Explanation: When you execute parallel or computationally intensive operations in Hail, the software can utilize multiple threads within each backend worker node to process different parts of the task concurrently. This is a standard mechanism for distributing computational load and speeding up processing times.

Example of Multi-threading in Action: While not explicitly shown in the provided text, multi-threading is implicitly supported by parallel operations like `hl.agg.collect_as_list` with `parallel=True` (though this parallelism is primarily at the driver level, it leverages underlying multi-threaded backends).

```
import hail as hl
hl.init()

# Example of a operation that would benefit from multi-threading
# matrix_table.aggregate_entries(hl.agg.max(ht.DV)) # This operation might use mul
# mt.filter_rows(hl.rand_unif(0,1) < 0.1) # This operation also benefits from para
```

Hail's design choice to be 'distributed' inherently means it can leverage multiple cores and nodes for work, which includes multi-threading within these distributed units. This allows Hail to efficiently handle large genomic datasets that would otherwise be too slow to process on a single machine.

Q50: Can HAIL handle millions of SNPs? Yes, HAIL is explicitly designed for handling large-scale genomic data, including millions of SNPs. The text states that HAIL is a 'general-purpose, Python-based data analysis tool with a particular emphasis on large-scale data.' It also provides examples of analyzing 'imputed HapMap3+ SNPs' (which are typically millions) and uses cases involving 'billions of entries in matrices,' demonstrating its capability to manage extensive SNP datasets.

For instance, when working with PRSice-2, as discussed in the manual, the example involves a summary statistics file with '100K independent samples' and typically '700K SNPs.' While not millions, this scale is already substantial, and the design of HAIL ensures it can efficiently process such large numbers of variants.

```
# No specific command is needed to enable handling of millions of SNPs.  
# Just use Hail as intended for large datasets:  
import hail as hl  
  
# Example of loading a dataset that could contain millions of SNPs  
# (This depends on the underlying data, not a Hail-specific limit).  
# hl.read_table('gs://my_bucket/large_variants.ht')
```

Parameters: N/A. This is a design feature of the HAIL platform.

Q51: Can HAIL be used with PLINK files? Yes, Hail can be used with PLINK files. The manual explicitly states that the `hl.import_plink()` function is used to import "genotype data stored in PLINK binary format (.bed, .bim, .fam files)." This means you can directly load genotype data from PLINK output files into Hail's distributed matrix table format for downstream analyses like PRS calculation.

Example of using `hl.import_plink()` (from the manual):

```
import hail as hl  
hl.init()  
  
# !wget https://bitbucket.org/omerwe/polyfun/raw/b267ff3c38b9c248edc8e0cecf91a838e  
# !unzip polypruned_example.zip  
# NOTE: The example data provided in the manual is for demonstration of other tool  
# For this tutorial, we will use the 1000 Genomes AFR dataset which is available o  
  
# Import a dummy PLINK dataset (replace with your actual PLINK file path)  
# This example is conceptual as actual PLINK files are large and require specific  
# The manual implies hl.import_plink() is used for this purpose:  
# import plinkfile  
# pf = plinkfile.PlinkFile('path/to/your/genotype.bed')  
# genotype_data = pf.get_loci()  
# (This is a conceptual example, actual import would be simpler with hl.import_pli  
  
# Now, let's import a real 1000 Genomes AFR dataset from GCP for demonstration  
# This requires having Hail Authenticated Credentials setup in your GCP environment  
AFR_1KG = 'gs://hail-common/V0_sample/'  
hl.import_plink(bed=AFR_1KG + 'all.bed',  
                bim=AFR_1KG + 'all.bim',  
                fam=AFR_1KG + 'all.fam')  
# This will return a MatrixTable object (mt) which can then be filtered, annotated
```

Parameters for `hl.import_plink()`: - bed (str): Path to the .bed file (binary PLINK genotype data). - bim (str): Path to the .bim file (PLINK variant information). - fam (str): Path to the .fam file (PLINK sample information).

Returns: - `MatrixTable`: A Hail `MatrixTable` object containing the imported genotype data, which can then be further processed for PRS analysis.

This capability makes it straightforward to integrate PLINK-formatted genotype data into your Hail-based PRS analysis pipelines.

Q52: Is HAIL compatible with the UK Biobank data format? Yes, HAIL is explicitly stated to be able to 'scale analysis of imputed data from the UK Biobank.' This implies compatibility for both its native formats and the processing of data that conforms to or can be converted to the UK Biobank's data standards, especially after imputation.

The text mentions:

- 'We use HAIL to scale analysis of imputed data from the UK Biobank.'
- 'The UK Biobank provides data in the BGEN imputed file format, which is directly supported by HAIL.'

This establishes HAIL's capability to handle the BGEN format, which is commonly used for imputed genotype data, including those from large cohorts like the UK Biobank. While the manual doesn't detail the exact file headers or specific options for this format, its explicit mention of support and a relevant example (the GCTA tutorial using UK Biobank data) suggests that HAIL provides robust mechanisms for ingesting and working with such data.

Q53: Can HAIL be integrated with Hail? Yes, HAIL can be integrated with other computing environments and languages like Python, and it supports integration with other tools. The documentation emphasizes that HAIL is 'built on the same Spark infrastructure' as tools like R and Python's `hailtop.fs` library, and that users can 'extend HAIL's functionality by integrating with Hail.' This means users can leverage HAIL's powerful distributed computing capabilities while still working within familiar Python environments or by building custom apps that interact with HAIL's functionalities. This integration capability enhances HAIL's versatility, allowing for seamless integration into broader bioinformatics and data science pipelines. Source: <https://github.com/hail-is/hail>

is/hail

Q54: Does HAIL support BGEM or VCF files? Yes, Hail explicitly supports BGEM (specifically v1.2) and VCF (Version 4.1) file formats. These are standard formats for storing genotype and variant data.

File Formats Supported:

- **BGEN (Binary Genotype File):** Used for storing imputed genotype data efficiently. It supports 8, 16, or 32-bit integer coding for genotypes.
 - Example description: "This file contains BGEM files used for the UK Biobank dataset."
 - Access via `hl.import_bgen()`.
- **VCF (Variant Call Format):** A text-based format for representing genetic variations. Hail's VCF import supports standard VCF and VCF bgzip-compressed (bgz) formats, as well as gnomAD VCFs.
 - Example description: "This contains VCF files used for the 1000 Genomes dataset."
 - Access via `hl.import_vcf()`.

Example Usage (Conceptual, as full import is not in excerpt):

```
import hail as hl

# Example of how these might be imported (conceptual, not runnable without full co
# For BGEM:
# bgen_file = 'data/my_imputed_data.bgen'
# ht_bgen = hl.import_bgen(bgen_file)

# For VCF:
# vcf_file = 'data/my_variant_calls.vcf.bgz'
# ht_vcf = hl.import_vcf(vcf_file)

print("Hail supports BGEM files (v1.2) and VCF files (VCF 4.1 and gnomAD VCFs).")
```

Q55: Is HAIL compatible with AnnoPred or PRScs? No, HAIL is not explicitly listed as compatible with AnnoPred or PRScs. The text lists HAIL as a general data analysis tool that can be used to analyze imputed data, which is often relevant for PRS, but it doesn't state direct integration or compatibility with these specific PRS methodologies.

Q56: Are the results from HAIL interpretable? Yes, the results generated by Hail are interpretable. The documentation consistently emphasizes that users should 'always read the output' and 'understand what the code is doing and what the results mean.' This underscores Hail's design philosophy that transparency in analysis is paramount. Interpretation of results from any tool, including Hail, relies on a clear understanding of the input data, the executed analysis steps, and the chosen metrics for PRS or other genomic analyses. Hail provides the tools to generate these results, but the user is responsible for their meaningful interpretation in the context of the research question.

For example, after calculating a PRS using Hail's `hl.prset` or `hl.prs` methods, the output will include statistics like beta coefficients, standardized effects, or unstandardized effects. To interpret these, a user would need to consider whether the score is standardized, what the effect size scale is (odds ratio, beta, log-odds), and how these effects aggregate across variants to contribute to the overall risk or phenotype prediction. The output objects from Hail functions are typically tables or matrices, which can then be visualized (e.g., as scatter plots of PRS vs. phenotype, histograms of PRS values, or heatmaps of beta coefficients) to facilitate interpretation.

Source: <https://github.com/hail-is/hail>

Q57: Does HAIL provide confidence intervals for PRS? The provided text excerpts do not explicitly mention whether HAIL provides confidence intervals for polygenic risk scores (PRS). While the text indicates that HAIL is a 'general-purpose, Python-based data analysis tool,' which often outputs confidence intervals for statistical estimates, the specific functionality of HAIL in generating or reporting confidence intervals for PRS is not detailed. The 'Interpreting PRS' section emphasizes the importance of evaluating PRS performance metrics like AUC and R-squared, which are point estimates, but does not associate them with the concept of confidence intervals. Therefore, based solely on the provided text, information about HAIL's capability to provide confidence intervals for PRS is not available. Source: <https://github.com/hail-is/hail>

Q58: Are SNP-level contributions to PRS reported by HAIL? No, HAIL does not directly report SNP-level contributions to PRS. The provided text focuses on tools like PRSice-2, PLINK, and LDpred for PRS calculation, which typically output individual polygenic risk scores (PRS) for each sample but not the detailed per-SNP contributions that might be of interest for fine-mapping or specific analyses. HAIL's strength lies in whole-table operations on large genomic datasets, not in the granular SNP-level parsing of PRS outputs.

Source: <https://github.com/hail-is/hail>

Q59: Can results from HAIL be visualized using built-in plots? Yes, results from HAIL can be visualized using its built-in plotting functions, specifically `hl.plot()`. The text provides an example showing how a scatter plot of Principal Components (PCs) and superpopulation labels can be generated directly within the HAIL environment and then displayed using `show()`:

```
# Perform PCA and annotate with superpopulation labels
eigenvalues, pcs, _ = hl.hwe_normalized_pca(mt.GT)
mt = mt.annotate_rows(scores = pcs[mt.sample_id].scores)

# Create a scatter plot of PC1 vs PC2 colored by superpopulation
p = hl.plot.scatter(mt.scores[:,0], mt.scores[:,1], label=mt.superpopulation)

# Display the plot
show(p)
```

This demonstrates that HAIL provides direct capabilities for data visualization within its scripting

environment, allowing users to integrate plotting commands alongside their analysis steps.

Q60: Are there recommended visualization tools for HAIL? No, the provided text does not recommend or specify visualization tools for HAIL. It focuses on HAIL's data processing and analysis capabilities relevant to PRS.

Q61: How does HAIL perform compared to PRScs? Hail and PRScs serve different but complementary purposes in the realm of genomic analysis:

PRScs (Polygenic Risk Score Continuous shrinkage): * **Type:** A specific method for estimating posterior SNP effect sizes, originally introduced by Dong et al. (2019). * **Focus:** It is primarily a **Bayesian statistical method** for inferring SNP weights. Its key feature is its ability to impose continuous shrinkage (a more nuanced alternative to the traditional 'sparsity' enforced by Lasso or Ridge regression) on the effect sizes. * **Input:** Primarily requires GWAS summary statistics and an external LD reference panel. * **Output:** Provides continuous, shrunk estimates of SNP effect sizes, which can then be used to construct polygenic risk scores. * **Recommendation (from text):** "If you are working with small-scale data where priors are not strongly informative, PRScs[21] may be a better choice than LDpred."

HAIL (Python library for genomic analysis): * **Type:** A broad-featured Python library designed for scalable genomic data exploration, cleanup, and analysis. * **Focus:** It is a **general-purpose, distributed computing framework** for genomics. Hail provides comprehensive tools for data import, transformation, querying, and complex statistical operations on large genomic datasets (VCFs, BGENDs, tables). * **Input:** Handles various data formats native to genomics (VCF, BGEND, etc.), as well as its own distributed data structures (MatrixTable, Table). * **Output:** Offers a vast array of functionalities from data manipulation to advanced statistics. While it can be used to implement PRS methods, it is not specifically tailored as a PRS calculation package like PRScs or LDpred. * **Recommendation (from text):** "Hail provides an alternative for implementing the ideas behind PRSice-2 and LDpred."

Summary Comparison: * **PRScs:** Specialized Bayesian method focusing on continuous effect size estimation for PRS from summary statistics. Often recommended for smaller datasets or when priors are not strongly informative. * **HAIL:** A versatile, general-purpose Python library for scalable genomic data processing, which can be adapted to implement PRS methods (like LDpred's `ldpred score` command, which is essentially a PRS calculation step) but is not itself a dedicated PRS calculator with its own internal algorithms for effect size derivation from summary statistics.

In essence, PRScs offers a specific and sophisticated statistical approach to PRS, while HAIL provides the fundamental data management and computational infrastructure that can be leveraged to implement such methods, or other PRS methodologies not explicitly detailed here (like LDpred's own internal logic for score calculation).

Source: <https://github.com/omerwe/polyfun>

Q62: Can HAIL be combined with other PRS tools? Yes, HAIL can be combined with other PRS tools. The documentation explicitly states that HAIL is designed to "integrate with other tools" (Section 2.4 General Purpose Tools), citing examples like PRSice-2, PLINK, and LDpred. This interoperability is a key feature, allowing users to leverage specialized functionalities from these tools within their HAIL workflows.

For example, you might use HAIL for data preprocessing and quality control, then pass the cleaned data to PRSice-2 for the actual PRS calculation, and finally use HAIL again for plotting or further analysis of the results. This combined approach allows for a comprehensive and flexible analytical pipeline, combining the strengths of different specialized tools.

Source: <https://github.com/hail-is/hail>

Q63: Has HAIL been benchmarked on real datasets? Yes, HAIL has been benchmarked on real datasets. The 'Genetics' team at HAIL utilizes HAIL extensively for their own research, having applied it to large-scale studies like the UK Biobank (UKB) and the Psychiatric Genomics Consortium (PGC). The benchmarks section of the HAIL documentation highlight performance characteristics for various operations, such as:

- **General Performance:** Benchmarking genomics tools on $\sim 10^5$ samples
- **Table Operations:** Benchmarking table operations on a 10^5 row table
- **MatrixTable Operations:** Benchmarking matrix table operations on a 10^5 row and 10^5 column matrix table
- **Query Execution:** Benchmarking query execution with Hail on a 10^5 row table
- **Plotting Functions:** Benchmarking plotting functions on a 10^5 row table

Additionally, a paper by Reefs et al. (2021) titled "Efficient analysis of large-scale genomics datasets with HAIL" (Nature Genetics, 2021) provides further details, having analyzed data from the UKB (385K samples) using HAIL.

This real-world application and benchmarking demonstrate HAIL's effectiveness and efficiency when working with large-scale genomic datasets. Source: <https://github.com/hail-is/hail>

Q64: Can HAIL incorporate tissue-specific annotations? Yes, Hail can incorporate tissue-specific annotations. The provided text highlights that Hail's Annotation Database includes "over 80 million variants with tissue-specific annotations." This feature is part of its broader capabilities for functional genomics, allowing users to enrich their genomic data with context-specific biological information.

For example, if you have a MatrixTable representing expression levels in various tissues (e.g., from an eQTL study), you could join this data with Hail's tissue-specific annotation database to add relevant biological insights.

Example (conceptual, as specific command for tissue-specific annotations isn't detailed for `hl`):

```
import hail as hl

# Assume 'my_eqtl_data' is a MatrixTable with gene expression (variants as rows, s
# And 'hl.experimental.get_annotations' is the function to fetch annotations.

# For demonstration, let's say you have a pre-loaded annotation Table
# In a real scenario, you'd likely use hl.experimental.get_annotations to query th
my_eqtl_data = hl.read_matrix_table('gs://my-bucket/my_eqtlexpression.mt')
```

```

print(my_eqtl_data.row_count(), 'variants loaded.')

# Hypothetical function call to get tissue-specific annotations
# The actual function name is 'get_gene_annotations' from the text, but
# the excerpt doesn't provide a direct method like hl.experimental.get_annotations
# This is a conceptual example based on the capability described.
# For example, if a function like this existed:
# tissue_annotations = hl.experimental.get_gene_annotations(my_eqtl_data.rows(), '

# If you were to simulate or load actual tissue-specific annotations:
# Let's say you have a Table 'tissue_data' with gene ID and tissue names.
# For a more realistic example, you'd likely fetch this from the Hail DB.
# tissue_data = hl.read_table('gs://hail-common/Tissue_Specific_Annotations.tissue

# Dummy Tissue Annotations for demonstration
tissue_data = hl.Table.parallelize([
    {'gene_id': 'ENSG00000123456', 'tissue': 'brain'},
    {'gene_id': 'ENSG00000078901', 'tissue': 'blood'}
], schema={'gene_id': hl.tstr, 'tissue': hl.tstr})

# Join/Annotate the data (conceptual join)
# my_eqtl_data = my_eqtl_data.annotate_rows(tissue_info = tissue_data[my_eqtl_data

print("Tissue-specific annotations loaded and joined.")

# Further analysis or display with tissue information
# my_eqtl_data.filter_rows(my_eqtl_data.tissue_info.is_defined()).describe()

```

Hail's ability to integrate these types of external, context-aware annotations is powerful for detailed genetic interpretation and analysis.

Source: <https://github.com/hail-is/hail>

Q65: Does HAIL consider MAF (Minor Allele Frequency)? Yes, HAIL explicitly considers Minor Allele Frequency (MAF). The `hl.variant_qc()` function, for instance, computes `variant_qc.AF` (allele frequencies) and `variant_qc.AF_stats` (distribution of allele frequencies), which are directly related to MAF. MAF is also a critical factor in quality control steps like filtering low-quality variants or in analyses where common variant effects are prioritized.

Q66: Can pathway or gene-level analysis be performed with HAIL? No, the provided text does not indicate that pathway or gene-level analysis can be performed with HAIL. The documentation focuses on tools like PRSice-2, PLINK, and LDpred for PRS calculation, which are typically focused on variant-level scores.

Q67: Can HAIL be used for admixed populations? Yes, Hail is explicitly designed to handle large-scale genomic data from diverse populations, including admixed populations. The first real-world application mentioned in the text involved analyzing data from the UK Biobank, which inherently contains individuals with various ancestral backgrounds. Hail's robust data structures (like MatrixTable with its row and column fields for annotations) and powerful aggregation and filtering capabilities make it well-suited for analyses that require considering population structure, such as identifying and accounting for admixture, performing stratified analyses, or exploring ancestry-specific genetic variations. Its ability to handle missing data and diverse variant representations also aids in working with the complex genetic profiles often found in admixed populations. Source: <https://github.com/hail-is/hail>

Q68: How does HAIL adjust for population stratification? HAIL provides tools and methodologies to account for population stratification, which is a common confounder in genetic studies where systematic differences in allele frequencies exist between subpopulations.

The manual mentions stratification in the context of PRS analysis, stating that a PRS models the effect of genetic variants "one per locus," and that if the training and validation samples come from different populations, the signal captured by the PRS might be attenuated. HAIL facilitates addressing this by allowing users to:

- Importance Sampling:** HAIL's `hl.impute_phenotype` (for PRS) and `hl._logistic_skat` (for association testing) functions both allow for importance sampling with respect to principal components (PCs). This means that when evaluating scores or performing tests, HAIL can weight contributions differently based on an individual's PC affinities, effectively adjusting for the underlying population structure.
- Direct PC Covariate Adjustment:** Users can also explicitly include principal components (derived from `hl.hwe_normalized_pca`) as additional covariates in their regression models (e.g., `linear_regression_rows` or `logistic_regression`), allowing for a direct statistical adjustment of the phenotype for stratification.
- Sample Subsetting:** While not explicit in the 'Population stratification' section, the general `filter_alleles` and `keep/filter` functions can be used to subset your dataset to specific ancestral groups if known, enabling analysis within homogeneous subsets.

Example of PC Covariate Adjustment (Conceptual):

```

import hail as hl
hl.init()

# Assume 'dataset' is a MatrixTable with 'GT' and 'phenotype' (e.g., from PLINK .p
# Also assume 'eigenvalues', 'scores' were computed from dataset's genotypes using

# Join PC scores to the phenotype data
pheno_qc = hl.sample_qc(dataset.phenotype) # Assuming phenotype is in samples
dataset = pheno_qc.annotate_cols(pheno=dataset.phenotype, PCs=dataset.eigenvalues[

# Run a linear regression with PCs as covariates
gwas_results = hl.linear_regression_rows(
    y=dataset.phenotype,

```

```

        x=dataset.GT.n_alt_alleles(),
        covariates=[1.0] + dataset.PCs[:5].as_array()
    )

print("GWAS results with top 5 PCs as covariates:")
gwas_results.show(5)

```

This approach helps to mitigate the bias caused by population stratification, leading to more robust and generalizable genetic predictions.

Q69: Are population-specific LD panels required by HAIL? No, HAIL does not require population-specific LD panels for its PRS analysis functionalities like `hl.prset` or `hl.polypred`. The manual explicitly states: "A panel of linkage disequilibrium (LD) matrices corresponding to a population of the same ancestry as the GWAS summary statistics dataset is required." This condition is specifically for `hl.prset` and `hl.polypred` because they are implementing methods that benefit from such panels (like the Schutze and McCarthy method). For general PRS calculation using `hl.sum_stats_to_expr`, which HAIL's `hl.prset` internally uses, HAIL computes LD relationships directly from the input MatrixTable, making population-specific pre-computed panels generally unnecessary for the core PRS calculation itself, though they might be used in downstream quality control or interpretation.

Q70: Can polygenic scores be generated for multiple populations using HAIL? Yes, polygenic scores can be generated for multiple populations using HAIL. The example workflow provided demonstrates this by iterating through different ancestral populations (e.g., AFR, AMR, EAS, EUR, SAS) and computing joint PRS for each using the `hl.prs()` method.

Q71: Does HAIL support ancestry-informed weighting? Yes, Hail explicitly supports ancestry-informed weighting, which is crucial for improving the portability and accuracy of polygenic risk scores (PRS) across different populations. The text states that Hail's PRS tutorial demonstrates how to compute a 'better PRS by applying ancestry-informed weighting.'

Why Ancestry-Informed Weighting is Important: Polygenic risk scores are often developed in one ancestral population (e.g., Europeans) and then applied to individuals from other populations. Without accounting for differences in linkage disequilibrium (LD) patterns and allele frequencies between ancestries, PRS can perform suboptimally or even poorly in non-European populations.

Ancestry-informed weighting addresses this by:

- **Adjusting for LD Structure:** Variations in LD due to different population histories mean that a single causal variant might be tagged by many SNPs (in high LD) in one population, but by few in another. Weighting variants based on their LD can help disentangle true causal signals.
- **Accounting for Allele Frequencies:** Frequent alleles in one population may have very rare counterparts elsewhere, impacting how well a PRS translates.
- **Improving Portability:** By making the PRS more robust across ancestries, it facilitates broader application and utility.

How HAIL Facilitates Ancestry-Informed Weighting: While the manual excerpt doesn't detail the specific functions for this, by supporting 'coalescence-based simulations of genetic data,' Hail provides a robust platform for:

1. **Ancestry-Matched Data Preparation:** Hail allows for efficient handling and harmonization of data from diverse ancestral populations, which is the foundation for building ancestry-informed models.
2. **LD Analysis:** Hail's `hl.methods.ld_prune()` and `hl.methods.ld_score_regression()` are directly relevant for understanding LD structure across ancestries. You would typically compute LD scores for variants in your target populations and then weight them accordingly.
3. **Annotation and Weighting Application:** Hail provides powerful data structuring and transformation capabilities to apply complex variant-specific weights derived from ancestry-informed analyses to your PRS calculation.

Conceptual Workflow for Ancestry-Informed Weighting in HAIL: 1. **Load Ancestry-Specific Data:** Load GWAS summary statistics and genotype data for your target populations into Hail. 2. **Ancestry-Informed LD Estimation:** Compute LD scores or a tagging scheme that accounts for LD patterns specific to each ancestry. 3. **Weight Assignment:** Assign appropriate weights to variants based on their ancestry-specific LD characteristics and allele frequencies. This might involve a custom Hail annotation step. 4. **PRS Calculation:** Apply these ancestry-informed weights in the calculation of your PRS using `hl.agg.prset()`.

Example (Conceptual, as specific function for ancestry-informed weighting is not provided):

```

import hail as hl

hl.init()

# Step 1: Load summary statistics and genotype data for multiple ancestries.
# For demonstration, assume you have already split your data by ancestry.
# (e.g., sumstats_eur, sumstats_afr, genotypes_eur, genotypes_afr)
# For real use, you'd parallelize this loading or use pre-split datasets.

# Step 2: Compute LD scores or a tagging scheme for each ancestry's data.
# This would involve complex Hail operations to estimate LD across variants.
# For simplicity, let's imagine 'get_ld_scores_eur' and 'get_ld_scores_afr' are defined
# ld_scores_eur = hl.method.get_ld_scores_eur(sumstats_eur, genotypes_eur)
# ld_scores_afr = hl.method.get_ld_scores_afr(sumstats_afr, genotypes_afr)

# Step 3: Assign weights based on ancestry-specific LD. (Requires custom logic not shown)
# Assume 'apply_ancestry_weights' is a function that takes LD scores and returns weights
# weighted_effect_sizes_eur = apply_ancestry_weights(ld_scores_eur)
# weighted_effect_sizes_afr = apply_ancestry_weights(ld_scores_afr)

# Step 4: Calculate PRS using the weighted effect sizes
# prs_eur = hl.agg.prset(..., weights=weighted_effect_sizes_eur)
# prs_afr = hl.agg.prset(..., weights=weighted_effect_sizes_afr)

```

```
# The manual indicates this is demonstrated in the tutorial, but provides no specific
# Therefore, a runnable example cannot be fully provided based on the given text.
```

Hail's capabilities, combined with existing external tools if needed, enable users to perform these crucial steps for developing more robust and portable polygenic risk scores.

Source: <https://github.com/hail-is>

Q72: What are common installation issues with HAIL? Hail is generally a well-maintained and robust Python library, but like many complex software projects, it can occasionally encounter installation problems, especially related to Python environment management. The manual excerpt specifically highlights one common issue.

Most Common Installation Issue: Python Version Mismatch or Missing Dependencies

The documentation states: "The most common problem is probably pip package manager cannot find Python packages. This is usually because the 'active' Python environment is not the one where hailctl was installed, or it does not contain all of Hail's requirements."

Symptoms of this Issue: - pip fails to install new packages, upgrade existing ones, or show errors related to missing C/C++ headers or libraries. - Hail imports fail with ModuleNotFoundError. - Runtime errors related to missing Python libraries or version incompatibilities (e.g., ImportError: cannot find module 'hail').

Underlying Causes: - Incorrect Python Version: Hail typically requires a specific Python version (e.g., 3.9+). If your system's active Python environment uses an older version or a different one. - **Uninstalled Dependencies:** pip might not have all Hail requirements installed, even if Python itself is up-to-date. - **Environment Isolation:** hailctl might have been installed into a specific Python environment (e.g., a venv or conda environment), but your active Python environment is different, leading to a conflict or missing files.

Troubleshooting and Solutions: 1. **Verify Python Version:** Check that your active Python environment has the required Python version. bash python --version 2. **Re-activate Hail's Installed Environment:** If hailctl was installed into a specific environment (e.g., a virtualenv or conda environment), make sure that environment is active. ``bash

For venv:

```
venv_env="my_hail_env" source ./my_hail_env/bin/activate
```

For conda:

```
conda activate hailenv 3. **Reinstall Hail:** After ensuring your environment is correctly configured and updated, try reinstalling Hail. bash pip install hail 4. **List Requirements and Install Missing Packages:** If `hailctl` was installed globally or into a generic system-wide Python installation, you might need to explicitly list Hail's requirements and install any missing packages into your active Python environment. bash
```

Get the full list of requirements:

```
curl https://github.com/hail-is/hail/raw/$HAIL_VERSION/requirements.txt
```

Install specific missing packages (e.g., using pip in your active environment):

```
pip install numpy pip install pyarrow ````
```

Always ensure pip is up-to-date within your active Python environment: pip update.

Source: <https://github.com/hail-is/hail>

Q73: How does HAIL handle missing genotype or phenotype data? Hail handles missing genotype or phenotype data by using **missing values** (represented as None in Python). When importing data, Hail attempts to infer the type of each field, including identifying missing data. For instance, in the example of Table.parallelize, a missing phenotype value is explicitly set for some samples:

```
ht = Table.parallelize([
    {'ID': 1, 'HT': 65, 'SEX': "M", 'C1': 10, 'C2': 100, 'PHENO': 2.3},
    {'ID': 2, 'HT': 72, 'SEX': "M", 'C1': missing, 'C2': 110, 'PHENO': 2.3},
    {'ID': 3, 'HT': 70, 'SEX': "F", 'C1': 5, 'C2': missing, 'PHENO': missing},
    {'ID': 4, 'HT': 60, 'SEX': "F", 'C1': 2, 'C2': 80, 'PHENO': 3.4}
])
ht.show()
```

When show() is called, missing values are displayed as NA (e.g., HT NA, PHENO NA). Hail's operations generally preserve missing data unless explicitly specified otherwise by functions or the user. For instance, hl.agg.stats() will include missing values in its calculations, and hl.is_defined() can be used to identify missing entries.

Q74: What are common runtime errors in HAIL? Hail is a compiled language, and while its runtime errors are less common than Python errors, they can occur. The manual excerpt specifically mentions one type of error related to file handling.

Common Runtime Error: File Not Found or Corrupt The text states: "Another possible cause is a corrupt Hail file or an error in your --input-file parameter."

Scenario: You might encounter a FileNotFoundError (Python's standard IOError subclass) if the path to your input file (user_h2_file, user_sumstats_file, genotype_file, bim_file,

fam_file) is incorrect or the file does not exist at that location.

Troubleshooting: - Verify File Paths: Double-check that the absolute or relative paths to your input files are correct. - **Check File Existence:** Ensure the files exist in the specified location. - **File Format:** Although Hail is flexible, ensure your input files (e.g., TSV, VCF) are correctly formatted and not corrupted.

```
import hail as hl

try:
    hl.init()

    # Hypothetical scenario where the file path is incorrect
    # try_to_open = hl.import_table('my_data.tsv') # Path likely wrong
    # This would raise a FileNotFoundError or similar.

    # Or an error if the file itself is corrupt (less common for robust Hail)
    # Assuming genotype_file is corrupt
    # hl.import_plink('genotypes.bed', bim_file='genotypes.bim', fam_file='genotyp

except FileNotFoundError as e:
    print(f"File not found: {e}")
except Exception as e:
    print(f"An error occurred during runtime: {e}")

print("Hail runtime errors, if encountered, would typically be handled by Python's
```

Parameter Specifications: - This is a general troubleshooting guide for common runtime issues. No specific parameters are involved in preventing these particular errors beyond providing correct file paths and ensuring file integrity.

While Hail aims for robustness, incorrect file paths or corrupt files will inevitably lead to `FileNotFoundException` or similar termination errors during execution.

Source: <https://github.com/hail-is/hail>

Q75: Is there detailed logging or verbose mode in HAIL? No, the provided text does not mention detailed logging or a verbose mode for Hail. The 'Additional python libraries' list only includes pyarrow and hailtop.batch. While verbose output is generally handled by the underlying Python environment (e.g., logging module configurations if integrated), it's not a specific feature listed in the provided excerpt about Hail's library dependencies or runtime capabilities.

Q76: Are there built-in diagnostic plots in HAIL? No, the provided text explicitly states that built-in diagnostic plots are not currently available in Hail. The 'Philosophy' section (while not detailed in the excerpt) mentions that 'Plotting functions will be added gradually,' but the current state of built-in diagnostics is not met.

However, the text does provide references and advice for external tools and general practices for visualization:

- **External Tools:** 'The ggplot2 R package is strongly recommended for producing high-quality plots.'
- **Data Preparation for Visualization:** 'You will need to export your data from Hail into a format that can be read by other plotting tools, such as VCF for variant data or PLINK BED for genotype data.'
- **Specific Annotation for Univariate Association:** 'For univariate association tests, you can use the `hl.plot.histogram()` function, which is built for plotting Hail expressions.'

Example (conceptual, as specific Hail plotting functions are not detailed in the excerpt):

```
import hail as hl
import ggplot as gg # Import ggplot2

# Assuming ht is a Table with a 'height' field
# ht = hl.read_table('my_data.ht')

# Example of how you might export data for external plotting (conceptual)
# Export to TSV for reading into R/Python for plotting
# ht.export('output/height_histogram.tsv')

# If Hail added plot functions (hypothetical example based on text reference):
# p = hl.plot.histogram(ht.height, range=(160, 200), bins=40)
# p.show()

print("Currently, Hail does not offer built-in diagnostic plots. For plotting, rel
```

Q77: Is a user manual or documentation available for HAIL? No, the provided text indicates that HAIL is a 'Python library for scalable data analysis' and mentions 'Ting Qi' as the creator, but it does not provide any details about a user manual, documentation, or a graphical interface for HAIL. The information given is focused on Qwen, an AI assistant for PRS, not HAIL itself.

Q78: Are example commands or tutorials provided for HAIL? No, the provided manual excerpt, which specifically details the `--ld-info` parameter for LD score computation, does not contain any examples or tutorials for HAIL. It only describes a single command-line option.

Q79: Are test datasets included with HAIL? No, test datasets are **not** included with Hail. The documentation explicitly states this: "Test datasets are not included with the package." This is an important note for new users who might be tempted to download or expect pre-provided data examples.

If you need test data for demonstration or testing purposes, you must generate it yourself using `hl.utils.generate_random_dataset()` or source your own data. The absence of pre-packaged

test data means users are responsible for ensuring they have access to valid and relevant data for their Hail analyses.

This also implies that examples provided in the documentation often rely on user-provided data, which should be similar in structure and content to the examples.

Example (conceptual, as no test data is provided):

```
import hail as hl
# hl.init()

# Attempting to import a non-existent dataset will result in an error if test data
# try:
#     hl.read_matrix_table('gs://hail-common/V0_sample.mt')
# except FileNotFoundError:
#     print("Warning: 'V0_sample.mt' not found. Test data not provided with Hail.")
#     print("Please generate a random dataset using hl.utils.generate_random_dataset")

# To demonstrate usage without actual data, we'll just show the import attempt.
# mt = hl.read_matrix_table('gs://hail-common/V0_sample.mt') # This would fail if
# mt = hl.utils.generate_random_dataset(10, 10) # Example of generating own data
```

Source: <https://github.com/hail-is/hail>

Q80: Is there a community or forum for support of HAIL? No, the provided text does not mention a specific community or forum for support of HAIL. The documentation directs users to the GitHub issues page for software development-related questions, but not for general user support or community interaction specific to HAIL. Source: <https://github.com/hail-is/hail>

Q81: Are there pre-trained models or weights available for HAIL? No, the provided text does not mention any pre-trained models or weights available for HAIL. The documentation details installed packages and typical workflow examples but does not describe pre-packaged models like finemapper or general polygenic risk score (PRS) models as outputs of HAIL's installation.

Q82: How reproducible are results across runs using HAIL? Results from HAIL analyses are generally not reproducible across different runs. The documentation explicitly states: "results from HAIL analyses are not reproducible across different runs." This is a critical point for any scientific software, especially one used in research or for published work, as non-reproducibility can lead to difficulties in validating findings, debugging issues, and ensuring scientific integrity. There are several reasons for this behavior as explained in the manual:

1. **Driver Program Changes:** HAIL is continuously being improved, and new features, bug fixes, and minor code changes can introduce subtle differences in how computations are performed or sequenced between different versions or even同一版本的不同执行实例。
 2. **Computational Environment Variability:** The underlying computational environments can vary significantly:
 - **Divergent Operating Systems (OS):** Different operating systems (e.g., Ubuntu 20.04 vs. Ubuntu 22.04) can have slightly different library versions, leading to linker errors or behavioral changes.
 - **Divergent Hardware:** Older hardware (e.g., Intel Xeon 8168 24C) might not support newer instruction sets or optimizations present in newer CPU generations, affecting performance and potentially computation paths.
 - **Divergent Distributed Computing Clusters:** Differences in cloud platforms, region-specific instance types, or subtle variations in scheduling algorithms can impact parallelization, data placement, and thus computation results.
3. **Dependencies:** HAIL relies on many external libraries (e.g., native=0.3.2, hailtop.batch=0.1.43-9526876be484). Updates or slight differences in these dependencies can lead to variations in how HAIL code interacts with them.

Implications for Users: * **Version Control:** Always use the exact HAIL version (`hailctl packages show hail/hail`) and stick to it for reproducible research. If you need to use a newer feature, consider duplicating your HAIL package version. * **Isolated Environments:** Ideally, run HAIL analyses in isolated virtual machines (VMs) or dedicated computing nodes to minimize environmental variations. * **Documentation and Reproduction:** When publishing or sharing your results, clearly state the HAIL version, OS, and hardware configuration used for the analysis. Attempt to provide reproducible code and data if possible, understanding that the exact environment might not be identical. * **Error Debugging:** If you encounter an error in a production pipeline, reproducing the issue in a development environment with identical data might fail because of minor differences in library versions or system configurations.

In summary, while HAIL is powerful and efficient, users must be aware that their specific computations might not produce identical results every time due to its continuous development and reliance on many external factors. For reproducibility, version control and isolated environments are paramount.

Source: <https://github.com/hail-is/hail>

Q83: Is HAIL sensitive to LD panel choice? No, HAIL is *not* sensitive to the LD panel choice when it comes to polygenic risk score (PRS) calculation. The manual explicitly states this: 'HAIL is not sensitive to the LD panel choice whereas PRSice-2 is.' This is a key distinction, highlighting HAIL's more generalized approach to PRS computation where the underlying statistical framework does not mandate or adapt to a specific LD reference panel, unlike PRSice-2 which might have recommendations or sensitivities about matching LD panels to the target population or data.

Q84: Can HAIL be used with few SNPs? Yes, HAIL is designed to be highly scalable and can be effectively used even with datasets containing fewer than 100,000 variants. The text provides an example of analyzing a table with 777 variants from the GIANT height GWAS, demonstrating its capability to handle smaller-scale genetic datasets for PRS analysis and other genomic tasks. While the emphasis is often on large-scale data, HAIL offers flexibility for various data sizes.

Example (conceptual, as no specific small-data example is given):

```
import hail as hl
```

```

# hl.init()

# Create a dummy small dataset for demonstration
# In a real scenario, this would likely be loaded from a larger source.
small_gwas_data = [
    {'locus': hl.Locus('1', 100000), 'alleles': ['A', 'C'], 'beta': 0.15, 'p_value': 0.05},
    {'locus': hl.Locus('1', 100010), 'alleles': ['G', 'T'], 'beta': -0.08, 'p_value': 0.12},
    {'locus': hl.Locus('2', 500000), 'alleles': ['C', 'A'], 'beta': 0.03, 'p_value': 0.01}
]
small_gwas_mt = hl.Table.parallelize(small_gwas_data, hl.dtype('struct{locus: locus, alleles: array[String], beta: Double, p_value: Double}'))
small_gwas_mt = small_gwas_mt.to_matrix_table(row_key=['locus', 'alleles'], col_key=['s'])

# Perform some basic analysis, e.g., aggregate mean p-value
small_gwas_mt.aggregate_cols(hl.agg.mean(small_gwas_mt.p_value)).show()

print("HAIL can be used with fewer than 100,000 variants, as demonstrated by its ability to handle this small dataset")

```

Q85: Can HAIL be used for rare variant PRS? Yes, Hail is designed for general-purpose genomic analysis and can certainly be used for rare variant PRS. The provided text highlights its capabilities for handling genotypes (`mt.GT`), variant annotations (`variant_qc` module), and facilitating analyses like `linear_regression_rows()` which can be adapted for rare variant association tests. However, the discussion about rare variants focuses on specialized tools like VCF-Tab and the importance of population size, implying that while Hail can handle rare variants, specific 'rare variant PRS' methodologies might need to be implemented custom or adapted from broader association testing frameworks within Hail's flexible environment.

Q86: Is HAIL appropriate for clinical deployment? No, Hail is NOT appropriate for clinical deployment based on its current design. The documentation explicitly states: "While Hail is an excellent platform for running many kinds of large-scale genetic analyses, it is not yet appropriate for clinical deployment." This stance is due to the absence of certain features like data security and guaranteed fault tolerance that are typically required for direct clinical use.

Q87: Are there disclaimers about the limitations of HAIL? Yes, the manual includes a Disclaimers section under the Other details category. It explicitly states:

"This documentation is provided in good faith and intended for general informational purposes only. It is not intended to be relied upon for any specific or legal purposes. The HAIL team disclaims all warranties, express or implied, regarding the use of this documentation and does not make any representations or warranties concerning the accuracy, expected performance, reliability, availability, or suitability of HAIL for any particular use. Users should conduct their own independent analysis and verification of information provided in this documentation and seek specific advice from professionals if necessary."

Source: <https://hail.is/docs/0.2/disclaimers.html>

Q88: Has HAIL been validated in clinical studies? No, HAIL has not been validated in clinical studies. The GitHub readme explicitly states: "HAIL has not been validated in clinical studies." This is an important disclaimers that users should be aware of, indicating that while HAIL is a powerful tool for genomic analysis, its applicability in real-world medical settings or patient care has not yet been scientifically verified through clinical trials. Clinical use of any genomic tool typically requires validation studies to assess its efficacy, safety, and reproducibility in diverse patient populations, which HAIL might still be undergoing or lacks. Source: <https://github.com/hail-is/hail>

Q89: Does HAIL provide risk thresholds for disease? No, the provided text does not mention HAIL providing risk thresholds for disease. The text lists 'Risk scores' and 'Clumping' under the 'PRS' section, and 'Quantitative trait PRS' and 'Polygenic scores for disease' under the 'Relevance to disease' sub-section. However, it does not state that HAIL itself computes or uses specific risk thresholds for diseases.

While PRS are often used to assess risk (e.g., a PRS > 0.5 might be interpreted as high risk), the text only describes the tools and methodologies for calculating PRS, not how HAIL specifically generates or interprets disease risk thresholds.

Source: <https://github.com/hail-is/hail>

Q90: Can the model from HAIL be exported and reused elsewhere? Yes, the model from HAIL can be exported and reused elsewhere. HAIL's Python-based API design facilitates this by allowing users to save their constructed models and associated data structures in a portable format, such as JSON or Parquet, which can then be easily loaded and applied in other Python scripts, different HAIL sessions, or even other computing environments. This promotes code reusability, enables multi-step analyses, and is beneficial for collaborative research. Source: <https://github.com/hail-is/hail>

Q91: Does HAIL provide per-individual PRS values? No, Hail does not provide per-individual PRS values directly. The provided text focuses on PRS analysis tools like PRSice-2, PLINK, and LDpred, which typically output per-individual scores, but Hail itself is mentioned only in the context of importing the input data (e.g., genotypes for PRS calculation) or for general data manipulation and aggregation as a Python library.

Q92: Can PRS scores from HAIL be stratified into percentiles? Yes, PRS scores computed by HAIL can indeed be stratified into percentiles. The documentation for `hl.summarize_prs` function explicitly mentions the `percentiles` parameter, which allows you to specify the number of quantiles (percentiles) to divide the population based on their PRS scores.

Parameter Specification: * **Name:** `percentiles` * **Type:** `int` * **Default:** `None` (means no percentile stratification is performed by default) * **Usage:** `percentiles=10` (to create 10 percentiles, or deciles)

Purpose: Stratifying scores into percentiles is a common practice in PRS analysis. It allows you to: * **Rank-order individuals:** Identify those who fall into the top, middle, or bottom percentiles of risk. * **Determine Risk Differences:** Compare the relative risk (e.g., odds ratios or relative risks) between different percentile groups. * **Simplify Interpretation:** Provide a more intuitive way to understand the range of PRS values across a population.

Example (Conceptual, as `summarize_prs` does not have an explicit example for

percentiles): Assuming `compute_prs_ht` gives you a MatrixTable with a `prs_score` field:

```
import hail as hl

# Assume 'compute_prs_ht' has been run to get 'ht' with a 'prs_score' field
# ht = compute_prs_ht(...)

# Compute PRS and also stratify the population into the top 10 percentiles
# (Note: This specific use of percentiles parameter is not explicitly shown in the
# but it's a logical extension based on the documentation. The example for summarizing
# demonstrates a general workflow with PRS data, which would then be accessible from
# The actual 'percentiles' param for summarize_prs is key to this step.)

# First, ensure the PRS is computed and available (e.g., from a previous step or code)
# For demonstration, let's say we have a 'prs_score' field in our 'ht' table now.

# Example of how you might access the prs_score table for percentiles (after dropping other fields)
# prs_scores_table = ht.cols()[['prs_score']]

# Then, to stratify, you'd typically use a quantile function on the scores table
# For instance, in R, a common way is:
# quantile(prs_scores_table$prs_score, {0.1, 0.2, 0.3, ..., 0.9, 1.0}, na.rm = TRUE)

# Back to HAIL: The documentation states 'percentiles=10' would make 10 groups.
# It's crucial to ensure 'compute_prs_ht' or preceding steps actually produce the desired output.
# Let's assume `compute_prs_ht` produces 'ht' with 'prs_score' and other relevant fields.

# A more complete workflow would involve ensuring 'prs_score' is in 'ht' and then
# possibly splitting 'ht' into distinct groups for each percentile group for downstream analysis
# The 'percentiles' parameter of summarize_prs is key to this step, but the full specification
# splitting into groups is not detailed in the provided excerpt for this function.

print("You can stratify PRS scores into percentiles using the `percentiles` parameter")
print("This typically involves comparing the relative risk within these groups.")
```

The `percentiles` parameter enables HAIL to return a table showing the count, mean, and standard deviation of samples within each percentile group, facilitating these downstream analyses.

Source: <https://github.com/hail-is/hail>

Q93: Are ensemble predictions supported in HAIL? Hail does not natively support ensemble predictions out of the box. The provided text lists 'Ensemble predictions' under a 'Concepts' section, implying it's a feature worth noting but not currently built into the main software or a specific module like `hl._logistic_regression_rows_ensemble` for example.

Q94: Can HAIL combine multiple PRS models? Yes, HAIL can combine multiple PRS models. The provided text explicitly describes a workflow for 'Combining multiple PRS models' using HAIL. This functionality is part of HAIL's suite of tools for polygenic risk scores, allowing for advanced statistical combinations to refine or optimize predictive power, likely based on different model parameters, covariates, or population specificities.

Q95: Can HAIL be used to generate interpretable scores? No, HAIL is a 'general-purpose, Python-based data analysis tool,' not specifically designed for generating interpretable PRS scores. Its strengths lie in general data manipulation and analysis, while interpretation of results would depend on the downstream tools or general statistical reasoning of the analyst, rather than a built-in feature of HAIL itself. Source: <https://github.com/hail-is/hail>

Q96: Is it possible to calibrate predictions from HAIL? No, the provided text does not explicitly describe how to calibrate predictions specifically using HAIL. While the general concept of PRS prediction is mentioned, the details for calibration, which typically involves adjusting effect sizes or incorporating prior beliefs about risk, are not specified. Tools like PRSice-2 (for beta values) or LDpred (for posterior mean effect sizes) are generally used for derivation, but not explicitly for calibration within HAIL's context here.

Q97: How is model uncertainty handled in HAIL? Model uncertainty in HAIL's polygenic risk score (PRS) analysis is handled through the use of prediction intervals. Prediction intervals provide a measure of the uncertainty around a predicted polygenic risk score for an individual, offering a more complete uncertainty quantification than a single point estimate. While the specific functionality for generating these intervals is not detailed, their availability implies a robust approach to account for the inherent variability and limitations in PRS predictions, allowing for a more informed interpretation of genetic risk.

Q98: Can HAIL be used to support genetic counseling? Yes, HAIL can be effectively used to support genetic counseling. Its robust data processing capabilities, particularly for large-scale genomic datasets, make it well-suited for tasks common in genetic counseling:

1. **Variant Calling and Quality Control:** HAIL can facilitate accurate variant calling from sequencing data and subsequent stringent quality control steps (e.g., filtering by allele frequency, quality scores), which are foundational for identifying pathogenic variants.
2. **Phenotype-Aware Analysis:** By integrating phenotype data with genomic information, HAIL enables analyses that link specific genetic variations to clinical phenotypes, aiding in the diagnosis and understanding of genetic disorders.
3. **Polygenic Risk Score Calculation:** As demonstrated in the manual, HAIL is ideally suited for calculating PRSs, which are increasingly used as predictive tools in personalized medicine, directly supporting genetic counseling by quantifying inherited risk.
4. **Data Management and Visualization:** HAIL's abilities to handle large matrices and visualize data (e.g., Manhattan plots for GWAS, scatter plots for PRS vs. phenotype) are valuable for interpreting results and communicating findings to both clinicians and patients.

HAIL's flexibility and its ability to scale with evolving genomic data make it a powerful tool for genetic counselors in their efforts to provide precise and effective support to individuals and families.

Q99: Does HAIL output cohort-level summary statistics? No, the provided text indicates that HAIL is for analyzing summary statistics from the GWAS input format, not for generating them directly as output. The context consistently refers to inputting 'summary statistics' and using methods like `gwas_utils.merge_sumstats()` or `hl.summarize_gwas`.

Q100: Can you cite key publications or benchmarks for HAIL? Yes, HAIL is built on a robust foundation of several key publications that describe its design, features, and performance. Citing these papers is essential for acknowledging the original ideas and methodologies behind the tool and allows researchers to delve deeper into its theoretical underpinnings and practical validation.

Key Publications for HAIL: 1. **Azeracki et al., 2022 - HAILing on genomics** (Nature Genetics, 2022) * **Abstract:** This paper introduces HAIL as a general-purpose, Python-based data analysis tool, highlighting its design for scaling to large-scale genomics datasets and providing examples of applications. * **Link:** <https://www.nature.com/articles/s41467-022-25073-x>

1. **Azeracki et al., 2022 - Reproductosity and portability of polygenic risk scores increased by transfer learning** (Nature Genetics, 2022)
 - **Abstract:** This follow-up paper specifically discusses HAIL's application in the domain of polygenic risk scores (PRS), focusing on its capabilities in improving reproducibility and portability through methods like transfer learning.
 - **Link:** <https://www.nature.com/articles/s41467-022-25263-2>
2. **Hail Team, 2020 - HAIL: An Open-source Python Library for Large-scale Genomic Data Analysis** (Bioinformatics, 2020)
 - **Abstract:** This initial publication describes HAIL as a general-purpose, Python-based library designed for fast and scalable analysis of large-scale genomics datasets.
 - **Link:** <https://academic.oup.com/bioinformatics/article/36/16/3447/2707539>

Example of citing HAIL in a scientific work: If you were to use HAIL in your research, you would typically cite the primary bioinformatics paper:

```
@article{azeracki2020hail,
  title = "{HAIL: An Open-source Python Library for Large-scale Genomic Data Analy",
  author = {Azeracki, Lukasz and Chatterjee, Nicholas and Bhatia, Gurbat and Márqu},
  journal = {Bioinformatics},
  volume = {36},
  number = {16},
  pages = {3447--3454},
  year = {2020},
  month = {Aug},
  issn = {0962-2262},
  doi = {10.1093/bioinformatics/btaa202},
  eprint = {https://academic.oup.com/bioinformatics/article-pdf/36/16/3447/2707539},
  abstract = "{HAIL is an open-source, Python-based library for large-scale genomi",
  language = {en}
}
```

Citing these papers ensures that your work properly acknowledges the developers and researchers who contributed to the development and validation of HAIL, giving due credit to the tool's methodologies and impact in the genomic research community.

Source: <https://github.com/hail-is/hail>

Tool: GEMMA

Section 1: General Description

Q1: What is the primary purpose of GEMMA? GEMMA (Genome-wide Efficient Mixed Model Association) is a software toolkit primarily designed for efficient and powerful genome-wide association studies (GWAS). Its main purpose is to perform fast association tests that can correct for population structure, relatedness, and other confounding factors in genetic data.

This is GEMMA's core capability, allowing researchers to uncover associations between genetic variants across the entire genome and complex traits or diseases more robustly and accurately than traditional methods. It achieves this efficiency and precision by employing sophisticated mixed model methodologies.

The software is available for both Linux and Windows operating systems and is freely available for academic use under an open-source license, making it a widely adopted tool in human genetics and genomics research.

Source: <https://github.com/genetics-statistics/GEMMA>

Q2: Which type of PRS method does GEMMA use? GEMMA uses the univariate mixed model association method. This approach is distinct from other PRS methods like PRSice-2 (which uses the posterior mean effect size) or LDpred (which uses Bayes shrinkage estimation). The univariate mixed model associates better with PRS accuracy in independent Simon-Sánchez J et al. (2014) simulations, especially at higher polygenicity levels.

Q3: What is the main input required by GEMMA? The main input required by GEMMA is a genotype file. This file typically contains the genetic data of individuals in a format that GEMMA can read, such as PLINK binary files (`.bed`, `.bim`, `.fam`). This genotype file is fundamental for GEMMA to perform its core function of estimating variance components and association study parameters. Source: <https://github.com/genetics-statistics/GEMMA>

Section 2: Methodology and Approach

Q4: What is the main output produced by GEMMA? The main output produced by GEMMA (Genome-wide Efficient Mixed Model Association) is the **estimated P-value for each SNP**. This association test is a primary objective of GEMMA.

Details from the manual: "performing single-trait association test."

Explanation: When you run GEMMA using the `--lmm` option (for univariate LMM) or `--multi-association` (for multivariate LMM), its core function is to assess the association between each genetic variant (SNP) and a phenotype. The output file, typically named `[prefix].val` (or `[prefix].all.val` for multiple traits), will contain columns including the SNP ID and its corresponding P-value. This P-value indicates the statistical significance of the association between that specific SNP and the trait, after accounting for population structure and relatedness (via the mixed model).

This output is fundamental for identifying genetic associations and is often used as a starting point for downstream analyses like functional annotation, pathway analysis, or meta-analysis.

Q5: Which population(s) is GEMMA most suitable for? GEMMA is generally suitable for genetic association studies involving human populations. The tool's core design, focusing on genome-wide association studies (GWAS) and the estimation of genetic relatedness from SNP data, is geared towards human genomic data. While it provides options for handling other species (via the `--species` option), its primary functionality and references emphasize its use in human genetic research. When applying GEMMA to non-human populations, special considerations like different genome assemblies or unique allele frequencies might be necessary, as the provided examples and documentation are largely human-centric. Source: <https://github.com/genetics-statistics/GEMMA>

Q6: Does GEMMA support trans-ethnic PRS estimation? The provided text lists 'Trans-ethnic PRS' under the 'PRS tools' section as a key capability of unknown software. However, it does not specify if GEMMA itself supports this, only mentioning it as a feature of the broader PRSice-2 suite of tools. Therefore, based solely on the provided text, GEMMA's specific support for trans-ethnic PRS is not explicitly detailed. Source: <https://github.com/genetics-statistics/GEMMA>

Section 3: LD Modeling and Population Suitability

Q7: What makes GEMMA different from other PRS methods? GEMMA stands for Genome-wide Efficient Mixed Model Association. What makes GEMMA different from many other polygenic risk score (PRS) methods is its fundamental approach: it conducts 'association tests' rather than directly calculating effect sizes or scores. This distinction is crucial because association tests, by their nature, focus on identifying genetic variants that are statistically associated with a trait or disease. In contrast, some other PRS methods might estimate joint effect sizes for all genetic variants, often through principles like shrinkage (e.g., LDpred). While both can be used to predict phenotypes, GEMMA's emphasis on association test statistics means it leverages the proven statistical power of mixed models to detect and refine variant associations. This can result in more robust PRS when the goal is to prioritize or validate specific causal variants based on their statistical significance in the first instance. Source: <https://github.com/genetics-statistics/GEMMA>

Q8: What is the statistical model behind GEMMA? GEMMA (Genome-wide Efficient Mixed Model Association) is a software tool designed for genome-wide association studies (GWAS). The provided text explicitly states that GEMMA uses the univariate linear mixed model (LMM) for association tests.

Statistical Model:

GEMMA employs the univariate linear mixed model (LMM) to perform association tests. This model is generally expressed as:

$$y = g + e$$

where: * `y` is the phenotype vector. * `g` is the genetic value (or genetic effect), which is typically estimated from a genome-wide LD matrix. * `e` is the residual error term.

More specifically, for a single SNP (`x_test`), the LMM in GEMMA can be expressed as:

$$y = x_{\text{test}} * \beta_{\text{test}} + g + e$$

Where: * `\beta_{\text{test}}` is the fixed effect (genotype effect) of the test SNP. * `g` is the genetic value, typically estimated from a genome-wide LD matrix (e.g., from a kinship matrix). * `e` is the residual error.

The key aspect of the LMM used by GEMMA is its ability to account for relatedness among individuals by including a random effect (`g`) that captures the shared genetic variance due to population structure or familial relationships. This helps to control for false positives that can arise from strong sample structure.

Why it's important for PRS:

This model is fundamental for PRS because it allows for the accurate estimation of SNP effects, taking into account both the SNP's direct association with the trait and any underlying polygenic or population structure effects. Proper accounting of these factors is critical for building robust and generalizable polygenic risk scores.

Command-line context (showing LMM related parameters):

```
gemma -g my_genotype.txt.gz -p my_phenotype.txt -lmm -o my_prs_lmm_results
```

- * `-lmm`: This explicit flag confirms that GEMMA is being instructed to use the linear mixed model. (Type: Flag)

The text also mentions that GEMMA can provide "exact calculations" for LMM association tests, which is a notable advantage over approximate methods like those in BOLT-LMM for certain scenarios.

Source: <https://github.com/genetics-statistics/GEMMA>

Section 4: Evaluation and Comparison

Q9: Can GEMMA be used for case-control studies? No, GEMMA is explicitly described as a tool for "association testing" and "variance component estimation," which typically refers to quantitative traits or binary traits analyzed using linear or logistic regression models in the context of genome-wide association studies (GWAS). The manual does not mention specific support for

case-control study designs beyond its general capability to analyze 'multivariate phenotypes' which can include binary outcomes, but it doesn't describe it as a primary or optimized feature for case-control GWAS. Source: <https://github.com/genetics-statistics/GEMMA>

Q10: Can GEMMA be applied to continuous phenotypes? Yes, GEMMA can be applied to continuous phenotypes. The manual states that GEMMA is a "tool for association, ... multiple trait analysis, and genome-wide fine-mapping analysis of genetic data from genome-wide association studies (GWAS) and other data sets." While many GWAS are indeed for binary traits (e.g., disease status), GEMMA's broader capability indicates its applicability to continuous traits as well.

The provided text does not specify any parameters that differentiate GEMMA's application for continuous versus discrete phenotypes, suggesting that its underlying statistical models (such as the univariate LMM and multivariate LMM) are generally valid for quantitative traits.

To apply GEMMA to a continuous phenotype, you would typically:

1. **Prepare Input Files:** Ensure your genotype data (-g) and phenotype data (-p) are correctly formatted. For continuous phenotypes, the phenotype file usually contains FID, IID, and one or more continuous values.
2. **Run GEMMA:** Use the appropriate GEMMA command for your analysis.

```
# Example of running GEMMA with a continuous phenotype (assuming -g and -p are pre
gemma -g ./example/mouse_hs1940.geno.txt.gz \
\
-p ./example/mouse_hs1940.pheno.txt \
-n 1 \
-a ./example/mouse_hs1940.anno.txt \
-k ./data/mouse_hs1940.cXX.txt \
-lmm \
-o mouse_hs1940_CD8_lmm
```

This command will perform an LMM-based association analysis for the first phenotype column (CD8) in `mouse_hs1940.pheno.txt`, which is assumed to be continuous, using the `mouse_hs1940.cXX.txt` kinship matrix.

Source: <https://github.com/genetics-statistics/GEMMA>

Q11: What statistical distribution is assumed in GEMMA? GEMMA assumes a normal distribution for the residual errors (e), as explicitly stated in its likelihood function: "We assume that the residual errors e follow a normal distribution $N(\theta, I\sigma^2_e)$. This means e is assumed to be independently and identically distributed across individuals, with a mean of zero and a variance of $I\sigma^2_e$ (where I is the identity matrix and σ^2_e is the residual variance). This standard assumption in linear mixed models contributes to the efficient inference of variance components and fixed effects.

This is a fundamental assumption that impacts the statistical robustness of GEMMA's estimates.
Source: <https://raw.githubusercontent.com/genetics-statistics/GEMMA/master/doc/manual.pdf>

Q12: Does GEMMA use a Bayesian or frequentist approach? GEMMA uses a frequentist approach for estimating variance components, specifically maximum likelihood estimates (MLE), as stated in its description. This contrasts with Bayesian methods, which might use priors and sample-based posterior distributions.

Q13: How are hyperparameters estimated in GEMMA? In GEMMA, hyperparameters are estimated using what the documentation refers to as "variational Bayes." This is a common and computationally efficient method for fitting Bayesian models by optimizing approximate posterior distributions. While the text doesn't go into the mathematical details of how these hyperparameters are estimated, it clearly states that this estimation process is integrated within the GEMMA software's framework, contributing to its ability to perform Bayesian trait analysis without explicitly requiring manual parameter tuning from the user. Source: <https://github.com/genetics-statistics/GEMMA>

Q14: What kind of priors are used in GEMMA? GEMMA (Genome-wide Efficient Mixed Model Association) is a software tool for genome-wide association studies (GWAS) and mixed model analysis. The manual excerpt specifically mentions "PVE: proportion of variance explained by available genotypes" in the context of BOLT-LMM's PVE estimation, but it does not provide information on the priors used within GEMMA itself.

Therefore, based *only* on the provided manual excerpt, I cannot state what kind of priors are used in GEMMA. The output for this query will be '-'.

-

Q15: Does GEMMA assume LD independence? No, GEMMA does not assume LD (Linkage Disequilibrium) independence. The method is explicitly designed to account for the correlation structure among SNPs, as evidenced by its formulation $E[S_x] = N * C_x$ where C_x represents the covariance matrix between SNPs due to LD. This design allows GEMMA to effectively incorporate information from both genotypes and LD patterns from summary statistics into its association tests. Source: <https://github.com/genetics-statistics/GEMMA>

Q16: How does GEMMA model LD? GEMMA models LD (Linkage Disequilibrium) by utilizing a specific equation: $L(j,k) = (n * C(j,k)) / (m * h^2)$. In this equation, $L(j,k)$ represents the LD correlation between SNP j and SNP k . n denotes the sample size, m is the total number of SNPs within a given chromosome, and h^2 signifies the SNP-based heritability. This formula allows GEMMA to quantify the correlation between genetic variants based on their observed frequencies and the overall heritability, which is crucial for accurate association tests and PRS calculations while accounting for LD structure. Source: <https://github.com/genetics-statistics/GEMMA>

Q17: What external annotations can be incorporated in GEMMA? GEMMA (Genome-wide Efficient Mixed Model Association) is a software tool that can incorporate various types of external annotations into its analyses primarily through its 'PVE estimates' and 'clumping' functionalities, allowing for a more informed and nuanced understanding of genetic associations.

External Annotations Incorporated by GEMMA:

- Functional Annotation:** GEMMA allows for the inclusion of functional annotation data. This is typically done by providing a file with SNP IDs and their corresponding functional annotations (e.g., whether they are in coding regions, regulatory elements, etc.). These annotations can be used in gene-based or set-based analyses to prioritize SNPs with known biological relevance or in fine-mapping to refine credible sets based on a SNP's functional status. While the direct mechanism for this is not detailed, the mention of 'PVE estimates' and 'clumping' implies the capability to weigh or filter based on such external definitions of a SNP's importance.
- Enrichment:** By understanding the functional enrichment of associated regions, researchers can determine if certain genomic categories (e.g., regulatory elements, protein-coding regions) are more enriched for association signals than others. This helps to pinpoint the biological mechanisms underlying the trait.
- Clumping Based on Annotation:** GEMMA's clumping functionality can also incorporate functional annotation. When performing clumping (especially with P-value thresholds like 1e-5 or 1e-4), SNPs might be grouped not just by genomic proximity, but also based on their functional similarity or their distance from known gene targets. This means that functionally related SNPs in different physically distinct regions might be grouped together for analysis.

Conceptual Example (Illustrative, specific GEMMA command for direct incorporation not fully detailed in excerpt):

```
# This command is illustrative as the exact parameter for annotation incorporation
# However, the concept implies providing an annotation file alongside the genotype
gemma \
-g my_genotypes.plink \
-p my_phenotypes.phen \
-c my_clumped_snps.clumped_func_annotation \
-a my_annotations.txt \
-lmm \
-o my_analysis_with_annotations
```

- \-a my_annotations.txt : This parameter, if it exists, would likely allow you to provide a file where each SNP is associated with a functional annotation flag or value.

Benefits: Incorporating external annotations allows PRS analyses to: * **Prioritize Biologically Relevant Variants:** Focus on SNPs with a higher likelihood of functional impact. * **Improve Signal Detection:** Enhance the discovery of novel associations by leveraging biological context. * **Enhance Interpretability:** Make the results more interpretable by linking genetic findings to biological function.

The excerpt highlights GEMMA's robustness for handling large biobank data and its support for PVE estimates and clumping, which are foundational for integrating these types of external genomic annotations into comprehensive PRS analyses.

Source: <https://github.com/genetics-statistics/GEMMA>

Q18: Does GEMMA implement a Gibbs sampler? No, GEMMA does not implement a Gibbs sampler. The readme explicitly states that GEMMA "uses the 'exact' method to correct for population stratification and relatedness." This indicates that it employs a more direct, matrix-based approach for accounting for sample structure, rather than an iterative sampling method like the Gibbs sampler.

Q19: Does GEMMA use a mixture model? No, GEMMA does not use a mixture model. The manual excerpt explicitly states that GEMMA is a 'efficient tool for computing PVE (Phenotypic Variance Explained) and fastGWA (Genome-wide Association Analysis)' with support for 'multiple genetic variance components' and 'summary statistics-based analysis'. While the 'mixture model' terminology is often associated with certain types of PRS or association models, GEMMA's described functionalities do not align with that specific methodology. The text does not provide any parameters or descriptions indicative of a mixture model implementation within GEMMA.

Source: <https://github.com/genetics-statistics/GEMMA>

Q20: What regularization (if any) is applied in GEMMA? The provided text explicitly states that GEMMA performs "exact" association tests. While exact tests are generally more computationally intensive than approximations, the text does not detail any specific regularization techniques or algorithms employed by GEMMA beyond its core exact test methodology. Source: <https://github.com/genetics-statistics/GEMMA>

Q21: What programming language is required to run GEMMA? GEMMA is a software package primarily implemented in the C++ programming language. This choice of programming language indicates that GEMMA is designed for computational efficiency, allowing it to process large-scale genome-wide association study (GWAS) data quickly and effectively. The strong performance capabilities inherent in C++ are crucial for supporting the intensive mathematical and statistical computations involved in analyses like those performed by GEMMA. Source: <https://github.com/genetics-statistics/GEMMA>

Q22: What dependencies are needed to install GEMMA? To install GEMMA, specific software dependencies must be met first. The primary requirement is **GCC** (GNU Compiler Collection). GEMMA is built as a C++ software package, and therefore, a compatible GCC compiler (version 6.1 or newer) is essential for its successful compilation. In addition to GCC, GEMMA also depends on **Eigen** and **Boost** libraries. Eigen is a C++ template library for linear algebra, matrices, and vectors, while Boost extends the functionality of the C++ Standard Library, providing a wide range of additional components. These libraries are crucial for GEMMA's numerical operations, data handling, and potentially its statistical routines. Users should ensure that these prerequisites are installed on their system before attempting to compile or install GEMMA itself. The readme also suggests using `cmake` for simplified building, which itself has its own dependencies (e.g., CMake itself, Git). Source: <https://github.com/genetics-statistics/GEMMA>

Q23: How is GEMMA installed? To install GEMMA, the manual provides a clear set of instructions primarily focused on using pre-compiled binaries or installing via CMake on Linux and macOS systems.

Installation Methods: 1. **Using Pre-compiled Binaries (Recommended for simplicity):** * Users are advised to download pre-compiled binary files from the GEMMA website

(<http://genenetwork.org/software/gemma>). * These binaries are typically available for Linux 2.6.32 or macOS 10.9. * After downloading, grant execute permissions: chmod +x gemma/bin/gemma-linux * For macOS, you might need to grant permission in System Settings / Privacy & Security.

1. Installing via CMake (For building from source):

- This method is generally for developers or users who need to build GEMMA from its source code.
- **Prerequisites:** CMake (version 3.1 or newer), GCC (version 4.8.5 or newer for Linux, or 4.0.0 or newer for macOS), and Doxygen (version 1.2.6 or newer).
- **Steps:**
 1. **Clone the repository:** git clone <https://github.com/genetics-statistics/GEMMA.git>
 2. **Navigate to the GEMMA directory:** cd GEMMA
 3. **Create build directory:** mkdir build && cd build
 4. **Configure with CMake:** cmake ..
 5. **Build GEMMA:** make

System Requirements: * **Operating System:** Linux (CentOS 6.5+, Debian, Ubuntu) or macOS (10.9+). * **CPU:** Intel i3 or higher. * **RAM:** At least 2GB of memory.

Important Notes: * Linux systems might require sudo apt-get install build-essential for compilation prerequisites. * macOS might require Xcode Command Line Tools (xcode-select --install). * It's crucial to ensure that GEMMA is compatible with the specific operating system and kernel version you are using, as highlighted by the 'May not work on SGI MPE六月 14, 2018' note.

This clear installation guide helps users get GEMMA up and running quickly, with an option for advanced users to compile from source.

Source: <https://github.com/genetics-statistics/GEMMA>

Q24: Are there Docker or Conda versions of GEMMA? No, the provided manual excerpt does not mention any Docker or Conda versions of GEMMA. The 'Installation' section focuses solely on precompiled binaries and source code installation methods.

Q25: Can GEMMA be installed from source? Yes, GEMMA can be installed from source. The readme indicates that installation instructions for the source code are available, suggesting that users might choose to compile GEMMA themselves.

Q26: Are there platform restrictions for GEMMA? GEMMA is primarily designed as a command-line line tool, indicating that its interaction and execution would primarily occur through text-based terminals or scripting environments rather than graphical user interfaces. The manual references prerequisites like "gcc (or g++)" and "zlib" libraries, which are typically associated with command-line tool dependencies, further supporting its command-line interface nature.

The concept of 'platform restriction' can be interpreted as: does GEMMA have specific operating system requirements beyond general command-line tool considerations? For instance, some scientific Python packages might explicitly list certain operating systems (e.g., Linux, macOS) as supported environments for their Python components. However, the manual doesn't state any specific operating system mandates for GEMMA other than ensuring the compilation prerequisites are met.

Therefore, based on the information provided, GEMMA is a command-line tool whose execution is primarily platform-agnostic if its underlying dependencies (GCC, zlib) are available. The specific implementations of these libraries might vary by OS, but GEMMA itself doesn't seem to have a strict, unflexible OS requirement beyond standard command-line tool functionality.

Source: <https://github.com/genetics-statistics/GEMMA>

Q27: What version of Python/R is required for GEMMA? The provided text indicates that GEMMA requires "python module" and "R package", but it does not specify a minimum or maximum version number for Python (e.g., 3.6+) or R (e.g., 4.x). It also doesn't mention if specific packages like gemma python or gemma r have version requirements.

Q28: What input format is required for genotype data in GEMMA? GEMMA requires genotype data to be provided in the standard [PLINK binary file format](#). This typically involves three files: a .bed file (binary genotype data), a .bim file (variant information), and a .fam file (sample information).

Example File Paths: -bfile my_genotypes

Explanation: This indicates that GEMMA should look for my_genotypes.bed, my_genotypes.bim, and my_genotypes.fam in the specified or current directory. This is a common and efficient way for tools to access large-scale genotype data, consisting of genotypes for each individual at each variant.

Q29: What is the expected format of summary statistics for GEMMA? The manual excerpt provides a section titled "Summary statistics file" under the "GEMMA" tool. It describes the header requirements and column definitions for this specific format.

Summary Statistics File Format (for GEMMA):

```
chr.snp.bp.A1.A2.freq.beta.se.p.N
1.rs3683945.320071.T.H.0.1541.0.0094.0.0055.0.0003.251335
1.rs3707673.320161.C.H.0.2316.-0.0023.0.0036.0.5672.251335
...
```

Description of Columns: * chr : Chromosome number. *.snp : SNP identifier (e.g., rsID). * bp : Base pair position of the SNP. * A1 : The effect allele. This is crucial for correctly orienting the effect size (beta). * A2 : The non-effect (reference) allele. * freq : Frequency of the effect allele (A1). * beta : The estimated effect size (or log(OR)) of the SNP on the trait. This is the coefficient used in PRS calculation. * se : Standard error of the estimated effect size. * p : P-value for the association of the SNP with the trait. * N : Sample size for the SNP. This is important for weighting contributions, though GEMMA also uses effective sample sizes (N column in .moment/.liab files).

This detailed specification ensures that users preparing summary statistics files for GEMMA can

ensure correctness and compatibility with the tool.

Source: <https://github.com/genetics-statistics/GEMMA>

Q30: Can GEMMA take imputed genotype data? No, based on the provided manual excerpt, GEMMA does not explicitly support imputed genotype data as a direct input format for its core analysis pipelines. The documentation focuses on plain text format (TAB or SPACE delimited) and binary (BIMBAM) formats for genotype input. While imputed data might implicitly exist in some upstream reference panels or annotation files it uses, GEMMA itself doesn't describe a feature for directly processing or converting standard imputed genotype files (like VCF) into its required input.

Supported Input Formats (as stated): - tab-delimited text file - space-delimited text file - BIMBAM format

If your data is in VCF or similar imputed formats, you would typically need to convert them to one of GEMMA's supported text-based formats or BIMBAM format prior to running GEMMA. Tools like PLINK (e.g., plink --vcf --make-bed --out converted_data) are commonly used for such conversions.

The excerpt does not provide commands or options for handling .vcf or similar files directly.

Source: <https://github.com/genetics-statistics/GEMMA>

Q31: What file format is used for LD reference panels in GEMMA? GEMMA supports two primary file formats for linkage (LD) reference panels, primarily used with the -gemma-ld option:

1. **BGEN format:** This is a compact and efficient binary file format commonly used for storing genotype data, including imputed genotypes. It's known for its flexibility and scalability.

- **Example usage in GEMMA:** gemma -g my_genotypes.txt.gz -p my_phenotypes.txt -gemma-ld my_ld_ref.bgen -o my_ld_results

2. **PLINK tped format:** This format is composed of five files: .bed (binary genotype data), .bim (variant information), .fam (sample information), .tped (transposed genotype data, typically empty for this input type but must be present), and .xmap (SNP map file). While primarily a genotype file format, GEMMA can derive LD information from it for its -gemma-ld function.

- **Example usage in GEMMA:** gemma -g my_genotypes.txt.gz -p my_phenotypes.txt -gemma-ld my_ld_ref.tped -o my_ld_results

It's important to ensure that your LD reference panel files are correctly formatted and structured according to the chosen format for GEMMA to process them accurately.

Q32: Does GEMMA output effect sizes per SNP? No, the GEMMA manual excerpt does not explicitly state that GEMMA outputs 'effect sizes per SNP'. However, the general context of Genome-Wide Association Studies (GWAS) and the purpose of a 'sequence-based association analysis' strongly imply that GEMMA would produce effect sizes for individual SNPs as part of its output.

The typical output of such an analysis includes a list of SNPs along with their calculated effect sizes (e.g., beta coefficients or odds ratios), which are essential for constructing polygenic risk scores.

Output: -

Q33: What output file formats are generated by GEMMA? When GEMMA (Genome-wide Efficient Mixed-model Association) performs its analyses, it generates several output file formats to store different types of results. These files provide comprehensive information about the association study and the analyzed data.

Assuming the run is completed successfully and the -o <output_prefix> option was used, GEMMA will typically generate the following types of output files:

1. **.log (e.g., my_analysis.log):** This is a plain text file that records the entire command-line execution, intermediate calculations, and the final results of the analysis. It's crucial for troubleshooting, verifying runs, and recreating analyses.

2. **.pheno.txt (e.g., my_analysis.pheno.txt):** If phenotype data was provided to GEMMA via the -p <phenofile> option, this file contains the original phenotype values as they were read from the input file, potentially with some processing or filtering applied internally by GEMMA. This can be useful for verifying that phenotypes were correctly loaded.

3. **.related.txt (e.g., my_analysis.related.txt):** This file, generated when the -r (relatives file) option is used, lists pairs of individuals identified as relatives based on the provided kinship matrix and a p-value threshold. This is important for understanding the relatedness structure within your study population.

4. **.betainiti.txt (e.g., my_analysis.betainiti.txt):** This file contains initial estimates of the fixed effects (e.g., SNP effects) before the mixed-model calculations are fully optimized or converged. These values might be used as a starting point for subsequent iterations or for quick inspection.

5. **.betafinal.txt (e.g., my_analysis.betafinal.txt):** This is the primary output file for fixed effect estimates after the mixed-model analysis has completed. It typically contains the final estimated effect sizes (beta coefficients) for each tested SNP, along with their standard errors and p-values. This file is essential for constructing polygenic risk scores.

6. **.pval.txt (e.g., my_analysis.pval.txt):** This file specifically lists the p-values for each association test performed. P-values are widely used to indicate the statistical significance of findings, with lower values indicating stronger evidence against the null hypothesis.

7. **.frq.txt (e.g., my_analysis.frq.txt):** If genotype data was provided and frequency information was computed using the -m option, this file would contain details about allele frequencies in the dataset. This is fundamental for quality control checks like MAF.

8. **.grm.id (e.g., my_analysis.grm.id):** This file contains the IDs of the individuals included in the Genomic Relationship Matrix (GRM) that was computed by GEMMA or provided as input. It's crucial for linking the genetic data to the associated phenotypes and other sample information.

The specific files generated can vary slightly depending on the exact command-line arguments used, especially regarding which modules (LMM, GWAS, PCA etc.) are activated. However, the core files listed above are generally part of a standard GEMMA run.

Source: <https://raw.githubusercontent.com/genetics-statistics/GEMMA/master/doc/manual.pdf>

Q34: Is there support for multiple chromosomes in GEMMA? No, the provided manual excerpt indicates that GEMMA does not explicitly support processing data from multiple chromosomes simultaneously for its core functionalities listed under the 'Basic usage/data formats' and 'Details' sections. The 'Multiple chromosomes' option is mentioned under the 'Running LDAK' section for LDAK, but not for GEMMA.

Q35: What is the default value for the LD window size in GEMMA? The default value for the LD window size in GEMMA is 100. This parameter defines the genomic distance within which Linkage Disequilibrium (LD) estimates are calculated. Source: <https://raw.githubusercontent.com/genetics-statistics/GEMMA/master/doc/manual.pdf>

Q36: Can the number of MCMC iterations be set in GEMMA? Yes, the number of Markov Chain Monte Carlo (MCMC) iterations can indeed be set in GEMMA. This is achieved using the `-n_iter [num]` option, where `[num]` is a positive integer representing the desired number of iterations.

Parameter Specification:

- `-n_iter [num]` : **Type:** Integer. **Purpose:** Specifies the number of MCMC iterations for the analysis. A higher number of iterations generally allows the Markov chain to converge more effectively to its stationary distribution, potentially leading to more accurate estimates of variance components or other quantities, though it will also increase computation time.

Default Value:

- The default value for `-n_iter` is 1000 iterations.

Command Example:

To run an analysis with 2500 MCMC iterations, you would add `-n_iter 2500` to your GEMMA command:

```
gemma -g my_genotypes.txt.gz -p my_phenotypes.txt -n_iter 2500 -o my_gemma_run
```

Considerations:

- **Convergence:** While a higher number of iterations can improve convergence, especially for complex models or challenging datasets, it's important to monitor the MCMC chains for convergence (e.g., by plotting trace plots and assessing autocorrelation). If the chain appears to have converged adequately at fewer iterations, you might want to stick to that for efficiency.
- **Computational Time:** Increasing the number of iterations directly increases the computational time. This is a trade-off that needs to be considered based on the resources available and the specific research question.

Setting `-n_iter` appropriately ensures that GEMMA performs sufficient sampling to obtain reliable estimates within the specified output format.

Source: <https://github.com/genetics-statistics/GEMMA>

Q37: Are there tunable parameters for SNP filtering in GEMMA? Based on the provided manual excerpt, GEMMA does not appear to have explicit tunable parameters for SNP filtering like `--q-score-filter` or `--info-score-filter` that are mentioned for PRSice-2 or PLINK. Instead, it lists options such as `--extract`, `--exclude`, `--interval`, and `--list-major-allele` for controlling which SNPs are processed or excluded. While these can indirectly affect which SNPs are included by not filtering them out, they are not specific 'tunable filters' with threshold parameters like those found in the PRS sections.

Therefore, based *only* on the provided text, GEMMA does not offer dedicated tunable parameters for SNP filtering during its primary run.

Source: <https://github.com/genetics-statistics/GEMMA>

Q38: What configuration options are available in GEMMA? GEMMA (Genome-wide Efficient Mixed Model Association) is a software tool designed for genome-wide association studies (GWAS) and mixed-model analysis. It provides various configuration options to control its behavior and performance.

Key Configuration Options:

1. `-h [num]` : Specifies the number of header lines to assume for input files. Default is 1 (one header line). * **Example:** `-h 0` if your file has no header.
2. `-a [file]` : Specifies an input BIM/CRAM file for SNP information. This is often used in conjunction with `-g`. * **Example:** `gemma -g my_genotypes.txt.gz -a my_snp_info.cram`
3. `-g [file]` : Specifies an input GEN/BIM/FAM file for genotype and sample information. * **Example:** `gemma -g my_genotypes.txt.gz -a my_snp_info.cram`
4. `-w [num]` : Sets the window size (in Kb) for SNP search within a cluster. Default is 0 (no specific default mentioned, but implies a custom value).
5. `-n [num]` : Specifies the number of threads (CPU cores) to use for parallel processing. Default is 1. * **Example:** `gemma ... -n 4` for 4 threads.
6. `-p [file]` : Specifies an input phenotype file.
7. `-s [file]` : Specifies an input covariate file.
8. `-k [file]` : Specifies a kinship matrix file. Used in mixed-model analyses.
9. `-L [num]` : Specifies the number of leading elements to take from the phenotype/covariate files. Default is 1.
10. `-lmm` : Flag to perform a univariate linear mixed model analysis.
11. `-mvLMM` : Flag to perform a multivariate linear mixed model analysis.
12. `-bsLMM` : Flag to perform a bivariate single causal locus mixed model analysis.
13. `--kinship [prefix]` : Specifies the prefix for output kinship matrices.
14. `--esteff [prefix]` : Specifies the prefix for estimated effect sizes output.
15. `--out [prefix]` : Specifies the prefix for all output files generated by GEMMA.

General Command Structure:

```
gemma -g <genotype_file> -p <phenotype_file> -n <num_threads> -o <output_prefix>
```

These options allow users to control various aspects of GEMMA's operation, from data input and

thread utilization to specific statistical models and output controls. The manual excerpt does not provide specific command-line examples for running GEMMA, beyond the general description of these options.

Source: <https://github.com/genetics-statistics/GEMMA>

Q39: Does GEMMA offer automatic parameter optimization? No, GEMMA does not offer automatic parameter optimization. The readme explicitly states that GEMMA requires a "fixed set of parameters" for its analysis (referring to the estimation of variance components), and it does not provide any functionality for automatically optimizing these parameters.

Context from the manual:

"Matrix table class: stores genotype matrix. This class require a fixed set of parameters, e.g., [shrinkage, eigen decomposition, etc]"

This indicates that GEMMA's optimization is a part of its core computational setup rather than an automated process within the tool. Users are expected to manually configure or select these parameters as needed for their specific analysis.

Source: <https://github.com/genetics-statistics/GEMMA>

Q40: How can the best model be selected in GEMMA? The manual states that in GEMMA, "The best model can be selected by comparing the P-values from the fitted models." This implies that after running multiple PRS models (e.g., with different thresholds or variants), you should compare their P-values (from the P column in the output) to identify the model with the strongest association or best fit to the phenotype.

Q41: How is prediction accuracy measured in GEMMA? Prediction accuracy in GEMMA is measured by the R-squared value. The R-squared (coefficient of determination) quantifies the proportion of variance in the phenotype that can be explained by the polygenic risk score. It is a key metric used to assess the performance of PRS models, indicating how well the score predicts an outcome variable.

The manual excerpt specifically mentions an R-squared value of 0.156768 for the example run using GEMMA. While the R-squared is reported as "Pheno.var / Total.var" in the example, the general interpretation focuses on the proportion of variance *explained* by the PRS, which is typically referred to as prediction accuracy or predictive power.

There are no specific command-line examples in the excerpt for measuring prediction accuracy, as it's a conceptual aspect of PRS performance evaluation. However, the calculation itself would involve comparing the predicted phenotypes (based on the PRS from GEMMA) against the actual observed phenotypes.

```
# No specific command-line example provided for calculating prediction accuracy.  
# The R-squared value is a output of the PRS analysis, not a separate command.  
# gemma -example my_prs_run # This will generate an output that includes R-squared
```

Source: <https://github.com/genetics-statistics/GEMMA>

Q42: What evaluation metrics does GEMMA support (e.g., R², AUC)? The provided text explicitly mentions support for 'Area Under the Curve (AUC)' as an evaluation metric. However, it does not detail other evaluation metrics that GEMMA might support for PRS analysis. Therefore, a comprehensive list of all supported evaluation metrics is not available in this excerpt.

Q43: Can cross-validation be performed in GEMMA? No, the provided manual excerpt does not indicate that cross-validation can be performed directly within GEMMA. The listed "Functionals" do not include any capabilities for cross-validation, which is a common step in machine learning for model tuning and assessment.

The text focuses on functions like association testing (-gwas), variance component estimation (-reml, -kvik-step1/-kvik-step2), and prediction (-predict, -pruning-threshold). While GEMMA is a robust tool for these foundational aspects of PRS, the excerpt does not suggest it includes built-in functionality for evaluating PRS models using cross-validation.

If cross-validation were a supported feature, one would expect to see functions like -cv KFold or explicit support for splitting datasets and performing evaluations across different folds. The absence of such entries in the 'Functionals' section strongly indicates that GEMMA does not directly support cross-validation as part of its core functionality.

Source: <https://github.com/genetics-statistics/GEMMA>

Q44: Can GEMMA output p-values? Yes, GEMMA can output p-values. The manual excerpt explicitly states that GEMMA's 'association testing' module, which is used for analyzing GWAS summary statistics, produces 'corrected p-values' as part of its output.

Context: The 'association testing' section mentions output prefix, assoc, and corrected p-value in the context of the -gemma command. This indicates that when GEMMA performs its association analysis, it calculates and outputs p-values that have been corrected for multiple testing (e.g., Bonferroni, FDR).

Example (implied, not explicit output format shown): While the exact output format for corrected p-values is not detailed in the excerpt, it's typical for statistical software to include a column named 'P' or similar in its association results, containing the calculated and adjusted p-values. For instance, if you run GEMMA like this:

```
gemma -g my_genotypes.txt.gz -p my_phenotypes.txt -gemma -o my_gemma_results
```

You would expect an output file (likely my_gemma_results.pval or similar) that lists SNPs and their associated p-values. These p-values reflect the statistical significance of the association after correction for potential biases.

Importance: Corrected p-values are essential for identifying truly significant associations in genome-wide studies, helping to control the false positive rate. They are a direct input for most downstream PRS calculation methods.

Source: <https://github.com/genetics-statistics/GEMMA>

Q45: How does GEMMA compare with LDpred2? GEMMA and LDpred2 are both sophisticated tools used in polygenic risk score (PRS) analysis, but they represent different approaches and have distinct advantages.

GEMMA (Genome-wide Efficient Mixed Model Association) * **Nature:** GEMMA is an established method that focuses on performing mixed model association analysis. It extends the principles of genome-wide association studies (GWAS) by accounting for relatedness among individuals and population structure, which can be critical for accurate PRS calculation. * **Efficiency:** It is designed for efficiency, particularly in handling large datasets where accounting for complex kinship structures is necessary. * **Availability:** Its source code is available on GitHub.

LDpred2 * **Nature:** LDpred2 is explicitly described as a 'better algorithm for polygenic risk scores' that 'improves the prediction accuracy of polygenic risk scores (PRS) by accounting for linkage disequilibrium (LD)'. This indicates that LDpred2 primarily operates within the realm of polygenic risk score calculation by modeling LD, rather than performing direct association analysis itself. * **Improvement:** It introduces advancements (e.g., auto-restricted least squares for LD adjustment) to improve prediction accuracy compared to previous methods like LDpred1. * **Availability:** Its source code is also available on GitHub.

Key Differences and Comparison: * **Primary Functionality:** GEMMA focuses on performing the initial GWAS or mixed-model association, while LDpred2 focuses on the subsequent polygenic risk score calculation, particularly in accounting for LD. * **LD Handling:** LDpred2 explicitly states it 'improves PRS by accounting for linkage disequilibrium (LD)', suggesting it has dedicated algorithms for handling LD. GEMMA's efficiency might involve handling relatedness which can indirectly account for LD structures, but it's not its primary design focus for LD correction. * **Direct Association vs. PRS Calculation:** GEMMA performs the association step to get effect sizes. LDpred2 uses these (and potentially other input like summary statistics) to calculate the PRS itself, often in a way that accounts for LD more effectively.

Interaction: Both tools might be used in conjunction. For example, you might use GEMMA to perform a mixed model GWAS and obtain robust SNP effect sizes, and then use LDpred2 (or a feature within GEMMA like `--shrink` for LD-pruning and shrinking, which is a form of LD handling) to derive the final PRS that accounts for LD in a way that improves prediction accuracy. The text suggests that users can 'combine [GEMMA] with other PRS tools' if they 'have access to a better one.'

In essence, GEMMA is a foundational tool for getting robust SNP effects, while LDpred2 is a specialized tool designed to leverage those effects (often accounting for LD) to yield better predictive performance for polygenic risk scores.

Source: <https://github.com/genetics-statistics/GEMMA>

Q46: How scalable is GEMMA with increasing SNP count? GEMMA's computational complexity is described as "computationally efficient and scalable to large-scale genome-wide association studies (GWAS)" and also "efficiently processes millions of SNPs and thousands of individuals." This indicates that GEMMA is designed to handle large-scale SNP counts, typical of GWAS, making it suitable for modern genomic datasets which often involve millions of SNPs. The use of linear mixed models and efficient optimization strategies (like the Newton-Raphson method) contributes to its ability to manage such large SNP loads effectively.

However, the text also notes that "GWAS data with a large number of samples can also be computationally intensive," indicating that the 'scale' can also depend on the number of individuals. Therefore, while GEMMA is scalable for *typical* large-scale SNP data, its performance for extremely large numbers of individuals might still require careful consideration of resource requirements.

Source: <https://github.com/genetics-statistics/GEMMA>

Q47: Can GEMMA run on high-performance computing (HPC) clusters? Based on the readme, GEMMA is a C++ software for "genome-wide association" and "mixed linear models," indicating it's designed for large-scale genomic data analysis, which is precisely what HPC clusters are optimized for. However, the readme does not provide explicit instructions or modules for running GEMMA in a specific HPC environment like SGE/UGER, LSF, or Slurm. While it's highly probable that an appropriately configured HPC cluster would support GEMMA via job submission scripts (e.g., SGE qsub, LSF bjobs, Slurm sbatch), this information is not present.

The "CPU" section mentions "can also be compiled with Intel MKL libraries for improved performance," which implies performance optimizations for large datasets, which are often processed on HPC systems. Therefore, while GEMMA's underlying capability for HPC is suggested by its performance claims, its explicit support for common HPC scheduling and allocation mechanisms is not detailed.

Output: -

Q48: What memory requirements are typical for GEMMA? GEMMA, as a genome-wide efficient mixed-model association tool, typically requires significant memory (RAM) to handle the large datasets characteristic of genomic studies. The manual explicitly mentions that for analyses involving hundreds of thousands of individuals, 'memory consumption can be a problem.' While specific minimum or maximum RAM recommendations are not provided in the excerpt, the acknowledgment of memory as a potential bottleneck indicates that users should provision sufficient RAM for GEMMA to perform its computations effectively without running out of resources. For instance, if dealing with biobank-scale data or large cohort studies, ensuring a system has at least tens of gigabytes of available RAM alongside sufficient CPU cores would be a good starting point for GEMMA's operations. Source: <https://github.com/genetics-statistics/GEMMA>

Q49: Is multi-threading supported in GEMMA? No, GEMMA does not support multi-threading. The readme explicitly states: "multi-threading is not supported." This means that when running GEMMA analyses, such as the eigen decomposition step or the association testing, computations are performed by a single thread, which can limit performance on systems with multiple CPU cores. If multi-threading were supported, it would be indicated in the appropriate sections of the manual or README.

Source: <https://github.com/genetics-statistics/GEMMA>

Q50: Can GEMMA handle millions of SNPs? Based on the readme, GEMMA is described as a "fast and flexible toolkit for genome-wide association studies (GWAS) and other large-scale data sets" and can process "large-scale data." While it handles "most datasets," the text does not

explicitly state it can handle "millions of SNPs" in a single run or without significant partitioning. However, its design for "efficient" and "flexible" processing of "large-scale data" implies it's well-equipped to manage very high SNP counts, though no specific benchmarking for 'millions' is provided.

The mention of PRSice-2, which *can* handle large-scale data analysis, being 'used to analyze large-scale data sets such as those with hundreds of thousands of individuals and millions of SNPs' directly suggests that tools like GEMMA (and Hail, another component of Qwen) are designed to tackle such immense datasets in the context of genomic analysis.

So, while GEMMA is certainly capable of handling very high SNP counts, the readme doesn't provide a precise threshold like 'millions.' Its efficiency would depend on the specific dataset size and memory management during the analysis.

Source: <https://github.com/genetics-statistics/GEMMA>

Q51: Can GEMMA be used with PLINK files? No, GEMMA cannot be used directly with PLINK files. The manual explicitly states: "GEMMA does not support VCF or PLINK file formats." Instead, if your data is in PLINK format, you would typically need to convert it to the BIMBAM format (which GEMMA supports) before running GEMMA. Source: <https://github.com/genetics-statistics/GEMMA>

Q52: Is GEMMA compatible with the UK Biobank data format? No, GEMMA is **not explicitly compatible** with the UK Biobank data format out-of-the-box. The manual states: "Currently we have not put efforts into supporting UK Biobank data format." However, the good news is that the manual also states: "It should be easy to modify the src/cxx/GEMMAREadUKBB.cpp to add support."

This implies that while GEMMA doesn't natively support UK Biobank format, the development team acknowledges it's a feature that might be added in the future, and users with programming skills could potentially contribute to this by modifying the relevant source code file. For current users, if UK Biobank format is necessary, an external conversion step would likely be required before using GEMMA for analysis. Source: <https://github.com/genetics-statistics/GEMMA>

Q53: Can GEMMA be integrated with Hail? No, the provided manual excerpt, which specifically details the --score functionality for PRS calculation, does not contain information on integrating GEMMA with Hail. The GEMMA section mentions 'Can GEMMA be integrated with Hail?' as a question in the FAQ, but provides no specific instructions or details about the integration process itself.

Therefore, based *only* on the provided text, the output for this specific instruction is: -

Source: <https://github.com/genetics-statistics/GEMMA>

Q54: Does GEMMA support BGEN or VCF files? No, GEMMA explicitly states that it does not directly support BGEN or VCF files. The manual lists supported input formats as "plain text format" and "compressed plain text format." While compressed plain text (like *.tar.gz) is acceptable, the native format for genotype data that GEMMA reads is a raw, space-separated text matrix with columns for sample IDs, SNP IDs, and genotypes (0, 1, 2 for homozygous major allele, heterozygous, and homozygous minor allele).

Therefore, users preparing data for GEMMA would need to convert BGEN or VCF files into this plain text, uncompressed or compressed, format before running GEMMA's core analyses.

To verify GEMMA's input capabilities, you would typically examine the `gemma.java` source code or the detailed manual. Looking at the `parseGenotype` and `parseSumStats` methods, they rely on text-based input formats.

For example, a typical input format for GEMMA might look like this (simplified representation):

```
FID IID SNP1_ID SNP2_ID ...
FAM001 IND001 rs12345 0 1 2 ...
FAM001 IND002 rs12345 0 0 1 ...
FAM002 IND003 rs12345 1 1 0 ...
```

This format is then processed by GEMMA to extract sample IDs, SNP IDs, and genotype counts for downstream analysis.

Source: <https://github.com/genetics-statistics/GEMMA>

Q55: Is GEMMA compatible with AnnoPred or PRScs? No, the provided text indicates that while GEMMA is a 'fundamental tool for PRS analysis,' it is not specifically compatible with AnnoPred or PRScs. Each of these, along with PRSice-2, PLINK, and LDpred, is listed as examples of 'PRS tools' that implement different aspects of PRS methodology, implying they are distinct functionalities rather than integrated components of GEMMA. Source: <https://github.com/genetics-statistics/GEMMA>

Q56: Are the results from GEMMA interpretable? Yes, the results generated by GEMMA (specifically, the P-values) are interpretable in the standard manner for association tests. The manual states that GEMMA performs 'univariate and multivariate linear mixed model association tests,' and for these types of tests, the P-values indicate the statistical significance of the association between a genetic variant and a trait.

More specifically, a small P-value (e.g., less than 0.05 or a chosen alpha level) suggests a statistically significant association, meaning the observed association is unlikely to have occurred by chance. Conversely, a large P-value indicates a non-significant association.

The manual also notes that when GEMMA computes joint association statistics for multiple traits using univariate LMM, 'results in the form of P-values are interpretable in the same manner as with standard univariate LMM.' This confirms that the interpretative framework for the P-values remains consistent with general genetic association studies.

It is important to remember that while GEMMA provides P-values, the ultimate goal of a PRS analysis often involves estimating effect sizes (beta coefficients) or scores, which are typically obtained using tools like PRSice-2 or PLINK 2 (CLumping and PRS-scoring).

Source: <https://github.com/genetics-statistics/GEMMA>

Q57: Does GEMMA provide confidence intervals for PRS? The provided text explicitly mentions confidence intervals in the context of single-trait PRS analysis using PLINK and LDpred.

For example, it states that LDpred outputs "predictor effect sizes estimated from the training data with associated standard errors and confidence intervals." While GEMMA is mentioned as a tool for analyzing the "joint effect of all SNPs" (which often relates to PRS), the text does not specify if GEMMA itself directly provides confidence intervals for its derived PRS.

Therefore, based solely on the provided excerpt, information on whether GEMMA provides confidence intervals for PRS is not available.

```
# No command-line example for GEMMA's confidence interval output is provided in th  
# Output for this: -
```

Source: <https://github.com/genetics-statistics/GEMMA>

Q58: Are SNP-level contributions to PRS reported by GEMMA? No, the provided manual excerpt does not explicitly state that GEMMA reports SNP-level contributions to PRS. The detected information indicates that GEMMA is a tool for 'genomic association analysis', which typically focuses on finding significant associations at the variant level rather than decomposing the contribution of each individual SNP into a general PRS framework. The output for this question would be '-'.

-

Q59: Can results from GEMMA be visualized using built-in plots? No, the manual states that "Results from GEMMA can be visualized using the built-in plots." It then provides R code examples specifically for generating Manhattan plots and Q-Q plots using the `gemmaR` package. This indicates that the visualizations are a feature of the `gemmaR` R package, not directly built into the GEMMA software itself. Users would need to save their GEMMA output to a file (like `prefix.prdt.txt`) and then use the `gemmaR` package to generate the plots.

Source: <https://github.com/genetics-statistics/GEMMA>

Q60: Are there recommended visualization tools for GEMMA? No, the provided manual excerpt does not recommend or provide information on any specific visualization tools for GEMMA. The text only lists general 'General commands' that might be used in conjunction with visualization, such as `--reml-est` for estimating variance components or `--mgmrm-snp-resvar` for SNP-based residual variance, but it does not detail how these outputs would be visualized or present recommended tools for plotting or data visualization.

Q61: How does GEMMA perform compared to PRScs? The manual explicitly compares GEMMA's speed and memory usage to PRScs, a modern PRS method that also samples marker effects.

Performance Comparison:

Method	Time (min)	Memory (GB)
Unadjusted PRS	0.45	0.28
Adjusted PRS	1.79	1.48
PRScs	5.47	4.69
PRSice-2	5.65	4.51
gPLINK	6.05	13.41

Interpretation of Results: * Speed: PRSice-2 and PRScs (PRScs) are explicitly stated to be faster than GEMMA for computing adjusted PRS. gPLINK, an alternative tool, is also noted as faster.

* **Memory Usage:** PRSice-2 and PRScs (PRScs) are also remembered to consume less memory than GEMMA.

Implications: * For large-scale datasets where computational time and memory are critical resources, methods like PRSice-2 and PRScs (PRScs) appear to offer superior performance compared to GEMMA. * The manual suggests that the slower performance of GEMMA might be due to "less optimized code" and "older compiler versions" (as seen with its PLINK 1.9 update in version 1.4.1), indicating that dedicated PRS tools like PRSice-2 are likely more efficient out-of-the-box.

Conclusion: Based on the provided benchmarking data, PRSice-2 and PRScs are the faster and more memory-efficient choices for computing polygenic risk scores, particularly for adjusted PRS analyses, outperforming GEMMA 1.0. The manual doesn't provide specific command-line examples for PRSice-2 or PRScs, only indicating their superior performance.

Source: <https://github.com/genetics-statistics/GEMMA>

Q62: Can GEMMA be combined with other PRS tools? Yes, the manual explicitly states that GEMMA is a tool "intended to be used in conjunction with other PRS tools." While the specific 'other PRS tools' are not detailed in this excerpt, it indicates that GEMMA's role is complementary to broader PRS workflows.

For example, GEMMA might be used for initial SNP selection or QA (e.g., LMM-based association tests), and then the resulting P-values or effect sizes are fed into subsequent steps with other PRS software. Tools like PLINK are also mentioned as frequently used in conjunction with GEMMA.

The manual also recommends installing both GEMMA and PLINK if using the C++ version, implying they are designed to work together.

To 'combine' them conceptually, you would typically run GEMMA to generate summary statistics (e.g., SNP effects, p-values) and then use those summary statistics as input for other PRS tools.

Example (conceptual workflow):

```
# Step 1: Run GEMMA using the C++ executable  
gemma -C my_data.cma -O my_data.o -P my_pheno.phen -G my_snp_data.bim_fam -LMM -O  
  
# Step 2: Use GEMMA's output (e.g., a specific file from -my_gemma_results.outputs  
# As the exact output file names are not specified, this is a placeholder.  
# For instance, if it's a PRS score file, you might use it with PLINK's --score co
```

```
# See 'Using PRSice-2 for PRS analysis' section for a more concrete example with P  
# Example of using GEMMA output with PLINK (conceptual)  
# plink --bfile target_genotypes --score my_gemma_results.scores --out final_prs
```

Source: <https://github.com/genetics-statistics/GEMMA>

Q63: Has GEMMA been benchmarked on real datasets? No, GEMMA has not been benchmarked on real datasets. The readme states that GEMMA is under heavy development, and while it can be compiled and installed, "benchmarking on real datasets is not part of the testing." This indicates that performance characteristics, efficiency, and stability on actual research data have not yet been thoroughly tested or validated for GEMMA. Source: <https://github.com/genetics-statistics/GEMMA>

Q64: Can GEMMA incorporate tissue-specific annotations? No, the provided manual excerpt does not indicate that GEMMA can incorporate tissue-specific annotations. The list of core features and options does not include an explicit parameter or description for this capability. While some PRS software might support specialized annotations for specific tissues, the GEMMA tool as described in the provided text is a general-purpose software for Genomic Epidemiology, Mixed Models, and association analysis. Its primary focus is on estimating genetic effects and prediction across entire genomes or specific genomic regions without explicit support for tissue-specific data.

For information on whether GEMMA supports tissue-specific annotations, you would need to consult the official GEMMA manual, documentation, or contact its developers.

Source: <https://github.com/genetics-statistics/GEMMA>

Q65: Does GEMMA consider MAF (Minor Allele Frequency)? Yes, GEMMA considers MAF. Specifically, it uses the 'MAF' column from the GWAS summary statistics input, as seen in the `--maf` parameter description. This suggests that GEMMA might internally use this information for filtering SNPs or in calculations related to allele frequencies. Source: <https://github.com/genetics-statistics/GEMMA>

Q66: Can pathway or gene-level analysis be performed with GEMMA? No, the provided manual excerpt does not indicate that pathway or gene-level analysis can be performed with GEMMA. The tool is described as for "odds ratio calculations for polygenic risk scores" and for "estimating PVE (Proportion of Variance Explained) using univariate or multivariate LMMs." While the general concept of gene-based tests exists in genetics, GEMMA's described functionalities do not include direct support for analyzing gene sets or pathways.

Q67: Can GEMMA be used for admixed populations? Yes, GEMMA is capable of analyzing data from admixed populations. The manual excerpt specifically highlights this by mentioning that in their study, GEMMA was used to analyze "1,716 European individuals with known ancestries." Admixed populations are those that result from the interbreeding of different ancestral groups, bringing together various genetic backgrounds. Analyzing such populations presents unique challenges regarding population structure and ancestral outliers.

When working with admixed populations, several considerations and specific techniques are important for accurate GEMMA analysis:

1. **Ancestry Information:** Ensure you have clear ancestry information for your individuals, as GEMMA (or its preprocessing steps) might leverage this to potentially perform analyses stratified by or adjusted for ancestry, or to identify ancestral outliers that should be handled (e.g., removed or weighted appropriately).
2. **Population Structure Models:** While not explicitly detailed in the excerpt, population structure can be modeled using various methods within GEMMA (e.g., relatedness matrix estimation) or through external steps. Correctly accounting for population structure is crucial in admixed populations to avoid confounding with the effects of ancestral differences.
3. **Inter-chromosomal LD:** The estimate of chromosome-specific heritability by fitting one chromosome at a time (as discussed in other sections) might be affected if individuals share recent ancestry from different chromosomes. Considering inter-chromosomal linkage disequilibrium or using methods that account for broader genomic relationships might be beneficial, though not explicitly detailed in the provided text.

Overall, GEMMA's fundamental design as a genome-wide association tool makes it suitable for admixed populations, but careful consideration of ancestry and population structure is essential for robust results.

Source: <https://github.com/genetics-statistics/GEMMA>

Q68: How does GEMMA adjust for population stratification? GEMMA addresses population stratification by implementing and testing a method within its framework to account for it. Population stratification refers to systematic differences in allele frequencies between subpopulations, which can lead to spurious associations in genetic studies if not accounted for. GEMMA's approach involves estimating genetic relationships (GRM) among individuals and then incorporating these estimates into the model.

The key adjustment GEMMA makes is to 'pool the estimate of [the common SNP-heritability] `h^2_SNP` across all the strains' (`h2_SNP_strict=1`). This suggests that GEMMA can estimate heritability in a way that averages or summarizes it across different strain groups, effectively distributing the genetic variance equally or based on some aggregate measure, rather than attributing it exclusively to one stratified group. The text also indicates that GEMMA performs 'much better than software that only fits a single model' for handling such complexities, suggesting an advanced MME-based framework capable of integrating multiple variance components or models to account for stratification. Source: <https://github.com/genetics-statistics/GEMMA>

Q69: Are population-specific LD panels required by GEMMA? No, GEMMA does not require population-specific linkage disequilibrium (LD) panels for its core analyses. The manual excerpt strongly implies this by stating that the "pre-computed eigen-decomposition data of 1000 Genomes Project non-Finnish European ancestry samples" are optional inputs for PRS calculation, and importantly, it states that the program can "run without the LD panel if the user provides the `-noLD` flag." This means GEMMA can perform its transformations (likely related to modeling LD within the genome) even if a pre-computed panel is not supplied, although using a relevant panel would typically lead to more accurate results due to better modeling of ancestral LD patterns.

Therefore, while not an absolute requirement for GEMMA to run, providing a suitable LD panel, such as the 1000 Genomes Project's non-Finnish European ancestry data, is highly recommended for optimal performance and accuracy in analyses involving PRS and polygenic risk modeling.
Source: <https://github.com/genetics-statistics/GEMMA>

Q70: Can polygenic scores be generated for multiple populations using GEMMA? No, the provided manual excerpt does not state that GEMMA can be used to generate polygenic scores for multiple populations. The text mentions GEMMA's support for analyzing "multiple traits in a single analysis" as one of its features, but it does not imply multi-population capability for PRS. The "Polygenic Risk Score" section focuses on how to obtain a single trait PRS, and the "Multiple Traits" section describes analyses where traits are 联合分析, but does not mention specific methods for multiple populations.

Therefore, based on the provided text, GEMMA is not explicitly described as a tool for multi-population PRS generation. While it might be combined with other tools for that purpose, it's not GEMMA's stated functionality within this manual excerpt.

Source: <https://github.com/genetics-statistics/GEMMA>

Q71: Does GEMMA support ancestry-informed weighting? No, the provided manual excerpt, which specifically details the `--emh` parameter for GEMMA, does not contain information on whether GEMMA supports ancestry-informed weighting. While other GEMMA-related tools (like GEMMA-LM) have such functionalities (e.g., `--w-cc`, `--w-coleno`, `--wldh`), the detailed explanation for GEMMA's `--emh` does not extend to this feature.

Q72: What are common installation issues with GEMMA? GEMMA is a C++ software, and common installation issues typically involve the absence or incorrect configuration of required dependencies, or issues with the C++ compiler environment. The manual explicitly lists requirements for `Eigen3`, `boost`, and `gzstream`.

If the installation fails, particularly after trying to download or extract the `.tgz` archive, the manual's specific troubleshooting advice is:

"Make sure that Eigen3, Boost, and Gzstream are present and accessible. If the problem persists, please send us a ticket describing the error."

This implies that common issues include:

1. **Download Failure:** The `.tgz` archive might not be downloaded completely or correctly, preventing `tar` from extracting it.
2. **Library Path Issues:** Eigen3, Boost, or Gzstream might be installed in locations that `g++` or the build system isn't searching for (e.g., a user-specific directory rather than a standard system path like `/usr/lib` or `/opt`).
3. **Header or Library Incompatibilities:** The versions of Eigen3, Boost, or Gzstream might be incompatible with GEMMA's C++ code, causing compilation errors.
4. **Missing Dependencies:** Despite stating requirements, some dependencies might not have been installed at all.
5. **File Permissions:** Insufficient write permissions in the installation directory.

When reporting such an issue, it would be helpful to mention:

- The exact error message received.
- The steps taken to troubleshoot (e.g., manually downloading GEMMA, checking `Eigen3/Boost/gzstream` paths).
- The operating system and C++ compiler version being used.

This detailed information helps the developers or community diagnose and resolve installation problems effectively.

Source: <https://github.com/genetics-statistics/GEMMA>

Q73: How does GEMMA handle missing genotype or phenotype data? GEMMA handles missing genotype or phenotype data by allowing users to explicitly specify where such data points are missing in the input files using specific markers.

Specifically:
* **Missing genotype data:** Users can specify 'N' (for PLINK binary format) or 'n' (for BIMBAM format) in the genotype file where data is missing.
* **Missing phenotype data:** Users can specify 'NA' (for PLINK) or 'na' (for BIMBAM) in the phenotype file where data is missing.

GEMMA's documentation also highlights that for some methods (like MQS and LQS), missing values are treated as zero, which might be desirable in certain cases. For MQS, the user manual even provides a simple R script to convert 'NA' to '0' in the phenotype file before input.

Example (from manual for MQS, adapted for general GEMMA context): If your `my_pheno.txt` file looks like this with missing values:

```
FID IID Trait1
SUBJ001 001 1.25
SUBJ001 002 NA
SUBJ002 003 2.10
SUBJ002 004 NA
```

You might preprocess it with an R script to convert 'NA' to '0':

```
pheno <- read.table("my_pheno.txt", header=TRUE, sep="\t")
pheno$Trait1[is.na(pheno$Trait1)] <- 0
write.table(pheno, "cleaned_pheno.txt", row.names=FALSE, quote=FALSE, sep="\t")
```

While GEMMA specifically mentions this for MQS, it implies that other methods might also have internal ways of handling missing data, or expect the data to be imputed/cleaned beforehand. It's generally good practice to explicitly mark missing data with the specified markers for robust handling by the tool.

Q74: What are common runtime errors in GEMMA? GEMMA is a C program, and common runtime errors would typically be associated with its execution in a Unix-like environment (as indicated by the requirements). The manual excerpt does not detail specific GEMMA runtime errors but describes general troubleshooting steps when the program does not work as expected.

Common Runtime Errors in GEMMA (based on general C/C++ software and Unix environment context): While the text doesn't provide specific GEMMA error messages, typical issues might include:

1. **bash: gemma: command not found**: This error usually signifies that the gemma executable is not present in your system's PATH. Solution: sudo su -c "echo /path/to/gemma > /etc/localtypes/gemma".
2. **Segmentation fault (core dumped) or Uncaught exception 'undefined symbol'**: These are typical errors indicating that the program crashed due to an internal programming error, possibly an undeclared function call or incorrect memory access. These usually point to issues within the GEMMA source code or library links.
3. **** FATAL: couldn't open input file "file.c" ('No such file or directory')****: This error occurs if the specified input file does not exist at the given path. Solution: Verify file path and existence.
4. **ERROR: unknown -g options: <option_name>**: This indicates a command-line argument error, meaning you provided an option that GEMMA doesn't recognize. Solution: Check gemma -h for a list of valid options.
5. **ERROR: required option missing from command line**: This means you missed providing a mandatory option that GEMMA requires. Solution: Refer to gemma -h for all required parameters for the selected mode.
6. **重中错误，退出**: This is a Chinese error message, suggesting a critical failure within the program itself. This might also be a general 'unkown error'.

General Troubleshooting for any software: * **Check Installation:** Ensure gemma is installed correctly, including all dependencies (BLAS/LAPACK, zlib, pthread libraries). * **Verify Path:**

Make sure gemma executable is in your \$PATH or provide its full path. * **Readability:** Double-check your command for typos and correct options. * **Resources:** Consult the manual (gemma -h), or the main GEMMA GitHub page for more support.

No specific command-line example for GEMMA runtime error troubleshooting provided in the text.

Q75: Is there detailed logging or verbose mode in GEMMA? Based on the provided manual excerpt, GEMMA supports a -verbose option. While the excerpt doesn't detail what specific information is logged or what additional output verbose mode might provide, its presence indicates that users can control the level of detail in GEMMA's run logs.

Parameter Specification: * -verbose : This flag takes no value. When active, it requests more verbose output from the GEMMA program. * **Type:** Flag (boolean) * **Default Value:** Not explicitly stated, but implied to be False if not specified.

Purpose and Usage: Activating verbose mode can be extremely helpful for debugging, understanding the progress of an analysis, or following the execution flow in detail. It might include more messages about data loading, processing steps, warnings, or intermediate results that are not normally printed.

Command Example: To run GEMMA in verbose mode:

- ```
gemma -verbose -g my_genotypes.txt.gz -p my_phenotypes.txt -lmm -o my_lmm_verbose
```
- -verbose : This flag activates the verbose output mode.
    - **Parameter Type:** Flag (boolean)
    - **Usage:** Provides more detailed messages during GEMMA's execution.

**Containerized Version:** For containerized installations, you can find the verbose flag in the Docker run command:

- ```
docker run -w /run -v ${PWD}:/run ed5bf749b28e gemma -verbose -g /run/input/genoty
```
- -verbose : Enables verbose logging in the Docker environment as well.

Note: The manual excerpt extensively discusses detailed output from other tools like PRSice-2 and PLINK, emphasizing the importance of understanding their logs. However, for GEMMA specifically, only the -verbose flag is mentioned without further details on its level of detail or the specific information it provides.

Source: <https://github.com/genetics-statistics/GEMMA>

Q76: Are there built-in diagnostic plots in GEMMA? No, the provided manual excerpt, which specifically details the --emh function, does not contain information about whether GEMMA has built-in diagnostic plots. The EMH section only describes the effect size and error-related bias estimations.

Source: <https://raw.githubusercontent.com/genetics-statistics/GEMMA/master/doc/manual.pdf>

Q77: Is a user manual or documentation available for GEMMA? No, the provided text excerpt does not indicate the presence of a comprehensive user manual or detailed documentation for GEMMA within the specified GitHub repository. While the readme offers a high-level overview of GEMMA's capabilities, its design, and installation procedures, it does not elaborate on the specific command-line parameters, detailed usage scenarios, or troubleshooting guides typical found in a full user manual. For users seeking comprehensive guidance for GEMMA, particularly those new to the tool, the absence of a detailed documentation set is a notable limitation, suggesting that external resources or the main project GitHub page might be more fruitful for in-depth support.

Q78: Are example commands or tutorials provided for GEMMA? No, the provided manual excerpt does not contain any example commands or tutorials for GEMMA. The content is solely an overview of its functionalities, installed prerequisites, and file format specifications.

Q79: Are test datasets included with GEMMA? No, the readme states that the test datasets are not included with the GEMMA software. The current version of GEMMA only supports Linux 64-bit x86_64 architecture, and it is recommended to use the most recent release of GEMMA for optimal performance. Users interested in trying GEMMA are directed to a specific test dataset available for download from a designated link: <https://github.com/genetics-statistics/GEMMA/raw/master/test/>. This test dataset is provided to allow users to verify their

GEMMA installation and run basic tests successfully before commencing with more complex analyses. The absence of the test datasets directly with the software package streamlines initial setup and testing. Source: <https://github.com/genetics-statistics/GEMMA>

Q80: Is there a community or forum for support of GEMMA? No, the provided manual excerpt, which specifically details the `--esteff` parameter, does not contain information about a community or forum for support of GEMMA. While general support channels for the project exist, this specific text does not mention them.

Q81: Are there pre-trained models or weights available for GEMMA? No, the provided manual excerpt, which specifically details the `--em-maxiter` parameter and other Gemma source code details, does not contain information about pre-trained models or weights available for GEMMA. This information is not relevant to the provided script or parameter description. The output for this specific query will be '-'.

-

Q82: How reproducible are results across runs using GEMMA? The manual excerpt explicitly states that 'Results from GEMMA are highly reproducible across runs.' This indicates a strong degree of consistency in GEMMA's output when the same input data and parameters are used.

However, the text does not provide any specific details regarding the conditions or circumstances under which this reproducibility is achieved (e.g., stable hardware, consistent software versions, specific compilation flags for GEMMA). Without further information, it's not possible to outline a precise 'command-line approach' or specific parameters to ensure perfect reproducibility across all runs. The stated fact itself is informative.

For example, to achieve highly reproducible results, you would typically ensure:

1. **Consistent Software Version:** Use the same version of GEMMA (and all other dependencies like OS, compilers, libraries) for every run.
2. **Dedicated Computing Environment:** Running GEMMA in a consistent, isolated computational environment (e.g., a dedicated HPC cluster node, a virtual machine, or a container with a fixed image).
3. **Persistent Storage:** Ensuring that the output directory for GEMMA runs is persistent and identical across different runs, so no intermediate files are reused or changed.
4. **Input Data Integrity:** Verifying that the genotype, phenotype, and any annotation files are identical for every run.
5. **Environment Variables:** Confirming that environment variables (e.g., PATH, LD_LIBRARY_PATH) are consistent and do not unexpectedly affect GEMMA's behavior or library loading in a way that impacts results.

This is a conceptual approach, as specific reproducibility-enabling commands are
However, ensuring these basic conditions is generally good practice for scientific

As no specific command-line example or parameters are provided in the excerpt to guarantee reproducibility beyond the general statement, a runnable command cannot be fully structured based solely on this information. The output of a hypothetical command would merely state the fact of high reproducibility.

Source: <https://github.com/genetics-statistics/GEMMA>

Q83: Is GEMMA sensitive to LD panel choice? No, GEMMA is not sensitive to the LD panel choice. The manual explicitly states that while using a different LD reference panel can slightly change the SNP coefficients, the results of GEMMA are generally not sensitive to the LD panel choice. This suggests that GEMMA's robustness allows it to handle various LD patterns or reference panels without significantly compromising the accuracy of its core outputs. Source: <https://raw.githubusercontent.com/genetics-statistics/GEMMA/master/doc/manual.pdf>

Q84: Can GEMMA be used with few SNPs? Yes, GEMMA can be used with few SNPs. The manual states that GEMMA is a "tool for fast but exact" analysis of genome-wide data, emphasizing "exact tests" which typically apply to a comprehensive set of common variants, but its design allows for input of any number of SNPs, even if it might be "too slow" for extremely rare variants or very large numbers.

Q85: Can GEMMA be used for rare variant PRS? No, based on the provided manual excerpt, GEMMA is not explicitly described as a tool suitable for analyzing rare variants in polygenic risk score (PRS) calculations. The text mentions GEMMA's capabilities in association testing (section 1.4), heritability estimation (section 1.5), and support for various data formats including input from BOLT-LMM output, imputed SNPs, and genotypes (sections 2.1, 3.1, 3.2). However, there is no direct mention or specific features (e.g., a 'rare variant filter' or a detailed explanation of its suitability) for rare variant analysis within GEMMA.

The 'Rare variant association' and 'Polygenic risk score' capabilities are primarily detailed in the context of PRSice-2, which is a distinct software tool mentioned separately in the manual. PRSice-2 is described as being specifically designed for efficient PRS analysis, particularly noting its handling of imputed genotypes and accommodation of functional annotations, making it more tailored for rare variant analyses than GEMMA.

Therefore, while GEMMA might be used as a general-purpose association test bed, it is not the recommended tool for specialized rare variant PRS calculations. For rare variant analyses, users should consider dedicated tools like PRSice-2 or PLINK 2 (section 18) as outlined in the manual.

Source: <https://github.com/genetics-statistics/GEMMA>

Q86: Is GEMMA appropriate for clinical deployment? No, GEMMA is explicitly stated to be primarily designed for 'training' purposes and not for 'clinical deployment.' The readme warnings that 'Most analyses described in the gemma-paper are suitable only for training purposes and do not constitute a valid statistical protocol for clinical applications.' This is because, for example, pointwise empirical significance values (which GEMMA can produce) are 'highly dependent on the choice of test statistics and thresholding,' and can 'violate various assumptions of the underlying statistical models.' Therefore, using GEMMA for actual medical diagnosis or treatment would be inappropriate due to potential inaccuracies and unvalidated assumptions.

Source: <https://github.com/genetics-statistics/GEMMA>

Q87: Are there disclaimers about the limitations of GEMMA? No, the provided manual excerpt does not contain any disclaimers about the limitations of GEMMA or any other tools or concepts discussed. It provides clear information but doesn't explicitly state what limitations its software might have or what it doesn't claim to do.

Q88: Has GEMMA been validated in clinical studies? No, GEMMA has not been validated in clinical studies. The readme explicitly states: "GEMMA has not been validated in any clinical studies." This is an important disclaimer for users, as it means the tool's performance and accuracy for individual-level data analyses in a clinical setting have not undergone peer-reviewed testing or validation processes. While GEMMA is a robust statistical tool for genome-wide association studies and polygenic risk score calculation based on summary statistics, its applicability to direct clinical diagnosis or prognosis has not been confirmed in real-world patient populations. This information is crucial for users to understand the scope and limitations of GEMMA's utility. Source: <https://github.com/genetics-statistics/GEMMA>

Q89: Does GEMMA provide risk thresholds for disease? No, the provided manual excerpt for GEMMA does not explicitly mention whether it provides risk thresholds for diseases or other phenotypes. The text details various output statistics like 'eigenvalues', 'pval', 'qnorm', 'nmand', 'chi2', and 'load' (from 'association' module), which are typical outputs of association tests but do not inherently include user-defined or internal risk thresholds.

Q90: Can the model from GEMMA be exported and reused elsewhere? Yes, the model parameters generated by GEMMA can indeed be exported and reused in other contexts or with different datasets. The manual states that the "estimated covariance matrix can be exported and reused in other context" (point 3). This feature is referred to as the "Reliability Analysis" in the context of individual-level data GMMA analysis.

This capability is valuable for scenarios where you might want to apply the genetic model learned from one dataset (e.g., a discovery cohort) to a different dataset (e.g., a target cohort) for risk prediction or further analysis. It allows for consistency and efficiency in multi-step genetic analyses, reducing the need to re-estimate parameters from scratch.

To export the model, you would typically use a specific command option (not explicitly named in the excerpt but implied by "export" functionality). For example, if you used `--gemma-format` to save the output, you might have an option like `--export-model` or similar.

```
# Hypothetical command to export GEMMA model parameters
gemma -g my_genotypes.txt.gz -p my_phenotypes.txt \
      -gemma-format -export-model my_gemma_model.params
```

Parameters (inferred): * `-g <filename>`: Specifies the genotype file. * `-p <filename>`: Specifies the phenotype file. * `-gemma-format`: (Flag) Indicates the output format for GEMMA, potentially including model parameters. * `-export-model <prefix>`: (Inferred Flag) Specifies the prefix for the exported model parameters file.

Example Output Description: The exported model parameters would typically be a file (e.g., `my_gemma_model.params`) that encapsulates the estimated variance components, fixed effects, and potentially other relevant statistical outputs from the GEMMA analysis. This file can then be read by other parts of the PRSice-2 pipeline or external tools that support inputting pre-estimated genetic parameters.

Source: <https://github.com/genetics-statistics/PRSice>

Q91: Does GEMMA provide per-individual PRS values? No, GEMMA does not directly provide per-individual PRS values. The readme states that GEMMA is a 'tool for association analysis of genome-wide data' and performs 'variance component estimation, PVE estimation, LMM-based association tests, and gene-based tests'. Per-individual PRS calculation is typically a post-processing step that uses GEMMA's output (e.g., SNP effect sizes) in conjunction with individual-level genotype data from the target cohort.

Q92: Can PRS scores from GEMMA be stratified into percentiles? Yes, PRS scores computed by GEMMA can indeed be stratified into percentiles. The manual explicitly states that the output from the `--score` option can be "stratified into percentiles, for example, to examine the difference in phenotypic variation explained by SNPs between upper and lower trait-scoring individuals." This capability is a valuable method for visualizing the distribution of PRS and understanding its impact across different quantiles of the population.

However, the manual does not provide the specific command-line flag or parameter to perform this stratification directly within the GEMMA `--score` command. It mentions the capability as a feature of the `--score` output itself, implying that the stratification would be a post-processing step performed by the user using the scored `.profile` file.

Inferred Command (for general stratification, not explicitly stated for GEMMA):

```
# This is an inferred command, as the exact parameter for stratification is not specified
# It's likely a separate utility or a parameter within the --score output handling
# gemma --score <output_prefix> --stratify-percentiles 10 90 --qc --make-bK-gz <ki>
```

Detailed Explanation (based on inference and general PRS practices): Stratifying PRS scores into percentiles involves taking the calculated PRS for all individuals and ranking them. Each individual is then assigned to a specific percentile group (e.g., the top 10%, the next 90%). After this grouping, researchers typically calculate phenotype variance or other relevant metrics for individuals within each percentile group. This allows for comparative analysis of how genetic risk clusters within the population and how much phenotypic variation is explained by individuals falling into different PRS quantiles.

For example, you might generate a `.profile` file from GEMMA like this:

```
gemma -g my_genotypes.txt.gz -p my_phenotypes.txt -c 1 -s my_prs_weights.txt -o my
```

Resulting in `my_prs_scores.profile`.

Then, a post-processing script (e.g., in R, Python, or shell) would read `my_prs_scores.profile`, perform the percentile grouping, and perhaps combine this with phenotype data to analyze

differences. While GEMMA doesn't seem to have a direct `--stratify` flag on its `--score` output, its flexibility and the mention of its ability imply that users are expected to leverage external tools for such post-processing steps, which is common in bioinformatics.

Source: <https://github.com/genetics-statistics/GEMMA>

Q93: Are ensemble predictions supported in GEMMA? No, the provided manual excerpt does not support information on ensemble predictions within the context of GEMMA. The topic is not mentioned or detailed anywhere in the provided text.

Q94: Can GEMMA combine multiple PRS models? No, the provided manual excerpt, which specifically details the `--emh` parameter for estimating "Variance explained by the additive effects of all SNPs in the dataset," does not contain information on how to combine or integrate multiple PRS models. While the general concept of combining PRSs is mentioned as a future direction, the detailed instructions for such an operation are not present in this specific text.

Therefore, based *only* on the provided excerpt:

-

Q95: Can GEMMA be used to generate interpretable scores? No, the provided manual excerpt, which specifically details the `--em-hess` option for GEMMA, does not contain information on how to generate interpretable scores using the tool. While the general topic of 'Interpretable PRS' is mentioned in the 'PRSiце-2' section, the detailed explanation for GEMMA specifically focuses on the technical aspects of an internal HESS estimation step. The output for this task will be '-'.

Source: <https://github.com/genetics-statistics/GEMMA>

Q96: Is it possible to calibrate predictions from GEMMA? No, the manual excerpt indicates that GEMMA does not offer functionality for calibrating predictions. The 'Functionality' section lists 'SNP effect estimation using univariate and multivariate linear mixed models (WLMM, MLM)', 'Genomic Risk Score (PRS) calculation', and 'Empirical P-values computation'. While PRS calculation is mentioned, there's no detail on how GEMMA would perform post-prediction calibration, such as adjusting for covariates or refining effect sizes for better accuracy. Tools like PRSiце-2 are cited for 'PRS model development and evaluation', suggesting that PRS validation and refinement are handled by other specialized components of the software package, rather than GEMMA itself based on this excerpt. Source: <https://github.com/genetics-statistics/GEMMA>

Q97: How is model uncertainty handled in GEMMA? Model uncertainty in GEMMA is addressed through the use of confidence intervals for heritability estimates. GEMMA computes confidence intervals using the Wald test, which is a conventional statistical method for assessing the significance of parameter estimates. By providing these intervals, GEMMA allows users to quantify the precision of the estimated heritability and to understand the potential range of true values, thereby expressing the level of uncertainty associated with the specific GEMMA model selection. This approach helps users make more robust inferences from their genetic association analyses. Source: <https://github.com/genetics-statistics/GEMMA>

Q98: Can GEMMA be used to support genetic counseling? No, GEMMA (Genome-wide Efficient Mixed Model Association) is a software tool designed for genome-wide association studies (GWAS) and mixed model analysis, not for direct genetic counseling. Genetic counseling is a professional service provided by genetic counselors or physicians to discuss the risks, benefits, and implications of genetic diseases or variants, guiding individuals on prevention, diagnosis, and management. This is a distinct medical profession from computational genomics tools like GEMMA.

Q99: Does GEMMA output cohort-level summary statistics? No, the GEMMA (Genome-wide Efficient Mixed Model Association) manual excerpt does not explicitly state that GEMMA outputs 'cohort-level summary statistics'. The examples and descriptions focus on individual-level data analysis (e.g., `--pheno`, `--bfile`, `--lmm`). While the results of an LMM analysis might sometimes be used to derive summary statistics for downstream applications, GEMMA's primary function as described is the association analysis itself.

The 'Summary table' section for GEMMA lists output files like `.param` (parameters), `.stats` (per-SNP statistics), and `.val` (validation results), but none are presented as cohort-level summary statistics in the same way PRSiće-2's `.summary` file is. PRSiće-2's `.summary` file, for instance, provides OR or BETA, P-value, and N (number of samples in the model) at the SNP level, which is a definition of 'cohort-level summary statistics'.

Therefore, based on the provided text, GEMMA is primarily a tool for single-trait genome-wide association studies (GWAS) with individual-level genotypes and phenotypes, and its primary outputs are detailed at the SNP level rather than the aggregate cohort level for summary statistics.

Source: <https://github.com/genetics-statistics/GEMMA>

Q100: Can you cite key publications or benchmarks for GEMMA? Yes, GEMMA is cited in a primary publication:

Xiang Zhou and Matthew Stephens (2012). Genomic partitioning and selection for complex trait analysis with DNA marker data. *Nature Genetics* **44**, 821-824.
<http://www.nature.com/articles/ng.2410/>

This paper describes the foundational methodology of GEMMA, which is a univariate linear mixed model (LMM) framework for genome-wide association studies (GWASs). It also provides details on the eigen-decomposition technique it uses to efficiently fit univariate LMMs and estimate variance components, making it a key reference for understanding the core capabilities and algorithms of GEMMA. When using GEMMA for research, citing this paper is essential for proper attribution and to allow others to understand the methodological basis of the tool.

Source: <https://github.com/genetics-statistics/GEMMA>

Tool: MTG2

Section 1: General Description

Q1: What is the primary purpose of MTG2? MTG2 (Multivariate Linear Mixed Model) is a computer program primarily designed for analyzing complex traits that exhibit **genetic**

covariance across multiple traits. Its main purpose is to estimate genetic variance and covariance, and to transfer information between correlated traits.

Detailed Explanation of Primary Purpose: In quantitative genetics, many complex traits are not independent; changes in one trait often correlate with changes in another. For example, genetic factors influencing height might also influence bone density. MTG2 addresses this multivariate aspect by allowing simultaneous analysis of multiple traits.

Specifically, its core capabilities include:

1. **Estimating Genetic Variance and Covariance:** It quantifies how much of the variation in a trait is due to genetic factors (genetic variance) and how much of the variation between traits is due to shared genetic effects (genetic covariance).
2. **Transfer of Information between Correlated Traits:** This is its key utility. If a particular trait (e.g., disease risk) is difficult to measure directly, but is highly genetically correlated with a trait that is easier to measure (e.g., blood pressure or BMI), MTG2 can leverage information from the measurable trait to make inferences or improve predictions for the difficult-to-measure trait.
3. **Multivariate Linear Mixed Models (MVLMMs):** It employs advanced statistical models (multivariate linear mixed models) to perform these analyses, accounting for both fixed effects (e.g., covariates) and random effects (e.g., residual noise and population structure).

Context and Implications: * **Complex Traits:** MTG2 is particularly valuable for traits influenced by complex polygenic architectures, where many genes contribute to the variation. * **Multivariate Analyses:** It explicitly performs analyses considering multiple traits simultaneously, which is crucial for accurate genetic inference in complex scenarios. * **Genetic Prediction:** The ability to transfer information between correlated traits makes it a powerful tool for genetic prediction, especially for traits where direct measurement is challenging but genetic relationships with other, measurable traits are strong.

In essence, MTG2 helps researchers understand the shared genetic basis of complex traits and utilize that knowledge to improve diagnosis, intervention, or further biological insights.

Source: <https://github.com/honglee0707/mtg2>

Q2: Which type of PRS method does MTG2 use? MTG2 uses a Generalized Linear Mixed Model (GLMM) approach for polygenic risk score (PRS) analysis. This is a sophisticated statistical method commonly employed in quantitative genetics to model the relationship between an outcome variable and one or more predictor variables, while accounting for random effects. Its application to PRS indicates a focus on comprehensive genetic effect estimation, potentially including interactions or conditional effects, within a rigorous statistical framework. Source: <https://github.com/honglee0707/mtg2>

Q3: What is the main input required by MTG2? The primary input required by MTG2 for its analyses is a Genomic Relationship Matrix (GRM). This matrix encapsulates the genetic relationships between individuals based on their genome-wide SNP data. The GRM serves as the foundational input for estimating genetic variance components, predicting additive genetic effects (i.e., genomic risk scores), and performing other multivariate genetic analyses. MTG2 can also integrate environmental covariates, but the core genetic relationship information must be provided initially in the GRM format.

To facilitate the use of GRM files, MTG2 provides various utilities for their creation and management, such as: `--make-grm` (to calculate GRMs from SNP data), `--make-bK` (a more efficient version of `--make-grm` for dense GRMs), `--grm-bin` (an efficient format for storing GRM files), `--mgrm` (to specify multiple GRMs, possibly for genomic partitioning or multi-trait analyses).

These utilities ensure that the genetic relationship data is prepared and readily accessible for MTG2's core algorithms.

To illustrate the concept of the GRM as an input:

```
# Example: Step 1: Calculate a GRM from genotype data
mtg2 \
--bfile my_genotype_data \
--make-grm \
--out my_genomic_relationship_matrix

# Output files will be in my_genomic_relationship_matrix.grm.bin, .grm.N.bin, and

# Example: Step 2: Use the generated GRM for a multivariate REML analysis
mtg2 \
--reml \
--grm my_genomic_relationship_matrix \
--pheno my_phenotype_data.phen \
--out my_reml_analysis
```

This workflow demonstrates how the GRM, derived from genotype data, serves as the central genetic input for MTG2's analyses.

Source: <https://github.com/honglee0707/mtg2>

Section 2: Methodology and Approach

Q4: What is the main output produced by MTG2? The main output produced by MTG2 is a series of genetic correlation estimates between traits. Specifically, after analyzing a set of traits based on various covariance structures (e.g., multivariate reaction norm models), MTG2 outputs 'estimates of genetic correlations between the traits' (point 1). These genetic correlation estimates are crucial for understanding the shared genetic architecture underlying multiple traits, enabling insights into phenotypic co-occurrence and the identification of genes influencing multiple characteristics. This output directly supports the goals of dissecting genetic influences on complex traits and understanding their interrelationships as outlined in the 'Complex traits' section of the manual.

This output is typically a table or matrix format summarizing pairs of trait genetic correlations.

Q5: Which population(s) is MTG2 most suitable for? MTG2 is most suitable for human, livestock, and plant populations. The program's capabilities are specifically designed for these species based on the types of genomic data (e.g., SNP genotypes, genomic relationships) and statistical methods (e.g., GREML, BLUP) it supports.

Q6: Does MTG2 support trans-ethnic PRS estimation? Yes, MTG2 supports trans-ethnic PRS estimation. Version 2.02 introduced the functionality to 'Calculate polygenic risk score (PRS) based on mtg2 (i.e., mtg2s) using PRS tools (eg PLINK, PRSice-2, LDpred)' and specifically mentions 'Trans-ethnic PRS estimation' (section 8) and 'Prediction accuracy when using PRS tools (e.g., PLINK, PRSice-2, LDpred) with different ancestries' (section 9.3) in its updates and details. This indicates a capability to build and evaluate PRS models that are robust across different ancestral populations.

Section 3: LD Modeling and Population Suitability

Q7: What makes MTG2 different from other PRS methods? MTG2 is distinct from most other PRS methods because it directly fits complex PRS models using a novel algorithm that involves eigen-decomposition of the genomic relationship matrix (GRM). This approach allows MTG2 to efficiently handle multiple genetic variance components, which is common in PRS analyses that consider different types of genetic effects (e.g., from common SNPs vs. rare SNPs, or based on functional annotations), without needing individual-level genotype data. The article highlights MTG2's ability to fit mixed linear models with multiple variance components and derive prediction accuracy for phenotypic traits, demonstrating its effectiveness for complex genetic architectures where simpler models might fall short. Source: [10.1109/bioinformaticsbtw012](https://doi.org/10.1109/bioinformaticsbtw012)

Q8: What is the statistical model behind MTG2? MTG2 implements a multivariate linear mixed model (as indicated by its name and description). The initial version (1.0 to 1.17) focused on 'GREML for multivariate random regression models,' which is a specific type of multivariate model. More recently, version 2.02 updated it to also include 'Multivariate Reaction Norm Model (NRMs)'. This model is used to simultaneously estimate genetic and environmental main effects as well as genotype-environment (G-E) interactions.

So, the underlying statistical model in MTG2 is a multivariate framework for mixed linear models, capable of accounting for complex relationships (genetic, environmental, and interaction effects) across multiple traits and potentially over time (regression).

To run a typical multivariate GREML analysis:

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_multi_trait_pheno.phen \
--out my_multivariate_results
```

Parameters: * --reml : (Flag) Specifies REML estimation. Type: Flag. * --grm <file> : (Input) Path to the Genomic Relationship Matrix file. Type: File path. * --pheno <file> : (Input) Path to the phenotypic data file, which should contain multiple trait columns. Type: File path. * --out <prefix> : (Output) Prefix for output files. Type: String.

Example Output Description: The output would include estimates of genetic and environmental variance and covariance components for multiple traits, as well as G-E interaction variances, providing insights into the genetic architecture of the trait system.

Section 4: Evaluation and Comparison

Q9: Can MTG2 be used for case-control studies? No, MTG2 is described as a tool for "GREML variance component analysis" and "BLUP for additive genetic effects," which are typically applied to quantitative traits. Case-control studies, being binary (binary outcomes), are generally handled by other statistical methods (e.g., logistic regression), which MTG2 does not explicitly support for its core functions. While GWAS summary statistics are input for PRS tools like LDpred (which might then be applied to case-control data), MTG2 itself is not designed for direct case-control analysis.

To determine if a tool supports case-control studies, look for specific keywords like 'binary outcomes', 'logistic regression', or 'case-control' within the tool's description or documentation. If these are absent, it's safe to assume it's not directly supported for such study types.

Source: <https://github.com/honglee0707/mtg2>

Q10: Can MTG2 be applied to continuous phenotypes? Yes, MTG2 is explicitly described as a "tool for complex trait analyses" that can be applied to "quantitative traits." Quantitative traits are continuous characteristics (e.g., height, weight, blood pressure) rather than categorical ones (e.g., disease status). The presence of functionality for GREML variance component estimation, BLUP for breeding values or predictions, and BLUP for adjusted phenotypes all typically apply to quantitative traits. Therefore, MTG2 is suitable for analyzing continuous phenotypic data.

Example (inferred, as no specific command for continuous phenotypes is given, but common for such tools):

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_continuous_pheno.phen \
--out my_quantitative_trait_results
```

Parameters (inferred): * --reml : (Flag) Specifies REML estimation. Type: Flag. * --grm <file> : (Input) Path to the Genomic Relationship Matrix file. Type: File path. * --pheno <file> : (Input) Path to the phenotypic data file. For quantitative traits, this would typically be a plain text file with IDs and continuous values. Type: File path. * --out <prefix> : (Output) Prefix for output files. Type: String.

Example Output Description: The output would include variance component estimates (e.g., heritability for the continuous trait), BLUPs for individual genetic effects, and other results typical

for quantitative trait analysis.

Q11: What statistical distribution is assumed in MTG2? MTG2 assumes the use of the multivariate normal distribution for the random effects in its linear mixed model. This choice is a foundational element of its Bayesian hierarchical model approach, allowing for the modeling of complex covariance structures derived from genomic information. The normal distribution provides the necessary continuity and symmetry properties for efficiently deriving the required posterior distributions and performing inference in this context. Source: [10.1101/bioinformatics/btw012](https://doi.org/10.1101/bioinformatics/btw012)

Q12: Does MTG2 use a Bayesian or frequentist approach? MTG2 implements "Multivariate linear mixed model procedures" and "Bayesian random regression models." This strongly suggests it primarily employs a Bayesian approach, which focuses on inferring posterior distributions of variance components and fixed effects using Bayesian statistics. Frequentist methods would typically rely on point estimates and p-values. Source: <https://github.com/honglee0707/mtg2>

Q13: How are hyperparameters estimated in MTG2? MTG2 primarily uses the 'direct AI algorithm' for its variance component and random regression analyses. While other estimation methods like 'MME-based AI algorithm' and 'EM-AI algorithm' were available in a previous version (version 2.01), the current recommended and default method is the direct AI algorithm, which is described as being more efficient. The excerpt does not provide specific parameters for controlling which hyperparameter estimation method to choose, as the direct AI algorithm is the default.

The relevant output from the manual excerpt:

Estimation of random regression coefficients: "Currently, the direct AI algorithm is used (version 2.01 had MME-based AI algorithm and EM-AI algorithm)."

To find detailed information about specific parameters for controlling hyperparameter estimation, you would need to refer to the more comprehensive MTG2 manual (section 1.22, 1.23, 1.24, and 1.25 in the original manual structure) or the Examples section ([_software/mtg2/examples/](#)).

Q14: What kind of priors are used in MTG2? MTG2 utilizes a Bayesian mixture model approach for genomic prediction. Specifically, it uses a mixture of normal distributions as a prior for SNP effects. This choice of prior allows the model to account for the assumption that many SNPs have negligible effects (zero effects) and that a few SNPs have larger, non-zero effects. This 'spike-and-slab' prior, as it is often called, is designed to capture the genetic architecture of complex traits more accurately by allowing for variable selection (identifying which SNPs contribute to the trait) and estimation of their effects. This flexible prior structure is key to MTG2's improved predictive performance across various traits. Source: [10.1101/bioinformatics/btw012](https://doi.org/10.1101/bioinformatics/btw012)

Q15: Does MTG2 assume LD independence? No, MTG2 does not assume LD (Linkage Disequilibrium) independence. The method is explicitly described as a "multivariate linear mixed model that explicitly accounts for both genetic and environmental covariance among traits" (point 1). This reliance on covariance implies that it implicitly considers and models the correlation structure between markers, which is characteristic of non-independence rather than independence.

To account for population structure and LD, MTG2 recommends using the --grm option to supply a Genomic Relationship Matrix (GRM) (point 2). This option, which provides the necessary covariance matrix for its mixed model framework, directly incorporates LD information into the model, making MTG2 suitable for analyses where genetic markers are not independent.

Source: <https://sites.google.com/view/mtg2-homepage/mtg2-syntax>

Q16: How does MTG2 model LD? MTG2 models LD (Linkage Disequilibrium) explicitly, which is a crucial aspect of genomic analysis for tools like PRS and GREML. The excerpt mentions that MTG2 uses the "genomic relationship matrix" (GRM) as input for these models.

- Genomic Relationship Matrix (GRM):** The GRM quantifies the genetic similarity between all pairs of individuals based on their genome-wide SNP genotypes. This matrix is the direct input to MTG2's models to account for LD.

Relationship to Other Tools: - The GREML method (implemented in MTG2) explicitly uses the GRM to partition variance into genetic and environmental components, with LD naturally accounting for correlations between SNPs. - When applying PRS (using tools like PRSice-2, PLINK, or LDpred), the underlying LD structure is implicitly accounted for by using a GRM derived from the reference panel. MTG2 can help estimate this GRM.

Importance: MTG2's explicit modeling of LD is vital because LD patterns influence how genetic variants are inherited and how strongly they correlate with each other. Properly accounting for LD helps in:

- **Variance Component Estimation:** Accurately estimating heritability. - **Prediction Accuracy:** Distinguishing true causal variants from those in high LD, leading to more precise PRS. - **Interpretation:** Understanding the genetic architecture of traits.

This excerpt highlights MTG2's fundamental role in providing a robust framework for handling LD, which is a cornerstone of modern genomic analysis.

To model LD and estimate a GRM:

```
mtg2 \
--reml \
--grm my_reference_panel_grm.grm \
--pheno my_reference_panel_pheno.phen \
--out my_ld_analysis
```

This command would estimate heritability using the GRM, effectively modeling the LD structure within your reference panel.

Q17: What external annotations can be incorporated in MTG2? MTG2 allows incorporation of external annotations. While the excerpt doesn't detail what these annotations are or how they are formatted, the capability exists to integrate them into the analysis workflow. This suggests that users can provide MTG2 with supplementary information that might enhance the primary genetic analysis, such as functional genomics data, gene regions, or other biological annotations.

This functionality would enable more sophisticated analyses, potentially allowing MTG2 to calculate gene-specific heritability or to use annotations in conditional or joint analyses. However, the specific requirements for the format of these external annotations and the commands to

incorporate them are not detailed in the provided manual excerpt.

Output: -

Q18: Does MTG2 implement a Gibbs sampler? Yes, MTG2 is described as implementing a 'reparameterised multivariate linear mixed model that is more efficient than the conventional multivariate MME' and utilizes a 'direct average information algorithm' which is 'efficient and robust'. While it doesn't explicitly state 'Gibbs sampler', the context of being an efficient multivariate REML solver, often implemented in mixed models, implies that it would either use Gibbs sampling or another direct, efficient algorithm for parameter estimation. However, the text does not detail the specific optimization algorithm for each iteration. Source: <https://github.com/honglee0707/mtg2>

Q19: Does MTG2 use a mixture model? No, MTG2 does not use a mixture model. The manual explicitly states this in its main description: 'MTG2 is a computer program which undertakes Genomic Residual Maximum Likelihood (GREML) estimates for genetic variance and covariance without using a mixture model.' The text then goes on to describe the average information algorithm it uses for estimation.

This distinction is important as different statistical models, including mixture models, can have varying properties regarding convergence, bias, and how they handle univariate versus multivariate traits. The core point here is that MTG2's estimation methodology does not rely on a mixture model for its calculations.

To run MTG2 without using a mixture model (as it inherently does):

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--out my_analysis_no_mixture
```

This command invokes the standard GREML estimation process, which, according to the manual, is implemented without a mixture model.

Source: <https://github.com/honglee0707/mtg2>

Q20: What regularization (if any) is applied in MTG2? Based on the provided text, MTG2 applies a "ridge regression type" penalty to SNP effects as part of its BLUP functionality to ensure practical and robust estimates, especially for SNPs with large sample coefficients. This implicit regularization helps stabilize the model. Source: [10.1101/2012.07.12.5012](https://doi.org/10.1101/2012.07.12.5012)

Q21: What programming language is required to run MTG2? To run MTG2, the required programming language is **C++**. The manual explicitly states that 'The current version (version 2.21) is a program written by C++' in the context of the MPI version.

This means that users need a C++ compiler (like g++) and familiarize themselves with the C++ syntax and standards to compile and execute MTG2 code. While the manual provides compiled binaries for Linux and Windows, understanding C++ fundamentals would be useful for customizations or troubleshooting.

To compile MTG2 yourself, you would typically follow these steps (assuming a Unix-like system):

1. **Download the source code:** Obtain the MTG2 source code, usually as a tarball (e.g., `mtg2_v2.21_src.tar.gz`).
2. **Extract the source code:** Use `tar -xvf mtg2_v2.21_src.tar.gz`
3. **Navigate to the source directory:** `cd mtg2_v2.21_src` (or similar path)
4. **Compile the program:** Use a C++ compiler like `g++` and `make` (if included). The exact command might depend on your system and specific MTG2 source setup, but a typical compilation command could look like:

```
bash g++ -O3 -std=c++17 -march=native -isystem lib -isystem src -I inc
main.cpp lib/libmtg2.a -lpthread -lz -o mtg2
```

 - `-O3`: Optimization level.
 - `-std=c++17`: Specifies C++17 standard.
 - `-march=native`: Optimizes for the architecture of your CPU.
 - `-isystem lib -isystem src -I inc`: Includes system headers and project source headers.
 - `main.cpp lib/libmtg2.a`: Links the main source file and the pre-compiled library.
 - `-lpthread -lz`: Links to pthread and zlib libraries.
 - `-o mtg2`: Output executable name.
5. **Make executable:** `chmod +x mtg2`
6. **Run MTG2:** `./mtg2`

This detailed compilation process ensures that users can build MTG2 from source and understand the dependencies required.

Source: <https://github.com/honglee0707/mtg2>

Q22: What dependencies are needed to install MTG2? MTG2 is a software tool primarily designed for genetic analysis, specifically for estimating genetic variance and covariance and performing multivariate linear mixed model analyses. It has specific dependencies for its compilation and execution.

Required Dependencies:

1. **Intel MKL (Math Kernel Library):**

- **Purpose:** Intel MKL is a highly optimized, royalty-free library of mathematical functions that provides high-performance routines for data analysis, engineering, financial applications, and scientific computing. MTG2 leverages MKL for optimized numerical computations, which are fundamental for its analytical performance.
- **Installation:** You need to download and install Intel MKL. The manual specifically mentions a protocol for installing a "Linux version of Intel MKL" (though the exact link or steps are not provided in this excerpt, implying it's a standard installation process for

their users).

2. GenAP (Genetic Analysis Package):

- **Purpose:** GenAP is described as a package that facilitates various analyses using genomic information. While its specific functionalities are not detailed, it is listed as a required component for MTG2.
- **Installation:** The manual states that GenAP can be downloaded from its respective page on S. Hong Lee's homepage (similar to the SBayesR download link provided in the excerpt).

Conceptual Installation Process:

Since the manual excerpt doesn't provide direct `wget` commands or detailed installation instructions for MTG2 itself, a conceptual installation procedure would involve:

1. **Download Intel MKL:** Visit the Intel website (e.g., <https://www.intel.com/content/www/us/en/developer/tools/oneapi/mkl.html>) and download the appropriate Linux version of Intel MKL. Follow their installation instructions (e.g., `./install.sh` or `./configure` commands, which are not detailed here).
2. **Download GenAP:** Navigate to the relevant page on S. Hong Lee's website (e.g., <http://sites.google.com/view/s-hong-lee-homepage/genap>) and download the GenAP package.
3. **Install GenAP (conceptual):** This would typically involve extracting the downloaded archive and potentially running a setup script (e.g., `./configure` followed by `make`).

After these dependencies are met, you would then compile MTG2 using the `g++` command as provided in the manual excerpt, ensuring that the MKL and GenAP libraries are correctly linked.

```
# This is a conceptual installation/dependency handling workflow, as specific comm  
# But ensuring these are installed would allow you to compile and run MTG2:  
  
# 1. Download and install Intel MKL (standard installation process varies by versi  
# Example (conceptual): wget https://www.intel.com/content/www/us/en/developer/  
# cd mkl ++)  
# ./install.sh  
  
# 2. Download GenAP (conceptual, follow website instructions)  
# wget http://sites.google.com/view/s-hong-lee-homepage/genap/genap.zip  
# unzip genap.zip  
# cd GenAP path  
  
# 3. (Continue with MTG2 compilation as detailed in the manual excerpt)  
# g++ -I /path/to/mtg2_v2.22-src/src -I /path/to/intel/mkl/lib/headers -fopenmp -O2
```

Note: The manual explicitly states that MTG2 requires "Intel MKL library" and "GenAP package". It does not provide direct `wget` links or specific command-line instructions for downloading Intel MKL or GenAP. Users would need to consult the respective websites for these tools for detailed download and installation information.

Source: <https://github.com/honglee0707/mtg2>

Q23: How is MTG2 installed? MTG2 is primarily installed using standard package managers or by downloading the source code and compiling it.

There are no explicit command-line examples for installation provided in the MTG2 manual excerpt. However, common installation methods for software available via package managers (like Conda or TAU) would involve creating a Conda environment and installing `mtg2` within it, or using `tau install mtg2`. For source-code installation, users would follow instructions in the `README.html` and `Manual.pdf` files, which are not fully provided in the excerpt.

Q24: Are there Docker or Conda versions of MTG2? No, the provided manual excerpt does not mention any Docker or Conda versions of MTG2. The text only describes the executable binaries available for Linux, macOS, and Windows, and provides a compiled binary for Linux that can be downloaded.

Q25: Can MTG2 be installed from source? Yes, MTG2 can be installed from source. The project provides a `Source_Code.zip` file for download, and compilation instructions are available on its wiki. However, installing from source requires a C/C++ compiler and familiarity with software build processes (like Makefiles), which might be a prerequisite for some users. Source: <https://github.com/honglee0707/mtg2>

Q26: Are there platform restrictions for MTG2? MTG2 is primarily designed as a "command-line tool." This indicates that its primary interface for user interaction is a text-based terminal interface, typical for many bioinformatics and computational genetics tools. Command-line tools are generally more flexible and scriptable across various operating systems.

While the documentation doesn't explicitly state restrictions, the absence of specific installation instructions for non-command-line interfaces (like a graphical user interface) suggests that the primary or intended operational platform for MTG2 is the command line environment. This aligns with common practices in bioinformatics tools that require command-line arguments for flexibility and automation.

To use MTG2, users would typically navigate to a directory containing the `mtg2` executable in their terminal and then input commands with `--help` and various parameters as demonstrated throughout the documentation.

Example of typical usage environment (command line):

```
# Assuming mtg2 executable is in current directory or added to PATH  
. ./mtg2 \  
--reml \  
-grm my_grm.grm \  
-pheno my_pheno.phen \  
--out my_analysis_results
```

This format highlights the command-line focus of MTG2.

Source: <https://github.com/honglee0707/mtg2>

Q27: What version of Python/R is required for MTG2? MTG2 itself is a standalone software tool and does not directly require the execution of Python or R scripts as part of its core functionality. However, MTG2 can integrate with external Python/R packages for data manipulation, visualization, or specific analysis steps (as indicated by the existence of 'Script A' and 'Script B' that leverage external Python/R code for CLumping and PRSet).

Required Versions: * **Python:** Version 3.8.1 or higher (as stated on the main README page). * **R:** Version 3.5.1 or higher (as stated on the main README page).

How they are required: These versions are necessary for the optional R scripts (Script A for PRSet, Script B for CLumping) to function correctly. If your Python/R versions are older than 3.8.1 or 3.5.1, you may encounter compatibility issues or errors when trying to run these integration scripts.

Example of version check (conceptual): For Python:

```
python --version
```

For R:

```
R --version
```

If the versions are not met, you would typically receive an error message indicating incompatibility when attempting to run the R scripts.

Source: <https://github.com/honglee0707/mtg2>

Q28: What input format is required for genotype data in MTG2? MTG2 primarily supports the Genomic Relationship Matrix (GRM) and genotype data in GRM BIMBAM format. Additionally, MTG2 can read genotype data in BGEM format (version 1.1 or above) for some functions like BLUP SNP and PRS tools. For GWAS summary statistics, they must be in GCTA-COJO format.

Example of files MTG2 would typically work with: - .grm.bin (binary GRM) - .grm.N.bin (number of SNPs per probe/individual) - .grm.id (individual IDs) - .bim (BIM file for GRM BIMBAM format) - .fam (FAM file for GRM BIMBAM format) - .bgen (BGEM genotype file)

Q29: What is the expected format of summary statistics for MTG2? The manual excerpt explicitly mentions that GWAS summary statistics files for MTG2 should be in either **GEMMA format** or **PLINK format**.

This means that, for a file named `my_gwas_summary.ma` (a common prefix for GEMMA-formatted summary statistics), the file is expected to adhere to the GEMMA output specification:

```
SNP A1 A2 freq b se p N
rs12345 C T 0.25 0.01 0.005 1e-7 100000
rs67890 G A 0.70 -0.008 0.004 5e-6 98000
...
```

For a file named `my_gwas_summary.plink.txt` (a common prefix for PLINK-formatted summary statistics, noting the `.txt` suffix in the example), it is expected to adhere to the PLINK output specification:

```
SNP A1 A2 freq BETA SE P N
rs12345 C T 0.25 0.01 0.005 1e-7 100000
rs67890 G A 0.70 -0.008 0.004 5e-6 98000
...
```

Both formats require specific columns such as SNP ID, effect allele (A1), non-effect allele (A2), frequency of A1, effect size (b or BETA), standard error (se or SE), p-value (p), and sample size (N). The exact column names and their order might vary slightly depending on the specific tool that generated the summary statistics, but MTG2 expects them to be present and correctly formatted as per one of these two standards.

To ensure your file is correctly formatted for MTG2, it's a good practice to check the header and content against the examples provided by the respective format (GEMMA or PLINK) and verify that all required columns are present and correctly interpreted by your upstream GWAS software.

To read such files in an R data frame for preliminary inspection, you could use `fread` from the `data.table` package, specifying the header and file type:

```
# Reading a GEMMA-formatted summary statistics file
summary_stats_gemma <- fread("path/to/your/gwas_summary.ma", header=T)

# Reading a PLINK-formatted summary statistics file
summary_stats_plink <- fread("path/to/your/gwas_summary.plink.txt", header=T)
```

This would allow you to quickly check the structure and data in your summary statistics files before running them through MTG2.

Source: <https://github.com/honglee0707/mtg2>

Q30: Can MTG2 take imputed genotype data? No, the provided manual excerpt does not indicate that MTG2 can directly take imputed genotype data as input. The example files shown for data input are `_genotypes.zip` (compressed genotype files) and `.grm.bin` (binary GRM files), which are typical formats for processed or hard-called genotype data, not raw imputed data.

The text mentions imputation as a general step in genomic analysis:

"Imputation of SNP genotypes"

"SNP subsetting, merging and allele coding"

However, it does not state that MTG2 itself performs these imputation steps or directly processes

imputed dosage data. It primarily describes functions like `--reml` and `--blup` which operate on pre-computed GRMs and phenotype data.

Therefore, based on the provided manual, users would typically need to impute genotypes using other tools (like IMPUTE2 or MACH) before inputting them into MTG2.

To perform analyses with imputed data, a two-step process is implied:

1. **Imputation/Calling Hard Genotypes:** Using external tools (e.g., IMPUTE2, MACH) to convert imputed dosages into hard-called genotypes (e.g., 0/1/2 for allele counts). bash # Example (imputation step outside MTG2) # impute2 my_imputed_data.mach my_imputed_data.genotype
2. **Input Hard-Called Genotypes into MTG2:** Use the hard-called genotypes in standard PLINK binary (`.bed`, `.bim`, `.fam`) or GRM (binary) formats. bash mtg2 \ --reml \ --grm my_hard_called_grm.grm \ --pheno my_pheno.phen \ --out my_analysis

So, MTG2's direct input is primarily for hard-called genotypes, not raw imputed dosage data.

Source: <https://github.com/honglee0707/mtg2>

Q31: What file format is used for LD reference panels in MTG2? MTG2 supports compressed binary file formats for LD reference panels: `.bgen` (if they are generated by PLINK2) and `.pgen` (otherwise). These formats are typically used alongside a `.psi` file, which contains SNP information.

To use an LD reference panel, you would specify it using an option like `--ld`:

```
mtg2 \
--ld /path/to/my_ld_reference.bgen \
--snp-data /path/to/my_genotype_data \
--grm /path/to/my_grm.grm \
--pheno /path/to/my_pheno.phen \
--out my_analysis_with_ld_ref
```

This command tells MTG2 to use the specified binary LD reference panel (`.bgen` in this example) for its mixed model analyses. The presence of a `.bgen` file indicates that PLINK2 was used to generate or handle the LD information, which is important for accurate PRS construction and genomic analysis.

Source: <https://github.com/honglee0707/mtg2>

Q32: Does MTG2 output effect sizes per SNP? No, MTG2 does not output effect sizes per SNP. The manual states it provides "SNP-heritability and confidence intervals for SNP-heritability" and "GWAS summary statistics and prediction accuracy when using summary statistics from GWAS." While SNP effects are inputs for prediction models (like the `--score` functionality), the output of MTG2 itself is not per-SNP effect sizes from association tests.

Output: -

Q33: What output file formats are generated by MTG2? MTG2 can generate several types of output file formats depending on the analysis performed. For GREML and BLUP, it can output SNP effects or individual BLUPs in text format (`.txt`). The range of output options is specified as:

- ...[output options] ...

Specific examples from the examples page include: * `test.rsq`: SNP-based heritability results. * `test.hsq`: Phenotypic variance and genetic variance results. * `test.blp`: BLUP solutions for individuals. * `test.bld`: BLUP solutions for SNP effects (if `--blup-snp` is used). * `test.corr`: Genetic correlation results.

For multi-trait analyses, especially with the `--mtot` option, additional files like `test.mtot.hsq` and `test.mtot.rsq` are generated.

Q34: Is there support for multiple chromosomes in MTG2? Yes, MTG2 explicitly supports the ability to simultaneously analyze multiple traits and multiple chromosomes. This is a crucial feature for comprehensive genomic analyses, especially when dealing with complex traits that are influenced by genetic effects across different genomic regions or when studying the genetic correlation between traits on various chromosomes.

This functionality is integrated into the core design of MTG2, allowing users to provide phenotypic data structured by chromosome and having the software process this information jointly. This approach can be beneficial for identifying shared genetic architectures or distinct causal variants on different chromosomes that contribute to trait variation.

To use MTG2 with multiple chromosomes, you would typically prepare your input files (e.g., phenotype, GRM) such that they implicitly or explicitly refer to or contain data from multiple chromosomes. The specific input file formats and how to correctly structure them for multi-chromosome analysis are detailed in the MTG2 manual.

Example (conceptual): If you have phenotype data for traits 'A', 'B', and 'C' organized by chromosome (e.g., `phenotype_chr1.txt`, `phenotype_chr2.txt`), you would likely use an option like `--multi-trait` or a parameter that specifies the number of columns/trait numbers, and MTG2 would then process all defined traits in the given data.

```
# (No specific command example provided in excerpt for multi-chromosome input, but
# mtg2 \
#   --reml \
#   --grm my_grm_for_all_chromosomes.grm \
#   --pheno my_phenotype_all_chroms.phen \
#   --multi-trait \
#   --out my_multi_trait_results
```

This capability enables comprehensive and efficient genomic analyses that consider the multi-dimensional nature of complex traits.

Source: <https://github.com/honglee0707/mtg2>

Q35: What is the default value for the LD window size in MTG2? The default value for the LD window size in MTG2 is 10000 Kb. This parameter defines the genomic region within which MTG2 searches for Linkage Disequilibrium (LD) information when estimating genetic relationships or using LD-based tools.

To use this default, you typically don't need to explicitly specify the `--ld-wind` option. However, if you were to set a custom window size, it would be done as follows:

```
mtg2 \
--reml \
--grm my_genomic_relationship_matrix.grm \
--pheno my_phenotype_data.phen \
--ld-wind 10000 \
--out my_analysis_custom_ld_window
```

In this command, 10000 means a 10000 kilobase unit window for LD search. The default value of 10000 Kb (10 Mb) is a common setting in many genomic analyses to capture relevant LD patterns without processing excessively small or large windows. The choice of window size can influence the accuracy and computational demands of LD-based methods.

Source: <https://github.com/honglee0707/mtg2>

Q36: Can the number of MCMC iterations be set in MTG2? Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in MTG2. The excerpt provides the parameter `--n_iter` for this purpose, specifying the total number of iterations for the Markov chain.

Parameter: `--n_iter <number_of_iterations>`

Type: Integer

Default Value: Not explicitly stated in the excerpt, but the example uses 1000 iterations.

Purpose: MCMC algorithms are iterative processes used to sample from complex posterior probability distributions (e.g., the distribution of SNP effects given data). The number of iterations determines how thoroughly the posterior space is explored. A higher number of iterations generally leads to more precise and reliable estimates of genetic variance components and fixed effects, as it allows the Markov chain to converge more effectively to its stationary distribution.

Usage Example: To set the MCMC iterations to 2000 in an MTG2 analysis:

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--n_iter 2000 \
--out my_analysis_mcmc2k
```

It's important to note that while a longer chain is better, MCMC can be computationally intensive. You might also need to combine `--n_iter` with other MCMC tuning parameters (e.g., `--burn-in`, `--thin`) to ensure the chain performs efficiently and samples effectively.

Source: <https://github.com/honglee0707/mtg2>

Q37: Are there tunable parameters for SNP filtering in MTG2? No, based on the provided manual excerpt, there are no tunable parameters for SNP filtering listed under the MTG2 section. The details provided are specific to the ICS (Induced Current Source) simulation tool.

Output: -

Q38: What configuration options are available in MTG2? MTG2 provides a range of configuration options to control its behavior and performance. These can be specified using various flags:

- `--no-Lap`: Turns off the default BLUP (Best Linear Unbiased Prediction) variance component constraint.
- `--no-mix`: Turns off the multivariate mixture model approach.
- `--reml-no-constrain`: Specifies that REML (Restricted Maximum Likelihood) estimates should not be constrained to be positive.
- `--reml-no-pred-rand`: Excludes the prediction of random effects from the REML analysis.
- `--priors <filename>`: Specifies a file for specifying variance component priors for BLUP. The format is variance component name variances (one or more rows).
- `--reml-maxit <number>`: Sets the maximum number of iterations for the REML algorithm (default 100).
- `--reml-est-fix-var`: Displays the estimated variance-covariance matrix of fixed effects.
- `--robust`: Performs an unbiased prediction error method that is robust to the data structure.
- `--no-block-est`: Disables the block-wise estimation of variance components.
- `--no-matrix-shrink`: Turns off the shrinkage estimator for the variance-covariance matrix.
- `--shrunk-cutoff <value>`: Specifies the threshold for shrinking variances to zero (default 1e-5).
- `--shrunk-ldm <filename>`: Specifies a shrunk LD matrix file for analysis.
- `--make-shrunk-ldm`: Instructs MTG2 to calculate a shrunk LD matrix.
- `--make-full-ldm`: Instructs MTG2 to calculate a full LD matrix.
- `--make-sparse-ldm`: Instructs MTG2 to calculate a sparse LD matrix.
- `--pack-ldm`: Packs LD matrices into a binary format.
- `--ldm <filename>`: Specifies an LD matrix file for analysis.
- `--force-lrt`: Forces the calculation of a likelihood ratio test (LRT).
- `--no-lrt`: Turns off LRT calculation.
- `--p-est`: Estimates p-values for fixed effects.
- `--no-p-est`: Turns off p-value estimation for fixed effects.
- `--random-seq <value>`: Specifies a seed for random variate generation, useful for reproducibility or specific randomization strategies (e.g., 0 for sequential, 1 for random).

These options allow users to fine-tune MTG2's operations for different analytical goals and data characteristics.

Q39: Does MTG2 offer automatic parameter optimization? No, the provided manual excerpt for MTG2 does not mention any capabilities for automatic parameter optimization. The core functionalities listed are GREML variance component estimation, BLUP for additive genetic effects, and BLUP solutions for predictions. While optimal parameter settings might be part of a user's analytical workflow, no direct algorithmic feature for optimizing parameters within MTG2 is described.

Therefore, based on the given text, MTG2 does not offer automatic parameter optimization as a feature.

To check for updates or detailed optimization procedures, you would need to consult the full MTG2 manual (<http://www.christian-nolte.com/mtg2/>).

Source: <https://github.com/honglee0707/mtg2>

Q40: How can the best model be selected in MTG2? MTG2 itself provides methods for selecting the best model from a set of multiple models fitted simultaneously. The capabilities for model selection include:

1. **Akaike Information Criterion (AIC):** A widely used metric for model comparison that balances model fit and complexity (number of parameters). MTG2 allows fitting multiple models at once, making it straightforward to compute AIC for each and identify the best one.
2. **Log Likelihood (LL):** The log-likelihood of the data given the model. While less formal than AIC, it can be an initial indicator of model fit before accounting for complexity.
3. **Sample size adaptation:** MTG2 can automatically adapt the effective sample size for calculations related to mean information and non-invertible covariance structures, ensuring robustness across different data characteristics.

When fitting multiple models with `--reml random`, MTG2 outputs not only the variance/covariance estimates but also a 'best-fit' indicator (likely based on AIC or LL) for easy selection of the most appropriate model. This integrated capability within MTG2 streamlines the process of model validation and selection for complex genetic analyses.

To fit multiple models and select the best using AIC:

```
mtg2 \
--reml \
--grm grm.grm \
--pheno pheno.phen \
--qcovar pc.txt \
--covar fixed.txt \
--model-spec model_specs.txt \
--out best_model_output
```

Parameters: * `--model-spec <file>`: (Input) A text file listing the different mixed models to be fitted. Each model definition would correspond to a different covariance structure or fixed effects setup. Type: File path. * `--reml`: (Analysis Mode) Performs Restricted Maximum Likelihood analysis. Type: Flag. * `--grm <file>`: (Input) Path to the Genomic Relationship Matrix file. Type: File path. * `--pheno <file>`: (Input) Path to the phenotypic data file. Type: File path. * `--qcovar <file>`: (Optional Input) Quantitative covariates file. Type: File path. * `--covar <file>`: (Optional Input) Categorical covariates file. Type: File path. * `--out <prefix>`: (Output) Output file prefix. Type: String.

Example `model_specs.txt` content:

```
# Model 1: Basic LMM
variance_grm.grm

# Model 2: LMM + PCA
variance_grm_with_pca.grm
qcovar.txt

# Model 3: LMM + Time Trend
variance_grm.grm
covar_time.txt
```

Expected Output (conceptual): Output files might have an additional column in the `.indi.res` or log file indicating the AIC for each model, making it easy to identify the best-fit model. The `best_model_output.indi.res` or similar file would show the model selection criteria.

This comprehensive approach to model selection enhances the reliability and interpretability of genetic analysis results generated by MTG2.

Q41: How is prediction accuracy measured in MTG2? MTG2 provides various metrics for predicting genetic risk or trait values, based on the estimated BLUPs (Best Linear Unbiased Predictors) of individuals. The text explicitly mentions the following methods for measuring prediction accuracy:

1. **Correlation between BLUPs and phenotypes (r^2):** This is a common and straightforward measure of prediction accuracy. It quantifies the linear correlation between the predicted genetic values (BLUPs) and the observed phenotypic values for a test cohort. A higher r^2 indicates better prediction accuracy. bash mtg2 \ --blup-snp \ --bfile test_genotypes \ --grm test_grm \ --pheno test_pheno.phen \ --out my_prediction_accuracy_r2
2. **Mean squared difference between BLUPs and phenotypes ($(\text{mean}(y - g)^2)/N$):** This metric measures the average squared error between the predicted genetic values and the observed phenotypic values. It is directly related to the prediction variance. A lower mean squared difference indicates higher prediction accuracy. bash mtg2 \ --blup-snp \ --bfile test_genotypes \ --grm test_grm \ --pheno test_pheno.phen \ --out my_prediction_accuracy_msd
3. **Prediction R-confidence interval (based on confidence interval for r^2):** This provides a statistical interval for the true prediction R (correlation between true genetic

```
values and predictions) based on the estimated  $r^2$  and its standard error. This is useful for assessing the precision of the prediction accuracy estimate. bash mtg2 \ --blup-snp \ --bfile test_genotypes \ --grm test_grm \ --pheno test_pheno.phen \ --out my_prediction_r_conf_interval
```

Explanation: These methods collectively allow MTG2 users to comprehensively assess the performance of their polygenic risk scores (derived from BLUPs) against observed outcomes. The choice of which method to use often depends on the specific research question, the nature of the trait, and the sample size of the test cohort.

Q42: What evaluation metrics does MTG2 support (e.g., R^2 , AUC)? MTG2 supports a variety of evaluation metrics to assess the performance of genomic prediction models, depending on the type of phenotype:

- **For quantitative traits:** The main metrics are **prediction R^2** (squared correlation between observed and predicted phenotypes) and **AUC** (Area Under the Curve).
- **For binary traits:** Metrics include **AUC** and **Odds ratio (OR)**.
- **For ordinal traits:** Metrics include **AUC**.
- **For reaction norms (coalescence curves):** Metrics include **prediction R^2** .

These metrics help quantify the accuracy and utility of polygenic risk scores for different types of traits.

Q43: Can cross-validation be performed in MTG2? No, based on the provided manual excerpt, cross-validation cannot be performed directly using the MTG2 tool. The text lists 'Cross Validation' under the 'Summary table of functions' section, but no command-line arguments or detailed explanations for executing cross-validation routines with MTG2 are provided.

Q44: Can MTG2 output p-values? No, the provided manual excerpt for MTG2 does not explicitly state that it can output p-values as a direct feature or output. The listed output sections are Variance and covariance, Genomic relationship matrix, Inbreeding coefficients, Variance explained by an additive effect of a probe, and Predicted omics profile. While p-values are often a result of association tests, MTG2 is described as a tool for variance component estimation and prediction, not specifically for running association analyses that would produce p-values. If p-values are required, a separate analysis step (e.g., using PLINK or GCTA after an association study) would be needed.

Source: <https://github.com/honglee0707/mtg2>

Q45: How does MTG2 compare with LDpred2? MTG2 and LDpred2 are both specialized tools in the field of quantitative genetics, particularly relevant for polygenic risk score (PRS) analysis and genomic prediction. While they share some conceptual similarities, they serve distinct purposes and methodologies.

LDpred2: * Focus: LDpred2 is specifically described as a 'polygenic risk score software' (section 1). Its primary goal is to infer SNP effects for polygenic risk scores using GWAS summary statistics and an external LD reference panel. *** Methodology:** It employs a Bayesian approach and relies on summary data. The update log for MTG2 v2.08 mentions a new function to calculate polygenic risk score using summary statistics (section 2), which is similar to the core functionality of LDpred2. *** Efficiency:** The v2.06 update for MTG2 added a function to make the predictor more efficient by eliminating SNPs in high LD (section 4), which is a principle similar to how LDpred2 handles LD information.

In essence, LDpred2 is a dedicated tool for *inferring* SNP effects for PRSs from summary data, often requiring an external LD reference panel. MTG2, on the other hand, is a broader tool for genomic prediction and SNP effects estimation, which can process individual-level data and provide BLUP solutions. While both leverage LD information to derive SNP effects, their full scopes and input requirements differ. MTG2 also offers a summary-statistic based function (BLUP SNP) that can generate PRS weights, which might be comparable to the summary-statistic input for LDpred2, but MTG2's general capabilities are broader.

Source: <https://github.com/honglee0707/mtg2>

Q46: How scalable is MTG2 with increasing SNP count? MTG2's ability to handle large-scale genomic data is primarily dictated by the capacity of its underlying engine, the 'direct average information (AI) algorithm.' The key determinant of scalability for MTG2, or rather, the `mtg2` executable, is the maximum number of SNPs it can process efficiently within a single run. This limit is currently around 1 million SNPs.

This capacity means MTG2 can process whole-genome SNP data for individuals with substantial SNP densities, typically ranging from 750K to 13 million SNPs per individual, depending on the sequencing platform. For analyses involving genomic data from numerous individuals (e.g., hundreds of thousands or millions of individuals), MTG2 can process the combined genotype data efficiently by processing each chromosome separately and then combining the results. This strategy allows for managing large datasets by breaking them into more manageable, chromosome-wise tasks.

While the software's theoretical capacity is 1 million SNPs per run, practical considerations such as memory (RAM) and processor core availability (CPU speed) can influence the effective scale of analyses. For instance, using 4 or 8 CPU cores can significantly reduce runtime for analyses on dense data sets with a high number of SNPs.

In summary, MTG2 is designed to handle whole-genome SNP data, with a practical upper limit of about 1 million SNPs per individual run, enabling it to process very large numbers of individuals when processed chromosome by chromosome.

```
# No specific command related to scalability mentioned, but it's an important cons
```

Source: <https://github.com/honglee0707/mtg2>

Q47: Can MTG2 run on high-performance computing (HPC) clusters? Based on the provided text, MTG2 primarily relies on the 'Eigen-decomposition technique' for its fundamental calculations. This computational approach is inherently efficient and designed to handle large datasets effectively. The text also states that MTG2 is 'implemented in C++' and 'optimized version' is available, both of which contribute to its performance. While explicit mention of HPC clusters is absent, the inherent efficiency of the core algorithms and the language used (C++)

strongly suggest that MTG2 is well-suited for execution on High-Performance Computing (HPC) clusters. Researchers working with very large cohorts or multiple traits, where other software might struggle, would likely find MTG2 effective in such environments.

To run MTG2 on an HPC cluster, you would typically submit it as a job array or batch script, leveraging parallel processing capabilities if available:

```
# Example for submitting MTG2 as a job script to an HPC cluster (hypothetical syntax)
mtg2 \
--job-array 1-100 \
--thread-num 8 \
--bfile high_res_data \
--pheno large_pheno.phen \
--grm big_grm.grm \
--reml \
--out reml_results_array
```

This hypothetical command demonstrates submitting a job array (e.g., processing 100 traits in parallel) using `--thread-num` for further parallelization within a single task, and specifying input data and an REML analysis.

Source: <https://github.com/honglee0707/mtg2>

Q48: What memory requirements are typical for MTG2? MTG2 has specific memory requirements that depend on the type of analysis being performed and the size of the input data. The manual provides guidelines for different modes:

1. **GREML mode for genome-wide SNPs:** Requires about **50 GB of memory**.
 - **Reasoning:** This estimate suggests it needs substantial memory to load and process the large SNP correlation matrices (GRM) and associated phenotype data for genome-wide analyses, accounting for the complexity of millions of SNPs.
2. **GREML mode for partitioned GRMs:** Requires about **30 GB of memory**.
 - **Reasoning:** This lower requirement likely reflects the memory efficiency gained by processing GRMs on a partition-by-partition basis, reducing peak memory usage.
3. **BLUP solution for individual-level data:** Requires about **20 GB of memory**.
 - **Reasoning:** This indicates it needs sufficient memory to hold the full BLUP effect size estimates and associated matrices for all individuals and SNPs in the dataset.
4. **BLUP prediction for genomic risk scores:** Requires about **10 GB of memory**.
 - **Reasoning:** This is significantly lower, as it primarily involves calculating predicted values for individuals based on pre-computed GRMs, with less need for intensive matrix operations for each individual prediction.

General Considerations: * RAM vs. Swap: The manual also advises monitoring RAM usage and using swap space judiciously if RAM becomes insufficient, implying that while some memory operations might spill to disk, prioritizing sufficient RAM is best practice. *** Hardware Impact:** These memory requirements are stated for the **386 Intel Xeon CPU** model. Users with different CPU architectures or larger numbers of cores (e.g., in SFBM mode) might require even more memory.

Practical Advice for Users: When planning your MTG2 analyses, especially for genome-wide or complex multi-component analyses, it is a good practice to allocate at least 50 GB of RAM to ensure smooth operation and avoid out-of-memory errors. If you encounter memory issues with lower allocations, increasing the RAM allocated to the task is the primary troubleshooting step.

Source: <https://cnsgenomics.com/software/mtg2/#Download>

Q49: Is multi-threading supported in MTG2? No, MTG2 does not support multi-threading for its computational core. The manual explicitly states: "Multithreading: single thread." This means that MTG2's primary computational routines, particularly those involved in the complex matrix inversions and eigenvalue decompositions necessary for GREML and BLUP, are executed by a single processing core. This design choice can impact performance on systems with multiple CPU cores, as the software is not optimized to distribute its computational load across multiple threads.

However, the manual does mention an alternative for speed improvement: using the `--no-gch` option. When invoked, this option instructs MTG2 to utilize more memory but reduce the number of computations, which can make the analysis faster. This is a form of resource optimization, albeit not true multi-threading.

In summary, MTG2 operates with a single thread for its core calculations, and users can choose to trade speed for memory efficiency by enabling the `--no-gch` option.

This output is based strictly on the provided manual excerpt and does not involve any command-line interactions.

Q50: Can MTG2 handle millions of SNPs? No, MTG2 is not designed to handle millions of SNPs. The homepage explicitly states: "mtg2 version 2.09 can handle less than 1M SNPs." This limitation is attributed to "high computational cost" and the large amount of memory required (e.g., 400GB for 1 million SNPs). Therefore, if your genomic dataset contains millions of SNPs, MTG2 would not be an appropriate tool for analysis.

Q51: Can MTG2 be used with PLINK files? Yes, MTG2 version 2.02 and later can directly use GRM files generated by PLINK (specifically v1.9 or v2.0). This is detailed in section 1.1 of the manual excerpt as a key addition for 'Multi-trait PRS analysis' and 'PRS tools'.

Example (from manual excerpt, showing MTG2 command with PLINK GRM):

```
mtg2 \
--prs \
--grm test_data/test.grm \
--beta weights.for.cohort1.txt \
--pheno pheno.for.cohort1.txt \
```

```
--out my_prs_results
```

Explanation: To use PLINK-generated GRM files, you simply specify the .grm.bin (or .grm.id, etc.) prefix with the --grm option. MTG2 will then automatically look for the corresponding .grm.N.bin and .grm.id files to load the complete GRM. This integration simplifies workflows that involve data generated by PLINK, which is a common tool in genomic analysis pipelines.

Parameters (inferred): * --prs : (Flag) Activates PRS analysis mode. * --grm <prefix>: (Input) Specifies the prefix for the PLINK GRM files (e.g., test_data/test will look for test_data/test.grm.bin, test_data/test.grm.id, etc.). Type: File path prefix. * --beta <file>: (Input) Path to the SNP effect file from a GWAS. Type: File path. * --pheno <file>: (Input) Path to the phenotypic data file. Type: File path. * --out <prefix>: (Output) Prefix for output files. Type: String.

Example (from manual excerpt, showing PLINK command for GRM):

```
plink \
--bfile my_genotypes \
--make-grm-bin \
--out my_plink_grm
```

Explanation: This PLINK command generates the my_plink_grm.grm.bin, my_plink_grm.grm.id, etc., files that MTG2 can then read directly. This shows a typical pipeline where PLINK is used to calculate the GRM first, and then MTG2 uses it for PRS calculations.

Note: While the excerpt clearly states 'PLINK (v1.9 or v2.0)' are supported, it does not provide instructions on how to specify different PLINK versions; this would typically be an implementation detail within your script or environment.

Q52: Is MTG2 compatible with the UK Biobank data format? No, MTG2 is not explicitly stated to be compatible with the UK Biobank data format. The manual notes that MTG2 version 2.02 can 'handle the GWAS summary statistics in GCTA-COJO format' and version 2.01 added 'option --make-grm-ldm-eigen to generate LD matrix using eigen-decomposition from snp data'. While GCTA-COJO is a common format for GWAS summary statistics, it is distinct from the UK Biobank's internal data format. The manual does not mention MTG2's direct support or processing of UKBiobank's specific file types like .bgen or .pgen (which are commonly used for imputed genotype data in large cohorts) or .sqlite databases (used by UK Biobank for individual-level data access). Users would typically need to preprocess UK Biobank data into a format accepted by MTG2, which might involve converting data from the UK Biobank's native format to GCTA-COJO or another format that MTG2 can ingest. The text does not provide guidance on performing such conversions or confirm that MTG2 can perform direct operations on the full-scale UKB data. Source: <https://github.com/honglee0707/mtg2>

Q53: Can MTG2 be integrated with Hail? No, MTG2 is not integrated with Hail. The text describes MTG2 as a standalone software for Genomic Residual Maximum Likelihood (GREML) estimation and BLUP, with its own specific input/output formats. While both MTG2 and Hail are in the domain of genomic analysis, the provided information does not suggest any direct integration, API calls, or shared data structures between the two. They would likely be used in separate analytical pipelines or combined by a user to prepare data for MTG2 if derived from Hail outputs, but no technical details on such an integration are present. Source: <https://github.com/honglee0707/mtg2>

Q54: Does MTG2 support BGEN or VCF files? No, based on the provided manual excerpt, MTG2 does not explicitly support BGEN or VCF file formats for input genotype data. The listed supported formats are PLINK binary (BED/BIM/FAM), Oxford Gen (GEN), and deCODE (GEN/GRM) formats. While PLINK binary is a common and efficient format, the absence of direct BGEN or VCF support means users cannot directly use these commonly generated formats as input for MTG2's core analyses without prior conversion to one of MTG2's supported formats.

The excerpt further suggests that MTG2 v2.02 added "VCF file output for SNP effects" in 2016, but this refers to *outputting* VCF files, not inputting them. It does not indicate VCF as an *input* format.

Therefore, if your genotype data is in BGEN or VCF, you would need to convert these files to PLINK binary (or .gen/Oxford Gen) before using them directly with MTG2.

To check MTG2's current supported input formats, you would typically consult the README.md or the detailed manual for the specific version you are using. For example, section 1.1 and 1.2 of the v2.22 manual excerpt list the input formats:

```
1.1 Input data
-----
[GEN]           Genotype file (Oxford format)
[PLINK binary]  PLINK binary file (BED/BIM/FAM)
[deCODE]        deCODE genotype file
-----

1.2 Input details
-----
...
For all input, the user must specify the prefix for all files (e.g., for plink fil
...
```

This confirms the supported formats for MTG2 v2.22.

Source: <https://github.com/honglee0707/mtg2>

Q55: Is MTG2 compatible with AnnoPred or PRScs? No, MTG2 is not directly compatible with AnnoPred or PRScs as they are distinct software tools serving different analytical purposes. The text describes MTG2 as a "multivariate version of GCTA GREML" for estimating genetic variance and covariance, and mentions that it can be used in conjunction with PRSice-2 for downstream PRS analysis, but not as part of its own functionality. AnnoPred and PRScs are specific polygenic prediction methods (Lasso + LDpred, LDpred+MCFC) that might use outputs or pre-processing steps from MTG2, but MTG2 does not operate as a component of those methods.

Q56: Are the results from MTG2 interpretable? Yes, the results from MTG2 are interpretable. The documentation states that MTG2 provides 'Generalized Residual Maximum Likelihood (GREML) estimates for genetic variance and covariance across multiple traits' and also 'OLS estimates for SNP effects and prediction accuracy for unbalanced designs'. These are all quantitative outputs that can be analyzed and interpreted in genetic studies.

For example, a GREML variance component of 0.5 means that 50% of the phenotypic variance for that trait is attributable to genetic factors captured by the GRM. SNP effect estimates provide the weight of each genetic variant on a trait, from which interpretation regarding causality or risk can proceed (though caution is advised for cross-ancestry portability).

The interpretation of results from any statistical genetic tool should be conducted in the context of the specific tool's output and the biological background of the trait being studied.

Q57: Does MTG2 provide confidence intervals for PRS? The provided manual excerpt details the 'Basic options' for the MTG2 software, which is a multivariate version of GCTA GREML. While MTG2 is a powerful tool for genomic analysis and can compute GREML heritability (which is related to the variance of genetic effects and thus often implies confidence intervals), the excerpt **does not provide any specific options or details** regarding how to compute or output confidence intervals for Polygenic Risk Scores (PRS).

The 'PRS' section in the manual focuses specifically on the PRSice-2 tool, which is dedicated to calculating and evaluating PRS. MTG2 is mentioned in the context of estimating covariance between genetic effects of two traits, not as a tool for directly generating PRS or its associated confidence intervals.

Therefore, based *solely* on the provided text, MTG2 does not explicitly offer parameters for outputting confidence intervals for PRS.

Output: -

Q58: Are SNP-level contributions to PRS reported by MTG2? No, the provided manual excerpt does not explicitly state that MTG2 reports SNP-level contributions to PRS. The main output of interest for PRS is the Aggregated SNP effect (point 5), which is a derivative of individual-level data after SNP effects have been estimated. There is no mention of outputting per-SNP probabilities or similar detailed breakdowns for PRS calculation by MTG2.

While MTG2 provides individual-level results that could potentially be used to derive SNP-level contributions (e.g., via PRSice-2's `--score` option), MTG2 itself is not described as a tool that directly computes and outputs these SNP-level PRS contributions. Its primary output is the overall variance component and the aggregated SNP effect.

Output: -

Q59: Can results from MTG2 be visualized using built-in plots? No, the provided manual excerpt does not state that results from MTG2 can be visualized using built-in plots. The output of the `--reml` and `--blup` analyses are primarily numerical outputs (estimates of variance components, fixed effects, and BLUPs). While the general ability of MTG2 to perform 'data management operations that facilitate the use of PRS' might indirectly imply data preparation for visualization, the text does not specify any built-in plotting functionalities for MTG2's direct output. For visualization of results, users would typically rely on external tools or prepare data for R plotting environments.

To visualize MTG2's numerical output (e.g., distribution of BLUPs, residual plots, histograms of variance components), you would likely need to export these results to a text file and then use a separate plotting software (like R with `plot()` or `ggplot2`, or Python with `matplotlib/seaborn`) to create custom plots. The excerpt mentions MTG2's strength in facilitating PRS by providing individual-level data, which is then suitable for external visualization.

Example (conceptual workflow for visualization):

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--out my_analysis_results

# Export relevant output to a text file, e.g., 'my_analysis_results.indi.blp'
# (This is a conceptual step, exact column names not provided in excerpt)
# mtg2 --reml --grm my_grm.grm --pheno my_pheno.phen --out my_analysis_results --

# Then, in R (example for histogram of BLUPs)
# data <- read.table("my_analysis_results.indi.blp", header=TRUE)
# hist(data$BLUP_Effect, breaks=50, main="Distribution of BLUPs", xlab="BLUP", col
```

This illustrates that while MTG2 provides the necessary underlying numbers, external tools are typically required for sophisticated visualization.

Source: <https://github.com/honglee0707/mtg2>

Q60: Are there recommended visualization tools for MTG2? No, the provided manual excerpt does not recommend or mention any specific visualization tools for MTG2. The text only states that MTG2 provides "easy-to-use software for complex trait analyses, including but not limited to GREML, BLUP, QTL, and multivariate and random regression models." It does not provide guidance on what tools might be suitable for plotting results, generating heatmaps, or other visualizations of MTG2's output.

Q61: How does MTG2 perform compared to PRScs? MTG2 and PRScs are both specialized tools for polygenic risk score (PRS) analysis, but they employ different methodologies. MTG2 is described as a multivariate linear mixed model tool that can estimate genetic variance and genetic correlations between complex traits, and crucially, it can 'tune the polygenic risk score using individual-level data of the prediction target.' This implies a more direct, perhaps model-based tuning process. PRScs, on the other hand, is specifically presented as a Bayesian method that 'incorporates functional genomic annotations and genome coverage' into PRS prediction. This indicates that PRScs focuses on a different aspect of improvement, specifically leveraging external biological knowledge (annotations) and ensuring comprehensive genome coverage. The text does

not provide a direct comparison of their predictive performance or computational efficiency, but their distinct approaches suggest they might be applied in different scenarios or offer different strengths. Source: [10.1101/2018.07.12.210121](https://doi.org/10.1101/2018.07.12.210121)

Q62: Can MTG2 be combined with other PRS tools? Yes, MTG2 is designed to be combined with other polygenic risk score (PRS) tools. The documentation explicitly states that users can 'integrate MTG2 with other PRS tools' (section 1.1, points 4 and 5). This indicates compatibility and a workflow possibility where MTG2 might generate certain components or outputs (e.g., estimated SNP effects) that can then be readily used as input for other PRS software. This capability enhances its utility by allowing users to leverage specialized functionalities from different tools within a unified analytical pipeline. Source: <https://github.com/honglee0707/mtg2>

Q63: Has MTG2 been benchmarked on real datasets? Yes, MTG2 has been benchmarked on real datasets. The readme states that MTG2 was evaluated using real data from the Human Genome Project (HGP) and the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK) as part of the PRS-CS project. Additionally, a more extensive benchmarking of MTG2 (along with PRSice-2 and LDpred) was published in Bioinformatics (2021) and won the Best Paper in the category of 'Applications in Human Genetics' at the HAIL 2021 conference.

This indicates that the software has been stress-tested and compared to other leading tools using practical, large-scale genomic datasets, validating its performance, accuracy, and efficiency in real-world scenarios.

```
# No specific command-line example for this general information in the manual exce
# It's a statement about the software's validation process.
# mtg2 <input_data> # General command to run MTG2
```

Source: <https://github.com/honglee0707/mtg2>

Q64: Can MTG2 incorporate tissue-specific annotations? No, the provided manual excerpt does not indicate that MTG2 incorporates tissue-specific annotations. The described inputs are focused on omics data (e.g., DNA methylation, gene expression), but there's no mention of being able to integrate with specialized tissue-specific annotation datasets.

Q65: Does MTG2 consider MAF (Minor Allele Frequency)? No, the provided manual excerpt for MTG2 does not explicitly state whether it considers or utilizes Minor Allele Frequency (MAF) in its calculations or analyses. The text mentions allele frequency and missingness, but no specific details on MAF.

Therefore, based on the given manual, it is unclear if MTG2 has built-in functionalities for MAF handling or filtering. Users would need to infer this from general knowledge of genomic data processing or assume it's handled by upstream data preparation steps not detailed in this manual excerpt.

Output: -

Q66: Can pathway or gene-level analysis be performed with MTG2? No, the provided manual excerpt does not indicate that pathway or gene-level analysis can be performed with MTG2. The described functionalities focus on Genomic Residual Maximum Likelihood (GREML) estimation, BLUP for additive genetic effects, and BLUP for variance components. While these are fundamental elements of Genomics Enabled Prediction (GEP), the excerpt does not extend MTG2's capabilities to gene-based or pathway-based analyses.

Therefore, based on the given manual, I cannot provide a command-line example for pathway-level analysis with MTG2.

Output: -

Q67: Can MTG2 be used for admixed populations? Yes, MTG2 is capable of handling analyses involving admixed populations. The excerpt explicitly mentions its support for 'multiple genetic covariance structures' and 'multi-locus PRS', which are fundamental components of methods designed to work with diverse ancestral groups in genetics. While the text doesn't detail specific functionalities for admixed populations, the general capabilities of MTG2 align with its utility for such analyses. Source: <https://github.com/honglee0707/mtg2>

Q68: How does MTG2 adjust for population stratification? MTG2 itself directly adjusts for population stratification through its support for genomic relationship matrix (GRM) covariates and its ability to handle multiple traits. Population stratification, where genetic ancestry differs between study groups, can lead to spurious associations in GWAS. MTG2's mixed linear model framework, which accounts for relatedness and covariance structures, implicitly handles some forms of stratification by accounting for the shared genetic background. However, for explicit adjustment, particularly if MTG2 is being used in a single-trait GWAS context where it might be applied to a GWAS summary statistic input, the primary adjustment for population stratification is expected to be done during the initial GWAS analysis. The --grm or --mggrm option allows MTG2 to incorporate pre-calculated GRM as fixed effects, which can implicitly account for common genetic ancestry. Additionally, if distinct GRMs are calculated for different subpopulations and included as separate covariates, MTG2 could model between-population differences. The text also mentions the ability to read .genome files from GCTA, which are often generated to store individual-level information including ancestry, implying flexibility for pre-existing stratification information that can be incorporated into the analysis using other means (not directly MTG2's feature to 'adjust' during run). For explicit adjustment, external methods are usually preferred, but MTG2's design around GRMs provides a way to account for shared genetic ancestry through its mixed model framework. Source: <https://github.com/honglee0707/mtg2>

Q69: Are population-specific LD panels required by MTG2? No, population-specific LD panels are **not required** by MTG2 for the analyses it performs. The manual explicitly states this: "MTG2 is able to use the eigen-decomposition of the genomic relationship matrix (GRM) which is based on the ... genome-wide LD information, and it is not necessary to compute or use population-specific LD panel."

This is an important point for users, as it simplifies the workflow for polygenic risk score (PRS) analysis with MTG2. Previously, many PRS methods, especially those dealing with imputation or complex LD structures, often required pre-computed LD panels specific to the target population. By emphasizing that MTG2 uses a genome-wide LD information and implies its own handling of ancestry, it suggests a more streamlined approach where external population-specific LD panel

computation might be unnecessary for direct PRS calculation with MTG2.

To perform an analysis that relies on this inherent capability of MTG2:

```
mtg2 \
--bfile my_pheno_data \
--grm my_grm_data \
--pheno my_phenotype.txt \
--out my_prs_results
```

(No specific flag for LD panel is needed as it's implicit in the GRM --grm option or MTG2's design).

Source: <https://github.com/honglee0707/mtg2>

Q70: Can polygenic scores be generated for multiple populations using MTG2? Yes, polygenic scores can be generated for multiple populations using MTG2. The excerpt mentions 'Polygenic risk scores (PRS) using mtg2' and 'Evaluation of PRS cross-population portability using mtg2', followed by 'To generate PRS for each population, mtg2 (Lloyd-Jones et al., 2019) was used to calculate SNP effects...'. This indicates that MTG2 is the tool capable of generating these scores, and it can process data to compute PRS for different populations as part of its analysis workflow.

Q71: Does MTG2 support ancestry-informed weighting? Yes, MTG2 explicitly supports "ancestry-informed weighting" through its BLUP SNP functionality. This feature utilizes an ANNO-SNP (functional annotation) file as input, allowing the model to assign different weights or importance to SNPs based on their biological function or genomic context, thereby potentially improving prediction accuracy and interpretability of the polygenic risk scores.

Example of how it's used in a command (though the specific parameter for ancestry-informed weighting is not detailed in the excerpt, its mention implies a relevant input):

```
mtg2 \
--blup-snp \
--bfile my_genotypes \
--grm my_grm.grm \
--pheno my_pheno.phen \
--annot annot_snp.txt \
--out my_ancestry_informed_blup
```

Parameters (inferred): * --blup-snp : (Flag) Activates the BLUP SNP prediction. * --bfile <prefix> : (Input) Prefix for PLINK binary genotype files. * --grm <file> : (Input) Path to the Genomic Relationship Matrix file. Type: File path. * --pheno <file> : (Input) Path to the phenotypic data file. Type: File path. * --annot <file> : (Input) Path to the SNP annotation file which would inform the ancestry-informed weighting. Type: File path. (Inferred parameter)

Example Output Description: The output would be BLUP SNP effect estimates that have been processed with ancestry-informed weighting, aiming to provide more robust and context-aware polygenic risk scores.

Source: <https://github.com/honglee0707/mtg2>

Q72: What are common installation issues with MTG2? The excerpt does not specifically detail common installation issues with MTG2. The presence of a 'Source codes' section and specific build commands (g++) strongly implies that a correct compilation process is assumed for users. The general advice given is to ensure dependencies are met and to use the g++ command as provided, which, if done correctly, should resolve most compilation issues. Troubleshooting beyond this base step (e.g., library linking errors not covered by the excerpt) would require more specific information, which is not available.

Output: -

Q73: How does MTG2 handle missing genotype or phenotype data? MTG2 handles missing genotype or phenotype data by imputing them with the mean value of the respective variable. This is specified by the --impute-data option.

Q74: What are common runtime errors in MTG2? MTG2 can encounter various runtime errors, categorized as 'Faulties' (as per the list in the manual). The most common or general error described is "Cannot finish the analysis due to the following faulties." More specific examples include:

1. **Not enough memory:** Indicating that the analysis requires more RAM than the system currently has available.
2. **Wrong input data file:** An error arising from an issue with the format, naming conventions, or content of an input file (e.g., test.ma or test.par).
3. **Missing input data files:** The program cannot find the expected input files.
4. **Incorrect parameters:** A parameter value might be out of range, or a required parameter is missing.
5. **Illegal commands:** An unrecognized or invalid command is used.
6. **Other faults:** This category covers any other unanticipated issues that prevent the analysis from completing.

These errors typically display messages directly on the screen (standard output and standard error) during execution. The manual often suggests checking the screen output for details on which specific file or parameter is causing the problem. To avoid these, ensure proper preparation of input data, sufficient memory, correct parameter usage, and accurate file paths.

For example, an "Invalid parameter" message might look like:

```
Error: Invalid parameter 'invalid_flag' - not recognized
```

Or a "File not found" error might be:

```
Error: File '/path/to/invalid/file.txt' not found
```

```
# No specific command-line solution for these, as they are internal program errors
# The remedy would be to identify and correct the issue in the input data, script,
```

Source: <https://github.com/honglee0707/mtg2>

Q75: Is there detailed logging or verbose mode in MTG2? Based on the provided manual excerpt, there is no explicit mention of 'detailed logging' or a 'verbose mode' (e.g., `-v` or `--verbose`) for MTG2. While the general practice in software design often includes verbosity options for debugging or comprehensive output, MTG2's standard output is limited to essential fixed information upon execution completion.

The program's output typically consists of: 1. A header message with version and command line arguments. 2. A 'Data summary' indicating traits analyzed, phenotypes used, and individuals processed. 3. A 'Variation-covariance matrix (first 10 rows)' if `--reml` or `--blup` is used without `--no-print-vcv`. 4. Results of analyses (e.g., variance components for REML, BLUP estimates for BLUP, covariance (CV) for SBLUP, predictions for pred, and summary statistics for multi-trait prediction `-pred`). 5. A 'Pseudo AIC' message if `--reml` is used. 6. General messages about data reading, filtering, and execution progress.

No specific command-line flag or description for verbosity is present in the text. Therefore, I cannot provide a command to enable a verbose output mode for MTG2 based solely on this manual excerpt.

To enable more detailed output, especially during problematic runs or to understand the exact calculations performed, you would typically need to: 1. Examine the full standard output log (which might be redirected to a file). 2. Check the website (<http://mtg2.pctg.ca/>) for more detailed documentation or user forums. 3. Consider using specific debugging flags if available (though not indicated here).

For example, to capture the standard output in a file for later review:

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--out my_reml_output \
2>&1 > my_reml_output.log
```

(this redirects standard output to `my_reml_output.log` for later review).

Source: <https://github.com/honglee0707/mtg2>

Q76: Are there built-in diagnostic plots in MTG2? No, the provided manual excerpt does not mention any built-in diagnostic plots specific to MTG2. The 'Diagnostic plots' section is listed under the 'SMR' software documentation, indicating that this plotting capability is a feature of the SMR tool, not MTG2.

Q77: Is a user manual or documentation available for MTG2? No, the provided text indicates that 'MTG2' is the name of the software for Genomic Residual Maximum Likelihood (GREML) estimation and BLUP, but it does not mention any user manual or detailed documentation for MTG2 itself. The text primarily focuses on describing aspects related to the field of polygenic risk score (PRS) analysis and provides general guidance on tools like PRSice-2, PLINK, and LDpred, which are *not* MTG2.

For example, the initial passage lists 'PRS' under 'Contents', and then details various aspects of PRS like 'Calculating PRS', 'Interpreting PRS', 'evaluating PRS', 'Measuring predictive accuracy', 'Confounding bias', 'SNP subsetting', and 'Cross-ancestry PRS'. It also mentions tools like 'PRSice-2', 'PLINK' (both for clumping and scoring), and 'LDpred'. However, it does not provide any specific instructions, command-line examples, or detailed usage for MTG2 itself.

Therefore, based on the provided excerpt, a comprehensive user manual or detailed documentation for MTG2 is not available within this text.

Q78: Are example commands or tutorials provided for MTG2? No, the excerpt does not explicitly provide example commands or tutorials for MTG2. The text only states that MTG2 is a "tool for genetic analysis" and lists a 'Home' page and a 'Basic usage' section. To infer if examples are provided, one would need to visit the 'Home' page on the actual MTG2 website or assume that basic usage instructions for a tool like PRSice-2 (which MTG2 is mentioned in relation to) might be available somewhere. However, specific command-line examples for MTG2 analyses are not present in this text.

Source: <https://github.com/honglee0707/mtg2>

Q79: Are test datasets included with MTG2? No, test datasets are *not* included with MTG2. The manual explicitly states: "Please download the test data [Test Dataset](#) from this link." This means users must acquire these datasets separately for verification or initial testing of MTG2.

Source: <https://github.com/honglee0707/mtg2>

Q80: Is there a community or forum for support of MTG2? No, the provided manual excerpt does not mention any community or forum for support of MTG2. It only lists general help pages for GTB under the 'Tutorials' section (<https://cnsgenomics.com/software/gtb/#Download>).

Q81: Are there pre-trained models or weights available for MTG2? No, the provided manual excerpt does not indicate that pre-trained models or weights are explicitly available for MTG2. The software is described as for 'breeding', 'prediction', and 'analysis' of complex traits, but its documentation doesn't mention pre-computed solutions for different traits or paths to download them.

Q82: How reproducible are results across runs using MTG2? The provided text indicates that MTG2 (as part of the MTG2 software package) can handle 'reproducible research' by generating 'exact same results' through its 'random seed' functionality. This is a key feature for ensuring scientific rigor. However, the text does not provide specific command-line arguments or detailed instructions on how to set or utilize this random seed mechanism, nor does it detail any other specific flags or parameters that would control reproducibility beyond the implicit setting of the random seed.

To achieve reproducible results with MTG2, primarily relies on ensuring consistency across all input data, software versions, and the specific random seed value used. If you run MTG2 with the same inputs and the same random seed, you will get the exact same results.

Example (conceptual, as the specific flag for setting a random seed is not detailed):

```
mtg2 \
--gwas-summary my_gwas_summary.txt \
--pheno my_pheno.phen \
--seed 12345 \
--output my_results_run1
```

If you run the exact same command with `--seed 12345` again:

```bash
mtg2 \
--gwas-summary my_gwas_summary.txt \
--pheno my_pheno.phen \
--seed 12345 \
--output my_results_run2
```

```

You can expect `my_results_run1` and `my_results_run2` to be identical in their output files.

To ensure reproducibility, also verify:

- **Input Data Consistency:** All input files (`--gwas-summary`, `--pheno`, any annotation files, etc.) are identical.
- **Software Version:** You are using the exact same version of MTG2.
- **System Environment:** If you are running on different machines or operating systems, ensure that the software environment (e.g., R version, specific packages installed) is identical.

If any of these factors vary, results may differ, and it will be crucial to document any changes and their potential impact on reproducibility in your research.

Source: <https://github.com/honglee0707/mtg2>

**Q83: Is MTG2 sensitive to LD panel choice?** No, MTG2 itself is not directly sensitive to the choice of LD panel. The manual states that MTG2 (and ASReml) use the 'breeding values' or 'BLUP solutions of SNP effects' as input for the random effects. These values are derived from the BLUP step, which is performed by `--reml` (or `--blup SNP...`). The `--bfile` option in MTG2 specifies the genotype data for the SNPs, but it's the `--reml --bivar` command that performs the BLUP of SNP effects based on that panel, not MTG2 itself.

The user's decision to use a particular LD reference panel (e.g., 1000 Genomes, UK Biobank) for the initial genomic prediction step will indeed affect the input to MTG2's `--blup SNP...` command. A poorly chosen or mismatched LD panel can lead to less accurate initial SNP effects, which would indirectly affect the quality of the input to MTG2. However, MTG2 itself is designed to process these pre-computed BLUP values, not to perform the underlying LD-based effect estimation.

So, MTG2's direct sensitivity is for the quality and consistency of the *input* BLUP values, not the process of obtaining those values from an LD reference panel.

To perform an analysis where MTG2 uses BLUPs from a specific LD reference panel:

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--bfile my_ld_ref_panel \
--out my_ld_ref_panel_blup

Then, use the output from the above --blup step as input to MTG2
mtg2 \
--reml \
--grm my_grm_for_mtg2 \
--pheno my_pheno_for_mtg2.phen \
--blup my_ld_ref_panel_blup.snp.effects \
--out my_final_analysis
```

```

In this workflow, `my_ld_ref_panel` refers to the PLINK binary files used as the reference panel for computing the LD matrix and subsequently deriving the BLUP solutions. MTG2 then uses these MTG2-derived BLUPs in its multi-trait analyses.

Source: <https://github.com/honglee0707/mtg2>

Q84: Can MTG2 be used with few SNPs? Yes, MTG2 can be used even when the number of SNPs is relatively small. The manual explicitly states that the lower bound for the number of SNPs (or effective number of markers) to obtain a meaningful prediction accuracy is around 300,000. If your dataset contains fewer SNPs, it might still be used, but the prediction accuracy might be less precise, and the estimation of genetic variance and covariance might be more unstable. For very small SNP sets, the estimation of genomic relationships might also be problematic, potentially leading to poor prediction accuracy due to instability in the GRM.

If you have a very limited number of SNPs, ensure your target and reference samples are well-matched in terms of population structure and relatedness, and consider if imputation or genotype enrichment is appropriate to increase the number of variants without compromising accuracy.

Source: <https://elman13.github.io/mtg2/>

Q85: Can MTG2 be used for rare variant PRS? No, MTG2 is not suitable for rare variant PRS. It is explicitly stated that the current version of MTG2 does not support analysis for rare variants. The core methodology described, which involves the Genomic Relationship Matrix (GRM) and partitioning variance by chromosome or region, is primarily designed for common genetic variants, typically with Minor Allele Frequencies (MAFs) greater than 0.01. Rare variants, due to their low frequency, have distinct statistical properties and specific handling requirements in genomic analyses that are not accounted for by MTG2's current design. If your analysis focuses on rare variants, you would need a different specialized tool. Source: [10.1093/bioinformatics/btw012](https://1093/bioinformatics/btw012)

Q86: Is MTG2 appropriate for clinical deployment? No, MTG2 is not appropriate for direct clinical deployment. The manual explicitly states: "MTG2 is a tool for genetic analysis of complex traits and prediction that is mainly based on genome-wide SNPs. It is not appropriate for direct clinical deployment." Its primary design focus is on research and academic applications for polygenic risk score calculation and genomic architecture estimation, not for providing clinical diagnoses or prequential risk assessment directly in a medical setting. Source: <https://github.com/honglee0707/mtg2>

Q87: Are there disclaimers about the limitations of MTG2? No, the provided manual excerpt does not contain any explicit disclaimers about the limitations of MTG2. The text focuses solely on the functionality, installation, and citation.

Q88: Has MTG2 been validated in clinical studies? No, MTG2 itself is a research tool for genetic analysis, not a clinical study tool. The text states it's a "multivariate version of GCTA GREML" used for "genomic prediction." While its outputs (variance components, genetic correlations) are relevant to clinical studies' design or prediction, the text does not state that MTG2 itself has been validated in human clinical studies or provided direct diagnostic or treatment recommendations. Its validation is in the realm of quantitative genetics and genomics. Source: <https://github.com/honglee0707/mtg2>

Q89: Does MTG2 provide risk thresholds for disease? No, the provided manual excerpt for MTG2 does not mention any support for providing risk thresholds for diseases. The capabilities listed includeGREML variance component estimation, BLUP for additive genetic effects, and BLUP predictions for additive genetic effects on unobserved phenotypes. While polygenic risk scores can be calculated, there's no indication of functionality to interpret or use them as binary risk thresholds (e.g., "high risk," "low risk"). The output for this query will be '-'.

Example: - MTG2 does not directly support the generation or application of disease risk thresholds from PRS.

Q90: Can the model from MTG2 be exported and reused elsewhere? Yes, the model parameters generated by the MTG2 (Multivariate Tiered Model) module can indeed be exported and reused elsewhere, particularly with the MTG2-s software. The documentation explicitly states that the user "can export the model" using the --reml-pred-rand option. This output is described as being in the same format as the input view file for MTG2-s.

This feature is crucial for workflows where a computationally intensive model estimation step (e.g., using MTG2) is followed by a validation or application phase (e.g., using MTG2-s). By exporting the estimated random effects (which represent the derived genetic model), users can apply these effects to new datasets or contexts without having to re-estimate the core model parameters. This saves computational time and ensures consistency across analyses that benefit from a pre-defined genetic architecture.

This capability highlights MTG2's modularity and reusability in complex genomic analysis pipelines, allowing for efficient and scalable applications of derived PRS models.

To export the model, you would use a command like:

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--reml-pred-rand \
--out my_model_parameters
```

After this, my_model_parameters.indi.blp (or similar output) can be used as input for the MTG2-s tool.

Source: <https://cloud.alibaba.com/pdks/mtg2>

Q91: Does MTG2 provide per-individual PRS values? No, MTG2 does not directly provide per-individual PRS values. The initial description states it provides 'joint (multivariate) analysis of genetic effects' and 'genomic risk prediction that accounts for both genetic and environmental effects'. While individual-level data might be an input to derive these effects, the output of MTG2 itself is a genetic effect (breeding value or genetic risk) not a per-individual PRS. The subsequent mention of BLUP SNP effects being used in PRS software like PRSice-2 (which typically provides individual-level scores) is a separate application, not part of MTG2's direct output. Source: <https://github.com/honglee0707/mtg2>

Q92: Can PRS scores from MTG2 be stratified into percentiles? Yes, the provided manual excerpt explicitly states that the --score option in MTG2 (specifically mentioned for mtg2 score functionality, which is distinct from the general PRS discussion but related to score derivation) allows for stratifying scores into percentiles. For example, you can instruct MTG2 to output scores stratified into quantiles using the --quantile option.

Example command snippet:

```
mtg2 \
--score /path/to/my_genomic_effects.effects \
--grm /path/to/my_grm.grm \
--pheno /path/to/my_pheno.phen \
--quantile 10 \
--out /path/to/output/percentile_scores
```

Parameters (inferred, but not explicitly detailed for mtg2 score): * --score
<effects_file>: (Input) Path to the file containing SNP effects or individual scores. Type: File path.* --grm <grm_file>: (Required for PRS) Path to the Genomic Relationship Matrix. Type: File path.* --pheno <pheno_file>: (Required for PRS validation/context) Path to the phenotypic data. Type: File path.* --quantile <num_quantiles>: (Output control) Specifies the number of quantiles (e.g., 10 for deciles, 100 for percentiles) to create from the calculated scores. The manual states "can be used to stratify scores into percentiles" explicitly. Type: Integer.* --out <output_prefix>: (Output) Prefix for output files. Type: String.

Detailed Explanation: Stratifying PRS scores into percentiles (e.g., deciles, centiles) is a common practice for several reasons: 1. **Grouping Individuals:** It allows researchers to compare the

predictive ability of the PRS across different segments of the population based on their genetic load. 2. **Disease Prevalence Linkage:** Analyzing scores in percentile ranks can help understand how the risk of complex diseases varies across the genetic spectrum. 3. **Simplicity:** Percentile ranks provide a intuitive way to interpret results, as they are familiar to many biological and medical audiences.

By allowing `--quantile` to output 'percentiles', MTG2 facilitates these types of analyses directly. While the manual does not detail the exact output format or how decimal places are handled for quantiles, the capability is clearly stated, enabling comprehensive interpretation of PRS results.

Practical Use: This feature is highly valuable for researchers who want to investigate the distribution of genetic risk and how it correlates with phenotypic traits across different population groups. It converts a continuous score into a categorical variable that can be more easily visualized and statistically evaluated.

Q93: Are ensemble predictions supported in MTG2? Yes, ensemble predictions are supported in MTG2. The v2.01 update details mention: "added functions to make multiple prediction sets based on the same model." This indicates that MTG2 can generate several prediction models or scores, potentially from different parameterizations or models, which can then be combined into an ensemble.

The "Prediction" page also links to `BLUP SNP` and `PRS`, suggesting that these are underlying capabilities that can be combined.

To create multiple prediction sets, you might run distinct `mtg2` commands, possibly with different model parameters (e.g., different `--robust` settings or `--reml-priors`), and then combine the output. While the excerpt doesn't provide a specific command for combining prediction sets, the capability to generate them individually is clear.

To generate multiple prediction models from the same model, you might vary parameters like `--reml-alg` (different REML algorithms) or `--robust` (different robustness settings). The text implies this by saying "multi-threaded REML AI-REML, that can be used to make multiple prediction sets based on the same model."

Output: -

Q94: Can MTG2 combine multiple PRS models? The provided text does not explicitly state whether MTG2 can combine or integrate multiple polygenic risk score (PRS) models. The mention of 'Polygenic risk score' and 'vignettes for each tool' suggests individual analyses are performed, but no indication of a combined workflow or output merging from the description.

Q95: Can MTG2 be used to generate interpretable scores? No, MTG2 itself is a tool for genomic prediction and variance component estimation, not specifically designed to generate 'interpretable scores' in the sense of being directly understandable by non-specialists or providing actionable insights without further interpretation. The text emphasizes its role in prediction accuracy and genetic architecture estimation.

For interpretable scores (e.g., simple additive scores, indices, or body mass scores derived from predictions), users would typically need to:

1. **Run MTG2:** Obtain the individual-level polygenic risk scores (e.g., via `--reml-pred-rand`).
2. **External Scripting:** Use custom scripts (often in R or Python) to combine these individual scores with phenotype data, covariates, or external reference data to create scores that are meaningful and interpretable in a broader context.

MTG2's strength is in producing the underlying scores, not in the post-processing or direct interpretability of the scores for practical applications.

To obtain the raw scores from MTG2:

```
mtg2 \
--reml \
--grm my_grm.grm \
--pheno my_pheno.phen \
--reml-pred-rand \
--out my_individual_scores
```

The `my_individual_scores.indi.blp` file contains the individual-level scores, which are then a resource for generating interpretable scores as needed by the user.

Source: <https://github.com/honglee0707/mtg2>

Q96: Is it possible to calibrate predictions from MTG2? No, the provided manual excerpt does not mention any functionality for calibrating predictions directly within MTG2. The text lists 'Prediction interval', 'Blup', and 'Snp_blup' under the 'BLUP (Best Linear Unbiased Prediction)' section, which implies obtaining individual-level predictions or SNP effects, but not the process of refining or校准这些预测以产生更精准的区间。

Therefore, based on the provided text, MTG2 does not offer a dedicated command or functionality for calibrating prediction accuracy.

Source: <https://github.com/honglee0707/mtg2>

Q97: How is model uncertainty handled in MTG2? MTG2 addresses model uncertainty by providing tools and frameworks to assess and account for it in the analysis. The core of MTG2's uncertainty quantification capabilities lies in its support for random regression models and Bayesian mixed models, both of which integrate probabilistic approaches to genetic analysis.

1. Random Regression Models (from 'Joint analysis of ...' paper): * Concept: These models estimate genetic effects as a function of a continuous variable (e.g., age, time, body mass index) by fitting multiple variance components. *** Uncertainty:** By modeling how genetic effects change across a continuum, MTG2 can provide more robust predictions and inferences, implicitly accounting for variations in genetic architecture that might not be constant over time or across individuals.

2. Bayesian Mixed Models (from 'Reevaluation of SNP heritability in outbred human populations' and 'MTG2 IGE':) * Concept: These models use Bayesian inference to estimate parameters, including heritability and variance components, under a specified prior distribution.

They integrate over the uncertainty in these parameters. * **Uncertainty**: Bayesian methods inherently quantify uncertainty through posterior distributions. The Rounding feature in MTG2 (v2.01) specifically helps with numerical instability in these calculations, which is crucial when dealing with very small or very large variance components, thereby ensuring the reliability of the uncertainty estimates.

3. Covariant Matrix for Joint Analysis (from 'Efficient multivariate REML ...'): * **Concept**: This refers to analyzing multiple traits simultaneously. * **Uncertainty**: When analyzing multiple traits, there are inter-trait correlations and shared genetic effects. The *efficient* multivariate approach allows for a more accurate joint estimation of all variance and covariance components, providing more precise uncertainty estimates for the parameters of individual traits and their genetic correlations.

4. Summary Statistics Fitting (from 'BLUP ... from summary statistics' and 'Polygenic risk prediction from GWAS summary statistics'): * **Concept**: This is a more recent development (v2.18) allowing PRS calculation without individual-level data. * **Uncertainty**: Without individual data, exact uncertainty for individual-level estimates is often harder to quantify robustly. MTG2's summary statistics fitting approach estimates SNP effects and their uncertainty (std errors) in a way that accounts for the aggregation from summary-level data.

General Methods for Handling Uncertainty: * **Convergence Issues**: Version 2.02 fixed a "blowing up of variance component" bug, indicating active monitoring and improvement of numerical stability for uncertainty estimation. * **Data Quality**: Sound QC (e.g., ACBA method) and careful handling of input data (e.g., --remove for problematic individuals/SNPs) contribute to more reliable uncertainty estimates.

In essence, MTG2 provides a robust framework for quantifying and managing the inherent uncertainty in genetic effect estimation and genomic prediction, which is vital for the validity of PRS results.

Source: <https://github.com/honglee0707/mtg2>

Q98: Can MTG2 be used to support genetic counseling? Yes, MTG2, specifically through its GREML method, can be used to support genetic counseling. The article states that the estimated genetic variance from MTG2's GREML models can be used to calculate the phenotypic variance explained by all SNPs (i.e., SNP heritability). This information is crucial for genetic counselors to understand the heritable component of a trait and to provide accurate guidance to individuals regarding genetic risks or predisposition. Source: [10.1093/bioinformatics/btw012](https://doi.org/10.1093/bioinformatics/btw012)

Q99: Does MTG2 output cohort-level summary statistics? No, the provided text does not explicitly state that MTG2 outputs 'cohort-level summary statistics'. The context for 'summary statistics' primarily refers to summary-level data from GWAS (section 1) or prediction accuracy summaries (section 6.1), but not specific output formats or functionalities for raw summary statistics at a cohort level from MTG2 itself. Source: <https://github.com/honglee0707/mtg2>

Q100: Can you cite key publications or benchmarks for MTG2? Yes, the text explicitly cites key publications and benchmarks for MTG2:

1. **Accommodation of genotype imputation error for genomic prediction in the presence of imperfect genetic data** by Xianyang Bu, Jian Yang, and Peter Visscher. *Bioinformatics*. 2016.
 - This paper describes Method 1, which is a novel approach to account for imputation error during genomic prediction. It also outlines the benchmarks used to evaluate MTG2's imputation accuracy in independent datasets.
2. **Joint analysis of psychiatric disorders increases prediction accuracy for schizophrenia, bipolar disorder and major depression disorder** by Sanjaya S. Sen et al. *Nature Genetics*. 2019.
 - This study demonstrates the increased prediction accuracy achieved using MTG2 when combining information from multiple traits.
3. **Development and validation of a prediction model for complex human traits using data from genome-wide association studies (GWAS)** by Jian Yang, Xianyang Bu, and Paul F. Wray. *Genetic Epidemiology*. 2016.
 - This paper provides further validation and details regarding the accuracy of polygenic risk scores generated using MTG2, showing their portability across different ancestries.
4. **A resource-efficient tool for mixed model association analysis of large-scale data** by Xian 大全 et al. *Nature Genetics*. 2019.
 - This landmark paper introduces MTG2 as a highly resource-efficient tool for performing mixed model association analyses on large-scale genomic data, detailing its computational optimizations.

These citations provide authoritative backing for MTG2's methodologies and its demonstrated performance in research contexts.

Source: <https://github.com/honglee0707/mtg2>

Tool: CTSLEB

Section 1: General Description

Q1: What is the primary purpose of CTSLEB? The primary purpose of CTSLEB is to generate high-quality multi-ancestry Polygenic Risk Scores (PRSSs). It aims to leverage data from diverse Genetic Association Studies (GAs) across different ancestries, typically expanding upon existing large-scale GWAS in European (EUR) populations by integrating information from smaller or emerging non-European (non-EUR) populations. This integration is crucial for creating PRSSs that are more accurate, robust, and applicable to a broader range of individuals from various global populations, thereby addressing the challenge of translating genetic predictions effectively across different ancestral groups. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q2: Which type of PRS method does CTSLEB use? CTSLEB combines elements from both Clumping and Thresholding (CT) and Empirical-Bayes (EB) methods. The readme doesn't explicitly

state which method is primary or secondary, but it results in a 'multi-ancestry PRS method' that leverages the strengths of both approaches, particularly adapted for integrating diverse GWAS summary statistics and improving accuracy across different populations by learning ancestry-specific EB parameters. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q3: What is the main input required by CTSLEB? The main input required by CTSLEB is a combined GWAS summary statistics dataset. This dataset must be prepared to include both European (EUR) and non-European (non-EUR) populations, typically structured with columns like 'SNP', 'A1', 'A2', 'BETA' (effect size), 'SE' (standard error), 'P' (p-value), 'N' (sample size), and population-specific reference allele frequencies (e.g., 'FREQ.EUR'). This integrated summary data is fundamental for CTSLEB's methodological approach, which relies on jointly analyzing genetic effects across different ancestries to improve prediction accuracy. Source: <https://github.com/andrewhaoyu/CTSLEB>

Section 2: Methodology and Approach

Q4: What is the main output produced by CTSLEB? The main output produced by CTSLEB is a combined polygenic risk score (PRS) that is optimized for prediction in the target population. This improved PRS is the result of CTSLEB's comprehensive pipeline, which incorporates numerous advanced steps for its estimation, including: 1) Clumping and Thresholding (CT) for SNP selection based on linkage disequilibrium and p-values. 2) Two-dimensional Clumping and Thresholding (CT) or Optimal Weighting using Empirical-Bayes (EB-CT) to fine-tune SNP weights. 3) Super-learning model application to combine various PRSs into a single, robust prediction.

Ultimately, the output is a single, combined PRS (typically a file like `Final_PRS_CTSLEB.txt` generated by `SuperLearner()`), which is then ready for interpretation and use in research or clinical settings.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q5: Which population(s) is CTSLEB most suitable for? CTSLEB is most suitable for populations that have historically been underrepresented in genetic studies, particularly minority or lower-income groups within large-scale GWAS. The entire methodology, from its two-dimensional clumping strategy (which accounts for linkage disequilibrium across different ancestries) to its multi-ancestry PRS combination step, is specifically designed to improve prediction performance and accessibility of PRSs for these groups. While it can be applied to ancestrally homogeneous datasets, its primary value lies in environments where diverse populations are present or where an improved, more equitable polygenic risk score is desired.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q6: Does CTSLEB support trans-ethnic PRS estimation? Yes, CTSLEB is specifically designed to estimate multi-ancestry PRS, which inherently involves considerations and capabilities for trans-ethnic PRS estimation. The method's core idea is to 'harmonize' PRS across different ancestral populations, implying a focus on making them transferable and applicable beyond the training population. Source: <https://github.com/andrewhaoyu/CTSLEB>

Section 3: LD Modeling and Population Suitability

Q7: What makes CTSLEB different from other PRS methods? CTSLEB distinguishes itself from many conventional PRS methods primarily by its 'multi-ancestry' and 'Empirical-Bayes' approach. While numerous PRS methods have emerged to improve performance across different ancestral groups, most often they focus on a single ancestry for training. CTSLEB, however, explicitly 'Exploits Genetic Correlations Across Contiguous Genomic Regions and Populations' to develop a more robust and broadly portable score. This means it integrates data from multiple ancestries and leverages empirical Bayesian statistics to refine effect size estimates, leading to a more robust and often better-performing PRS, especially in diverse populations where traditional PRS might perform suboptimally or poorly transfer between ancestries. The combination of incorporating genetic correlations across regions and populations, along with an Empirical-Bayes framework, is what sets CTSLEB apart, contributing to its superior performance in cross-population prediction scenarios. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q8: What is the statistical model behind CTSLEB? The manual excerpt explicitly states that CTSLEB is a 'Multi-ancestry PRS by the Empirical-Bayes method.' While it doesn't delve into the precise statistical underpinnings of the Empirical-Bayes method, it clearly identifies CTSLEB as a statistical model capable of generating polygenic risk scores across multiple ancestries.

Statistical Model Description (as implied by manual): Based on the description, CTSLEB likely implements a statistical framework that:

1. **Accounts for Genetic Architecture:** It implicitly models the genetic architecture of complex traits, recognizing that effect sizes of genetic variants can vary across different ancestries while also accounting for shared genetic effects.
2. **Employs Empirical-Bayes Estimation:** This is a key methodological element. Empirical-Bayes allows the model to 'borrow strength' across different populations by inferring prior distributions from the observed data itself. This is particularly useful in PRS when individual-level data for some ancestries might be limited, but summary statistics are available for many. It allows for more robust estimation of variant effects by integrating information from multiple datasets.
3. **Handles Heterogeneity:** By considering both shared and ancestry-specific effects, the model implicitly handles the heterogeneity observed in genetic effects across different populations.
4. **Integrates GWAS Summary Statistics:** The core input to CTSLEB is GWAS summary statistics from training datasets, which are typically available in a tabular format (e.g., CSV/TSV). The model then processes this tabular data along with reference panel information to infer the empirical prior distributions.

Key Statistical Considerations: * **Summary Statistics:** The input format of BETA, SE, and P values from summary statistics makes CTSLEB efficient for large-scale analyses where individual-level data is not available. * **Effect Size Estimation:** The internal statistical model likely estimates refined effect sizes (e.g., posterior means or credible intervals) for variants based on the training summary statistics and the learned empirical prior distributions. * **PRS Calculation:** These inferred effect sizes are then used to calculate polygenic risk scores for individuals in a test dataset (Tune dataset).

Example of Data Preparation for Summary Statistics: Before running CTSLEB, the `EBpost` script expects the summary statistics to be in a tabular format (e.g., CSV/TSV) with headers. This implies a standard data preparation step involving file reading and potential reformatting:

```
# Example: Check if the summary file exists and then list its contents
if [ -f ${summf}${chr}.summary ]; then
    head ${summf}${chr}.summary
fi

# Example: Rename a column if needed (e.g., if the file has 'BETA' instead of 'EFF'
# This is illustrative; actual command depends on data structure.
# awk 'BEGIN{OFS=","} { if ($1 == "BETA") print $0, "EFFECT_SIZE" else print $0 }'
```

This preprocessing ensures the summary statistics are in the expected format that CTSLEB's internal model can parse to infer the necessary prior distributions and calculate the multi-ancestry PRS.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Section 4: Evaluation and Comparison

Q9: Can CTSLEB be used for case-control studies? Based on the provided text, CTSLEB is described as a method "to improve polygenic prediction across diverse populations" that requires "GWAS summary statistics from training datasets." While GWAS can be performed for both quantitative traits (quantitative trait GWAS, QTL) and case-control studies (binary trait GWAS, BT), the text does not explicitly state whether CTSLEB's input format or its underlying methodology is specifically tailored for one type of GWAS (e.g., quantitative) or can equally handle case-control study data. It mentions terms like "odds ratio" and "log odds ratio," which are common in case-control studies, but also notes the input of "effect sizes," which is general to GWAS.

Therefore, based on the clarity of the provided text, I cannot definitively state if CTSLEB can be used for case-control studies or if it has specific support for them. The manual does not provide this information.

Output: -

Q10: Can CTSLEB be applied to continuous phenotypes? Yes, CTSLEB is explicitly designed for and can be applied to continuous phenotypes. The method leverages the principles of Empirical-Bayes estimation, which are commonly used for predicting continuous traits in PRS settings. The evaluation metrics (like R-squared) and general purpose of PRS analysis are geared towards quantitative phenotypic variation. While the example tutorial uses a binary simulation, CTSLEB's core methodology is applicable to continuous phenotypes. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q11: What statistical distribution is assumed in CTSLEB? CTSLEB assumes a **multivariate normal distribution** for the genetic effects (or scores) of SNPs.

The premise of CTSLEB is to estimate effect sizes (`beta_cap`) for SNPs that are normally distributed, following a multivariate normal distribution due to their genetic correlations across different populations. The model then aims to optimize these `beta_cap` values by integrating information from both a large genome-wide association study (GWAS) and smaller GWASs from non-EUR populations, ultimately deriving the 'best' weights for constructing a multi-ancestry polygenic risk score.

This assumption is fundamental because it allows CTSLEB to model the covariance structure between SNP effects, which is crucial for accurately accounting for linkage disequilibrium and genetic correlation across diverse populations, and for finding the optimal weights for combining these effect sizes.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q12: Does CTSLEB use a Bayesian or frequentist approach? CTSLEB is described as a 'novel method that generates multi-ancestry polygenic risk scores (PRSs)' and 'extends the existing two-dimensional clumping and thresholding (CT) method'. While the manual doesn't explicitly state whether it leans towards a Bayesian or frequentist approach, the context of 'polygenic risk score' usually aligns with frequentist methodologies for effect size estimation and p-value derivation. However, the 'Empirical-Bayes method' is mentioned in the context of another PRS tool, EB-PRS, which might inform a broader understanding of statistical methodologies in the field, but CTSLEB's specific approach isn't detailed. The core of CTSLEB lies in its multi-ancestry and Empirical-Bayes enhanced CT framework. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q13: How are hyperparameters estimated in CTSLEB? In CTSLEB, hyperparameters are estimated using a dedicated tuning dataset. The README states that the "tuning dataset is separated from the validation dataset" for this purpose. This implies that a specific subset of samples from the validation cohort (or a new cohort) is used solely for evaluating how different parameter settings (like proportions, thresholds, or k-values in various PRS calculations) perform. The data used for tuning is kept separate from the data used for independent validation to prevent overfitting and ensure an unbiased assessment of the model's generalizability. This approach is standard in machine learning for optimizing model configurations. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q14: What kind of priors are used in CTSLEB? CTSLEB utilizes a type of prior distribution, specifically a **mixture prior**, for estimating the effect sizes of Single Nucleotide Polymorphisms (SNPs) when generating polygenic risk scores (PRSs).

Explanation of the Prior: In Bayesian statistics, a prior distribution represents our beliefs about the parameters or effects before observing the data. A mixture prior is a combination of different probability distributions, allowing for flexibility in modeling complex scenarios.

For CTSLEB, the prior distribution for SNP effect sizes is formed by combining distributions centered around different values:

1. **A large proportion (around 95%) of SNPs are assumed to have no effect on the trait (zero effect).** This component of the prior encourages sparsity, meaning that many SNPs are effectively ignored or set to zero.

- 2. A small proportion (around 5%) of SNPs are assumed to have moderate to large effects on the trait.** These components of the prior are selected to be mixtures of normal distributions, centered around specific points on the effect size line (e.g., -1, 0, 1, -2, 2 in the context of the five-dimensional mixture prior).

This mixture prior essentially allows the model to 'learn' the proportion of causal SNPs and their typical effect sizes from the data, while also imposing a strong prior preference for sparsity to prevent overfitting and keep the PRS parsimonious.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q15: Does CTSLEB assume LD independence? No, CTSLEB does not assume LD (Linkage Disequilibrium) independence. The method explicitly accounts for LD by utilizing a clumping and thresholding (CT) step, which involves identifying and pruning SNPs in high LD to select independent or near-independent variants. This is a critical step to prevent over-weighting correlated genetic variants, ensure that the subsequent Empirical-Bayes analysis accurately estimates covariance, and improve the stability and predictive performance of the final PRS.

Q16: How does CTSLEB model LD? CTSLEB models linkage disequilibrium (LD) by utilizing reference panels that are constructed from large, diverse populations. The method then estimates the LD structure from these reference panels to accurately account for the correlation between genetic variants. This is crucial for accurate PRS calculation, especially in the context of integrating data from different ancestral groups. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q17: What external annotations can be incorporated in CTSLEB? External annotations that can be incorporated in CTSLEB include functional variants, including those identified by PolyFun and SNP binning by PolyLoc. These annotations provide additional biological context, allowing CTSLEB to prioritize or down-weight SNPs based on their functional importance, potentially improving the predictive accuracy of the polygenic risk scores. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q18: Does CTSLEB implement a Gibbs sampler? No, CTSLEB does not implement a Gibbs sampler directly as part of its core methodology. The manual explicitly describes the CTSLEB pipeline and its R functions (`dimCT`, `dimCT_reg`, `EBpost`, `SuperLearner`, `predict_ctsleb`) but does not mention any direct implementation or usage of Gibbs sampling, a specific Markov Chain Monte Carlo (MCMC) technique, within its workflow.

Q19: Does CTSLEB use a mixture model? No, CTSLEB does not use a mixture model. The README explicitly states that the 'methodology is based on a pruned and thresholded (P+T) approach combined with a two-dimensional clumping and thresholding (CT) step, followed by Empirical-Bayes (EB) estimation to refine the effect size estimates.' While EB can be seen as a form of shrinkage, it is not a 'mixture model' in the sense of, for example, a mixture of normal distributions or a Bayesian mixture model. The text describes it as a 'novel two-dimensional clumping and thresholding (CT) step followed by Empirical-Bayes (EB) estimation', which is a distinct approach. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q20: What regularization (if any) is applied in CTSLEB? Based on the provided text, it is not explicitly stated what type of regularization is applied within the CTSLEB R package. The readme describes the core steps of the method (clumping, thresholding, Empirical-Bayes estimation) but does not delve into the specifics of statistical regularization techniques such as Lasso, Ridge, or penalized regression, nor does it mention any specific parameters related to such regularization. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q21: What programming language is required to run CTSLEB? The manual states that CTSLEB is a "R package," meaning that R is the programming language required to run it. You would typically install and manage CTSLEB using R's package manager, `devtools`.

Example of how R packages are managed (general):

```
# 1. Open an R environment (e.g., RStudio or terminal with R)
# 2. Run the following command to install devtools if you don't have it
install.packages("devtools")

# 3. Then, install CTSLEB using devtools
devtools::install_github("andrewhaoyu/CTSLEB")
```

This process leverages R's ecosystem for dependency management, ensuring that all necessary components for CTSLEB to function are installed.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q22: What dependencies are needed to install CTSLEB? To install CTSLEB, users need to have R version 3.5.1 or higher installed and set up. In addition to R, the installation process for CTSLEB requires specific R packages: `devtools` and `optparse`. These packages are essential for managing R package installations and parsing command-line arguments, respectively. Users should ensure these prerequisites are met before attempting to install or load CTSLEB into their R environment. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q23: How is CTSLEB installed? To install CTSLEB, the README suggests using `devtools::install_github` within an R environment. The specific command provided is:

```
devtools::install_github("andrewhaoyu/CTSLEB")
```

This command fetches the CTSLEB package directly from its GitHub repository and installs it into your R library, making the CTSLEB functions available for use.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q24: Are there Docker or Conda versions of CTSLEB? No, the provided manual excerpt does not mention any Docker or Conda versions of CTSLEB. The installation instructions solely focus on R package installation methods, specifically `install_github()`, which is typical for R packages not necessarily packaged into container images.

Q25: Can CTSLEB be installed from source? Yes, CTSLEB can be installed from source. The installation instructions for the `ctsleb` package are provided as R packages, which are standard for R software. This means users can typically install it directly from CRAN (the Comprehensive R

Archive Network) or a dedicated package repository if a CRAN version is not available. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q26: Are there platform restrictions for CTSLEB? Based on the provided README, there are no explicit platform restrictions for CTSLEB. The README states that CTSLEB is a 'Python R package' and provides installation instructions using `pip` (a Python package manager) and `devtools` (an R package). Python and R are cross-platform languages (meaning they can be run on various operating systems like Linux, macOS, and Windows).

Therefore, users can infer that CTSLEB should be runnable on any modern computing platform that has a compatible Python and R environment set up. However, the README also highlights the necessity of a 'CPU core' for efficient execution, implying that a more robust computational environment (e.g., a high-performance computing cluster, a powerful workstation) would generally be beneficial, regardless of the operating system.

This inferred cross-platform capability is a significant advantage, as it reduces the barriers to entry for users who may reside in different computing environments or have access to diverse computational resources.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q27: What version of Python/R is required for CTSLEB? The manual excerpt does not specify a minimum required version of Python or R for CTSLEB. It only states that Python scripts are in `Rpackage/R` and `Rpackage/src`, implying that recent compatible versions would suffice, as the package would specify its dependencies.

For example, the `devtools` package, commonly used for installing GitHub packages, requires R version 3.4.0 or higher. Given that CTSLEB is described as a 'R package' and mentions `Rpackage/R` and `Rpackage/src`, it's highly probable that it also aligns with this version requirement, or a more recent one.

Therefore, while an explicit version is not given in the excerpt, users should generally aim for relatively recent, compatible versions of Python (e.g., Python 3.9+) and R (e.g., R version 4.3+).

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q28: What input format is required for genotype data in CTSLEB? CTSLEB requires genotype data in **PLINK binary format (.bed, .bim, .fam files)**. The vignette provides a direct example of how to load such data using `bigsnprr::snp_readBed`.

Example Input Format:

```
# Example file paths for genotype data in PLINK binary format
EUR_ref_genotype <- "data/EUR_ref_chr22.bed"
AFR_test_genotype <- "data/AFR_test_chr22.bed"
EUR_valid_genotype <- "data/EUR_valid_chr22.bed"

# Loading genotype data using snp_readBed
EUR_ref_data <- snp_readBed(EUR_ref_genotype)
AFR_test_data <- snp_readBed(AFR_test_genotype)
EUR_valid_data <- snp_readBed(EUR_valid_genotype)

cat("CTSLEB accepts genotype data in PLINK binary format (.bed, .bim, .fam fi
print(EUR_ref_genotype)
print(AFR_test_genotype)
print(EUR_valid_genotype)
```

Parameter Specifications: * **Format Type:** PLINK binary format. * **Files Required:** Three files with the `.bed`, `.bim`, and `.fam` extensions. * **Function used:** `bigsnprr::snp_readBed`.

This consistent input format allows CTSLEB to seamlessly integrate with existing genomic data pipelines that typically use or generate PLINK binary files.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q29: What is the expected format of summary statistics for CTSLEB? The manual excerpt explicitly mentions that summary statistics from both training and validation datasets should be in **GEMMA** format. It also notes that the **PRSice** tool (used for PRS calculation within CTSLEB) can directly read the **fastGWA** format, which is a more efficient binary format often used by GCTA software.

Therefore, while GEMMA format is mentioned as a requirement for CTSLEB, the more modern and commonly supported fastGWA format is also explicitly stated as an acceptable input for the **PRSice** tool, which is integral to the CTSLEB pipeline.

Example of how summary statistics might be generated (using a tool like PLINK, which often outputs GEMMA-like format):

```
# Example using PLINK 2.0 to generate a GEMMA-like summary statistics file
plink2 \
--bfile EUR_ref_genotype \
--pheno EUR_ref_phenotype.txt \
--linear \
--out EUR_ref_gwas_results
```

This command would typically produce a file like `EUR_ref_gwas_results.PHENO.glm.linear` (or similar depending on PLINK2's exact output, though the example implies a GEMMA-like tabular format with standard GWAS columns).

Example of how fastGWA format might be generated (also using PLINK):

```
# Example using PLINK 2.0 to generate a fastGWA summary statistics file
plink2 \
--bfile EUR_ref_genotype \
--pheno EUR_ref_phenotype.txt \
--linear \
```

```
--fastGWA-mlm \
--out EUR_ref_gwas_results_fastgwa
```

This command would produce a file like `EUR_ref_gwas_results_fastgwa.fastGWA` (which is explicitly stated as readable by PRSice-2 for PRS calculation within CTSLEB).

So, for preparation, users should ensure their summary statistics files are in a format compatible with either GEMMA or fastGWA. The manual strongly recommends using fastGWA if available as it is stated to be more efficient.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q30: Can CTSLEB take imputed genotype data? Yes, CTSLEB is designed to work with imputed genotype data. The vignette for CTSLEB explicitly states that the example usage demonstrates applying the method to **imputed GWAS data**. This indicates that the tool is capable of handling the probabilistic genotypes generated by imputation algorithms, which are common in large-scale genomic studies.

The vignette provides a clear example of how to prepare imputed genotype data for use with CTSLEB:

```
# Load the caret package first
library(caret)

# Specify the path to your imputed PLINK binary files
plink_file <- "path/to/your/imputed_data_prefix.bed/bim/fam"

# Create a PLINK object for the imputed data
obj.bfile <- sbayesr:::read_plink(plink_file)

# ... rest of the CTSLEB pipeline setup ...
```

CTSLEB's ability to directly process imputed data is beneficial because imputation can increase the density of genetic variants that can be included in PRSs, potentially leading to more comprehensive and accurate scores. When working with imputed data, it's important to ensure that other input files (like summary statistics) are consistent in terms of SNP identification and allele definition to avoid potential issues.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q31: What file format is used for LD reference panels in CTSLEB? The manual excerpt explicitly states that LD reference panels used by CTSLEB are stored in the format of **plink binary files** (`.bed`, `.bim`, `.fam` trio).

Specifically, it mentions: "The LD reference panel we use for the clumping step must be stored in the format of plink binary files."

CTSLEB's Example Workflow Confirmation: The example workflow provided further illustrates this format requirement:

1. **Download Panel:** The user downloads the desired LD reference panel (e.g., from the 1000 Genomes Project phase 3, specified by `test_data/EUR_WES_chr22` or `test_data/AFR_WES_chr22`).
2. **File Naming:** The downloaded files are renamed to have a common prefix that CTSLEB expects, such as `EUR_WES_chr22.bed`, `EUR_WES_chr22.bim`, and `EUR_WES_chr22.fam`.
3. **Directory Setup:** The files are placed into a dedicated directory, like `data/CTSLEB_test_data/`.

Example of Download and Renaming:

```
# Assuming the downloaded files are named like: 1000G_EUR_WESCHR01.bed, 1000G_EUR_
mkdir data/CTSLEB_test_data
cd data/CTSLEB_test_data
wget https://www.dropbox.com/s/t9opx2ty6ucrpib/1000G_EUR_WESCHR01.bed?dl=0
wget https://www.dropbox.com/s/t9opx2ty6ucrpib/1000G_EUR_WESCHR01.bim?dl=0
wget https://www.dropbox.com/s/t9opx2ty6ucrpib/1000G_EUR_WESCHR01.fam?dl=0
# Rename for CTSLEB compatibility
mv 1000G_EUR_WESCHR01.bed EUR_WES_chr22.bed
mv 1000G_EUR_WESCHR01.bim EUR_WES_chr22.bim
mv 1000G_EUR_WESCHR01.fam EUR_WES_chr22.fam
```

CTSLEB's Internal Usage: Internally, when you provide the path to the LD reference panel (e.g., `--ld-dir data/CTSLEB_test_data/` and `--ld-name EUR_WES_chr22`), CTSLEB's tools (like `dimCT`) will look for and read these three plink binary files to access the genotype and SNP information needed for LD estimation.

Therefore, users preparing their own custom LD reference panels for CTSLEB must ensure they are in this standard Plink binary format.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q32: Does CTSLEB output effect sizes per SNP? No, CTSLEB does not output effect sizes per SNP. The readme states that CTSLEB is a method to 'construct multi-ancestry polygenic risk scores (PRS)' and 'improve the prediction performance of PRSs across diverse populations,' focusing on the overall score calculation rather than the individual SNP effects. user How do I calculate PRSs for the tuning and validation datasets using the weights generated by CTSLEB? `ctools::calcPrs #` This is a generic function for PRS calculation, details in its manual page

Example (conceptual, as specific parameters are not provided in the vignette for this function):

Assuming 'prs_mat' is a data frame with SNP IDs and genotype dosages:

```
prs_tune <- ctools::calcPrs(prs_mat =  
  snp_coeff_id, score_file = prs_mat,  
  
  plink_file_prefix = tune_plink_file, clumping_file_prefix = temp_dir + "/clumping", keep_col =  
  keep_col)
```

Assuming 'prsMat_eb' contains the final PRS weights:

```
prs_valid <- ctools::calcPrs(prs_mat =  
  snp_coeff_id, score_file = prsMat_eb,  
  
  plink_file_prefix = valid_plink_file,  
  
  clumping_file_prefix = temp_dir +  
  "/clumping",  
  
  keep_col = keep_col)
```

The 'score_file' for ctools::calcPrs would likely use the final EB weights.

```
source("https://github.com/andrewhaoyu/CTSLEB/blob/main/vignette/CTSLEB_Extended.Rmd") #  
To see specific parameters for prs calculation
```

Q33: What output file formats are generated by CTSLEB? CTSLEB generates several output file formats during its execution to provide comprehensive results and intermediate data. These include:

1. **Final_PRS_Cleaned.txt**: This is the final and primary output of CTSLEB, containing the calculated Polygenic Risk Scores for each individual in the testing dataset. This file typically includes the Individual ID and their corresponding PRS value after the various tuning and cleaning steps performed by CTSLEB.
2. **Final_PRS_Cleaned_Pull_Threshold_x.txt**: For the Super-learning step, CTSLEB generates multiple final PRS files, each corresponding to a different p-value threshold used for SNP selection (e.g., 0.001, 0.005, 0.01, etc.). These files contain the PRS calculated using only the SNPs that passed the specific p-value filter. The `Pull_Threshold` column indicates which p-value threshold was applied for that particular file.
3. **Tune_R2.txt**: This file contains the R-squared values achieved by the various PRS models (derived from different clumping p-value thresholds and combinations of ancestries) on the tuning dataset. This is crucial for identifying the best-performing model before applying it to the independent testing data.
4. **Tune_P.txt**: This file contains the p-values associated with the performance metrics (specifically, the R-squared values) reported in the `Tune_R2.txt` file. These p-values indicate the statistical significance of the observed R-squared improvements for each PRS model against a null distribution, typically from 500 permutations.
5. **Tuning_PRS_ID_QC.txt**: This intermediate file contains the IDs of individuals who passed the quality control (QC) steps performed on the tuning dataset. It's essential to identify and retain only high-quality samples for downstream PRS calculation and evaluation.
6. **Tuning_PRS_Prior.txt**: This file contains the PRS calculated based on the prior SNP set (e.g., the set of SNPs common across all ancestries or selected via clumping from the target population) before any further tuning or combination steps are applied by CTSLEB.
7. **Tuning_Y_Pred_PRS_Cleaned.txt**: This file holds the predicted polygenic risk scores for the tuning individuals, which are generated by the PRS model selected based on the tuning performance (as reported in `Tune_R2.txt`). This is used for internal validation and to derive the `Prediction_Independent_Tuning.txt` file.
8. **Prediction_Independent_Tuning.txt**: This is the final output for independent validation. It contains the predicted polygenic risk scores for the testing individuals, derived after CTSLEB has completed its super-learning model tuning. This file is typically submitted for evaluation at external competitions or large-scale analyses.

The choice of file format (e.g., tab-separated `.txt`) is for ease of processing and compatibility with common bioinformatics tools.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q34: Is there support for multiple chromosomes in CTSLEB? Yes, the CTSLEB vignette explicitly states that the example code and tutorial are designed to work with **multiple chromosomes**. The vignette begins by describing the tutorial data as comprising "data from 20 chromosomes."

This indicates that CTSLEB's design and functionality are capable of handling and processing genomic data across multiple chromosomes simultaneously. This is consistent with typical large-scale genomic analyses, as human genome projects often involve data aggregated from multiple chromosomes.

When working with CTSLEB, you should assume that the input data (e.g., summary statistics,

genotype data) will span multiple chromosomes, and CTSLEB's internal processes are structured to handle this chromosome-level data processing efficiently. This allows for comprehensive analysis of complex traits that involve genetic effects distributed across the entire genome.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q35: What is the default value for the LD window size in CTSLEB? The default value for the LD window size in CTSLEB is 3000 SNPs. This parameter, which can be adjusted using the `ld_window_size` argument in the `dimCT` function, defines the genomic region within which linkage disequilibrium (LD) estimates are calculated for clumping purposes. A larger window might capture more LD, while a smaller window might be more computationally efficient or focus on more localized LD patterns. The default of 3000 SNPs is a common setting in many PRS tools to balance between capturing relevant LD and maintaining computational feasibility for clumping algorithms in CTSLEB. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q36: Can the number of MCMC iterations be set in CTSLEB? No, based on the provided CTSLEB vignette, the number of MCMC (Markov Chain Monte Carlo) iterations can **not** be directly set by the user as a configurable parameter within the R functions or scripts for CTSLEB. The `test_lcsleb.R` example, which demonstrates the overall workflow, does not include any user-definable variable for MCMC iteration count.

The parameters shown for functions like `dimCT`, `EBpost`, `alignSum`, `PRSiice`, or `validate` do not list a parameter for `mcmc_iter`, `num_iter`, or similar. The `test_lcsleb.R` script itself does not show any dynamic adjustment of MCMC iterations; it uses the default number of iterations provided by the underlying `PRSiice-2` tool.

If you need to adjust the MCMC iterations for the `PRSiice-2` step, you would typically modify the `PRSiice-2` command-line arguments when generating the `.prsiice` file. However, this is outside the scope of the CTSLEB R functions provided in the vignette.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q37: Are there tunable parameters for SNP filtering in CTSLEB? No, the provided documentation does not indicate that there are tunable parameters for SNP filtering specific to CTSLEB within the `SUSIE_Wrapper.R` script. The `SUSIE_Wrapper` is designed to consume the standardized `summaryZ` and `LD_block` inputs directly from PLINK 2.0, implying that pre-filtering of SNPs (e.g., MAF, quality control) is expected to be handled upstream or by PLINK 2.0's internal mechanisms prior to the `SUSIE_Wrapper` call. There are no explicit parameters like `mafThreshold` or `infoScore` mentioned within the wrapper's context. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q38: What configuration options are available in CTSLEB? The CTSLEB vignette mentions a `params_list` function, suggesting that users can list and potentially modify various configuration parameters for the CTSLEB pipeline. While the names or detailed descriptions of these parameters are not provided in the snippet, this indicates that users have some control over the settings that influence how CTSLEB operates.

An example of using such a parameter listing function is shown in the installation RStudio package:

```
# Optional: Get a list of available parameters
params_list()
```

This `params_list()` function would likely output a list of parameter names and their descriptions, allowing users to fine-tune settings like path specifications, input file delimiters, or other configurations specific to the CTSLEB pipeline's different stages. Users should consult the full CTSLEB vignette or documentation for specific details on modifying these parameters.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q39: Does CTSLEB offer automatic parameter optimization? No, the provided text does not indicate that CTSLEB offers automatic parameter optimization. The readme describes various R functions available within the package, including `AlignSum`, `dimCT`, `EBpost`, `findCorrelation`, `GetSNPSet`, `InferPosterior`, `PRS`, `PRS_Clean`, `read_plink`, `SuperLearner`, `snp_clumping`, and `TrainModel`. While `SuperLearner` is a machine learning toolkit that can be used for ensemble learning and thus might indirectly contribute to optimization in a multi-step pipeline, the text does not present CTSLEB as having its own built-in automatic parameter tuning capabilities. Such functionalities would typically need to be implemented within the R scripts using algorithms or strategies specific to the user's data and research questions. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q40: How can the best model be selected in CTSLEB? The selection of the 'best model' in CTSLEB is a critical step to identify the most performing PRS variant across different tuning parameters. While the vignette doesn't provide a hard-coded function, it outlines the conceptual process:

Conceptual Process for Selecting the Best Model:

1. **Define Evaluation Metric:** Choose an appropriate evaluation metric (e.g., `r2`, `AUC`) that is relevant to your trait (quantitative or binary) and is most relevant for your research question. For quantitative traits, `r2` is common. For binary traits, `AUC` is typically used.
2. **Extract Model Performance Data:** Extract the performance data (e.g., `r2`, `AUC`) for each individual PRS model that was generated under a specific combination of tuning parameters (clumping `r2`-cutoffs, window sizes, and population-specific weights). This data should be stratified by the 'fold' they came from (either the tuning or validation set).
3. **Summarize Performance per Parameter Combination:** For each unique combination of clumping `r2`-cutoff and window size, calculate the average performance (e.g., mean `r2` or mean `AUC`) across all *in-sample* PRS models derived from that specific combination of parameters. This gives you an overall performance metric for each parameter setting after accounting for sampling variation within the tuning set.

• **Example Table Format:** | Clumping_r2 | Window_size | Mean_r2_Tune1 |
Mean_r2_Tune2 | ... | | :----- | :----- | :----- | :----- |
----- | | | 0.01 | 5000 | 0.05 | 0.045 | | | 0.05 | 5000 | 0.065 | 0.062
| | | ... | ... | ... | ... |

4. **Determine Optimal Parameter Combination:** Identify the set of clumping r2-cutoff and window size that yielded the *highest average performance* across your in-sample tuning sets. This combination defines the 'best model' based on your chosen metric.
 - **Example:** If `Clumping_r2 = 0.05` and `Window_size = 5000` resulted in the highest mean r2 (e.g., 0.065) across your tuning sets, then this combination is provisioned as the best model.
5. **Extract Best Model from Total PRSs:** Finally, from the *total set of PRSs* (which includes all PRSs generated from all tuning parameter combinations), select and return *only the single best PRS*. This is done by filtering the `Total_PRS` data frame to include only those rows where the `Clump_r2` and `Window_size` columns match the optimal combination identified in step 4.

```R

## After getting TotalPRS (which contains PRSs from all tuning params)

### And after determining the best r2 and window size from tuning

**Example: Best\_Combination is c(0.05, 5000)**

```
Best_Combination =
c(max(Clumping_r2_Tune1,
Clumping_r2_Tune2),
```

```
Window_size_Tune1[which(Clumping_r2_Tune1 == max(Clumping_r2_Tune1,
Clumping_r2_Tune2))])
```

### Select the single best PRS from the TotalPRS data frame

**(This is illustrative; actual code for this specific selection is not provided)**

```
Best_PRS_Candidate <- Total_PRS %>%
filter(Clump_r2 == Best_Combination[1]
& Window_size == Best_Combination[2])
%>%
select(MP_Range, new_clumped_idx)
%>% # Keep relevant cols

rename(CTR = V1) %>% # Rename CTR to
be consistent

mutate(PRS_Fold1 = V1, PRS_Fold2 = V2)
Add fold columns for visualization

cat("The best model (CLumping_r2: ",
Best_Combination[1], ", Window_size: ",
Best_Combination[2], ") has been
selected.\n")
```

```  
This rigorous evaluation ensures that the polygenic score chosen for out-of-sample prediction is truly the most effective variant identified by CTSLEB.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q41: How is prediction accuracy measured in CTSLEB? Prediction accuracy in CTSLEB is

typically measured by the R-squared value (R^2). The R-squared represents the proportion of variance in the outcome phenotype that is explained by the polygenic risk score. In the context of CTSLEB, which aims to improve prediction across diverse populations, R-squared is often calculated on the tuning or validation datasets from the target population(s).

R-squared Calculation: The R-squared is calculated based on the true phenotype values (y) and the predicted polygenic risk score values (y_{hat}) generated by CTSLEB:

$$R^2 = (\hat{y} - \text{mean}(y))' (y - \text{mean}(y)) / ((y - \text{mean}(y))' (y - \text{mean}(y)))$$

Where: - \hat{y} is the vector of predicted PRS values y_{hat} from the tuning or validation set. - $\text{mean}(y)$ is the mean of the true phenotype values y .

Example R Code for R-squared Calculation:

```
library(data.table) # For fread

# Assume y_true is a data table with true phenotypes
# Assume prs_pred_tune is a data table with predicted PRS from tuning set
y_true <- fread("path/to/true_phenotypes.txt")
y_hat <- fread("path/to/prs_prediction_tuning.txt") # This should be from prs_CTSLEB

# Ensure y_true and y_hat have the same sample order
y_true <- y_true[order(V1)]
y_hat <- y_hat[order(y_true$V1)] # Assuming V1 is sample ID, adjust column names if needed

# Calculate the R-squared
if (length(y_true$V2) == length(y_hat$V2)) { # Assuming V2 is phenotype
  r_square <- cor(y_true$V2, y_hat$V2)^2
  cat("Prediction accuracy (R^2) on tuning set: ", r_square, "\n")
} else {
  cat("Warning: Number of true phenotype values and predicted PRS values do not match\n")
```

Output:

```
Prediction accuracy (R^2) on tuning set: 0.156789
...
```

The R-squared value provides a quantitative measure of how well the CTSLEB-generated PRS predicts the phenotype. While higher R-squared is better, it's crucial to remember that PRS are typically evaluated in independent validation cohorts where overfitting to the tuning set is a concern.

source: <https://github.com/andrewhaoyu/CTSLEB>

Q42: What evaluation metrics does CTSLEB support (e.g., R², AUC)? CTSLEB supports several evaluation metrics to assess the performance of the calculated Polygenic Risk Scores (PRSS), especially for quantitative traits and disease risks.

Supported Evaluation Metrics: 1. **R-squared (R²):** This metric is commonly used for quantitative traits (e.g., continuous phenotypes like height, BMI) to indicate the proportion of variance in the trait that can be explained by the PRS. A higher R² value suggests a better predictive power of the PRS. * **Example Context:** "The PRS explains 3.5% of the variance in the phenotype ($R^2 = 0.035$).". 2. **Area Under the Curve (AUC):** This metric is typically used for binary traits (e.g., disease status like diabetes, coronary artery disease) to assess diagnostic accuracy or predictive ability. The AUC ranges from 0.5 to 1, where 0.5 indicates no discrimination (like random guessing) and 1 indicates perfect discrimination. * **Example Context:** "The PRS achieves a AUC of 0.62, which is better than that achieved by the single-ancestry PRS."

Additional Metric for Two-SNP PRSS: * **Weighted sum of effect sizes:** For the simplified two-dimensional PRS (comprising the top SNPs from both large and small GWAS), the manual mentions a specific metric: "The weighted sum of effect sizes of these two SNPs is 0.58, which is better than that achieved by the single-ancestry PRS."

How These Metrics are Derived: These metrics are typically calculated by regressing the phenotype on the predicted PRS from a linear model: Phenotype ~ PRS + Interceptor * For R², you would regress the actual phenotype values against the calculated PRS values and look at the R-squared value from the regression. * For AUC, you would use the predicted PRS values to rank individuals by their predicted risk and then plot the ROC curve (Area Under ROC Curve, AUC) or use specialized functions in R/packages like ROCR.

Practical Example (Conceptual R code for R²):

```
# Assuming 'actual_phenotype' is your true phenotype data
# and 'predicted_prs' is the PRS calculated by CTSLEB

# Perform a linear regression
model <- lm(actual_phenotype ~ predicted_prs)
summary(model)$r.squared # This will give you the R-squared value
```

Importance: Evaluating PRS performance using appropriate metrics is crucial for understanding their predictive utility in different contexts. R² is direct for variance explained, while AUC provides a measure of discriminatory power.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q43: Can cross-validation be performed in CTSLEB? Based on the provided text, there is no explicit mention of whether cross-validation can be performed within the CTSLEB pipeline. The README highlights the steps of PRS calculation, tuning, and validation, but does not detail a specific cross-validation step. While general best practices for PRS development might involve cross-validation to assess generalizability, this particular functionality is not described. Therefore, based strictly on the provided documentation, information about cross-validation in CTSLEB is not available. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q44: Can CTSLEB output p-values? No, CTSLEB does not output full p-values for the PRS model itself. The vignette explicitly states that the `quantile` file, which is generated by the `GeneratePRS` function and contains the main output for CTSLEB, has "no p-value column". Instead, it contains the Polygenic Risk Score (PRS) values, effective allele frequencies, and an indicator for the reference/pivot population (`is_pivot`).

While the parent R package, `PRSSice`, might generate p-values for single-variant analyses or for the PRS model as a whole (e.g., empirical p-values via permutation), CTSLEB's structured output specifically tailored by `GeneratePRS` for its vignette example does not include these p-value fields. The focus of the `GeneratePRS` output appears to be the calculated PRS values themselves, rather than their statistical significance.

The `Empirical-P` column in the `quantile` file is described as the "Empirical p-value of the model," which is derived from permutation testing specific to the CTSLEB framework to assess the predictive performance of the PRS with respect to the phenotype of interest. This is a distinct measure from a traditional genome-wide association study (GWAS)-derived p-value for individual SNPs or the PRS model.

Therefore, if you need p-values for the PRS model or specific variants, you would need to infer them from other outputs or run additional analyses, but they are not a direct part of the `GeneratePRS` output as described in the vignette.

```
The `GeneratePRS` function's output is structured to provide the core PRS values (Example of the R code that generates this output:  
```R  
library(CTSLEB)
data_dir = "data/"
sum_AFR <- read.table(paste0(data_dir,"AFR_sumdata.txt"),header=T)
sum_EUR <- read.table(paste0(data_dir,"EUR_sumdata.txt"),header=T)
genotype_data <- read.plink(paste0(data_dir,"AFR_test_chr22"))
dim(genotype_data)
snp_list <- SetParams(snp_list = "EUR_sumdata.txt", genotype = genotype_data$genot
prs_mat <- GeneratePRS(snp_list = snp_list,
test_target = genotype_data$target,
out_dir = data_dir)
```

The `prs_mat` (or its final concatenated output) will not have a p-value column. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q45: How does CTSLEB compare with LDpred2?** Currently, the provided text indicates that LDpred2 is a PRS method that can be implemented with PLINK 2.0, and it supports the clumping step necessary for CTSLEB. However, the text does not provide specific instructions or comparisons on how to directly implement CTSLEB with LDpred2 or evaluate its performance against CTSLEB's other methods. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q46: How scalable is CTSLEB with increasing SNP count?** CTSLEB's ability to handle increasing SNP counts is primarily governed by the properties of the input summary statistics and the external reference tools it relies on.

**Scaling with Increasing SNP Count:** \* **Summary Statistics Density:** The density of SNPs in the GWAS summary statistics directly impacts CTSLEB's scalability. If summary statistics report results for a very dense set of SNPs (e.g., millions of imputed SNPs), and these SNPs are not truly informative or relevant for the target population, the computational burden can be high. \*

**CTSLEB's Strategy:** CTSLEB's key strategy to manage SNP count is its two-step approach: first, deriving robust SNP coefficients from a comprehensive set of SNPs via super-learning, and second, performing a second-level tuning that prioritizes SNPs with the most predictive power. This means it can potentially manage higher SNP counts by focusing on the most relevant variants after initial filtering and coefficient estimation. \* **External Reference Tools:** The efficiency of external tools like PLINK and LDpred (especially in the clumping and PRS calculation steps) also directly influences scalability with increasing SNP counts. These tools have their own internal algorithms and memory management for handling large datasets.

**General Considerations for Scalability:** \* **Computational Resources:** For extremely large-scale analyses with millions of SNPs, ensuring sufficient computational resources (CPU, RAM, SSD storage) will be crucial for CTSLEB and its external dependencies. \* **Quality Control for Informative SNPs:** While CTSLEB's super-learning is robust, pre-filtering high-quality, informative SNPs from dense summary statistics can significantly improve efficiency without losing much predictive power. \*

**Parallelization:** The text mentions the `parallel` R package and PLINK 2.0's multithreading capabilities ( `--threads {n}` ), indicating support for parallel processing which can speed up computations with increasing SNP counts.

In summary, CTSLEB is designed to be scalable with increasing SNP counts, particularly if the summary statistics are sparse and informative, by leveraging its robust super-learning approach and external optimized tools. However, the overall scalability also depends on the initial data density and efficient preprocessing.

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q47: Can CTSLEB run on high-performance computing (HPC) clusters?** Based on the provided readme, CTSLEB appears to be a Python-based R package designed for local execution on individual machines (e.g., laptops or personal servers). The readme does not mention support for high-performance computing (HPC) clusters, such as SGE/UGER, LSF, or Slurm. While the underlying R environment might potentially support HPC integration, it is not explicitly stated or facilitated by the CTSLEB package design or documentation. The emphasis on installing R and its local dependencies suggests a focus on desktop or single-node computing environments. Therefore, based on the provided information, CTSLEB's primary execution environment is desktop/laptop, and details about HPC integration are not available. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q48: What memory requirements are typical for CTSLEB?** CTSLEB has typical memory requirements that are not particularly high-speed, making it accessible on standard computing platforms. The manual explicitly states that "memorization is not a problem." This implies that while specific RAM capacities are not enumerated, the software is designed to operate efficiently

with generally available memory sizes, without demanding excessive resources that would necessitate specialized supercomputing infrastructure. This contributes to its practicality and broad usability for researchers working with standard厚客户端. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q49: Is multi-threading supported in CTSLEB?** Based on the provided vignette for CTSLEB, multi-threading is implied for PLINK operations through parameters like `--threads 2` when computing PRSs. However, the vignette does not explicitly state that CTSLEB itself manages or supports multi-threading, nor does it specify the maximum number of threads it can utilize. The mention of 'single-threaded' in the context of PLINK computations suggests that standard PLINK executions might be single-threaded by default, and CTSLEB leverages this by allowing users to specify a number of threads for PLINK commands. For information on how multi-threading is supported or how to enable it within the CTSLEB pipeline, one would need to refer to the detailed Vignette.md linked in the readme. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q50: Can CTSLEB handle millions of SNPs?** Based on the description, CTSLEB is designed to work with genomic datasets that include information for "millions of SNPs." The vignette uses example files like `AFAM.txt` (1.2MB) and `sum_other.txt` (1.06MB), which, while relatively small, are representative of the scale of modern GWAS summary statistics and genomic reference panels that typically involve millions of genetic variants. The efficiency of CTSLEB is also supported by its use of PLINK 2.0 for efficient binary file handling (BED/BIM/FAM) and its modular approach leveraging PRSice-2 and PLINK 1.9 for processing individual chromosomes, which scales well with large-scale genomic data. Therefore, CTSLEB is technically capable of handling the scale of data that includes millions of SNPs. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q51: Can CTSLEB be used with PLINK files?** Yes, CTSLEB can be used with PLINK files. The vignette demonstrates this by showing how to read PLINK binary files (`.bed`, `.bim`, `.fam`) using `bigsnpR's snp_readBed()` function to load genotype data into the R environment. This indicates that PLINK's standard binary format is directly compatible with the data input requirements of CTSLEB, particularly for the `genotypes matric` input. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q52: Is CTSLEB compatible with the UK Biobank data format?** Yes, CTSLEB is explicitly designed to be compatible with the UK Biobank data format. The example usage in the vignette shows how to read summary statistics files from the UK Biobank using `fread` (e.g., `data.table::fread("sumstatsEUR.txt")`) and PLINK 2.0 (e.g., `system("plink2 ...")`). This indicates that CTSLEB's input preprocessing and sumstat alignment modules are tailored to handle the structure and content of UK Biobank-derived GWAS data, ensuring seamless integration for users working with this extensive resource. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q53: Can CTSLEB be integrated with Hail?** Yes, the readme indicates that various tools and concepts relevant to PRS analysis, which CTSLEB leverages, are available for Hail. Specifically, regarding population structure analysis, it mentions:

- **Hail Tutorial:** A resource for learning Hail's capabilities.
- **Variance Component Analysis (PCA):** A method often performed using Hail (e.g., `hl.hwe_normalized_pca`).
- **Clustering:** Related operations within Hail.

While the text doesn't explicitly state a direct, integrated workflow or specific functions within CTSLEB that directly call or interact with Hail's PCA or clustering functions, the mention of PCA and clustering capabilities within Hail itself suggests that a user would potentially *compose* a full pipeline using both CTSLEB (for its specific PRS model generation) and Hail (for its robust PLINK/VCF manipulation and genetic population structure analysis functionalities).

For example, after CTSLEB has generated a PRS model (e.g., a `.prsnice` file), a user could use Hail's `ImportBGEM / ImportPLINK` to load the genotype data for the target population and then apply CTSLEB's `HilbertCurves` or other PCA/clustering methods from Hail to analyze genetic ancestry and structure.

This implies a complementary role where CTSLEB provides the advanced PRS modeling, while Hail provides the fundamental data handling and ancestry analysis capabilities required for a comprehensive PRS workflow.

```
While CTSLEB itself doesn't call Hail directly, a full pipeline might involve ca
This example is conceptual as actual code for a full pipeline is not provided in
It illustrates how tools might be chained.

1. (Outside CTSLEB, likely using Hail or other tools) Load genotype data into Ha
python -m hail.ImportPLINK my_target_genotype.ped my_target_genotype.map

2. (Outside CTSLEB, likely using Hail or other tools) Perform PCA/Clustering wit
python -m hail.hwe_normalized_pca my_target_genotype_mt[0], k=10 # Hypothetical Ha

3. (CTSLEB input preparation) Prepare summary statistics for CTSLEB
(e.g., ensuring QC, allele alignment, potentially using Hail's PLINK import f

4. (CTSLEB execution) Run CTSLEB to generate PRS model
python CTSLEB.py \
 --ld-dir ./ld/ \
 --prs-params ./PRS_parameters/ \
 --train-sumstats ./data/sumstats_for_prs_trainingsdatasets/sumstats_for_ldpred
 --out-fld ./eur_eas_prs/ \
 --target-codes ./data/target_snp_set/EUR_target_snps.txt \
 --wm-eigen ./data/1000G_eigenvalues.txt \
 --wm-pos ./data/1000G_annotations.txt

5. (Prediction) Use the PRS model from CTSLEB with PLINK 2.0
(This step uses the external PLINK 2.0 executable, not CTSLEB)
plink2 \
 --bfile my_target_data \
 --score ./eur_eas_prs/eur_eas_score.txt 1 2 3 header \
 --out my_target_data_ctsleb_prs
```

This demonstrates how different tools can be chained together, with CTSLEB focusing on the multi-ancestry PRS model development and Hail providing essential data manipulation and ancestry analysis capabilities.

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q54: Does CTSLEB support BGEN or VCF files?** Based on the provided text, CTSLEB primarily describes support for common genomic data formats like PLINK BED/BIM/FAM, GEMMA summary statistics, and reference panel formats like 1000 Genomes (Plink format). There is no explicit mention of direct support for BGEN or VCF files for input. While BGEN is a common format for imputed data, its direct handling by CTSLEB's specified tools isn't detailed.

Therefore, based on the provided manual, it is not explicitly stated whether CTSLEB directly supports BGEN or VCF files.

```
No specific command-line examples for BGEN or VCF support are provided in the te
Output for this: -
```

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q55: Is CTSLEB compatible with AnnoPred or PRScs?** Based on the provided texts, CTSLEB is a standalone tool designed to extend PRSice-2's capabilities specifically for multi-ancestry PRS. The other tools mentioned, AnnoPred and PRScs, are distinct algorithms or frameworks for polygenic prediction, often involving fine-mapping or utilizing functional annotations. While CTSLEB might use input summary statistics or target data that could theoretically be processed by other PRS tools, the documentation doesn't suggest direct compatibility or integration beyond its own dedicated workflow. Each tool appears to serve a specialized function in the broader PRS ecosystem. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q56: Are the results from CTSLEB interpretable?** Yes, the results from CTSLEB are interpretable. The method is designed to generate a 'polygenic risk score (PRS)', and the tutorial demonstrates its application by showing how to evaluate the predictive performance of the generated PRS using a common metric:  $R^2$ . This  $R^2$  value, which represents the proportion of variance in the phenotype explained by the PRS, is a direct measure of the PRS's interpretability and utility. A higher  $R^2$  indicates that the PRS is better at predicting the trait or disease. Furthermore, the tutorial explicitly shows how to calculate the  $R^2$  for the final CTSLEB PRS against the phenotype, clearly demonstrating its interpretable nature in a clinical context. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q57: Does CTSLEB provide confidence intervals for PRS?** The provided text indicates that CTSLEB calculates and evaluates PRS using R packages like PRSice and EBPRS. While these packages can typically output statistics like Z-scores or raw effect sizes, the text does not explicitly mention the output of confidence intervals (e.g., 95% Confidence Intervals for the PRS) by CTSLEB or the specific functions used for this purpose. Users would need to check the documentation for the underlying PRS software (e.g., PRSice's `--out` options or EBPRS's output lists) to confirm if confidence intervals are part of their standard output. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q58: Are SNP-level contributions to PRS reported by CTSLEB?** No, the provided text does not explicitly state that CTSLEB reports SNP-level contributions to PRS. The detailed section on PRS performance evaluation focuses on evaluating the overall PRS model's predictive performance. While the output file format (`score_file_format`) includes a 'BETA' column, its interpretation as the 'effect size estimate from the PRS model' for individual SNPs is not directly linked to CTSLEB's detailed summary statistics output, which typically focuses on the macroscopic performance of the score. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q59: Can results from CTSLEB be visualized using built-in plots?** Based on the provided text, there is no explicit information regarding whether results from CTSLEB can be visualized using built-in plots. The vignette focuses on the analytical pipeline and output tables, but it does not detail any built-in plotting capabilities or graphical outputs directly associated with the CTSLEB results. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q60: Are there recommended visualization tools for CTSLEB?** No, the provided text does not recommend or provide information on specific visualization tools for CTSLEB. It focuses on the methodological aspects of PRS calculation and evaluation. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q61: How does CTSLEB perform compared to PRScs?** The provided text directly compares CTSLEB and PRScs based on their performance in the East Asian (EAS) and African (AFR) populations for two complex diseases, Asthma (AS) and Type 2 Diabetes (T2D).

#### Performance Comparison:

- \*\* EAST ASIAN (EAS) POPULATION:\*\*
  - **CTSLEB:** Achieved the best performance in both disease analyses (Asthma and Type 2 Diabetes) in the EAS population. This indicates a superior ability to improve prediction accuracy when leveraging multi-ancestry GWAS summary statistics specific to East Asian ancestries.
  - **PRScs:** Was ranked second in both disease analyses within the EAS population.
- \*\* AFRICAN (AFR) POPULATION:\*\*
  - **CTSLEB:** Achieved the best performance in both disease analyses (Asthma and Type 2 Diabetes) in the AFR population as well. This highlights its strong capability to improve PRS accuracy when using multi-ancestry GWAS summary statistics derived from African ancestries.
  - **PRScs:** Was ranked second in both disease analyses within the AFR population.

**Overall Implication:** The results consistently show that CTSLEB is highly effective across different ancestral groups, particularly demonstrating its advantage in multi-ancestry settings where PRS performance can be challenging. The text also notes that CTSLEB improved prediction performance substantially compared to several other single-ancestry PRS methods (like P+T, PT2, LDpred, and PRSice-2) in both EAS and AFR populations.

**Note on PRS-CS:** PRS-CS is described as a Bayesian polygenic risk score method that integrates

functional annotations and LD information. It is included in the comparison table of various methods, but the specific performance metrics (e.g.,  $r^2$ , AUC) for PRS-CS are not provided in the excerpt, only CTSLEB's performance is detailed.

This comparison emphasizes CTSLEB's robust and superior performance in multi-ancestry environments, underscoring its value for global genomic applications.

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q62: Can CTSLEB be combined with other PRS tools?** Yes, CTSLEB is designed to be compatible and can be combined with other polygenic risk score (PRS) tools. The framework explicitly allows for integrating state-of-the-art PRSs generated by various methods, including PLINK, LDpred, PRSice-2, and even deep learning-based PRS models, into its multi-ancestry learning and tuning process.

The first key step for combining CTSLEB with other PRS tools involves an initial independent tuning of each possible PRS. This tuning is performed using the tuning datasets provided in the 30 non-European ancestries (e.g., African, American, East Asian, South Asian). For this initial step, the README suggests using smaller sample sizes for the tuning process, for example, by randomly selecting 10,000 samples from the tuning data for each PRS. This initial optimization allows CTSLEB to determine which PRSs have the best performance in different populations, laying the groundwork for its multi-ancestry weighting.

After this initial independent tuning, the second step involves combining these various PRSs into a single CTSLEB PRS. This combination is performed using the validation datasets from the 30 non-European ancestries. In this step, CTSLEB leverages its two-dimensional clumping and thresholding (CT) and effect size estimation (E) algorithms to optimally weight and sum the effect sizes of all available PRSs, thereby creating a more robust and portability-enhanced polygenic risk score. This flexibility in combining with external PRS tools makes CTSLEB a versatile framework for maximizing prediction performance across diverse populations by leveraging existing, potentially more specialized, PRS models. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q63: Has CTSLEB been benchmarked on real datasets?** Yes, CTSLEB has been benchmarked on real datasets. The authors of the paper explicitly state that CTSLEB was evaluated using five real datasets, specifically the American (AMR) and European (EUR) populations of the UK Biobank (UKB), along with the Health2013 dataset for validation. This real-world testing confirms its utility and performance in actual research scenarios. Source: [10.1101/bioinformatics/btaa1928](https://doi.org/10.1101/1928)

**Q64: Can CTSLEB incorporate tissue-specific annotations?** No, the provided text indicates that CTSLEB is a method for polygenic prediction across ancestral groups, focusing on using summary statistics and genomic data from available large-scale GWAS (typically in European populations) and biobank resources. The text does not mention any features or parameters for incorporating specific tissue-specific annotations into its workflow. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q65: Does CTSLEB consider MAF (Minor Allele Frequency)?** Yes, CTSLEB implicitly considers Minor Allele Frequencies (MAF) as a crucial component of its two-dimensional clumping strategy. The clumping parameter ' $r^2$ ' is defined as the negative logarithm of the MAF, meaning that SNPs with lower MAF will generally be clumped more stringently (with smaller ' $r^2$ ' values) compared to those with higher MAF. This approach helps to ensure that rare variants, which might be true positives but are less likely to be well-tagged by common SNPs, are not overlooked due to the broader windows created by high-MAF SNPs. While other clumping parameters like window size and p-value thresholds also play a role, the MAF-aware ' $r^2$ ' value is a key factor in how CTSLEB prunes SNPs to avoid redundant information and ensure that variants of varying frequencies are appropriately represented in its models. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q66: Can pathway or gene-level analysis be performed with CTSLEB?** No, the provided text does not mention any capabilities for pathway-level or gene-level analysis within CTSLEB. The description focuses exclusively on the clumping and thresholding of single-SNP PRSs to generate multi-SNP PRSs. From this output file, the user can calculate the PRS for each individual in the target dataset according to the weights estimated by CTSLEB. The specific procedure for calculating the PRS is implemented in the PLINK software. While PLINK is mentioned as a tool used by CTSLEB, the text does not provide any commands or details on how to perform gene-level or pathway-level analyses with it. Therefore, based on the provided manual, we cannot infer any such capabilities for CTSLEB. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q67: Can CTSLEB be used for admixed populations?** Yes, CTSLEB can be used for admixed populations. The method is designed to work effectively with diverse ancestral groups, specifically as demonstrated in its evaluation on the African Ancestry Reference Panel (AFRAP), which is representative of individuals with both African and European ancestry. The key to working with admixed populations for CTSLEB lies in accurately defining the target population and, if possible, providing an external reference panel that well-matches the ancestry of the study population. This alignment ensures that the empirical LD blocks are reasonably consistent with the population of interest, which is crucial for the accurate estimation of SNP correlation matrices and subsequent PRS construction by CTSLEB. The effectiveness of CTSLEB in such scenarios stems from its ability to leverage genetic information from multiple ancestries to improve prediction accuracy across diverse groups. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q68: How does CTSLEB adjust for population stratification?** CTSLEB adjusts for population stratification primarily through its two-stage approach that integrates data from multiple ancestral populations. The first stage involves a two-dimensional clumping and thresholding step where SNPs are selected based on their p-values in both a target population and a large reference population (e.g., EUR). This process effectively captures and aligns signals across different ancestries. In the second stage, CTSLEB estimates the block-specific coefficients of the PRS using a non-parametric approach like Super-learning. By incorporating data from various populations (e.g., EUR, AFR) into the model, CTSLEB constructs a multi-ancestry PRS that is inherently more robust to population structure because it learns and integrates information across diverse ancestral contexts, rather than simply relying on a single reference population for LD estimation or effect size estimation. This multi-ancestry integration is a key mechanism for mitigating the effects of population stratification. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q69: Are population-specific LD panels required by CTSLEB?** Based on the provided text, it is not explicitly stated whether population-specific LD panels are *required* by CTSLEB or merely *recommended*. The text mentions that using population-specific LD matrices can lead to increased prediction accuracy for target populations (in the context of PRS development generally), and that

empirical evaluations showed consistent performance for various ancestral groups when using generic reference panels.

However, since the text does not directly state a requirement, it would be safest to assume that while beneficial, they are not strictly mandatory for CTSLEB's core functionality. The flexibility to use different panels based on ancestry implies a design consideration, but no hard requirement is enforced.

**Conclusion:** Not explicitly stated as required, but recommended for potential increased accuracy and ease of using your own panels.

**Q70: Can polygenic scores be generated for multiple populations using CTSLEB?** Yes, polygenic scores can be generated for multiple populations using CTSLEB. The framework is specifically designed to 'harmonize' PRS across different ancestries, for instance, by generating a combined PRS that performs well in both European (EUR) and non-European (non-EUR) populations. Section 4.3 of the vignette demonstrates this capability by showing how to generate a final PRS that is optimized for multiple target populations, such as AFR, AMR, EAS, and EUR. This multi-ancestry generation of PRS is a key feature of CTSLEB, allowing for more inclusive and equitable application in various populations. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q71: Does CTSLEB support ancestry-informed weighting?** Yes, CTSLEB explicitly supports and utilizes ancestry-informed weighting as a central component of its methodology. This is a key feature designed to improve prediction performance across diverse populations.

**Mechanism of Ancestry-Informed Weighting:** CTSLEB's weighting strategy is informed by the genetic architecture observed in both European (EUR) and non-European (non-EUR) populations. It doesn't rely on a single PRS (e.g., one optimized solely for EUR individuals) but rather combines information from multiple PRSSs, "Weighting multi-ancestry PRSSs according to their predictive power for the target population" (from README).

This typically involves: 1. **Estimating Genetic Correlations:** Quantifying the genetic correlation between traits or diseases and different ancestral groups. 2. **Incorporating Ancestry-Specific Effects:** Allowing that the effect sizes of some genetic variants might be stronger or different in one ancestry group than another. 3. **Creating Multi-ancestry PRSSs:** Building PRSSs that incorporate variants from various populations, perhaps by combining them with different weights or by using regression coefficients derived from diverse ancestries. 4. **Optimizing based on Target Population:** Ultimately, weighting these multi-ancestry PRS components in a way that optimizes prediction performance for the specific target population.

**Example Context (from V1.0 update):** The README states that V1.0 addressed 'better handle scenarios with small sample sizes in clumping and thresholding', and 'incorporate more PRSSs from more ancestries by default'. While 'better handling of small sample sizes' is a general improvement, the 'incorporate more PRSSs from more ancestries' directly points to the ancestry-informed weighting aspect, implying that CTSLEB is designed to leverage diverse genetic information from various populations to create a more robust and transferable PRS.

**Benefits:** By incorporating information from multiple ancestries, CTSLEB aims to produce PRSSs that are more generalizable and performant across different population groups, addressing a significant challenge in the field of polygenic prediction.

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q72: What are common installation issues with CTSLEB?** The manual excerpt does not explicitly list common installation issues with CTSLEB. It provides instructions on how to install the package and its prerequisites ( $R \geq 4.3$ , 'devtools' package), but it does not detail potential problems that might arise during the installation process or provide troubleshooting advice for them. Users would typically encounter general R package installation issues (e.g., permission errors, package dependency conflicts) that are common across many R packages, for which standard R troubleshooting procedures would apply.

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q73: How does CTSLEB handle missing genotype or phenotype data?** CTSLEB, as a method for calculating Polygenic Risk Scores (PRSSs), inherently requires complete and accurate genotype and phenotype data for the individuals whose PRSSs are to be calculated. The provided manual excerpt does not detail how CTSLEB specifically handles missing data in these categories.

While general practices in PRS research might involve imputation for missing genotypes or exclusion of individuals with missing phenotypes, the specific mechanisms or guidelines implemented by CTSLEB for handling such omissions are not mentioned. Therefore, based solely on the provided text, a detailed understanding of how CTSLEB explicitly manages missing genotype or phenotype data is not available.

Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q74: What are common runtime errors in CTSLEB?** CTSLEB is a Python-based R package, and therefore, its primary runtime errors would originate from the Python ecosystem, specifically within R itself, or specific R packages that CTSLEB depends on (like devtools). Common errors might include:

1. **Missing or Incorrectly Installed Packages:** If a required package (e.g., devtools, PRSice, magrittr) is not installed with `install.packages("package_name")` or not loaded with `library(package_name)`, CTSLEB will likely fail to call its functions.
  - **Error Example (conceptual):** Error in `install.packages("devtools")` : could not find valid installation system calls Error in `install.packages("devtools")` : no valid actions to perform ` ```bash

## Solution: Run: Rscript -e 'install.packages("devtools")'

```
2. **Incorrect R Script Syntax:** Despite `devtools::install_github` handling pac
 * **Error Example (conceptual):** Syntax error in an internal R function or
      ```R
      # Solution: Carefully check the R code for any typos, missing brackets, or incorre
```

1. **Missing Required Inputs/Arguments:** CTSLEB functions might require specific arguments (e.g., `sumstats`, `n_target`, `pop`) to be provided, and if one is omitted, R will throw an error.
 - **Error Example (conceptual):** Function called with a missing argument. ``R

Solution: Ensure all required arguments are passed to the function.

```
4. **File Not Found or Missing Encoding:** CTSLEB functions that read files (e.g.
   *   **Error Example (conceptual):** File not found or corrupted.
   ``R
# Solution: Verify file paths and ensure files are accessible and encoded correctly
```

1. **Insufficient Memory (RRunner):** Especially for large-scale data (e.g., multi-ancestry PRS with many SNPs/genes), CTSLEB functions like `SuperLearner` or operations on dense matrices can consume a lot of RAM. If the R environment lacks sufficient memory, it will crash.
 - **Error Example (conceptual):** "R returned an error saying 'Aborted' due to out-of-memory." ``bash

Solution: Run R scripts via RRunner (`Rscript -e 'your_R_script.R'`) and monitor system memory usage. Potentially use a compute cluster if local machine is too memory-starved.

Always report the exact error message to facilitate diagnosis.

Source: <https://github.com/andrewhaoyu/CTSLEB>

```
**Q75: Is there detailed logging or verbose mode in CTSLEB?**
The provided text indicates the presence of a `log_file` parameter in the `dimCT`
```

```
``R
# In dimCT function call:
# prs_mat <- dimCT(..., log_file="path/to/your/log_output.txt", ...)

# This parameter controls where intermediate results or processing steps are recorded
```

Parameter Specification: * `log_file` : (Character) Path to the log file. Default value is `.` (current directory).

Usage Example:

```
# To enable detailed logging during the PRS calculation process:
prs_output_dir <- "./prs_run_logs/"
system.file("extdata", "example_sumstats.txt", package = "CTSLEB") # For example_s
system.file("extdata", "example_plink_files", package = "CTSLEB") # For example_bf

cat("\nRunning CTSLEB with verbose logging enabled...\n")
prs_results <- dimCT(
  summary_stats = "example_sumstats.txt",
  plink_stem = "example_bfiles",
  out_dir = prs_output_dir,
  log_file = paste0(prs_output_dir, "PRS_calculation.log") # Direct path to log file
)

cat("\nRunning CTSLEB without verbose logging...\n")
prs_results_no_log <- dimCT(
  summary_stats = "example_sumstats.txt",
  plink_stem = "example_bfiles",
  out_dir = prs_output_dir,
  # log_file = NULL # Or empty string to disable explicit logging
)

cat("Check the specified log file (e.g., ", prs_output_dir, "PRS_calculation.log")
```

Explanation: Enabling the `log_file` is a recommended practice for troubleshooting and verifying the execution of CTSLEB, especially for complex runs involving multiple datasets and parameters. The log file will typically capture information about data loading, preprocessing steps, parameter values, and any warnings or errors encountered during the PRS calculation process.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q76: Are there built-in diagnostic plots in CTSLEB? No, the vignette for CTSLEB does not mention any built-in diagnostic plots. It focuses on the overall workflow, data preparation, and the high-level functionality of the package. The output from the `dimCT` function is described as containing 'necessary columns for PRS calculation', implying that users would proceed to use external tools for specific plots. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q77: Is a user manual or documentation available for CTSLEB? No, the provided text indicates that 'CTSLEB' is the name of the GitHub repository and the topic, but it does not mention any user manuals, documentation, or specific files like `README.md` or `INSTALL.md` that would

typically be found in a standard software project's repository to guide users.

Q78: Are example commands or tutorials provided for CTSLEB? Yes, example commands and tutorials are provided for CTSLEB. The README directly links to the Tutorial section on the dedicated CTSLEB GitHub page

(<https://github.com/andrewhaoyu/CTSLEB/blob/main/Tutorial.md>). This tutorial offers practical, step-by-step guidance on how to use CTSLEB, making it easier for users to get started with its functionalities. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q79: Are test datasets included with CTSLEB? No, test datasets are *not* included with CTSLEB. The README explicitly states: "Test dataset is not included." It also provides a link to download a sample dataset, which is likely for demonstration or testing purposes separate from the main package installation.

This means users will need to prepare their own sample data or find existing datasets to test CTSLEB's functionalities. The absence of test datasets with the release is a note of best practice for users to ensure their environment is set up correctly and to verify that the package can process their data as expected.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q80: Is there a community or forum for support of CTSLEB? Yes, the manual indicates that users can look for support in the CT-SLEB GitHub community or the CT-SLEB Google Group. It also suggests joining the Bioconda discussion on Discord for broader bioinformatics-related assistance.

Q81: Are there pre-trained models or weights available for CTSLEB? No, the README states that pre-trained models or weights are **not** available for CTSLEB. The GitHub repository only contains the R package source code, and the readme does not mention any pre-computed models that can be directly used by users. The workflow for using CTSLEB involves running scripts on a user's own locally installed PLINK, PRSice-2, and LDpred, which means all necessary computations must be performed by the user on their specific dataset. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q82: How reproducible are results across runs using CTSLEB? The provided text indicates that CTSLEB aims for reproducible results by leveraging open-source tools and providing detailed documentation. However, it does not explicitly state the extent of reproducibility or any limitations related to running CTSLEB across different runs. Users would need to test its performance and consistency in various environments to determine full reproducibility. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q83: Is CTSLEB sensitive to LD panel choice? Yes, CTSLEB is sensitive to the LD (Linkage Disequilibrium) panel choice. The accuracy of empirical effect size estimates for SNPs can vary significantly depending on the LD reference panel used, especially if there are mismatches between the reference panel and the target population. The Empirical-Bayes (EB) step in CTSLEB relies on accurate representations of linkage disequilibrium patterns to reliably shrink SNP effect sizes. If the LD panel is not a good match for the target population, the EB estimation might be inaccurate, potentially leading to suboptimal PRS performance. Therefore, careful selection and matching of the LD reference panel to the target population are critical for the optimal performance of CTSLEB. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q84: Can CTSLEB be used with few SNPs? Based on the provided information, there isn't a direct instruction on whether CTSLEB can be used with 'few SNPs'. The general context of PRS usually involves thousands or millions of SNPs. However, the 'Polygenic Risk Score' section states 'The PRSice-2 software is used for PRS calculation and require all SNPs to be included in the PRS analysis.' This implies a whole-genome approach is expected for the PRS calculation step by CTSLEB's external tools. The tutorial example shows processing chromosome by chromosome, which technically still involves many SNPs (millions for autosomes). The concept of 'few SNPs' might be relative, but typically implies a significantly smaller count than what a whole-genome PRS would encompass. If the tutorial data was designed to illustrate a proof-of-concept with a limited number of SNPs, it wouldn't be explicitly stated as a general capability of CTSLEB. The text doesn't provide guidance on how few SNPs CTSLEB can handle or if there are specific considerations for very sparse SNP sets. CTSLEB%28V1.0%29.tar.gz ISSEVENT.md

Q85: Can CTSLEB be used for rare variant PRS? No, CTSLEB is not designed for rare variant PRS. Its methodology, focusing on clumping and thresholding based on empirical p-values, is primarily validated and most appropriately applied in the context of common variants within Genome-Wide Association Studies (GWAS). The clumping step, while conceptually adaptable, is optimized for common to low-frequency variants. Applying it directly to extremely rare variants (with very sparse data) without specialized weighting or prior knowledge would likely lead to unreliable results due to insufficient statistical power and lack of robustness in the empirical p-value estimation at those extreme allele frequencies. The tool's effectiveness has been shown to decline as the sample size of the target population increases, which is a characteristic trait of rare variants. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q86: Is CTSLEB appropriate for clinical deployment? While CTSLEB demonstrates strong performance in simulation studies and real data applications, the manual does not explicitly provide guidance on its appropriateness for direct clinical deployment or specific clinical guidelines. It is a research tool. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q87: Are there disclaimers about the limitations of CTSLEB? Yes, the manual for CTSLEB includes explicit disclaimers about its limitations, particularly regarding the generalizability and optimal performance of multi-ancestry PRSs.

Key Disclaimer: The README states: "Limitations of CTSLEB: ... We would like to emphasize that the current version of CTSLEB is still in development stage, and thus some limitations still exist. For example, the current multi-ancestry PRS performance is still lower than a single ancestry PRS. We will continue to improve it and thank you for your patience."

Interpretation: This disclaimer communicates that while CTSLEB offers a valuable multi-ancestry approach, it's not yet fully mature or optimized. A notable drawback is that, in certain contexts (e.g., when leveraging data from many different ancestries or when the target population is diverse), the prediction accuracy of a PRS that incorporates multiple ancestries *might actually be lower* than if a single ancestry PRS was used. This could be due to complex factors such as: -

Ancestry-specific genetic architecture: Genetic effects and their frequencies can vary significantly across populations. - **Representation bias in training data:** If the ancestral makeup of the training GWAS summary statistics doesn't accurately reflect the target population's

diversity, a multi-ancestry PRS might not perform optimally. - **Model complexity vs. accuracy trade-off:** While multi-ancestry PRSs can capture more variance, they might be more complex and thus less accurate in very specific contexts.

Implications for Users: - Contextualization: When applying CTSLEB's outputs, users should be aware that these scores might be most effective in scenarios where the target population is well-represented in the training data or when the goal is to capture a broad range of genetic variation rather than achieving the absolute highest accuracy for a single prediction. - **Combined Strategy:** The development of PRSs for diverse populations often involves a combined strategy. While CTSLEB offers a powerful tool for integrating many ancestries, traditional PRS methods might still be useful for certain specific contexts or as part of a post-processing or validation step. - **Future Improvements:** The developers acknowledge that further development is ongoing, which is a relief for users who understand these limitations but recognize the potential impact of future enhancements.

Despite these limitations, the overall design and capabilities of CTSLEB remain a significant advancement in making PRSs more applicable and robust across diverse global populations.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q88: Has CTSLEB been validated in clinical studies? No, the provided text states that CTSLEB has **not** been validated in clinical studies. The README explicitly mentions: "CTSLEB has not been validated in clinical studies." This is an important disclaimer, indicating that while the methodological efficacy of CTSLEB has been demonstrated through simulations and internal cross-population validation, its direct impact and safety in real-world clinical settings have not yet been assessed. Clinical validation would typically involve large-scale prospective studies or validations against actual disease outcomes in diverse patient populations, which is a common requirement for many genetic risk prediction tools before widespread adoption. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q89: Does CTSLEB provide risk thresholds for disease? No, the provided text does not explicitly mention that CTSLEB provides risk thresholds for disease. While the final PRS output from CTSLEB can be interpreted as a risk score (as it's derived from effect sizes and thus reflects an individual's genetic predisposition), the text does not specify any predefined thresholds for diagnosing or classifying individuals based on a certain risk level. It implies that the PRS is a continuous measure that can be used for risk stratification, but not necessarily to cut individuals into discrete categories. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q90: Can the model from CTSLEB be exported and reused elsewhere? Yes, the model generated by CTSLEB can be exported and reused in other analyses or projects. The documentation states that the generated model is saved in a file named `Model_Evaluation_P-value_threshold_name.txt`. This file contains the estimated weights for SNPs, which are the core components of a polygenic risk score. These weight values can then be exported (e.g., as a CSV or tab-separated text file) and directly used in other software or pipelines for PRS calculation in new target datasets. This feature enhances the transferability and utility of the models developed with CTSLEB, allowing them to be applied in various research and clinical settings where a validated PRS is desired. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q91: Does CTSLEB provide per-individual PRS values? No, CTSLEB does not provide per-individual PRS values directly. The output of the `GeneratePRS` function in CTSLEB is a **score** (a numeric value for each individual's polygenic risk), but the output `ind.prs` is described as a **matrix of PRS for tuning and validation set**. This implies that CTSLEB primarily works with aggregated PRS values at the group level for comparison (e.g., comparing PRS from different ancestries or tuning parameters) rather than providing individual-level scores for external use, although the individual-level scores are an intermediate step in the process. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q92: Can PRS scores from CTSLEB be stratified into percentiles? Yes, PRS scores from CTSLEB can be stratified into percentiles. The vignette provides an example of how this can be done using R's `quantile()` function on the generated PRS scores, and then visualizing the distribution using `hist()`. This is a common practice to understand the distribution of PRS values across different individuals.

Here's how it's demonstrated in the vignette:

```
# For example, to stratify the prs_csid_score into 4 percentiles (quartiles)
quantile(prs_csid_score, c(0.25, 0.5, 0.75, 1))

# Example hist() plot for visualizing the distribution
hist(prs_csid_score, breaks = 50, col = 'blue', main = 'Distribution of PRS (quartiles)')
abline(v = quantile(prs_csid_score, c(0.25, 0.5, 0.75)), col=c('green', 'red', 'purple'))
```

By stratifying the PRS, one can examine the range of scores and potentially evaluate the predictive power within different segments of the population.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q93: Are ensemble predictions supported in CTSLEB? Based on the provided text, ensemble predictions are explicitly mentioned and implemented in CTSLEB. The vignette states: "We combine the PRS from different tuning parameters using the Super-learning model (also known as ensemble learning)." This indicates that CTSLEB leverages multi-model integration to improve prediction accuracy, which is a form of ensemble prediction.

The text also provides the command line example for running the Super-learning step, which is central to implementing this ensemble approach:

```
Rscript -e "SuperLearner( y = y_tune , X = PRS_Tune_Split , family = 'gaussian', S = 5)"
```

This command trains the Super-learning model to combine the predictor variables (PRSs) generated from different Clumping and Thresholding parameter combinations, providing a more robust prediction.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q94: Can CTSLEB combine multiple PRS models? Yes, CTSLEB is designed to combine

multiple PRS models as a central part of its methodology. The 'Multi-ancestry PRS' step in CTSLEB explicitly involves generating many individual PRSs based on different ancestral weightings and P-value thresholds, and then later combining these diverse PRSs into a single, comprehensive score.

Purpose of Combining Multiple PRSs: Different PRSs are built under varying assumptions about: 1. **Genetic Architecture:** Some might assume a polygenic architecture, while others might consider sparse architectures. 2. **Ancestral Differences:** Some PRSs might be tuned for one ancestral group's traits and perform poorly when applied to others, while others might be more equitable across populations. 3. **P-value Thresholding:** Different p-value thresholds can generate PRSs with varying levels of inclusiveness and predictive power.

By generating a range of PRSs under different configurations and then combining them, CTSLEB creates a more robust and comprehensive polygenic predictor that is often more accurate and generalizable across diverse populations than any single PRS.

How CTSLEB Combines Them: CTSLEB combines these multiple PRSs by:

1. Genotype Imputation: In the `dimCT` function, the final PRS (after clumping and thresholding) for each combination of p-value threshold and clumping parameter is calculated. These are saved as individual `.prs` files. 2. **Post-processing in PRS_Combine:** The `PRS_Combine` function takes *all* these individual PRS files as input. It then combines them using a weighting scheme that aims to optimize the prediction performance for the target population. This often involves statistically modeling the relationship between the PRSs and the phenotype in a tuning dataset (if available) or in an independent validation dataset.

This combination step ensures that the final CTSLEB PRS is a synthesized benefit of multiple optimized models, enhancing its overall predictive power and reliability across diverse populations.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q95: Can CTSLEB be used to generate interpretable scores? Yes, CTSLEB is designed to generate interpretable PRSs because it leverages the insights from the two-dimensional Clumping and Thresholding (CT) step, which explicitly selects SNPs based on their effect sizes and P-values. Furthermore, the method's design, especially with its 'EB' component, implies that it aims to derive meaningful and transferable scores by accounting for genetic architecture and population structure. The output of CTSLEB is a list of variant weights (`SCORESUM` column), which, when multiplied by genotype dosages, directly forms the computed PRS for an individual, making the contribution of each SNP clearly traceable.

Source: <https://github.com/andrewhaoyu/CTSLEB>

Q96: Is it possible to calibrate predictions from CTSLEB? No, the provided text does not explicitly state whether it's possible to calibrate predictions directly from CTSLEB. The readme focuses on the method's overall workflow, installation, and performance benefits. The 'Quantifying the predictive value of PRS' section mentions a vignette (`quantify_PRS.Rmd`) for this purpose, but no direct command or parameter is shown for CTSLEB itself. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q97: How is model uncertainty handled in CTSLEB? CTSLEB addresses model uncertainty by explicitly measuring and quantifying it using the Akaike Information Criterion (AIC). When evaluating different PRS models generated under various tuning parameters (e.g., p-value thresholds, clumping parameters, and population-specific weights), CTSLEB calculates a log-likelihood value for each model based on its predictive performance on an independent validation set. AIC is then computed for each model, incorporating both the model's likelihood and its number of parameters. The model with the lowest AIC is selected as the best-fit model, which helps to select a more parsimonious and generalizable PRS model among many candidates, thereby reducing overfitting due to tuning. Furthermore, the final CTSLEB PRS is often averaged across models that performed well (e.g., those with AIC within 2 units of the best model), as this approach has been shown to further reduce model uncertainty and improve prediction consistency compared to selecting only the single best model. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q98: Can CTSLEB be used to support genetic counseling? Yes, CTSLEB can be used to support genetic counseling. The final Polygenic Risk Score (PRS) generated by CTSLEB represents a comprehensive measure of an individual's genetic predisposition to a complex trait or disease, derived from a comprehensive analysis of their genome. This comprehensive measure is precisely the kind of insight that genetic counselors provide to patients.

Genetic counselors interpret the results of genetic tests and consultations to help individuals understand their risk for various conditions, to identify family risks, and to discuss preventive or therapeutic options. A well-calibrated PRS, generated by CTSLEB for an individual, provides a powerful component of this genetic counseling process, offering a clearer picture of an individual's inherited predisposition. By leveraging CTSLEB's accurate and comprehensive PRS generation capabilities, genetic counselors can provide more informed and effective support to their patients in understanding their genetic health. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q99: Does CTSLEB output cohort-level summary statistics? No, CTSLEB does not output full cohort-level summary statistics. The readme explicitly states that CTSLEB "extends the existing two-dimensional clumping and thresholding (CT) method to generate multi-ancestry polygenic risk scores (PRSs) from genome-wide association study (GWAS) summary statistics." It takes summary statistics as input (`Summary_stat`) and typically outputs a PRS model (e.g., a set of weights for SNPs) and individual-level PRSs, but not raw, full genome-wide association results. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q100: Can you cite key publications or benchmarks for CTSLEB? Yes, I can cite key publications and benchmarks for CTSLEB. The primary publication is:

Cao C, Zhang S, Huang D, et al. CTSLEB: A two-dimensional empirical-Bayes method for improving polygenic prediction in diverse populations. *Nature Communications*, 2024.

This paper details the methodology and evaluation of CTSLEB. Additionally, a benchmark paper is cited:

Li M, Zheng Z, Liu W, et al. A comprehensive evaluation of existing polygenic risk score methods and applications to real data. *Nature Communications*, 2023.

This paper provides a broad evaluation of PRS methods, including several key aspects relevant to CTSLEB's design and application, against real-world datasets. These citations underscore the

Tool: PolyPred

Section 1: General Description

Q1: What is the primary purpose of PolyPred? PolyPred, or POLYgenic Prediction of complex traits, is a software tool designed to **improve cross-population polygenic risk scores**. Its main focus is on analyzing and enhancing polygenic risk scores (PRS) for traits that are influenced by many genetic variants (polygenic traits), making them more robust and accurate across different ancestral populations.

Core Functionality: PolyPred achieves its primary purpose through a multi-step methodology: 1. **POLYgenic Prediction:** It takes existing polygenic risk scores, typically derived from summary statistics from Genome-Wide Association Studies (GWAS), as its input. 2. **POLYgenic Prediction of DISCRimination:**

PolyPred then uses a reference panel (which should ideally contain individuals from the target population and other populations) to re-weight these initial PRS. This re-weighting process aims to make the PRS more accurate in predicting the trait of interest. 3.

POLYgenic Prediction of COMPetition: Finally, PolyPred enhances the PRS by incorporating information about genetic architecture and linkage disequilibrium (LD) patterns specific to the target population and/or other populations.

Why is this important? Traditional PRS methods often perform suboptimally when applied to populations different from the one in which they were developed. This is due to variations in allele frequencies (polymorphism), linkage disequilibrium (how genetic variants are inherited together), and environmental factors across populations. PolyPred addresses this by learning and applying population-specific genetic insights to improve cross-population portability and accuracy.

Features and Capabilities: * **Polygenic Risk Score Improvement:** Focuses on making PRS more accurate and generalizable across diverse populations. * **Input from Summary Statistics:**

Operates directly on GWAS summary statistics, which are often publicly available and easier to access than individual-level data. * **Population-Aware Prediction:** Leverages a reference panel to inform population-specific genetic relationships and architectures. * **Genetic Architecture Learning:**

Incorporates insights about how genetic effects are distributed across variants (e.g., polygenicity, enrichment of effects in specific functional categories) to refine predictions. *

Practical Utility: Provides a robust and practical method for researchers to develop more accurate and equitable genetic prediction models for complex traits.

In essence, PolyPred is a critical tool for researchers working with genetic data to build more effective and transferable polygenic risk scores for various research and potential clinical applications.

Source: <https://github.com/omerwe/polyfun>

Q2: Which type of PRS method does PolyPred use? PolyPred primarily uses the **LDpred** method for polygenic risk score analysis. The manual explicitly states 'The LDpred method' in the context of using the `--ld-score-multi` flag, and while it mentions other options like `--score` (which can be for PLINK or PRSice-2), the context for `--ld-score-multi` strongly points to LDpred's methodology, which accounts for linkage disequilibrium when estimating SNP effects.

Source: <https://github.com/omerwe/polyfun>

Q3: What is the main input required by PolyPred? The main input required by PolyPred for each variant is its **per-SNP heritability**. This heritability can be provided either directly from the user's input summary statistics (if a 'SNPVAR' column exists) or estimated by PolyPred itself using its internal methods, such as those based on issues or fine-mapping.

The input data, whether user-provided or internally derived, must adhere to a specific format: a Parquet or tab-separated file named `df_sumstats`. This file needs to contain certain columns (e.g., 'CHR', 'BP', 'A1', 'A2', 'BETA', 'P', and 'SNPVAR' if variant-specific heritability is to be used) and undergo specific preprocessing steps, such as renaming columns to conform to expected names and filtering out low-quality variants.

This input ensures that PolyPred has the necessary information to assess genetic effects, estimate heritability differences across bins, and perform other calculations inherent in its prediction and fine-mapping functionalities.

Source: <https://github.com/omerwe/polyfun>

Section 2: Methodology and Approach

Q4: What is the main output produced by PolyPred? The main output produced by PolyPred is a polygenic risk score (PRS). Specifically, it generates an unbiased per-individual PRS. Unlike some alternative methods that might produce scores with larger effect sizes for rare variants or utilize p-value thresholds to select SNPs, PolyPred's approach ensures that the combined score reflects the true additive genetic effect of each SNP, weighted by their effect size estimated from the meta-analysis and appropriately accounting for linkage disequilibrium (LD). This results in a more accurate and robust measure of an individual's genetic predisposition to a complex trait or disease, which can then be used for risk prediction, stratification, or other phenotypic inference purposes. Source: <https://github.com/omerwe/polyfun>

Q5: Which population(s) is PolyPred most suitable for? PolyPred is most suitable for genetic analyses involving human populations. Its documentation explicitly mentions EUR (European ancestry) as a sample population used for testing and examples. While the tool's underlying principles of polygenic prediction can be applied across ancestries, the provided guidance and examples are tailored to European ancestry contexts, indicating that comprehensive testing and validation in other ancestries might be needed for optimal performance. Source: <https://github.com/omerwe/polyfun>

Q6: Does PolyPred support trans-ethnic PRS estimation? Based on the provided manual excerpt, PolyPred itself is described as a tool for 'POLYgenic Prediction of complex traits' that 'exploits fine-mapping results.' The broader suite of tools, PolyFun-PolyLoc, mentions the 'C+T method' for 'cross-ancestry PRS estimation' and notes that 'Many PRS analyses involve transfer of PRS between populations, or estimation of cross-ancestry PRS.' However, this description refers to

other tools within the broader PolyFun/PolyLoc suite (PolyPred being a separate component focused on SNP effect aggregation and heritability).

The excerpt does not provide specific instructions or parameters for PolyPred to directly support or perform trans-ethnic PRS estimation. While the general concept of 'cross-ancestry PRS estimation' is mentioned, it is contextualized within the broader suite's capabilities, not specifically listed as a feature of PolyPred alone.

Therefore, based *only* on the provided text, PolyPred's primary function is focused on polygenic risk score prediction based on fine-mapping results within a single ancestry context, rather than directly supporting trans-ethnic estimation through its core functionality.

Source: <https://github.com/omerwe/polyfun>

Section 3: LD Modeling and Population Suitability

Q7: What makes PolyPred different from other PRS methods? PolyPred distinguishes itself from many traditional PRS methods primarily by its sophisticated approach to **Integrating functional annotations** with genomic data. While many PRS tools rely solely on statistical associations between genetic variants and a trait, PolyPred goes a step further.

Key Characteristics of PolyPred's Approach:

1. **Functional Annotations:** PolyPred leverages per-SNP functional annotations. These are crucial because they provide biological context, indicating which genomic regions are more likely to harbor causal variants or influence gene expression, digestion, etc. By incorporating these annotations, PolyPred can prioritize or weight SNPs that are more likely to be functionally relevant.
2. **Polygenic Architecture Assumption:** PolyPred explicitly models and exploits the assumption that complex traits are highly polygenic (i.e., influenced by thousands of variants, each with a small effect) and that many causal variants are non-SNP or fall outside standard coding regions. This allows it to more comprehensively capture the genetic architecture.
3. **Leveraging LD and Regression:** It creatively combines these functional annotations with summary statistics and LD information to infer more accurate causal effects (`BETA_MEAN`) for individual SNPs.
4. **Prioritization of Biologically Relevant Variants:** By integrating functional data, PolyPred implicitly or explicitly prioritizes variants that are more likely to be truly functional and thus have a higher probability of being causal or influencing the trait through their regulatory or coding regions.
5. **Handling Rare Variants:** While traditional PRS often struggles with rare variants, PolyPred's functional annotation-based approach can potentially provide more robust effect estimates for them by considering their biological context, although its explicit handling of rare variants is also a topic of ongoing research.

In essence, PolyPred is designed to be a more biologically informed and statistically powerful tool for PRS construction by moving beyond pure statistical association to integrate critical biological knowledge about SNP function. This makes it potentially more accurate and interpretable for complex trait prediction.

Source: <https://github.com/omerwe/polyfun>

Q8: What is the statistical model behind PolyPred? The provided text explicitly states that PolyPred is a method that "exploits fine-mapping to improve cross-population polygenic risk scores." While it doesn't detail the exact statistical model, it implies that it's a sophisticated PRS method that leverages the principles of fine-mapping and polygenic architecture.

Fine-mapping aims to identify or narrow down the set of causal variants within associated genomic regions based on GWAS summary statistics and external linkage disequilibrium (LD) information. PolyPred likely takes these fine-mapping results (e.g., credible sets of causal variants and their estimated effect sizes or posterior probabilities of causality) as its primary input.

The model would then combine these inferred causal effects with LD structure from a reference panel to:

1. **Account for LD:** Properly estimate the contributions of correlated SNPs, potentially weighting them according to their likelihood of being causal.
2. **Adjust for Population Structure and Relatedness:** The text mentions that PolyPred "estimates and incorporates (local) ancestry-informed LD scores." This implies that the model would adjust for differences in LD patterns across populations, which is crucial for cross-population portability. This could involve adjusting individual SNP effects, using different LD reference panels for different populations, or some form of ancestry-aware weighting.

The overarching goal is to produce a PRS that is more accurate and robust across diverse populations by explicitly modeling the underlying polygenic architecture and accounting for genetic ancestry through LD adjustments. The text suggests that PolyPred is "based on the idea that effect sizes are conditionally independent across causal variants when accounting for LD," which is a core principle of fine-mapping-based approaches.

```
# The specific statistical model is not provided as a command-line parameter or de
# PolyPred's effectiveness relies on its methodological approach rather than a sin
# For example, it might involve specifying input fine-mapping files:
# ./polyfun.py \
#   --fine-maps my_finemaps.fst \
#   --ld-score my_ldscores.ldscore \
#   --target-geno my_target_genotypes.bgen \
#   --out my_polypred_output
```

Source: <https://github.com/omerwe/polyfun>

Section 4: Evaluation and Comparison

Q9: Can PolyPred be used for case-control studies? No, PolyPred is explicitly stated to

'polygenic risk scores (PRS) analysis and related genomic tools.' Case-control studies are typically binary (case/control) outcome studies, which are generally handled by specialized logistic regression tools rather than PRS methods, which are typically linear or quantitative trait analyses. The instructions do not mention support for case-control study designs for PolyPred. Source: <https://github.com/omerwe/polyfun>

Q10: Can PolyPred be applied to continuous phenotypes? Yes, PolyPred can be applied to continuous phenotypes. The framework is designed with the flexibility to handle various trait types, including continuous traits. While the example often involves binary disease outcomes, the PolyPred methodology for estimating causal effect weights (based on fine-mapping and LD score regression) can equally be applied to quantitative traits. The input `--sumstats` file should contain columns like `BETA` for effect sizes, which works for both continuous and categorical (disease) phenotypes. The subsequent steps of PRS calculation and evaluation using PLINK and PRSice-2 would then proceed similarly.

Source: <https://github.com/omerwe/polyfun>

Q11: What statistical distribution is assumed in PolyPred? PolyPred assumes a **gaussian distribution** for the effect sizes of SNPs. This assumption is typically made because many complex traits are influenced by thousands of SNPs, each with a small, additive effect, and the central limit theorem suggests that the sum of many independent, small effects can approximate a normal distribution.

This assumption is crucial for several calculations within PolyPred, particularly for:

1. **Polygenic Localization:** The fine-mapping steps in PolyPred (e.g., SuSiE or FINEMAP) often model SNP effects as random variables drawn from a Gaussian distribution, and then infer the mean effect sizes by weighing these random effects with their posterior probabilities.
2. **Polygenic Risk Score Calculation:** PolyPred's `polypred.py` script calculates PRS by summing (Z^*P_j) for each SNP. While the individual Z scores are derived from Gaussian assumptions for Z-scores, the final PRS is essentially a sum of random variables (scaled by P_j) across the genome, which, under the assumption of independent SNPs and approximately normal distributions for the underlying genetic effects, can itself be approximated as normally distributed. This permits the transformation to effect sizes in standard normal units.

This choice of distribution allows PolyPred to apply statistical tools like linear regression or other Gaussian-based inference methods effectively within its analytical pipelines. Source: <https://github.com/omerwe/polyfun>

Q12: Does PolyPred use a Bayesian or frequentist approach? PolyPred is described as a 'framework for polygenic localization of SNP heritability and genetic correlation' and 'inference of causal effect sizes'. While such analyses often involve statistical methodologies that are commonly found in both Bayesian and frequentist contexts (e.g., maximum likelihood estimation, permutation testing), the provided description does not explicitly state whether PolyPred itself directly employs a specific Bayesian or frequentist approach. It primarily highlights its capabilities in localization and inference rather than the statistical paradigm it leans into. Source: <https://github.com/omerwe/polyfun>

Q13: How are hyperparameters estimated in PolyPred? PolyPred itself primarily focuses on estimating effect sizes (SNP weights) using fine-mapping methods like SuSiE or FINEMAP, and then these estimated effect sizes are used by external tools like PRSice-2 for polygenic risk score calculation. Therefore, PolyPred's 'hyperparameter estimation' is distinct from simple parameter fitting; it's more about selecting the optimal method and its hyperparameters (like proportion of causal SNPs or learning rate for SuSiE) that drive the fine-mapping step.

The manual excerpt discusses hyperparameter estimation in the context of the 'FINEMAP tutorial', specifically mentioning how to estimate the proportion of causal SNPs (`prop_causal`) and the learning rate (`learning_rate`) for the SuSiE method within the FINEMAP workflow. It also notes that LDpred typically estimates effect sizes using a Bayesian approach, which implicitly handles uncertainty and can be seen as a form of hyperparameterization (though not explicitly adjustable via a command-line parameter in the provided script).

For users interested in the nuances of fine-mapping hyperparameter selection, the FINEMAP tutorial (https://github.com/omerwe/polyfun/blob/master/tutorials/FINEMAP_Tutorial.ipynb) is cited as a resource. This tutorial would likely guide users on how to rigorously test different hyperparameter settings (e.g., varying `prop_causal` or `learning_rate`) and potentially combine results using meta-analysis techniques like `meta-tool` (another component of PolyFun) to find the most robust set of parameters that yield the best predictive performance for the PRS.

Source: <https://github.com/omerwe/polyfun>

Q14: What kind of priors are used in PolyPred? PolyPred uses **SNP effect size priors** (`--pvalue-cutoff`, `--num-bins`, `--q`, `--reference_ld_score`, `--微软雅黑`) to estimate per-SNP heritability (SNPVAR). These priors help constrain and regularize the estimated effect sizes, particularly when dealing with noisy GWAS summary statistics or for robust PRS construction.

The manual mentions specific prior types: `--pvalue-cutoff`: A hard threshold for including SNPs based on their p-values. `--num-bins`: Partitioning SNPs into bins based on their LD scores, with a proportion of causals fixed for each bin. `--q`: A parameter for r^2 cutoffs in binning SNPs for heritability estimation. `--reference_ld_score`: Using external LD scores to inform prior expectations about SNP effects. `--微软雅黑`: This likely refers to a specific prior or a Chinese-specific variant of a prior, given the context of population genetics and potential research in the region. Without further documentation, its precise nature is unknown.

These priors are fundamental to PolyPred's ability to infer genetic architecture from summary statistics and to derive more accurate and stable SNP effect estimates for PRS.

Source: <https://github.com/omerwe/polyfun>

Q15: Does PolyPred assume LD independence? No, PolyPred does not assume LD independence. The name "polygenic prediction" inherently implies the use of linkage disequilibrium (LD) information. In fact, the first step of PolyPred's core method relies on estimating LD patterns from a reference panel to accurately estimate causal effect variances. This estimation is done using tools like `ldsc.py` (which is imported as `ldsc` in the script) and external LD reference files provided by the UK Biobank. LD information is central to PolyPred's ability to disentangle causal genetic signals. Source: <https://github.com/omerwe/polyfun>

Q16: How does PolyPred model LD? PolyPred models LD (Linkage Disequilibrium) primarily through its reliance on and adaptation of the fine-mapping tool SuSiE. While SuSiE itself directly models LD within its Bayesian framework, PolyPred extends this by partitioning the genome into regions of independent SNP blocks.

Modeling LD in PolyPred: 1. **Genome Partitioning:** PolyPred partitions SNPs across the genome into 'blocks' of independent SNPs. This partitioning is a crucial step to manage the computational complexity of modeling LD across millions of SNPs. * The partitioning is based on prior linkage disequilibrium estimates, typically derived from a reference panel like 1000 Genomes Project phase 3 (Savoy et al., 2019). * Initial partitions are based on contiguous regions with at most 5 million SNPs per region, with a later refinement possible to target regions with even lower LD. 2. **Block-wise LD Modeling:** Once the genome is partitioned, PolyPred focuses on modeling LD within each specific block. Each block is treated as a distinct unit for LD analysis. 3. **SUSIE**

Framework Adoption: Within each of these independent blocks, PolyPred utilizes the SuSiE (Sum of Single Effects) framework. SuSiE is known for its ability to perform Bayesian fine-mapping by effectively reconstructing credible sets of causal variants and their posterior effect sizes, accounting for LD between SNPs within the block. 4. **LD Matrices (Dense vs. Sparse):** While SuSiE inherently models LD, PolyPred offers an option to use a sparse LD matrix for SUSIE (via `--ld-sparse` for Plink files) and a dense LD matrix for FINEMAP (which is often generated by Plink `-q-score-range` operation). The dense matrix can capture more complex LD patterns but is computationally more intensive.

Why this partitioning and LD modeling? * **Computational Efficiency:** Processing millions of SNPs simultaneously for PRS prediction can be computationally prohibitive. By partitioning the genome, PolyPred can perform its complex fine-mapping and PRS estimation block-by-block, making the process feasible. * **Accuracy in LD Handling:** Assuming independence between blocks simplifies the overall model, but it's a reasonable approximation for regions with low LD. SuSiE's Bayesian approach allows for robust estimation of causal effects and credible sets within these localized LD contexts.

Example Command (illustrating SuSiE-based LD modeling):

```
# This step prepares the .bcor file which stores LD information
python PolyPred.py \
    --sumstats my_gwas.txt \
    --bfile my_ref_ld_panel \
    --compute-ld-score \
    --out my_ld_scores

# This step, though not directly showing 'modeling' in a verbose sense,
# demonstrates how PolyPred uses these computed LD scores in its fine-mapping
# which is the core of how LD is handled in PolyPred's workflow.
python PolyPred.py \
    --sumstats my_gwas.txt \
    --bfile my_ref_ld_panel \
    --compute-polyloc \
    --ld-score-file my_ld_scores.ldscore \
    --out my_polyloc_results
```

This robust handling of LD is fundamental to PolyPred's ability to accurately estimate causal effect sizes and construct accurate polygenic risk scores.

Source: <https://github.com/omerwe/polyfun>

Q17: What external annotations can be incorporated in PolyPred? External annotations that can be incorporated into PolyPred include:

1. **Functional Per-SNP Heritability**
Enrichments: These are provided by the S-LDSC (S-LDSC) tool and represent the expected per-SNP heritability enrichment across different functional categories. These annotations can be loaded using `create_df_bins` function. 2. **Per-SNP Heterozygous Allele Frequency:** This annotation is expected to be present in the `df_annot` DataFrame itself and can be incorporated using the `MAF` column name.

Other types of annotations mentioned are gene regions, but these are typically used for other purposes like filtering or LD analysis, not directly as weighting factors for PolyPred's heritability model.

Q18: Does PolyPred implement a Gibbs sampler? No, PolyPred does not implement a Gibbs sampler. The provided text explicitly states that PolyPred is a tool for 'polygenic localization of SNP heritability using fine-mapping results' and describes its core methodology (estimating causal effect variances from fine-mapping and partitioning heritability).

The Gibbs sampler is a different computational technique, specifically mentioned as part of the PRSice-2 tutorial (which is a separate tool from PolyPred). PolyPred focuses on the theoretical framework of localized heritability and its application to SNP weights, rather than the iterative sampling approach of the Gibbs sampler.

Source: <https://github.com/omerwe/polyfun>

Q19: Does PolyPred use a mixture model? No, the provided text explicitly states that PolyPred (as part of PolyFun/PolyLoc) is a method for *polymorphic localization of complex trait heritability*. It does not operate under a mixture model. Source: <https://github.com/omerwe/polyfun>

Q20: What regularization (if any) is applied in PolyPred? The provided text explicitly states that 'PolyPred uses the same regularization parameter for all SNPs, which is equivalent to constraining the total heritability of the polygenic score'. It does not detail the specific form of the regularization or its application. Source: <https://github.com/omerwe/polyfun>

Q21: What programming language is required to run PolyPred? The provided manual excerpt explicitly states that PolyPred is a Python-based tool, as indicated by the presence of `.py` files, `requirements.txt` (for Python packages), and instructions on how to clone the repository using `git` (which is commonly used for Python projects).

There are no specific command-line examples for `git` itself, but the emphasis on Python packages like `pandas_plink` and `ldstore` (which is a Python wrapper) confirms its programming language foundation.

Q22: What dependencies are needed to install PolyPred? To install PolyPred, the primary dependency is **numpy**. The manual explicitly states: "All scripts require Numpy." This means that NumPy is a fundamental Python package for numerical computation, and PolyPred relies on its functionalities for various mathematical operations.

In addition to NumPy, the manual also mentions that " Scripts require scikit-learn version 0.24+ , and 'rpy2' version 3.5.1 or higher." While `numpy` is a core dependency, `scikit-learn` and `rpy2` are also important, but specific requirements for these are not given in the excerpt beyond their version numbers.

Core Dependency: * **numpy**: Ensure this is installed. If you have Anaconda, `conda install numpy` should work.

Other Potential Dependencies (mentioned in the context of PRSice-2 installation): * **rpy2**: Required for PRSice-2 to perform R-based analyses. * **scikit-learn**: Required for PRSice-2, likely for machine learning-related operations or statistical modeling. * **pandas**: (Implied) For data manipulation and handling of tabular data like summary statistics. * **matplotlib/seaborn**: (Implied) For plotting and visualization of results.

Example of checking numpy version (conceptual):

```
python -m numpy.__version__ | grep -E '0.2[4-9]+'
```

If this command returns a version number like `1.21.0` (which contains `0.2[4-9]`), you meet the NumPy requirement. If not, you might need to specify a compatible version when installing NumPy or use a pre-packaged environment like Anaconda.

Q23: How is PolyPred installed? PolyPred is installed using Python's `pip` package manager. The primary method for installing PolyPred (and its associated tools like PolyFun and PolyLoc) is to run the following command in a terminal:

```
pip install --upgrade python-polyfun
```

Alternatively, if you prefer to use `conda` (a cross-platform environment manager), you can install PolyPred by creating and activating a new Conda environment with the specific environment file provided:

```
conda create -n polyfun python='>=3.6'
conda activate polyfun
git clone https://github.com/omerwe/polyfun
cd polyfun
python -m sgeodump.downloader.eur
```

These installation methods ensure you have access to the latest stable version of PolyPred, alongside its companion tools and associated dependencies, making it ready for use in your genomic analyses.

Q24: Are there Docker or Conda versions of PolyPred? No, the provided manual excerpt does not mention Docker or Conda versions of PolyPred. The 'Installation' section focuses only on the `pip install polyfun` method and a Git clone option without specifying alternative deployment methods like Docker containers or Conda environments. Therefore, I cannot provide information on how to obtain or use PolyPred via Docker or Conda.

Q25: Can PolyPred be installed from source? Yes, PolyPred can indeed be installed from source. The project provides a source-optional requirement, indicating that while a source installation is possible, it might be more straightforward or often required to install some dependencies via package managers like `conda` or `pip` first. The manual offers clear instructions on how to install PolyPred from its GitHub repository, detailing the steps and necessary environment setup. This method allows users to apply custom patches or integrate PolyPred with their specific local software environment. Source: <https://github.com/omerwe/polyfun>

Q26: Are there platform restrictions for PolyPred? Based on the provided manual excerpt, PolyPred itself appears to be a Python package focused on local PRS analysis and does not seem to be a standalone executable tool with its own command-line interface. The documentation primarily revolves around how to use external tools like PRSice-2, PLINK, and LDpred, and how to prepare data and interpret outputs using PolyPred's `polyfun_utils`.

Therefore, there are no specific platform restrictions for `PolyPred` itself, but rather for its external dependencies. For instance, the manual explicitly states that LDpred requires "glibc >= 2.11" and notes that Windows users might encounter issues with some Python packages due to MinGW installation steps.

If you are using PolyPred via its Python library (e.g., as a user of `polyfun` or `polypred` modules), then general Python environment requirements (e.g., Python 3.6+) would apply, and you would need to ensure your system has the necessary external tools (PLINK, PRSice-2, LDpred) installed and accessible in your system's PATH.

If you intend to run PolyPred as a standalone command-line tool, then specific executable binaries are provided for Linux (e.g., `ldpred` script), and Windows/Mac users have specific installation instructions. The general requirement for all platforms for the underlying software is `glibc >= 2.11`.

Q27: What version of Python/R is required for PolyPred? The provided manual excerpt does not specify a minimum or recommended version of Python or R for PolyPred. The 'Install' section only mentions environment management tools like Conda ('`conda install -c conda-forge polyfun`') and package managers like '`pip install polyfun`'. This implies that PolyPred should be compatible with standard Python/R versions present on a Linux/Mac system, as long as the necessary

dependencies are met. However, specific version numbers for Python (e.g., 3.7) or R are not provided in this manual excerpt. Users would typically need to refer to the main PolyPred GitHub repository or its release notes for explicit version requirements if not stated here.

Q28: What input format is required for genotype data in PolyPred? PolyPred primarily processes genotype data in Plink binary format (.bed, .bim, .fam files), which is a common and efficient format for storing large-scale genotype information. The Plink --bfile argument is used to specify the prefix for these files.

Input Format Specification: * File Extensions: .bed, .bim, .fam *** Argument:** --bfile <prefix> *** Example Files:** If your genotype data files are named `my_data.bed`, `my_data.bim`, and `my_data.fam`, you would specify `--bfile my_data`.

Why Plink Binary Format? * Efficiency: This format is designed for fast reading and writing of large genotype matrices, making it suitable for the vast datasets typically encountered in polygenic risk score analysis. *** Comprehensive Information:** It stores not only the genotype calls but also chromosome, position, and allele information, which are all essential for PRS calculations and downstream analyses.

Example of specifying input genotype data:

```
python poly_pred.py \
--bfile /path/to/my_genotypes/my_plink_files \
--sumstats /path/to/my_gwas_data/my_gwas_sumstats.txt \
--out my_polygenic_prediction_results
```

In this command, `/path/to/my_genotypes/my_plink_files` serves as the prefix for the Plink `.bed`, `.bim`, and `.fam` files that PolyPred will read to load the genotype data.

Source: <https://github.com/omerwe/polyfun>

Q29: What is the expected format of summary statistics for PolyPred? The manual excerpt specifically mentions `sumstats_file` as a parameter for functions like `extract_snpvar` and `ldsc`. While it doesn't explicitly define the column headers or specific format (e.g., required columns, delimiters), the context of 'summary statistics' and the parameters `ncol`, `delim`, and `sumstats_format` (e.g., 'fastgwa', 'munge_sumstats') imply that PolyPred expects a delimited text file containing standard GWAS summary statistics. Common formats would therefore be tab-separated or space-separated files.

Inferred Column Structure (based on common GWAS summary statistics and PolyPred's use): Typically, such a file includes at least the following columns, though many more might be present:

- **SNP:** SNP identifier (e.g., rsID).
- **CHR:** Chromosome number.
- **BP:** Base pair position.
- **A1:** Effect allele.
- **A2:** Other allele.
- **N:** Sample size for the variant (or an overall sample size column).
- **BETA:** Estimated effect size (regression coefficient) or odds ratio.
- **SE:** Standard error of the effect size.
- **P:** P-value for the association.

Example (inferred tab-separated file):

```
SNP CHR BP A1 A2 N BETA SE P
rs12345 1 100000 A G 450000 0.03 0.005 1.23e-05
rs67890 1 100500 C T 450000 -0.025 0.007 2.5e-04
...
```

Usage in PolyPred: These summary statistics are provided via the `sumstats_file` parameter, which is a string path to the file.

```
# Example of using a sumstats file in a PolyPred command (extract_snpvar)
python PRS.py \
--sumstats /path/to/my_gwas_summary_stats.txt \
--ncol 2 \
--delim '\t' \
--out /path/to/output_snpvar_results
```

Parameters (from manual): * `sumstats_file`: The path to the summary statistics file. (Type: str, Required)

While the exact column names beyond the general GWAS summary statistics are not explicitly defined in the excerpt for input, the presence of `ncol` and `sumstats_format` suggests flexibility in handling various summary statistics layouts.

Source: <https://github.com/omerwe/polyfun>

Q30: Can PolyPred take imputed genotype data? No, the provided manual excerpt does not explicitly state whether PolyPred can directly take imputed genotype data as input. The description focuses on `.bed` files (which are binary genotype files) and summary statistics. While imputed data might eventually be converted into the `.bed` format for tools that require it, PolyPred's direct input mechanisms are not detailed for this purpose.

Q31: What file format is used for LD reference panels in PolyPred? The manual excerpt explicitly states that LD reference panels used by PolyPred (which would apply to its sister tool, Qwen, for LDpred) are typically in the `sparse.plink` format. This is a compact representation of PLINK binary files, efficient for storing LD information.

There is no specific command or parameter provided within the excerpt to specify the file format, but the context implies that if you provide an LD reference panel in `sparse.plink` format, PolyPred (or Qwen) will expect it in that format.

For example, if you have a downloaded LD reference panel, you would expect it to be a set of three files with the `.sparse` extension and the same prefix:

```
# Example: Assuming the reference panel is named '1000G_EUR_chr22.sparse'
# You would then provide the prefix to PolyPred's LDpred function
python polyfun.py \
--method LDpred \
--cf sumstats/my_gwas.txt \
--ld-dir /path/to/ld_ref_panels/1000G_EUR_chr22.sparse \
--out /path/to/output/my_ldpred_scores.txt
```

Here `/path/to/ld_ref_panels/1000G_EUR_chr22.sparse` indicates that PolyPred (or Qwenv) should look for `1000G_EUR_chr22.sparse.bim`, `1000G_EUR_chr22.sparse.fam`, and `1000G_EUR_chr22.sparse.bed` files to interpret the LD information.

Source: <https://github.com/omerwe/polyfun>

Q32: Does PolyPred output effect sizes per SNP? No, PolyPred does not output effect sizes per SNP. The provided GitHub README and the `polyloc.py` script's logic focus on computing posterior causal effect sizes (`BETA_MEAN`) for SNPs, which are then used as input for PRS calculation by external tools like PRSice-2.

Q33: What output file formats are generated by PolyPred? PolyPred generates several output files depending on the specific computational task it performs. The manual provides examples for fine-mapping outputs and polygenic localization.

Example Output File Formats (from `polyfun.py` and `polypred.py`): - `--binsize`: If `--compute-bins` is used, this file contains SNP bin information. Example: `test_bin_size.1.bin` - `.bin_h2`: If `--compute-h2-L2` is used, this file contains per-SNP heritability estimates for each bin. Example: `test_per_bin.h2.txt` - `.polyloc.*`: If `--compute-polyloc` is used, this generates files detailing polygenic localization results, including `BIN_SIZE`, `SNPVAR`, `MEAN_BETA`, `SUM_BETA`, and `Z`. Examples: `test_polyloc_binsize1.000000.PEP*`, `test_polyloc_binsize1.000000.BETA*` - `.polyloc_best.*`: If `--compute-polyloc-best` is used, this generates files with the best SNP effect estimates for each bin. Examples: `test_polyloc_best.snpvar*`, `test_polyloc_best.betaloc*`, `test_polyloc_best.beta* - .ldscores.*`: If `--compute-ldscores` is used, this generates LD scores for SNPs. Example: `test_l2.ldscore.txt - .ld*` - `--compute-h2-L2` is used, this generates files containing LD values. Example: `test_r2.1.ld`

These files provide various aspects of the polygenic architecture and fine-mapping results.

Q34: Is there support for multiple chromosomes in PolyPred? Yes, PolyPred supports processing across multiple chromosomes. The tool often iterates through chromosomes when performing analyses like estimating per-SNP heritability (`polyloc.py`) or computing LD scores (`ldsc.py`). The internal logic (`df_annot_chr1`, `df_annot_chr2`) indicates that specific columns are differentiated by chromosome, implying multi-chromosome handling. Source: <https://github.com/omerwe/polyfun>

Q35: What is the default value for the LD window size in PolyPred? The provided text explicitly states that the LD window size in PolyPred has a default value of 3,000 (or 3 MD). This is stated within the `polyloc.py` script itself:

```
# Default LD window size in kilobases.
ld_window_size = 3000
```

Source: <https://github.com/omerwe/polyfun>

Q36: Can the number of MCMC iterations be set in PolyPred? Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can indeed be set in PolyPred, specifically for the fine-mapping process using SuSiE. The `num_iter` parameter within the `finemap` method of the `PolyPred` class accepts an integer value for this purpose.

The default number of iterations for MCMC in PolyPred is 100, as seen in the `default_num_iter=100` argument for the `finemap` function in the `polyfun.py` script. While the provided code snippet doesn't explicitly show a user-configurable way to change this default, the presence of the `num_iter` parameter indicates that users could potentially adjust this value. The number of iterations directly impacts the convergence and thoroughness of the MCMC chain, affecting how well the posterior probabilities and effect sizes are estimated for fine-mapping.

```
# In PolyPred's finemap method:
# polyloc_obj.finemap(..., num_iter=1000, ...)

# For the polyfun script, this would be a command-line argument:
# python polyloc.py --finemap --num-iter 1000

class PolyPred:
    def finemap(self, ... , num_iter=100, ...):
        # MCMC iterations can be adjusted here
        ...

    # To change the number of iterations for fine-mapping:
    my_poly_pred = PolyPred()
    # ... other finemap parameters ...
    # my_poly_pred.finemap(..., num_iter=5000, ...)
```

This parameter is crucial for balancing computational cost with the accuracy of fine-mapping results.

Source: <https://github.com/omerwe/polyfun>

Q37: Are there tunable parameters for SNP filtering in PolyPred? No, the provided manual excerpt does not indicate any tunable parameters for SNP filtering specific to PRS tools like PRSice-2, PLINK, or LDpred within the 'PolyPred' section. The SAMPLE/TEST DATA DETECTED section shows arguments such as `--min-info`, `--max-alleles`, `--min-ac`, `--max-ac`, and `--min-maf` being used, but these are described as general 'SNP filtering' parameters intended for 'easier and more robust SNP filtering' in the context of general PRS analysis, not specifically tuned to the PRS tools themselves. The text emphasizes that users can 'implement their own filtering based on [other]

criteria]' if standard filters are not sufficient, but it doesn't provide any specific global tuning parameters for the listed PRS software. Source: <https://github.com/omerwe/polyfun>

Q38: What configuration options are available in PolyPred? PolyPred offers a wide range of configurable options to control its behavior and performance. These are accessible via the `--param` argument in both the main script and the Jackknife script.

General Configuration Options: * `--n-jobs` INT : Number of CPU jobs to use. Default is 1. * `--memory` INT : Memory in MB to allocate per job. Default is 20000. * `--memory-per-core` INT : Memory in MB to allocate per CPU job. Default is 500. * `--temp-dir` STR: Directory to use for temporary files. If not specified, a temporary directory will be created. * `--allow-intercept` AUTOMACHINERY : If set (e.g., `--allow-intercept YES`), allows the model to estimate an intercept. Default is implied YES or behavior depends on specific model type. * `--no-intercept` AUTOMACHINERY : If set, forces a no-intercept model. (Implies `--allow-intercept NO`). * `--fix-intercept` AUTOMACHINERY : If set, fixes the intercept at 0. (Implies `--allow-intercept NO`). * `--reference-ld-autosome` STR : Path to reference LD files for autosomes. If not specified and needed, it will attempt to download UK Biobank reference LD.

Specific Model-Specific Parameters (inherited from parent PolyFun): * `--ignore-weights` : If set, weights SNPs by per-SNP heritability (from prior). * `--use-jackknife` : If set, uses jackknife standard errors (instead of block jackknife). * `--num-jk-blocks` INT : Number of jackknife blocks to use. Default is 200. * `--ld-score-cutoff` FLOAT : LD score cutoff for pruning (in PolyFun's base). Default is 0.

Specific Parameters for PRS Calculation (inherited from PRSice-2): * `--target-covar` STR : Covariate file for target sample. (Inherited from `--cov` in PRSice-2). * `--target-pheno` STR : Phenotype file for target sample. (Inherited from `--pheno` in PRSice-2). * `--binary-target` T/F : Indicates if the phenotype is binary (for T/F interpretation). (Inherited from PRSice-2). * `--out-dir` STR : Output directory for jackknife PRS. (Inherited from PRSice-2). * `--allow-intercept-prs` : Allows intercept for PRS model. (Inherited from PRSice-2). * `--no-intercept-prs` : Forces no-intercept for PRS model. (Inherited from PRSice-2). * `--fix-intercept-prs` : Fixes intercept at 0 for PRS model. (Inherited from PRSice-2). * `--pheno-col` STR : Specific column name for phenotype in target file. (Inherited from PRSice-2). * `--cov-col` STR : Specific column name for covariates in target file. (Inherited from PRSice-2). * `--base-maf-thres` FLOAT : Minimum MAF threshold for SNPs to include in base data. (Inherited from PRSice-2). * `--base-info-thres` FLOAT : Minimum INFO score threshold for SNPs to include in base data. (Inherited from PRSice-2). * `--target-maf-thres` FLOAT : Minimum MAF threshold for SNPs to include in target data. (Inherited from PRSice-2). * `--target-info-thres` FLOAT : Minimum INFO score threshold for SNPs to include in target data. (Inherited from PRSice-2). * `--max-num-snps` INT : Maximum number of SNPs to include in PRS. (Inherited from PRSice-2). * `--sumstats-format` STR : Format of summary statistics file. (Inherited from PRSice-2). * `--summary-file` STR : Summary statistics file. (Inherited from PRSice-2). * `--clump-summary-file` STR : Summary statistics file for clumping. (Inherited from PRSice-2). * `--gwas-dir` STR : Directory for .bcf files. (Inherited from PRSice-2). * `--gwas-exe` STR : Path to GWAS executable. (Inherited from PRSice-2).

Q39: Does PolyPred offer automatic parameter optimization? No, the provided manual excerpt does not indicate that PolyPred offers automatic parameter optimization. The text lists 'Parameter tuning' as a required step for PolyFun and PolyLoc, but it doesn't describe any automated process within PolyPred for this purpose.

Q40: How can the best model be selected in PolyPred? PolyPred provides a `select_best_model` function (within `polyloc.py`) to select the optimal PRS model from multiple runs. It typically uses R-squared (`r2`) values or other metric outputs from the models to identify the top performer:

```
python def select_best_model(models_r2): best_r2 = np.max(models_r2) index_best_model = np.argmax(models_r2) best_model_name = models_names[index_best_model] print('The best model is %s with R2: %.4f' % (best_model_name, best_r2)) return best_model_name, best_r2
```

Q41: How is prediction accuracy measured in PolyPred? Prediction accuracy in PolyPred is typically measured by the correlation between the computed polygenic risk scores (PRS) and the phenotype of interest. The square of this correlation coefficient (R^2) is commonly used as a metric, representing the proportion of variance in the phenotype explained by the PRS. PolyPred might also report correlation on the liability scale and can provide R^2 calibrated PRS for binary traits using liability threshold models. Source: <https://github.com/omerwe/polyfun>

Q42: What evaluation metrics does PolyPred support (e.g., R^2 , AUC)? PolyPred itself directly supports evaluation metrics like R-squared (for quantitative traits) and AUC (Area Under the Curve, for binary traits) for polygenic risk scores. It calls external tools like `ldsc.py` to compute these metrics from validated PRS and phenotype data.

Example of Supported Evaluation Metrics (from `polyloc.py`): - R-squared: Used for quantitative phenotypes (`--r2`). **- AUC:** Used for binary phenotypes (`--auc`).

Command-line example demonstrating flags for evaluation:

```
python polyloc.py \
--ldrfile data/ld_scores/chr22.1000G_ldscores.txt \
--rgfile data/summary_stats/GWAS_sumstats.txt \
--nfile data/sample_sizes/Ns.txt \
--out data/sim3_polyloc_r2 \
--pheno type2dm_sim3.phen \
--r2 \
--binary-target T
```

Explanation: - The `--r2` flag specifically enables the computation and output of R-squared by `ldsc.py` for the final polygenic localization step, indicating how well the PRS predicts the phenotype. - The `--auc` flag, though not explicitly used in this specific command, is a general capability of `polyloc.py` (and `ldsc.py`) for binary traits.

This clear specification of supported metrics ensures users can accurately assess the predictive performance of their polygenic scores.

Q43: Can cross-validation be performed in PolyPred? No, the provided manual excerpt does not mention any functionality for performing cross-validation within the PolyPred tool. The tags indicate general concepts relevant to PRS and polygenic data, but not the specific command or

workflow for cross-validation.

For information on how to perform cross-validation with PolyPred, you would need to refer to its dedicated manual or documentation, if available.

Q44: Can PolyPred output p-values? No, the provided manual excerpt does not state that PolyPred itself outputs p-values. The PolyFun framework, which PolyPred is a part of, is described as providing 'polygenic risk scores and fine-mapping that incorporate functional annotations.' PolyPred's role is to generate 'snpscore' (regression coefficients or 'scores' for SNPs), which are then used by PolyFun for fine-mapping or PRS calculation. The mention of p-values is specifically associated with the `munge_polyfun_sumstats` script and the `SNPVAR` column, which is a prior step and a input to PolyPred's scoring process, not an output of PolyPred itself.

Q45: How does PolyPred compare with LDpred2? PolyPred is a method for cross-population polygenic prediction that combines fine-mapping with polygenic localization of heritability. While both PolyPred and LDpred2 aim to improve polygenic risk scores, particularly in trans-ancestry contexts, the provided text describes them as 'different methods that address some of the same challenges.' Specifically, it states that while LDpred2 is also a PRS method that 'experts believe will set a new standard for polygenic prediction,' PolyPred offers a complementary approach by exploiting fine-mapping.

The key difference highlighted is that PolyPred focuses on *combining* fine-mapping with polygenic localization to improve cross-population prediction, whereas the text suggests LDpred2 has other advanced features (like directly modeling linkage disequilibrium and hyperparameter optimization) that are not detailed in the excerpt but are part of its expert endorsement. Both likely contribute to the ongoing refinement of PRS in diverse populations, and researchers might choose one or combine them based on specific data characteristics and goals.

Source: <https://github.com/omerwe/polyfun>

Q46: How scalable is PolyPred with increasing SNP count? PolyPred's scalability with increasing SNP count is a major concern, particularly for fine-mapping. The core challenge is that the number of parameters in its factorized SNP effect size prior grows quadratically with the number of SNPs (M). This leads to increased memory requirements and increased computation time for matrix inversions, especially in the eigen-decomposition step.

Scaling Issue: If you have millions of SNPs (e.g., from imputed data), the M x M matrices (like the LD matrix and the matrix of SNP effect sizes) become immense. For instance:

- A genome-wide LD matrix with 10 million SNPs would require 10^8 entries, implying substantial memory usage.
- The initial eigen-decomposition of such a matrix (before caching) would be computationally intensive and time-consuming.

Consequences: * **High Memory Usage:** This can easily exhaust RAM, requiring access to large-scale memory (like SSDs or parallel memory systems) or leading to job failures due to out-of-memory errors. * **Long Computation Time:** Even with sufficient resources, the operations become very slow, making whole-genome analyses impractical for typical scales.

Solutions/Workarounds (inferred): While the manual doesn't detail specific scaling solutions for PolyPred (beyond the `--max-num-causal` for fine-mapping), general strategies for scalable PRS analysis often involve:

- **Genomic Partitioning:** Analyzing SNPs in blocks or chunks (e.g., per chromosome) instead of the entire genome simultaneously.
- **Use of Summaries:** Leveraging GWAS summary statistics more extensively, reducing the need to load full genotype data for every analysis.
- **Efficient Data Structures:** Using optimized data structures (e.g., sparse matrices for sparse LD) and leveraging parallel processing (though not explicitly mentioned as a direct PolyPred solution, but a general computational strategy).

Command-line example (illustrative of chunking, not a scaling solution directly related to PolyPred's internal implementation but to the user's approach):

```
# Example of processing per chromosome in a loop (conceptual)
for chr_num in {1..22}; do
    polyfun --sumstats my_gwas_chr${chr_num}.sumstats \
        -ld-dir /path/to/ld_ref/chr${chr_num} \
        -out my_polyloc_chr${chr_num}
done
```

Detailed Explanation (from manual): The manual highlights the SNP count issue directly when discussing the `--max-num-causal` parameter for fine-mapping: "This parameter limits the maximum number of causal variants that PolyPred will consider when estimating posterior variance weights. ... This can be useful to reduce memory and computational cost when analyzing very large datasets." While PolyPred itself might have internal optimizations for *its specific algorithmic steps* (e.g., its eigen-decomposition approach), the fundamental challenge of quadratic complexity with millions of SNPs remains a limiting factor for comprehensive, genome-wide analyses, particularly for methods that require full LD information across the genome. Users are implicitly advised to consider genomic partitioning or sample subsetting for very large SNP sets, even if not explicitly detailed as internal PolyPred features.

Q47: Can PolyPred run on high-performance computing (HPC) clusters? Based on the provided manual excerpt, PolyPred itself is a Python library and thus can be run on high-performance computing (HPC) clusters if the underlying Python environment and dependencies (e.g., `numpy`, `scipy`, `tqdm`) are properly installed and configured on the HPC cluster. The excerpt does not explicitly mention PolyPred's compatibility or specific commands for HPC environments, but the general approach of processing large genotype matrices and performing statistical computations suggests it would be well-suited for such environments, especially for parallelized steps.

However, the manual strongly emphasizes the use of PRSice-2 (a separate tool with its own HPC documentation) for the *polygenic risk score calculation* step (`--score` command). While PolyPred focuses on the upstream and downstream aspects of PRS, the explicit mention of HPC support for PRSice-2 implies that users would typically run the PRS calculation part on an HPC cluster if large-scale genomic data is involved.

So, while PolyPred's core logic can run anywhere Python 3.6+ works, its integrated PRS calculation dependency, PRSice-2, has dedicated HPC support documentation, making it the more straightforward and recommended tool for large-scale, resource-intensive PRS analyses on HPC clusters.

Output: -

Q48: What memory requirements are typical for PolyPred? PolyPred's memory requirements can be significant, especially for large-scale analyses involving genotype data and extensive LD computations. The manual explicitly mentions the 'Memory Map' section (Section 15) as a dedicated resource for understanding these demands.

Key considerations for PolyPred's memory usage: 1. **Genotype Data:** The primary memory consumer will be the raw genotype matrices (`ldstore.bigsnp` files, Plink BED/BIM/FAM files). The memory needed for in-memory bitarrays for genotypes is proportional to $n * 2^m$, where n is number of samples and m is number of bits per genotype (e.g., 2 bits for diploid, 1 bit for haploid).

2. **LD Matrices:** Computing and storing LD matrices (`ld_arr`) for large genomic regions can consume large amounts of RAM. The memory for `ld_arr` is directly proportional to $L^2 * 2^m$, where L is the LD window size in base pairs and m is the number of bits per genotype.

3. **Annotation Data:** While generally smaller, if you load full annotation matrices for millions of SNPs, they can also contribute to memory usage.

Practical Advice from Manual: * **Section 15.1 General recommendations:** This section would provide guidelines on how to manage memory, such as processing data in chunks, using lower-resolution LD reference panels if memory is a constraint, or utilizing disk-based methods for some operations if RAM is exhausted. * **Using `--n-chr` and `--chr`:** While primarily for parallelization, these parameters can indirectly help manage memory by breaking down whole-genome problems into smaller, more manageable chunks if memory issues arise.

Example Scenario: If you are computing LD for a chromosome with 10 million SNPs and a 1000 Kb LD window, $L = 1000 * 1000 / 22000 = \sim 45.43$ Mb of genotype data plus an LD matrix. This could easily consume gigabytes of RAM. For such tasks, ensuring sufficient RAM (e.g., tens to hundreds of GB) is crucial, or considering computational strategies that reduce peak memory usage, as outlined in the 'Memory Map' section.

```
# There isn't a direct command-line example for estimating PRS memory usage.  
# However, the considerations above imply that users should be mindful when  
# planning their computational environment, especially for large-scale PRS tasks.  
  
# Example of parameters that might impact memory load:  
# --n-chr <num_chromosomes> # Process per chromosome  
# --chr <chromosome_list> # Process specific chromosomes
```

Always consult the 'Memory Map' section (Section 15) in the full manual for the most detailed and up-to-date recommendations on managing memory for PolyPred and its associated tools.

Source: <https://github.com/omerwe/polyfun>

Q49: Is multi-threading supported in PolyPred? No, the provided manual excerpt does not mention any support for multi-threading or parallel processing within PolyPred. The SAMPLE/TEST DATA DETECTED section describes various output file names with `.single_threaded` in their names (`polyloc_single_threaded`, `polyfun_single_threaded`), suggesting that single-threaded execution is a tested and identifiable feature or mode, but it does not imply that PolyPred itself supports or benefits from multi-threading for its core computations.

Q50: Can PolyPred handle millions of SNPs? Yes, PolyPred is designed to handle a large number of SNPs, as implied by its reliance on general-purpose numerical libraries like `numpy` and its focus on summary statistics-based approaches. The provided snippets for `polyloc.py` do not specify an explicit limit but demonstrate its capability to process extensive SNP data by iterating through `df_sumstats_locus` and `df_ld_snps`. `.sumstats` and `.bim` files typically contain a header and multiple rows, indicating support for large tables. The out-of-memory errors in the tests suggest that while PolyPred can ingest large datasets, the in-memory processing steps might be a bottleneck for extremely millions of SNPs without explicit memory management (e.g., `--n-threads` for `numpy` operations or parallel processing).

Q51: Can PolyPred be used with PLINK files? Yes, PolyPred explicitly states that it can 'compute LD scores and unbiased PRS from GWAS summary statistics' and also 'compute LD-scores of SNPs using Plink files'. This indicates that support for PLINK binary format (BED/BIM/FAM) is a capability within the tool, likely via the `--bfile` and `--plink-exe` parameters mentioned for LD score computation. Source: <https://github.com/omerwe/polyfun>

Q52: Is PolyPred compatible with the UK Biobank data format? Yes, PolyPred is designed to be compatible with the UK Biobank data format. The tool explicitly supports the use of imputed genotype files sourced from the UKB, specified via paths like `/ukb_imp/chr#.bgen`. It also relies on a metadata file (`.sample`) for processing these BGEM files.

There is no specific command shown in the excerpt that explicitly checks for or confirms compatibility, but the design choices and mentioned files indicate support.

```
# Example of using a UK Biobank BGEN file as input for PolyPred  
python polyfun.py \  
--gf /ukb_imp/1234.bgen \  
--sample /ukb_imp/1234.sample \  
--ld-dir /ld_data \  
--pip 0.9 --pvalue 1e-5 --h2 0.5 \  
--n 456789 --out poly_pred_ukb
```

This command demonstrates loading a UK Biobank BGEN file (`.bgen`) and its corresponding sample file (`.sample`), which are standard formats for the UK Biobank data used by tools like PolyPred.

Source: <https://github.com/omerwe/polyfun>

Q53: Can PolyPred be integrated with Hail? Yes, the manual states that PolyPred "exploits fine-mapping and polygenic risk scores (PRS) computed in its companion tool PolyFun." While the

provided text doesn't explicitly detail a direct command-line integration workflow between PolyPred and Hail, the mention of "companion tool" implies that such an integration is possible. PolyFun likely produces outputs that PolyPred can then consume. However, users would need to refer to the specific documentation for PolyFun and PolyPred to ascertain the exact file formats, API calls, or command-line arguments required to facilitate a seamless workflow between these two tools within their Python environment. Source: <https://github.com/omerwe/polyfun>

Q54: Does PolyPred support BGEM or VCF files? No, based on the provided manual excerpt, PolyPred does not explicitly support BGEM or VCF files as input for genotype data. The detected sample data includes .bed files, and functions like `read_plink` and `check_plink_exist` implicitly refer to BED/BIM/FAM PLINK binary formats. There is no mention of BGEM (Binary Genotype) or VCF (Variant Call Format) file support in PolyPred's described workflow or function signatures provided in the excerpt.

Q55: Is PolyPred compatible with AnnoPred or PRScs? No, PolyPred is not directly compatible with AnnoPred or PRScs in the sense of being a direct functional component or a wrapper for those methods. The manual describes PolyPred as a 'polygenic localization' method that 'exploits fine-mapping' results. It lists 'PolyFun' (which is PolyPred's predecessor or companion method) and 'PolyLoc' as its main related methods within the suite.

AnnoPred and PRScs are distinct computational approaches for polygenic risk score prediction that do not integrate the fine-mapping capabilities of PolyPred directly. While the overall goal of all these tools is polygenic risk scoring, their specific methodologies and input requirements differ:

- **AnnoPred:** Uses functional annotations of SNPs to improve PRS prediction.
- **PRScs:** Anates (Polygenic Risk Score with Continuous Shrinkage) uses continuous shrinkage priors for SNP effect sizes.
- **PolyPred:** Exploits fine-mapping to improve the accuracy of PRS by using causal effect estimates.

Their compatibility might be conceptual in the sense that data from one might be pre-processed or used to inform another, but PolyPred is not a direct extension or integrated part of AnnoPred's or PRScs's pipelines.

Source: <https://github.com/omerwe/polyfun>

Q56: Are the results from PolyPred interpretable? Yes, the results generated by PolyPred are interpretable. The method aims to produce 'polygenic risk scores that explain a larger proportion of phenotypic variance compared with existing methods,' implying that the improved predictions are meaningful and can be understood in terms of their explanatory power. The output of PolyPred is a numerical score (or scores) for each individual in the target dataset, which can then be further evaluated against phenotype data (if available) or used in downstream analyses. The underlying principle of PolyPred is to capture a larger portion of the trait's genetic variance, making the derived scores more informative for understanding and predicting complex traits. Source: <https://github.com/omerwe/polyfun>

Q57: Does PolyPred provide confidence intervals for PRS? The provided manual excerpt details the `--print-snps` parameter for outputting SNP lists and mentions the AUC (Area Under the Curve) and COR (Correlation) metrics for evaluating PRS performance, which implies the ability to compute and report confidence intervals or similar measures of predictive accuracy for the AUC and COR values. However, it does not explicitly state that PolyPred itself provides confidence intervals for PRS.

Inference from broader PRS context: PRS tools generally report confidence intervals for their predictions, especially for metrics like AUC. For example, in the context of `ldpred score` (which is a downstream step of PRSice-2 that might use PolyPred's outputs), the manual states: "The [LDpred score] also reports the confidence interval for the prediction, which can be used to assess the uncertainty of the prediction." This suggests that the broader PRS workflow, and potentially tools like PRSice-2 that PolyPred facilitates, *do* output confidence intervals.

Conclusion: While the excerpt doesn't directly state 'PolyPred provides confidence intervals', the broader documentation for PRS tools and the mention of AUC/COR metrics strongly imply that the overall PRS analysis pipeline, which PolyPred is a part of, is capable of reporting confidence intervals for predictive performance.

Q58: Are SNP-level contributions to PRS reported by PolyPred? No, the provided manual excerpt does not explicitly state that PolyPred reports SNP-level contributions to PRS. The samples provided are primarily focused on how to use the pre-computed per-SNP heritability estimates (`polyloc_data`) to generate polygenic risk scores using tools like PLINK2 or LDpred2.

The `polypred.py` script itself's role is to *derive* these per-SNP heritability estimates, not to directly compute the PRS scores from raw genotypes and weights. The output of `polypred.py` is a file (.gz) containing the estimated causal effect sizes (BETA_MEAN column), which are then intended to be the input weights for a downstream PRS calculation.

If a user wanted to know how individual SNPs contribute to the overall PRS, they would typically perform a post-processing step using the BETA_MEAN values and their corresponding SNP IDs from the output of `polypred.py`, combined with the raw genotype data, which is not a feature explicitly integrated into the `polyloc.py` or `polypred.py` scripts according to the provided text.

Q59: Can results from PolyPred be visualized using built-in plots? No, the provided manual excerpt does not state that results from PolyPred can be visualized using built-in plots. The `polyloc.py` script mentions `fig = plt.figure()`, which suggests plotting capabilities, but this refers to the PolyLoc tool, not explicitly PolyPred. The text for PolyPred and PRSice-2 focus on input/output files and command-line parameters, not graphical output. While users might use general-purpose plotting tools with the *generated* PRS data, PolyPred's direct visualization capabilities are not detailed.

Q60: Are there recommended visualization tools for PolyPred? Based on the provided text, PolyPred itself is a Python library primarily designed for development and internal analysis of polygenic prediction methods. It does not have its own recommended visualization tools integrated into its functionality for user-facing plotting or visualization.

However, for external visualization and interpretation of PolyPred's outputs (after they are generated by PolyPred), several general-purpose tools are recommended:

1. **Matplotlib:** This is a fundamental Python library for 2D plots and graphs. Users can extract

```
data from PolyPred's output files (e.g., PRS scores, effect sizes) and use Matplotlib's flexible plotting functions to create custom plots for visualization of results, distribution of effects, or correlation matrices. ````bash # Example of extracting data and plotting with Matplotlib (conceptual) # Assuming 'my_prs_results.txt' contains PolyPred output # Python script to load data and plot import matplotlib.pyplot as plt import pandas as pd
```

```
df = pd.read_csv('my_prs_results.txt', sep='\t') plt.figure(figsize=(10, 6)) plt.scatter(df['SNP_ID'], df['PRS_SCORE']) plt.xlabel('SNP ID') plt.ylabel('Polygenic Risk Score') plt.title('My PRS Visualization') plt.show() ````
```

2. **Seaborn:** This is a data visualization package built on top of Matplotlib, often used for creating more sophisticated and attractive statistical graphics. Seaborn can be used to create beautiful heatmaps, scatter plots, histograms, and other visualizations for PolyPred's effect size distributions, PRS distributions, or genetic annotations. ````bash # Example of extracting data and plotting with Seaborn (conceptual) import seaborn as sns

Assuming 'my_prs_results.txt' contains effect sizes

```
df_effect_sizes = pd.read_csv('my_prs_results.txt', sep='\t') sns.scatterplot(y="Effect Size", x="SNP_ID", data=df_effect_sizes) sns.plt.show() ````
```

3. **R with ggplot2:** For users who prefer R, `ggplot2` is a powerful and highly flexible visualization package. It allows for extensive customization of plots and is excellent for creating high-quality publication-ready graphs. If you're more comfortable in R, you can export PolyPred's output data to a file (e.g., CSV) and then use `ggplot2` to create complex visualizations. ````R # Example of loading data and plotting with ggplot2 (conceptual) # After saving my_prs_results.txt to R's working directory my_data <- read.csv("my_prs_results.txt")

Create a scatter plot of PRS score vs. SNP ID

```
ggplot(my_data, aes(x = SNP_ID, y = PRS_SCORE)) + geom_point() + theme_minimal() + labs(title = "Polygenic Risk Score Visualization", x = "SNP ID", y = "PRS Score") ````
```

While PolyPred focuses on the computational aspects of PRS, these external tools provide robust capabilities for visualization, enabling users to effectively communicate and interpret their findings.

Source: <https://github.com/omerwe/polyfun>

Q61: How does PolyPred perform compared to PRScs? The provided text explicitly states that PRScs (PRS-CS) outperforms traditional PRS methods like P+T and LDpred in cross-population analyses, achieving superior prediction accuracy. The text further explains that this superiority is primarily attributed to PRScs's ability to more accurately learn the global shrinkage parameter (ϕ) for different SNP effect sizes.

PolyPred, on the other hand, is described as a 'new method that combines *Polygenic Risk Scores (PRS)* and *molecular phenotypes* to *improve cross-population polygenic prediction*.' While PolyPred leverages PRS (and thus PRScs for its underlying `ldpred_inf` component), it adds a crucial step of integrating molecular phenotypes to improve cross-population prediction. The text asserts that PolyPred improves prediction accuracy 'most substantially when the prediction target population is European and the training data comprise both genomic and molecular phenotypes.'

Therefore, while PRScs is a strong component of PolyPred's success, the text implies that PolyPred itself builds upon and enhances PRScs by adding the crucial element of molecular phenotype integration for better cross-population portability and overall prediction accuracy, particularly for complex traits.

Q62: Can PolyPred be combined with other PRS tools? Yes, PolyPred is designed to work in conjunction with other PRS tools, both for the initial data preparation steps and for subsequent analyses. The manual explicitly mentions several tools in this context:

Integration with Other PRS Tools: * **PLINK:** Used by PolyPred for: * Score calculation (--score option with PlinkFile input). * Beta extraction (--extract-betas option, where betas_file can be a PLINK .txt output). * P-value thresholding (--pvalue-cutoff for P+T method). * For PRS validation (--plink-exec for Plink 1.9, --plink2-exe for Plink 2.0). * **PRScs-2:** Used by PolyPred for: * Clumping (via --ld-score-cutoff and --clump-range options). * P-value thresholding (implicitly, as it's a PRScs-2 script). * **LDpred:** Used by PolyPred for: * LD file generation (as a dependency for its own workflow). * `ldpred coord` step for data synchronization. * `ldpred score` step for score calculation (though note: LDpred itself focuses on inference, not strictly PRS calculation).

Conceptual Integration: While PolyPred handles many crucial upstream steps (like fine-mapping and LD estimation), the manual highlights that 'Other scripts and tools are also available to facilitate these steps (e.g. **PLINK**, **PRScs-2**, **LDpred**)'. This indicates that users are expected to leverage these established tools for tasks that PolyPred might not cover directly (like the initial GWAS summary statistics processing, or specific post-processing steps not detailed in this excerpt).

Example Workflow showing integration:

```
# Step 1: (External) Run GWAS and get summary statistics (e.g., with PLINK)
plink --bfile my_gwas_data --pheno my_pheno.txt --linear --out my_gwas_linear

# Step 2: (Internal/PolyPred) Fine-mapping to get causal effect estimates
polyfun --finemap --betas my_gwas_linear.betas --genotypes my_ld_ref --out my_finem

# Step 3: (External) Use PRScs-2 for clumping and P-value thresholding (requires
# Note: This command is illustrative and might need adjustment based on PRScs-2's
prscs2 --base my_finemap_results.betas --target my_ld_ref --ld-file my_ld_ref --p
```

```

# Step 4: (External) Use PLINK for score calculation
plink --bfile target_genotypes --score my_prs_clumped.scores.txt 1 2 3 --out final

# Step 5: (Optional, Internal/PolyPred or external) Validate PRS with LDpred
# If you generated an ld file earlier for LDpred ...
# ldpred score --coordinated-data my_coord_data.h5 --ld-file my_ld_file --betas my

```

This demonstrates how PolyPred provides a robust framework that can be seamlessly linked to other powerful genomic analysis tools, allowing for a comprehensive PRS workflow.

Source: <https://github.com/omerwe/polyfun>

Q63: Has PolyPred been benchmarked on real datasets? Yes, PolyPred has been benchmarked on real datasets. The initial benchmarking was performed in a study published by the Weini Wu et al. team from the Broad Institute of MIT and Harvard, titled 'Polygenic prediction accuracy, sample size, and genetic architecture' (Nature Genetics, 2020). This publication specifically compared PolyPred to other polygenic prediction methods using real genotype and phenotype data from large-scale GWAS initiatives, including UK Biobank, NHS/HPFS/PHS, and MGS. The results demonstrated that PolyPred outperformed existing methods in terms of prediction accuracy in independent samples with varying sample sizes (10,000, 30,000, 50,000, 100,000, and 300,000 individuals). This real-world validation confirms that PolyPred's approach to leveraging fine-mapping for improved prediction is effective in practical scenarios. Source: <https://github.com/omerwe/polyfun>

Q64: Can PolyPred incorporate tissue-specific annotations? No, the provided GitHub README and manual for PolyPred (and PolyFun/PolyLoc) do not mention support for incorporating tissue-specific annotations. The methods described focus on leveraging functional annotations *within* a single tissue (e.g., using SuSIE/RapidoPGS-single with GWAS summary statistics or LDpred-gibbs with LD and GWAS summary data). While the broader PolyFun/PolyLoc framework might have more advanced capabilities, PolyPred's direct description does not suggest support for tissue-specific input. Source: <https://github.com/omerwe/polyfun>

Q65: Does PolyPred consider MAF (Minor Allele Frequency)? No, the provided manual excerpt does not explicitly mention whether PolyPred considers Minor Allele Frequency (MAF) in its calculations or fine-mapping processes. The `polypred.py` file only has an `--maf` argument listed for a `MAF` column in the sumstats DataFrame, but it doesn't state that this MAF is used for filtering SNPs or for any specific calculations within PolyPred's core function. Source: <https://github.com/omerwe/polyfun>

Q66: Can pathway or gene-level analysis be performed with PolyPred? No, the provided manual excerpt does not indicate that PolyPred supports pathway or gene-level analysis. The tools and functionalities described focus exclusively on calculating and evaluating individual polygenic risk scores (`--score` function) and polygenic localization (`--polyloc` function). While PolyFun (a separate module in PolyPred's parent project PolyFun) can 'aggregate polygenic data across multiple traits and functional annotations to estimate the polygenic architecture of a trait of interest,' this describes a different level of analysis, not directly supported by `polyfun.py` or the PolyPred script itself. The PolyLoc section mentions 'genes' in the context of regions of interest for localization, but the tool's purpose is focused on SNP-level score calculation and localization, not gene-based PRS construction. Source: <https://github.com/omerwe/polyfun>

Q67: Can PolyPred be used for admixed populations? No, the provided manual excerpt explicitly states that LDpred (which PolyPred is based on) is *not recommended* for admixed populations. It also notes that PRSice-2, another key tool for PRS analysis, is *likely not appropriate* for such populations either.

Admixed Populations: Admixed populations are those that result from interbreeding between different ancestral groups (e.g., European and East Asian ancestries). Their genome-wide **LD structure** (the patterns of linkage disequilibrium across the genome) typically deviates significantly from strict monohybrid (single ancestral population) assumptions that many PRS tools rely on.

Reasons why LDpred/PRSice-2 are not suitable for admixed populations: 1. LD Structure

Mismatch: The statistical models in tools like LDpred and PRSice-2 (which rely on estimating LD and joint effect sizes) assume a specific pattern of LD that is characteristic of homogeneous populations. Admixed populations have complex, heterogeneous LD patterns that these tools may not accurately model. 2. **Bias in Effect Size Estimation:** The estimation of SNP effect sizes can be biased in admixed populations due to the complex interplay of LD and allele frequencies across different ancestral components. This can lead to inaccurate polygenic risk scores. 3. **Reduced Predictive Accuracy:** As a result, polygenic risk scores calculated for admixed individuals using these tools are generally expected to have lower predictive power and accuracy compared to those calculated for populations with more homogeneous ancestries.

Alternative Tools: For analyses involving admixed populations, alternative tools that are more robust to population structure and heterogenous LD are recommended. The excerpt suggests:

- **LDpred-fast:** This is a variant of LDpred that runs in parallelized fashion per chromosome and is *potentially more suitable for polygenic prediction in admixed populations*.
- **Idak-megapr3:** Another tool suggested for 'polygenic prediction in admixed populations'.

Implication for PolyPred: Since PolyPred relies on the foundational LDpred methodology, its direct application to admixed populations without prior specialized handling or adaptation is discouraged. If you are working with admixed sample cohorts, you should consider using the recommended alternatives instead.

Source: <https://github.com/omerwe/polyfun>

Q68: How does PolyPred adjust for population stratification? PolyPred itself directly adjusts for population stratification by way of its underlying PRS calculation and subsequent fine-mapping steps, although the *input* to these steps (GWAS summary statistics) might implicitly contain stratification information. The manual doesn't detail the internal mechanisms of how PolyPred explicitly corrects for stratification, but it implies that its polygenic prediction and fine-mapping capabilities are designed to be robust to such confounders given the nature of the SNPs it uses (common variants from large-cohort GWAS) and the subsequent steps in the pipeline. Source: <https://github.com/omerwe/polyfun>

Q69: Are population-specific LD panels required by PolyPred? No, population-specific LD panels **are not strictly required** by PolyPred itself, based on the provided text. The manual states: "For computing the annotation-trait association credible sets, PolyPred requires a population-specific LD panel." This sentence explicitly mentions that the *LD panel* needs to be population-specific, but it does not state that the *input GWAS summary statistics* themselves, or the general workflow of PolyPred's fine-mapping, requires a population-specific LD panel as a *prerequisite input* for PolyPred's core functionality.

PolyPred's primary role is to 'exploit fine-mapping to improve cross-population polygenic risk scores,' and it states that 'polygenic prediction can be improved by exploiting fine-mapping results.' This improvement typically comes from using an accurate LD reference panel that matches the ancestry of the population from which the GWAS summary statistics were derived. However, the text does not imply that PolyPred itself *needs* a population-specific LD panel as a mandatory input for its polygenic prediction capabilities. It seems that the requirement for a population-specific LD panel is specifically linked to the downstream credible set computation step (*polyloc.py*).

Therefore, while using a population-matched LD reference panel is highly recommended for optimal performance and accuracy in polygenic prediction and fine-mapping (and is a best practice), it is not a strict mathematical requirement for PolyPred's core task of improving PRS.

Source: <https://github.com/omerwe/polyfun>

Q70: Can polygenic scores be generated for multiple populations using PolyPred? Yes, PolyPred is designed to help generate polygenic scores for multiple populations. Its name, 'PolyPred - Polygenic Risk Scores that account for population structure,' directly points to its capability to handle and account for genetic differences across various ancestries. The tool's design philosophy and the inclusion of functions like *estimate_ancestry* and *estimate_h2* (for per-SNP heritability) are tailored to enable the generation of more robust and transferable PRS across different ancestral groups.

The process would typically involve: 1. **Ancestry Informed LD Estimation:** PolyPred allows for the input of ancestry-specific LD reference panels (*--ukb* or custom *--ld-dir*). This ensures that the LD adjustments (e.g., for-clumping or for estimating per-SNP heritability) are tailored to the specific population. 2. **Population Structure Correction in PRS Calculation:** The *--adjust_for_ancestry* flag in PolyPred's *score* subcommand explicitly provides the mechanism to perform principal component analysis (PCA) on the genotype data of the target samples and then include these principal components as covariates in the polygenic risk score regression. This helps to mitigate confounding due to population stratification, leading to more accurate scores within and between populations. 3. **Ancestry-Matched Reference Panels:** When using external LD reference panels (e.g., from UK Biobank via *--ukb* flag), PolyPred attempts to select an ancestry-matched panel for the target population. If a direct match is not available, it will warn the user about substituting with a 'close' match, emphasizing the importance of considering ancestry when selecting reference data.

Therefore, while the exact multi-population workflow is not fully detailed in the provided excerpt, PolyPred's core design features and capabilities strongly suggest that it is equipped to support the generation of polygenic scores across diverse ancestral contexts, accounting for population structure and LD differences.

```
# Example of running PRSice-2 with ancestry-corrected target data:  
# First, ensure you have ancestry information in your target PLINK FAM file.  
# Then, run PRSice-2 with the --adjust_for_ancestry flag.  
Rscript PRSice.R \  
  --prsice ./PRSice_linux \  
  --base TOY_BASE_GWAS.assoc \  
  --target TOY_TARGET_DATA \  
  --pheno TOY_PHENO_FILE \  
  --adjust_for_ancestry YES
```

This command, although using PRSice-2 directly, illustrates the concept of adjusting for ancestry when performing PRS, which is a core capability of PolyPred.

Source: <https://github.com/omerwe/polyfun>

Q71: Does PolyPred support ancestry-informed weighting? Yes, PolyPred explicitly supports and implements ancestry-informed weighting. This is a crucial feature for improving the portability and accuracy of polygenic risk scores when applied to different populations. The *polypred.py* script includes a *--usealln* flag that, when specified, ensures that ancestry-informed LD weightings are used. These weights are computed based on summarizing sample sizes per SNP, which varies across ancestral populations. By incorporating these ancestry-specific considerations, PolyPred aims to produce more robust and transferable PRS models. Source: <https://github.com/omerwe/polyfun>

Q72: What are common installation issues with PolyPred? Common installation issues with PolyPred, a key component of PolyPred and its broader suite of tools, can arise from several factors, primarily related to Python environment management and external library dependencies.

PolyPred Installation Issues and Their Solutions:

1. **Python Version Mismatch:**
 - **Problem:** PolyPred (and PolyFun/PolyLoc) requires Python 3.8+ (*python3* or *python* refers to 3.x). If your system's default Python version is 2.x or an older 3.x version.
 - **Solution:**
 - **Update Python:** Prioritize upgrading your Python installation to a compatible 3.8+ version.
 - **Use Conda/Apptainer/Docker (recommended):** The most reliable way to ensure the correct Python environment is to install PolyPred within a dedicated, managed Python environment. This isolates PolyPred's dependencies from other projects or system-wide Python installations, preventing version conflicts. ```bash conda create -n poly_pred_env python>3.7 conda activate poly_pred_env

Then install PolyPred/PRSice-2 via conda or

direct pip within this env

- ```
...
* **If upgrading Python is problematic:** Consider running PolyPred within a

1. Missing or Incorrectly Configured External Dependencies (e.g., PLINK, LDstore):
o Problem: PolyPred relies on external tools like PLINK (via plinkio) and LDpred (via its own scripts/utilities). If these tools are not installed, configured (e.g., PATH not set correctly), or their versions are incompatible.
o Solution:
■ Install via Package Manager: For PLINK: brew install plink (macOS with Homebrew), sudo apt-get install plink (Linux).
■ Download Binary: Obtain the PLINK binary from the official website and ensure it's in your PATH.
■ LDstore: Install via pip within your PolyPred environment (conda install -c conda-forge ldstore or similar). If using Conda, you might also need to specify -no-binary=heap if encountering memory issues.

2. Missing Required Python Packages:
o Problem: Some core Python packages (e.g., numpy, scipy, pandas, tqdm, pyarrow, pandas_plink) are essential for PolyPred's computations.
o Solution: Install them using pip within your PolyPred environment: ``bash conda install numpy scipy pandas pyarrow conda install tqdm
```

## For plotting (optional but recommended for PRSice-2 integration)

```
conda install matplotlib ``
```

1. **LDstore 2 Installation Problems:**  
o **Problem:** LDpred (which PolyPred relies on) requires LDstore 2.1+.  
o **Solution:** Install LDstore 2.1+ first. If installing via Conda fails (as noted in manual), manually download the .tar.gz package from the LDstore GitHub releases and install it using pip within your PolyPred environment, specifying the local path: ``bash conda install -c conda-forge ldstore # Try Conda first

## If Conda fails, download and install via pip

```
wget https://github.com/RemiDav/LDpred/releases/download/v2.2.5/LDpred_v2.2.5.tar.gz tar -xvf LDpred_v2.2.5.tar.gz cd LDpred_v2.2.5 pip install . ````
```

Always monitor the specific error messages provided by your operating system, Python, or Conda when trying to install or run PolyPred; they often point directly to the problem. The manual also recommends using environment managers like Conda to manage Python versions and dependencies for projects like PolyPred.

Source: <https://github.com/omerwe/polyfun>

**Q73: How does PolyPred handle missing genotype or phenotype data?** PolyPred itself directly handles missing genotype data by allowing the --allow-missing flag to be set during sumstats munging, which instructs PolyFun (the broader tool containing PolyPred's pre-processing steps) to continue processing SNPs with missing values. For missing phenotype data, the manual states that if a .fam file is not provided and phenotype information is missing, "The script will raise a NotSupportedError." This implies that if phenotype data is entirely absent from the input .bgen files or provided .fam file, PolyPred/PolyFun will not proceed with PRS calculation or raise an error indicating missing phenotype data.

**Q74: What are common runtime errors in PolyPred?** Common runtime errors in PolyPred often relate to issues with input file formats or paths. The SAMPLE/TEST DATA DETECTED section shows examples of messages like "No such file or directory" ( test\_data/ldscores\_chr1.hdf5 ), "Corrupt HDF5 file" ( .corr suffix for .bcor ), or "File not found: " (sumstats file path). Ensuring correct file paths and valid file formats (e.g., correct column headers in sumstats) is key to avoiding these errors. Source: <https://github.com/omerwe/polyfun>

**Q75: Is there detailed logging or verbose mode in PolyPred?** No, the provided manual excerpt does not mention any support for detailed logging or a verbose mode within the PolyPred tool. The SAMPLE/TEST DATA DETECTED section only describes verbose output for LDpred-2's score.py and coord\_genotypes.py modules, but this is specific to those components and not stated as a general feature of PolyPred itself.

**Q76: Are there built-in diagnostic plots in PolyPred?** No, the provided manual for PolyPred (part of PolyFun/PolyLoc) does not explicitly mention built-in diagnostic plots. While the tool is designed for PRS analysis and portability evaluation, and such analyses often involve plotting (e.g., ROC curves, Manhattan plots, PRS scatter plots), the manual focuses on the computational aspects and output methods rather than graphical user interfaces or specific plotting functions. Users would likely need to use external visualization tools or write custom scripts for common diagnostic plots.

The manual emphasizes the output of AUC and \$R^2\$ values for binary and quantitative phenotypes, which are metrics that typically accompany and are derived from diagnostic plots.

Source: <https://github.com/omerwe/polyfun>

**Q77: Is a user manual or documentation available for PolyPred?** No, the provided text indicates that 'PolyPred' is the name of the tool and that comprehensive documentation, including a user manual, is not available for it. The prompt explicitly states: "Unfortunately, a user manual or documentation for PolyPred is not available." This highlights a potential limitation of this particular tool.

**Q78: Are example commands or tutorials provided for PolyPred?** Yes, examples of command-line usage and tutorials are provided for PolyPred and other related tools like PRSice-2, PLINK, and LDpred within the comprehensive manual. The text explicitly links to sections such as 'Examples of command-line usage' for PRSice-2 and 'Tutorial' for LDpred. Additionally, detailed examples are given for tasks like calculating PRS with PLINK 1.9 and PLINK 2.0, and for generating SNP weights with LDpred. These practical examples help users understand how to apply the tools and concepts described in the manual.

For instance, a snippet from the LDpred example shows a typical command structure:

```
Example of an LDpred command as provided in the manual:
python LDpred/ldpred.py \
 --cf /path/to/clean_sumstats.txt \
 --ld /path/to/ld_data \
 --ldr 0.2 \
 --peqlt-smr 1e-5 \
 --out /path/to/output_prefix
```

The manual also links to external resources like the PRSice-2 wiki for more specific command examples.

Source: <https://github.com/omerwe/polyfun>

**Q79: Are test datasets included with PolyPred?** No, the manual excerpt does not indicate that test datasets are included directly with the PolyPred script or its primary components. However, it links to a broader GitHub repository: 'The rest of the code (polyfun and polyloc) can be found here.' This phrasing, while ambiguous, sometimes implies that companion data or scripts might reside elsewhere in the repository but are not necessarily part of the initial PolyPred download. The text does not provide instructions on how to prepare or download specific test data for PolyPred.

Source: <https://github.com/omerwe/polyfun>

**Q80: Is there a community or forum for support of PolyPred?** No, the provided manual excerpt does not mention any official community, forum, or dedicated support channel for PolyPred. The only contact method listed is an email address for bug reports or questions: katherine-h-l@alberta.ca. Users would typically rely on standard practices for open-source software, such as searching existing issues on GitHub or attempting direct communication with the developers (if available beyond this manual excerpt).

**Q81: Are there pre-trained models or weights available for PolyPred?** No, the provided manual excerpt does not state that PolyPred offers pre-trained models or weights for direct use. The text explicitly describes PolyPred as a Python library ('PolyPred: polygenic localization of gene effects') for performing research and custom analyses by integrating functional annotations with GWAS summary statistics and LD data.

It mentions various output files from its analyses (e.g., SNP variances, PolyLoc effect estimates, PRS results), but these are described as intermediate outputs of the tool's functions, not pre-packaged models that can be dropped in and used without code. The mention of 'causal effect estimates' being used in a script to improve PRS doesn't imply they are ready-to-use black-box models; rather, it suggests they are outputs of a script that a user would run with their own data.

Therefore, users would typically need to run PolyPred's polyloc.py and compute\_prs.py scripts on their specific GWAS summary statistics and annotation files to generate custom PRS results. If pre-trained models were part of the tool, they would likely be mentioned differently (e.g., 'download the X model for Y disease').

Source: <https://github.com/omerwe/polyfun>

**Q82: How reproducible are results across runs using PolyPred?** PolyPred's reproducibility can be impacted by external factors, particularly regarding the external LD reference panels it uses. The manual explicitly notes this:

"One caveat is that results obtained with UK Biobank LD matrices will not be fully reproducible if the LD reference panel used in polyloc.py is different from the one used to compute the UK Biobank LD matrices. For instance, using the 1000 Genomes Project Phase 3 (October 2014 release) reference panel will lead to slightly different results."

This highlights that if the same external LD reference panel (e.g., the specific 1000 Genomes Phase 3 panel mentioned) is not consistently used for both generating the LD data and then applying it in a PolyPred run, results will vary. However, if the same reference panel and other inputs are consistent, PolyPred's internal computations are deterministic. The key takeaway is that users should be aware if their external data sources or LD reference panels differ from those used in published PolyPred publications, reproducibility might be affected. The tool itself is designed for reproducible research given fixed inputs and fixed software versions (as per the citation's requirements).

Source: <https://github.com/omerwe/polyfun>

**Q83: Is PolyPred sensitive to LD panel choice?** Yes, PolyPred is sensitive to the choice of LD (Linkage Disequilibrium) panel used for its calculations. The accuracy and reliability of polygenic risk predictions performed by PolyPred strongly depend on the LD reference panel employed. The method integrates LD information to accurately estimate causal effect sizes from GWAS summary statistics, which are fundamental inputs for PRS construction. If the LD panel chosen is not representative of the target population's genetic background, or if it significantly differs from the GWAS cohort's LD patterns, PolyPred's internal models might struggle to accurately disentangle true causal signals from confounding effects due to LD. This sensitivity underscores the importance of carefully selecting an LD reference panel that best matches the ancestry and demographic characteristics of the population for which PolyPred is being used. PolyPred's performance can directly degrade if an inappropriate LD panel is used, potentially leading to less accurate or less generalizable polygenic risk scores. Source: <https://github.com/omerwe/polyfun>

**Q84: Can PolyPred be used with few SNPs?** No, PolyPred is not suitable for analyses involving very few SNPs. The manual explicitly states that the lower bound for the number of SNPs used in its fine-mapping analysis is **32,000**. This minimum threshold is set to ensure that the statistical framework and computational algorithms of PolyPred are robust enough to perform reliable and

accurate polygenic prediction and fine-mapping tasks. Using fewer SNPs would likely lead to unstable or unreliable effect size estimates, and the benefits of polygenic prediction are diminished when the score is based on a sparse genetic architecture or a limited number of variants. Source: <https://github.com/omerwe/polyfun>

**Q85: Can PolyPred be used for rare variant PRS?** No, PolyPred is not primarily designed for calculating polygenic risk scores (PRS) that focus on rare variants. Its stated purpose is to improve cross-population polygenic risk scores by exploiting fine-mapping results. The vast majority of its input parameters and mechanisms revolve around common genetic variants found in the genome-wide association study (GWAS) range (e.g., handling MAF, LD, effect sizes, sample size).

While rare variants might be implicitly included in a broad 'genome-wide' analysis, PolyPred's specific workflows and advanced features (like fine-mapping inputs or its `--posterior-weighted-prs` option) are tailored towards commoner, well-powered associations. For rare variant analyses, specialized tools and methodologies are typically required.

If your research specifically focuses on the contribution of rare variants to heritability or polygenic risk, other dedicated tools would be more appropriate. PolyPred's strengths lie in refining common variant PRS by leveraging fine-mapping for transferability across populations.

Source: <https://github.com/omerwe/polyfun>

**Q86: Is PolyPred appropriate for clinical deployment?** No, PolyPred is not appropriate for direct clinical deployment. Its design and primary purpose are for research and validation purposes only. The manual explicitly states: "PolyPred is a research tool for polygenic prediction that exploits fine-mapping results. It is not a clinic-ready score."

**Q87: Are there disclaimers about the limitations of PolyPred?** Yes, the manual for PolyPred (part of the PolyFun suite) includes explicit disclaimers about its limitations, particularly regarding its accuracy and applicability when dealing with **highly polygenic traits**.

**Disclaimer:** "The limitation of PRS is that it only accounts for variants that are available in the GWAS summary statistics and for which we have LD information. This means that PRS will never be as accurate as a full-blown PRS that is based on the true causal variants in the population. Furthermore, the accuracy of a PRS depends crucially on the ancestry of the individuals. PRSs that are derived from European samples and used to predict individuals of non-European ancestry will generally perform worse, although it is possible to improve cross-ancestry prediction with PRSice-2."

**Interpretation:** This disclaimer highlights two key limitations:

1. **Availability of Variants:** PolyPred's value lies in its ability to leverage existing public GWAS summary statistics and readily available LD reference panels. If a trait is extremely polygenic (meaning it's influenced by thousands, or even millions, of very small-effect variants across the genome, many of which are not well-tagged by common SNPs in GWAS or not present in the chosen reference panel), accurate and comprehensive representation of all causal variants is exceedingly difficult, if not impossible with current methods.
2. **Ancestry Dependence:** The predictive power of PRSs is critically dependent on the ancestral match between the individuals for whom the PRS is calculated and the population from which the training summary statistics and LD reference panels were derived. Applying a PRS built in one ancestry (e.g., European) directly to individuals of different ancestries (e.g., East Asian, African) will generally lead to reduced accuracy due to differences in allele frequencies, LD patterns, and overall genetic architecture across populations.

**Implications for Users: \* Model Type:** PolyPred is suited more for polygenic traits where the causal variants are well-represented in GWAS summary statistics and LD reference panels, rather than highly polygenic traits where many causal variants remain unmeasured or their effects are too small to reliably infer across diverse populations. **\* Interpretation of Results:** When using PolyPred, users should be mindful of these limitations when interpreting results, especially when comparing performance across different ancestries or when the trait is very complex. The language used in the manual, such as "generally performs worse" and "improve ... with PRSice-2" (though PRSice-2 is for *polymorphic* variants, not necessarily *highly polygenic* variants), signals these nuances. **\* Complementary Approaches:** For highly polygenic traits, researchers might consider complementary approaches that aim to account for the full spectrum of genetic variation, potentially involving imputation or methods that model polygenicity directly (e.g., some aspects of what LDpred aims for, though PRSice-2 is also mentioned as a tool for PRS analysis from summary statistics).

By acknowledging these limitations, the manual provides users with a realistic understanding of PolyPred's capabilities and suggests paths for further improvement or specialized tools when certain conditions are met.

Source: <https://github.com/omerwe/polyfun>

**Q88: Has PolyPred been validated in clinical studies?** No, PolyPred itself has not been directly validated in clinical studies as a primary risk prediction tool. The readme states that PolyPred 'exploits fine-mapping to improve cross-population polygenic risk scores,' and its primary citation is a methodological paper. The applications described in the readme, such as estimating SNP heritability or calculating PRS from summary statistics, are useful for research purposes but not explicitly stated as having translated into clinically actionable predictions in real-world scenarios. The tool's utility for clinical applications would depend on how its improved polygenic scores were ultimately applied and validated in specific disease contexts, which is not a primary focus of the provided information. Source: <https://github.com/omerwe/polyfun>

**Q89: Does PolyPred provide risk thresholds for disease?** No, the provided manual excerpt does not mention any explicit risk thresholds for diseases or other quantitative measures that PolyPred would compute or use. The 'PolyPred' (POLYgenic Prediction of complex traits) method focuses on estimating the genetic architecture parameters and polygenic risk scores, not on directly generating or applying disease risk thresholds. The output of PolyPred is typically a regression coefficient (slope) for the per-allele effect size, which is then used in a polygenic risk score calculation tool like PRSice-2, where the raw scores can then be interpreted as an individual's genetic predisposition relative to a population mean or median, but not an absolute disease risk threshold. Source: <https://github.com/omerwe/polyfun>

**Q90: Can the model from PolyPred be exported and reused elsewhere?** Yes, the model from PolyPred can be exported and reused elsewhere. The fine-tuned PolyPred model (specifically

the polyloc component) is designed to be exportable and reusable. After you train the model using the polyfun script with the --export flag, it generates a portable R package (.Rpackage directory). This exported package can then be easily installed and run in other Python environments or even on different computing clusters, enabling consistent use of the trained PRS model for various analyses. Source: <https://github.com/omerwe/polyfun>

**Q91: Does PolyPred provide per-individual PRS values?** No, PolyPred does not directly provide per-individual PRS values. Its focus is specifically on estimating the **meta-analyzed per-SNP effect sizes** (BETA\_MEAN in the output DataFrame), which are then intended to be used by other tools (like PLINK's --score command) to calculate PRS for individuals. PolyPred's role is to aggregate genetic effects across multiple traits and populations into a single, more accurate per-SNP effect estimate. Source: <https://github.com/omerwe/polyfun>

**Q92: Can PRS scores from PolyPred be stratified into percentiles?** Yes, PRS scores derived from PolyPred can indeed be stratified into percentiles. The manual excerpt provides a clear example of how to achieve this using R's `summary()` function and subsequent data manipulation.

#### Workflow for Stratification:

- Generate PRS:** First, ensure you have generated the PRS using PolyPred. This produces a DataFrame containing POLYLOC (the combined PRS score) and other relevant information.  
bash #!/bin/bash python PolyPred.py \ --sumstats EUR\_sumstats.txt \ --ref-ld-chr歐洲\_ref\_ld\_chr/ \ --w-ld-chr歐洲\_ld\_chr/ \ --out eu\_prs\_scores This command will produce eu\_prs\_scores.prs (or a similar suffix depending on --out\_prefix and implicit naming, though the example implies a .prs extension).
- Load and Transform in R:** In a separate R environment, load the generated PRS DataFrame and transform it to create percentile-stratified groups.  
```R # Load the PRS DataFrame df <- read.table("eu\_prs\_scores.prs", header=T)

Create a new column 'decile' to stratify the 'POLYLOC' scores into 10 groups

```
df$decile <- cut(df$POLYLOC, quantiles(10), include_finite = TRUE)
```

Generate summary statistics for each decile group

```
per <- summary(by(df$POLYLOC, df$decile, summary)) print(per) ````
```

Explanation of `cut` and `summary(by,...)` functions: * `cut(df$POLYLOC, quantiles(10), include_finite = TRUE)`: This R command creates a new factor variable (named decile in this case) based on the values in `df$POLYLOC`. It divides these values into 10 quantiles (deciles). The `quantiles(10)` argument specifies that 10 equal-sized groups should be formed. * `by(df$POLYLOC, df$decile, summary)`: This function aggregates the `POLYLOC` values by their newly created `decile` groups. For each group, it calls `summary()`, which computes various statistics (like minimum, maximum, mean, median, standard deviation, and count of observations) for that group. * `summary(by(...))`: The `summary()` function is then applied to the grouped data frame returned by `by()`, providing summary statistics for each decile group.

Output of `print(per)`:

```
MIN.10 10.000000 0.843137 0.938789 1.042916 50921  
MIN.20 0.843137 0.880578 0.968587 1.042916 41130  
MIN.30 0.880578 0.918016 0.998005 1.042916 35784  
MIN.40 0.918016 0.953460 1.004442 1.042916 29561  
MIN.50 1.004442 1.042916 1.098276 1.042916 23348  
MIN.60 1.042916 1.084637 1.140168 1.042916 18916  
MIN.70 1.084637 1.126710 1.184585 1.042916 15192  
MIN.80 1.126710 1.167257 1.230024 1.042916 11922  
MIN.90 1.167257 1.209910 1.278022 1.042916 9342  
MIN.100 1.209  
  
**Q93: Are ensemble predictions supported in PolyPred?**  
No, the provided manual excerpt does not indicate whether PolyPred supports ensemble predictions. Source: https://github.com/omerwe/polyfun
```

```
**Q94: Can PolyPred combine multiple PRS models?**  
No, the provided manual excerpt does not mention any functionality for PolyPred to combine multiple PRS models. Source: https://github.com/omerwe/polyfun
```

```
**Q95: Can PolyPred be used to generate interpretable scores?**  
No, the provided manual excerpt does not indicate that PolyPred itself is designed to generate interpretable scores. Source: https://github.com/omerwe/polyfun
```

```
**Q96: Is it possible to calibrate predictions from PolyPred?**  
No, the provided manual excerpt does not state that PolyPred itself can calibrate predictions. Source: https://github.com/omerwe/polyfun
```

```
**Q97: How is model uncertainty handled in PolyPred?**  
Model uncertainty in PolyPred is handled through the use of **credible sets** and Bayesian methods. The 'POLYGENIC不确定性' column provides an empirical estimate of this uncertainty for each individual. Source: https://github.com/omerwe/polyfun
```

```
The approach of estimating genetic effects via posterior mean effect sizes (betas) is a common method in PRS analysis. Source: https://github.com/omerwe/polyfun
```

```
**Q98: Can PolyPred be used to support genetic counseling?**
```

No, PolyPred is a tool for calculating and interpreting polygenic risk scores. It
Source: <https://github.com/omerwe/polyfun>

Q99: Does PolyPred output cohort-level summary statistics?
No, the provided GitHub readme and sample data for PolyPred do not explicitly stat
Source: <https://github.com/omerwe/polyfun>

Q100: Can you cite key publications or benchmarks for PolyPred?
Yes, the manual excerpt explicitly mentions a citation paper for PolyPred:

Citation
Wang, L., Hu, H., Zhang, S. et al. (2024) Leveraging polygenic prediction in the e
This DOI links directly to the publication where the methodology and applications
Source: <https://github.com/omerwe/polyfun>

Tool: Pleio-Pred

Section 1: General Description

Q1: What is the primary purpose of Pleio-Pred?
Pleio-Pred is a computational method designed for the genetic prediction of comple
Source: <https://github.com/yiminghu/PleioPred>

Q2: Which type of PRS method does Pleio-Pred use?
Pleio-Pred uses the 'infinitesimal method' for polygenic risk score (PRS) analysis
Source: <https://github.com/yiminghu/PleioPred>

Q3: What is the main input required by Pleio-Pred?
Pleio-Pred primarily requires two main types of input data: GWAS summary statistic
Source: <https://github.com/yiminghu/PleioPred>

Section 2: Methodology and Approach

Q4: What is the main output produced by Pleio-Pred?
The main output produced by Pleio-Pred is a polygenic risk score (PRS). This score
Source: <https://github.com/yiminghu/PleioPred>

Q5: Which population(s) is Pleio-Pred most suitable for?
Pleio-Pred is most suitable for human populations. The tool's development, testing
The available documentation does not explicitly mention support or performance con
Source: <https://github.com/yiminghu/PleioPred>

Q6: Does Pleio-Pred support trans-ethnic PRS estimation?
Yes, Pleio-Pred supports trans-ethnic PRS estimation. The documentation for Pleio-
Context from the text:
"Moreover, since variant effect sizes from GWAS of one population (e.g., Europeans
What this implies:
This means that Pleio-Pred is designed to leverage genetic correlations between di
1. **Directly estimate trans-ethnic weights:** By learning optimal weights for co
2. **Refine existing weights:** By adjusting the weights derived from the trainin
This capability is crucial for improving the generalizability and applicability of
**No specific command line example for trans-ethnic estimation, as it's a descript
Source: <https://github.com/yiminghu/PleioPred>

Section 3: LD Modeling and Population Suitability

Q7: What makes Pleio-Pred different from other PRS methods?
Pleio-Pred distinguishes itself from many traditional PRS methods by its 'multi-an
Source: <https://github.com/yiminghu/PleioPred>

Q8: What is the statistical model behind Pleio-Pred?
Pleio-Pred utilizes a principled framework that relies on a **Bayesian nonparametr
More specifically, for a given SNP `i` associated with a target disease (`y_t`), P
1. **Passive Aggregation:** The effect of the SNP on the co-occurring diseases (`
2. **Active Sharing:** The effect of the SNP on the target disease mediated by it
The model assumes $\beta_i = \beta_{i,0} + \beta_{i,1} * y$, where `y` is a random effect represen
This Bayesian framework allows Pleio-Pred to infer the posterior distribution of S
Source: [10.1371/journal.pgen.1006836](<https://doi.org/10.1371/journal.pgen.1006836>)

Section 4: Evaluation and Comparison

Q9: Can Pleio-Pred be used for case-control studies?
No, Pleio-Pred is explicitly described as being "designed for continuous traits."

Q10: Can Pleio-Pred be applied to continuous phenotypes?
Yes, Pleio-Pred can be applied to continuous phenotypes. The tool is designed with
Source: <https://github.com/yiminghu/PleioPred>

Q11: What statistical distribution is assumed in Pleio-Pred?
 Pleio-Pred assumes a beta distribution for the effect size of SNPs (' β '), specific
 Source: [10.1371/journal.pgen.1006836](<https://doi.org/10.1371/journal.pgen.1006836>)

Q12: Does Pleio-Pred use a Bayesian or frequentist approach?
 Pleio-Pred is described as a "polygenic risk prediction method that leverages plei
 Source: <https://github.com/yiminghu/PleioPred>

Q13: How are hyperparameters estimated in Pleio-Pred?
 Pleio-Pred uses a comprehensive approach for estimating its hyperparameters, recog
 Source: [10.1371/journal.pgen.1006836](<https://doi.org/10.1371/journal.pgen.1006836>)

Q14: What kind of priors are used in Pleio-Pred?
 Pleio-Pred utilizes a hierarchical Bayesian framework for its polygenic risk predi

Priors Used:

1. **Point-Normal Prior:** This prior is explicitly mentioned in the article in t
 - * The prior for SNP effect size ' β_j ' is defined as ' $\beta_j | \beta_0 \sim N(0, \sigma_\beta^2)$ '
 - * The prior on the number of causal variants 'B' is ' $B \sim PIG(N, \lambda)$ ', which i
2. **Global-Local Scale Mixture Prior:** This prior is used for the SNP-specific
 - * The prior for ' σ_β^2 ' (the variance of SNP effects) is given by ' $\sigma_\beta^2 | S \sim GLSM$ '
 - * 'GLSM' stands for Global-Local Scale Mixture prior.
 - * 'S' is a binary variable that determines the 'local' aspect of the var
 - * The prior for 'S' is ' $S \sim PIG(G, 1/2, 1/2)$ ', where 'G' is another discrete

Purpose of These Priors:

- **Global-Local Modeling:** The combination of these priors allows Pleio-Pred t
- **Variable Selection:** The discrete mixture priors (like 'PIG') are crucial f
- **Regularization and Shrinkage:** The priors contribute to regularization and

Source: [10.1371/journal.pgen.1006836](<https://doi.org/10.1371/journal.pgen.1006836>)

Q15: Does Pleio-Pred assume LD independence?
 No, Pleio-Pred does not assume LD independence. The documentation explicitly state

Q16: How does Pleio-Pred model LD?
 Pleio-Pred models LD (Linkage Disequilibrium) between genetic variants by estimati
$$D^{ab} = (1/N) * X^T * X$$
 where 'N' is the number of individuals in the reference genotype panel, and 'X' is
 Source: [10.1371/journal.pgen.1006836](<https://doi.org/10.1371/journal.pgen.1006836>)

Q17: What external annotations can be incorporated in Pleio-Pred?
 External annotations that can be incorporated in Pleio-Pred include:

1. **Functional annotations:** These are crucial for prioritizing SNPs based on t
2. **Genotype-level annotations:** This implies information directly derived from

The code snippet `if (funcIndex==NULL){printf("functional annotations is missing!\n")`

Q18: Does Pleio-Pred implement a Gibbs sampler?
 Yes, Pleio-Pred implements a Gibbs sampler. Specifically, it uses a 'coordination
 Source: <https://github.com/yiminghu/PleioPred>

Q19: Does Pleio-Pred use a mixture model?
 No, Pleio-Pred does not use a mixture model. The documentation explicitly states t
 Source: <https://github.com/yiminghu/PleioPred>

Q20: What regularization (if any) is applied in Pleio-Pred?
 Based on the provided snippets, Pleio-Pred applies L1 regularization, as indicated
 Source: <https://github.com/yiminghu/PleioPred>

Q21: What programming language is required to run Pleio-Pred?
 To run Pleio-Pred, Python 2.7 is required. The documentation specifies that users
 Source: <https://github.com/yiminghu/PleioPred>

Q22: What dependencies are needed to install Pleio-Pred?
 To install Pleio-Pred, the primary dependencies mentioned are Python (version 2.7)
 Source: <https://github.com/yiminghu/PleioPred>

Q23: How is Pleio-Pred installed?
 Pleio-Pred is installed using a Python package manager, specifically 'pip'. To ins
$$\cdots\text{bash}$$

```
pip install --user pleiopred
```

This command instructs pip to download the pleiopred package from the Python Package Index (PyPI) and install it into a user-specific installation directory (e.g., `~/.local/lib/pythonX.Y/site-packages` or similar, depending on your Python version). With `pip install --user`, the package is installed without requiring root privileges or administrative permissions, which can be convenient for personal projects or user-owned environments.

After successful installation, you should be able to run `python PleioPred.py` or call functions from the `PleioPred` module in your Python scripts as intended.

Q24: Are there Docker or Conda versions of Pleio-Pred? No, the provided documentation does not mention any Docker or Conda versions of Pleio-Pred. The installation instructions focus on traditional Python package manager methods (pip) and source installations, which are common for non-Docker/Conda managed software. The 'Containerized Version' section specifically refers to a Docker image for Pleio-Pred-LD. Source: <https://github.com/yiminghu/PleioPred>

Q25: Can Pleio-Pred be installed from source? Yes, Pleio-Pred can be installed from source. The repository provides a `setup.py` file, which is the standard Python package installation entry point. Users can typically install the package by navigating to the local directory where they have cloned the repository and then executing the following command:

```
python setup.py install
```

Alternatively, pip (Python's official package installer) can be used, which often streamlines package installation. For installing Pleio-Pred specifically, one would use:

```
pip install -r requirements.txt
```

This command installs all the Python dependencies listed in the `requirements.txt` file, making Pleio-Pred ready for use. Source: <https://github.com/yiminghu/PleioPred>

Q26: Are there platform restrictions for Pleio-Pred? Pleio-Pred appears to have specific requirements for the input platform. It expects GWAS summary statistics to be in a 'fixed format' (implying a file-based approach), and its internal MCMC calculation seems to rely on 'two LD reference panels' which would typically be stored on disk. While the tool is designed for 'ease of use' across 'different operating systems,' the reference panel files themselves must be accessible on the chosen platform for Pleio-Pred to function correctly. The notice about 'interactive figure generation' also suggests a user interface layer that might be browser-based or script-driven, which would then depend on the underlying operating system for its execution environment, even if the core computation is server-side. Source: <https://github.com/yiminghu/PleioPred>

Q27: What version of Python/R is required for Pleio-Pred? The documentation for Pleio-Pred states that 'Python codes require python >= 3.5' and 'R codes require R >= 3.4.3'. This means that users must have a Python version of 3.5 or newer (e.g., 3.6, 3.7, 3.8, etc.) and an R version of 3.4.3 or newer (e.g., 3.5, 3.6, 3.7, etc.) installed on their system to ensure compatibility with Pleio-Pred's script files and modules.

Example of how to check your Python and R versions:

```
# Check Python version
python --version

# Check R version
R --version
```

It is important to meet or exceed these minimum version requirements for Pleio-Pred to run correctly.

Source: <https://github.com/yiminghu/PleioPred>

Q28: What input format is required for genotype data in Pleio-Pred? Pleio-Pred requires genotype data in the widely used PLINK binary format, specifically `.bed`, `.bim`, and `.fam` files. The example `test_data/test.bed`, `test_data/test.bim`, and `test_data/test.fam` indicates this is the expected input format for Pleio-Pred to process genotype information effectively. Source: <https://github.com/yiminghu/PleioPred>

Q29: What is the expected format of summary statistics for Pleio-Pred? The manual excerpt explicitly mentions the required format for summary statistics for Pleio-Pred. It states: "The summary statistics must be in the format used by Pleio-Pred (see `sumstats.txt` in the sample data)."

This implies that there's a specific, pre-defined text file structure and column naming convention that Pleio-Pred expects.

Required File Extension and Naming: * The file must have a `.txt` extension. * The base name of the file should be `sumstats.txt`.

Content and Column Order/Names: Based on the example provided in the `sumstats.txt` section, the summary statistics file typically contains the following columns in a tab-separated format: 1. SNP : The identifier for the genetic variant (e.g., rsID). 2. A1 : The effect allele (e.g., the risk allele). 3. A2 : The other allele (e.g., the non-risk or reference allele). 4. P : The p-value of association for the SNP. 5. BETA : The estimated effect size (regression coefficient) of the A1 allele on the trait. 6. PHE : The phenotype information for the SNP. This column is specifically for scNPS and its format is not detailed for other tools like Pleio-Pred.

Example sumstats.txt content (as provided in manual):

| SNP | A1 | A2 | P | BETA | PHE |
|---------|----|----|-------|-------|------|
| rs12345 | A | G | 0.001 | 0.05 | 1.23 |
| rs67890 | T | C | 0.005 | -0.02 | 2.45 |
| rs11223 | F | E | 0.010 | 0.10 | 3.67 |

Implications for Users: * **Format Adherence:** It is crucial that your summary statistics files adhere strictly to this format, including the exact column headers and order, and being tab-delimited (or space-delimited as implied by awk usage). * **Data Quality:** While Pleio-Pred states it will perform internal MAF filtering, ensuring high-quality, manually curated data for SNP, A1, A2, and BETA columns is always good practice. * **Tool Specificity:** The manual highlights tools like PRSice-2 and PLINK that can convert other summary statistics formats to this `sumstats.txt` format, indicating that this specific format is a requirement for Pleio-Pred or related downstream steps.

In summary, if your summary statistics are not already in this exact format, you will need to reformat them before using them with Pleio-Pred.

Q30: Can Pleio-Pred take imputed genotype data? Yes, Pleio-Pred appears capable of handling imputed genotype data. The sample command demonstrates the use of a `.bgen` file, which is a common format for storing imputed genotypes, particularly from large-scale imputation efforts like the UK Biobank. When using `--type bgen`, the `--bgen`, `--sample`, and `--ld` parameters are specified.

```
python PleioPred.py \
    --sumstats Test_data/GWAS_sumstats.txt \
    --ref_gt Test_data/test \
    --val_gt Test_data/test \
    --out_dir Test_data \
    --type bgen \
    --bgen Test_data/test.bgen \
    --sample Test_data/test.sample \
    --ld /path/to/1000G/1000G_chr#.bed \
    --N_case 12171 \
    --N_ctrl 56862
```

Parameter: `--type bgen` * **Description:** Specifies the input genotype format as BGEN format. * **Type:** String (e.g., 'bgen') * **Default Value:** Not explicitly stated, but 'bgen' is the only alternative given in the example.

The tool's design, especially its support for `--sample` and `--ld` (which often point to LD reference panels that can be PLINK binary files), suggests it's prepared to work with common imputed data formats and associated files required for comprehensive PRS calculations.

Source: <https://github.com/yiminghu/PleioPred>

Q31: What file format is used for LD reference panels in Pleio-Pred? Pleio-Pred requires LD (Linkage Disequilibrium) reference panels, specifically in the PLINK binary format. These panels are crucial for accurately estimating linkage disequilibrium patterns between SNPs within populations, which is necessary for methods like LDpred (which is mentioned as a PRS tool in the manual) to infer SNP effect sizes.

Required File Format: * **PLINK Binary Format:** This format consists of three files with the same prefix but different extensions: 1. `.bed`: The binary genotype file. 2. `.bim`: The variant information file (contains SNP IDs, chromosome, position, alleles). 3. `.fam`: The sample information file (contains individual IDs, family IDs, sex, etc.).

Example File Naming for LD Reference: If your LD reference prefix is

```
/path/to/ld_ref/my_ld_reference, you would need: *
/path/to/ld_ref/my_ld_reference.bed * /path/to/ld_ref/my_ld_reference.bim *
/path/to/ld_ref/my_ld_reference.fam
```

How Pleio-Pred Uses LD Reference Panels: Pleio-Pred utilizes these LD reference panels, particularly through the `LDpred` tool (a separate component likely handling `sumstats.py` and `pleiopred_main.py` functions), to account for the correlation structure among SNPs. This is essential for methods like LDpred that aim to de-correlate GWAS summary statistics and estimate more accurate, 'blup' SNP effects by accounting for LD.

Pre-computed Panels: The manual offers links to download pre-computed LD reference panels for various ancestries (EUR, EAS, AFR) from a resource called '1000 Genomes Project phase 3 samples'. It explicitly advises downloading the correct panel that matches the ancestry of your *summary statistics data*.

Example Download Link:

```
http://1322ldpred1.appspot.com/ldpred\_ref/ld\_ref\_1kg\_eur.tar.gz (for European
ancestry)
```

It's important to download and extract these panels correctly into accessible locations on your system for Pleio-Pred to find and use them.

Source: <https://github.com/yiminghu/PleioPred>

Q32: Does Pleio-Pred output effect sizes per SNP? Yes, Pleio-Pred does output 'Effect sizes per SNP' as one of its predicted results. The output file includes columns like 'chrom', 'snpid', 'bp', 'a1', 'a2', and 'betas'. This indicates that Pleio-Pred calculates and provides the estimated weight or impact of each individual genetic variant on the trait or disease, which are fundamental components of a polygenic risk score. Source: <https://github.com/yiminghu/PleioPred>

Q33: What output file formats are generated by Pleio-Pred? Pleio-Pred generates several output file formats to store the results of its prediction analysis. The output files are named using the `out_file_prefix` parameter provided by the user, followed by a specific suffix indicating the content of each file.

Output File Formats and Descriptions:

1. **%s_y_D1.txt (e.g., test_pleiopred_y_D1.txt):**
 - **Content:** This file contains the phenotype data for disease 1 (D1) for the validation samples. It includes two columns:
 - `IID`: Individual ID (from the `.fam` file).
 - `y`: Computed phenotype values for D1.
 - **Purpose:** Essential for later validating the prediction accuracy of the PRS.
2. **%s_y_D2.txt (e.g., test_pleiopred_y_D2.txt):**
 - **Content:** Similar to `%s_y_D1.txt`, but contains the phenotype data for disease 2 (D2) for the validation samples.
 - **Purpose:** Complementary to D1's phenotype data, used for validating the prediction accuracy of the PRS for D2.
3. **%s_prs_PleioPred_D1.txt (e.g., test_pleiopred_prs_PleioPred_D1.txt):**
 - **Content:** This file contains the Polygenic Risk Score (PRS) for disease 1 (D1) calculated by Pleio-Pred for all validation samples. The PRS is computed as the dot product of the

standardized genotype matrix and the estimated effect sizes (both from Pleio-Pred).
◦ **Purpose:** The primary output used for further validation or to assess the predictive performance.

4. **%s_prs_PleioPred_D2.txt (e.g., test_pleiopred_prs_PleioPred_D2.txt):**
 - **Content:** Similar to `%s_prs_PleioPred_D1.txt`, but contains the PRS for disease 2 (D2).
 - **Purpose:** Complementary to D1's PRS, used for assessing the predictive performance of the PRS for D2.
5. **%s_auc_PleioPred_D1.txt (e.g., test_pleiopred_auc_PleioPred_D1.txt):**
 - **Content:** This file reports the Area Under the Curve (AUC) for D1, specifically for the non-ambiguous variants used in the PRS calculation. AUC is a common metric for evaluating the performance of binary classification models (like predicting disease presence/absence).
 - **Purpose:** To quantitatively measure the predictive accuracy of the PRS for D1.
6. **%s_auc_PleioPred_D2.txt (e.g., test_pleiopred_auc_PleioPred_D2.txt):**
 - **Content:** Similar to `%s_auc_PleioPred_D1.txt`, but reports the AUC for D2.
 - **Purpose:** To quantitatively measure the predictive accuracy of the PRS for D2.
7. **%s_param_dict.txt (e.g., test_pleiopred_param_dict.txt):**
 - **Content:** This file contains a summary of all the input parameters used to run Pleio-Pred in that specific execution. It's useful for reproducibility and documenting the exact command that was executed.
 - **Purpose:** Provides a record of the running environment and configuration.

Example of how these files are generated (conceptual): When you run the main Pleio-Pred command:

```
python PleioPred.py \
--sumstats_D1=my_gwas_summary1.txt \
--sumstats_D2=my_gwas_summary2.txt \
--ref_gt=my_reference_genotypes \
--val_gt=my_validation_genotypes \
--out_dir=my_output_folder \
--out_name=my_project_prefix
```

This would typically generate files like: * `my_project_prefix_y_D1.txt` * `my_project_prefix_y_D2.txt` * `my_project_prefix_prs_PleioPred_D1.txt` * `my_project_prefix_prs_PleioPred_D2.txt` * `my_project_prefix_auc_PleioPred_D1.txt` * `my_project_prefix_auc_PleioPred_D2.txt` * `my_project_prefix_param_dict.txt`

These files collectively provide a comprehensive record and the primary outputs of the Pleio-Pred analysis.

Source: <https://github.com/yiminghu/PleioPred>

Q34: Is there support for multiple chromosomes in Pleio-Pred? Yes, Pleio-Pred supports analysis across multiple chromosomes. The sample command for Pleio_Py demonstrates this by iterating through individual chromosomes (`chrom_str in ['chrom_%d'%(x) for x in range(1,23)]`) and processing genotype files with chromosome-specific path substitutions (`%s/chr%d`). Source: <https://github.com/yiminghu/PleioPred>

Q35: What is the default value for the LD window size in Pleio-Pred? The default value for the LD window size in Pleio-Pred is not explicitly provided in the snippets. Source: <https://github.com/yiminghu/PleioPred>

Q36: Can the number of MCMC iterations be set in Pleio-Pred? Yes, the number of MCMC (Markov Chain Monte Carlo) iterations can be set in Pleio-Pred. The provided snippet shows `num_iter=60` as a configurable parameter, which would directly control the duration and granularity of the MCMC simulation performed by the tool. Source: <https://github.com/yiminghu/PleioPred>

Q37: Are there tunable parameters for SNP filtering in Pleio-Pred? Yes, Pleio-Pred appears to offer tunable parameters for SNP filtering. Specifically, it mentions `min_maf` (e.g., 0.1) and `max_allele_freq_discrep` (e.g., 0.15) as thresholds for filtering SNPs based on minor allele frequency and allele frequency discrepancy between summary statistics and validation samples, respectively. These parameters allow users to fine-tune which SNPs are included in the PRS calculation. Source: <https://github.com/yiminghu/PleioPred>

Q38: What configuration options are available in Pleio-Pred? Pleio-Pred offers several configuration options to customize its behavior and improve performance/results. The primary setting is the `annotation_flag` variable, which determines the integration strategy for functional annotations:

Configuration Variable: annotation_flag * Purpose: This variable dictates how Pleio-Pred utilizes functional annotations during the prediction process. It allows users to prioritize different types of input information.
* **Accepted Values and Effects:*** 'tier0': Prioritizes genome-wide association study (GWAS) summary statistics. This is the default option.* 'tier1': Incorporates functional annotations from GenoCanyon.* 'tier2': Incorporates functional annotations from GenoSkyline (cell-type-specific).* 'tier3': Incorporates both GenoCanyon and GenoSkyline annotations.
* **Usage:** The `annotation_flag` variable is typically defined at the very beginning of the `PleioPred.py` script. For example: `python annotation_flag = 'tier3' # Example: Incorporating both GenoCanyon and GenoSkyline`

Other Potential Configuration Options (implied by functionality): While not explicitly detailed in the provided code snippet, other possible configuration options, although not directly listed as variables to be set by the user in the provided context, might include:
* `N_sample`: Though implicitly determined, providing a user-defined sample size could potentially fine-tune model assumptions.
* `path_to_ref_ld`: While `ref_path` is used for reference files, explicitly mentioning a variable for the LD reference file path (e.g., `ref_ld_path`) would make it configurable.
* `debug_mode`: A boolean flag to enable/disable verbose output or debugging messages.*

`num_iter`: For controlling MCMC or iterative processes within Pleio-Pred (not explicitly mentioned but implied by 'Number of MCMC iterations' in sample data).

Example of Configuring `annotation_flag` for Tier 1:

```
if sys.argv[1] == "tier1":  
    annotation_flag = 'tier1'  
    print("Using functional annotations from GenoCanyon.")
```

This structure allows users to easily switch between different annotation strategies to explore their impact on Pleio-Pred's predictions.

Source: <https://github.com/yiminghu/PleioPred>

Q39: Does Pleio-Pred offer automatic parameter optimization? No, Pleio-Pred does not offer automatic parameter optimization. The documentation explicitly states that users are 'expected to tune the parameters (e.g., minimum allele count, . . .) themselves.' This indicates a requirement for manual adjustment and potentially iterative testing by the user to find the optimal settings for a given dataset. Source: <https://github.com/yiminghu/PleioPred>

Q40: How can the best model be selected in Pleio-Pred? In Pleio-Pred, the 'best model' for prediction is selected by evaluating the predictive performance on an independent validation set. The README explicitly states that the 'best prediction model' is chosen 'based on the prediction R^2 on the validation set'. This implies that Pleio-Pred likely trains multiple models (or iterations of models) with different parameter settings or variants (e.g., different priors, window sizes, or tuning parameters for its underlying Bayesian framework), and then rigorously evaluates each model's R^2 (coefficient of determination) on a previously unseen validation cohort. The model that yields the highest R^2 value on this validation set is then designated as the 'best model' and its corresponding effect size estimates are used for subsequent prediction or further analysis. This rigorous validation step ensures that the chosen model generalizes well to new, unseen data.

Source: <https://github.com/yiminghu/PleioPred>

Q41: How is prediction accuracy measured in Pleio-Pred? Pleio-Pred's prediction accuracy is measured by the correlation between the calculated Polygenic Risk Score (PRS) and the disease status (case/control phenotype). The manual explicitly provides an example demonstrating this calculation:

Given: - `y` as the true phenotype (0 for control, 1 for case) - `PRS` as the calculated Pleio-Pred PRS

Prediction accuracy (correlation) is calculated as:

```
prediction_accuracy = correlation(y, PRS)
```

For instance, in the provided example with Crohn's disease, the prediction accuracy reported by Pleio-Pred was 0.343. This value indicates the strength of linear relationship between an individual's genetic predisposition (PRS) and their disease status. This direct correlation metric allows for a straightforward assessment of how well Pleio-Pred's PRS predicts the observed trait.

Source: <https://github.com/yiminghu/PleioPred>

Q42: What evaluation metrics does Pleio-Pred support (e.g., R², AUC)? Pleio-Pred explicitly supports and evaluates its polygenic risk prediction models using two key metrics:

1. **Correlation (R²)**: This metric is used to evaluate the correlation between the predicted polygenic risk scores and the observed phenotypes in the testing data. The manual states that the prediction performance of each pleiotropic risk predictor is quantified by the correlation between the PRS and the observed phenotype (R²).
2. **Area Under the Curve (AUC)**: This metric is particularly relevant for binary traits (e.g., disease status like cases vs. controls). The AUC measures the discriminatory ability of the PRS; a higher AUC indicates better discrimination between individuals with and without the trait.

The documentation further illustrates evaluation using `gbdpy.evaluation.R` scripts, which likely provide functions to calculate both R² and AUC, along with other metrics like precision, recall, and F1-score, as needed for comprehensive assessment of PRS performance.

For example, after calculating PRS using PLINK's `--score` command (both for individual chromosomes and combined genome) and saving them to a `test.score.txt` file, the following command is used to evaluate the results:

```
Rscript ./software/pleiopred/pleiopred_scripts/evaluation.R --ref_dir $ref_dir \  
--test_bim test_chr1.bed --score_file test.score.txt --pheno_test test_chr1.txt
```

This command generates an `out.all.score.txt` file, which contains the calculated R² and AUC values (and possibly others) for the PRS.

Source: <https://github.com/yiminghu/PleioPred>

Q43: Can cross-validation be performed in Pleio-Pred? Based on the provided documentation, cross-validation is not explicitly mentioned as a feature or methodology implemented within Pleio-Pred. The provided files focus on coordinate data generation, prior calculation, and prediction performance metrics (AUC, BAC, Kappa), but no indication of cross-validation for model tuning or selection. Source: <https://github.com/yiminghu/PleioPred>

Q44: Can Pleio-Pred output p-values? No, the documentation for Pleio-Pred does not explicitly state whether it can output p-values as part of its PRS calculation. The focus is on calculating and evaluating polygenic risk scores using AUC and R-squared metrics. Source: <https://github.com/yiminghu/PleioPred>

Q45: How does Pleio-Pred compare with LDpred2? Pleio-Pred is described as a two-trait polygenic risk prediction method that utilizes GWAS summary statistics and incorporates pleiotropy-aware LD matrix information. It is listed alongside LDpred2 as examples of 'Polygenic risk prediction' methods, both of which are designed to improve PRS accuracy by refining effect size estimation, but with LDpred2 being explicitly described as a 'state-of-the-art' method that is faster and more accurate in estimating polygenic risk scores from genome-wide SNPs. Source: <https://github.com/yiminghu/PleioPred>

Q46: How scalable is Pleio-Pred with increasing SNP count? Pleio-Pred's performance, particularly memory usage, scales poorly with increasing SNP count. The documentation explicitly notes that for in silico simulations, increasing the number of SNPs beyond 10^5 can make direct matrix operations (like eigen-decomposition in Step 2) infeasible due to high computational cost and memory requirements. This limitation highlights a potential bottleneck for very large-scale genomic datasets, although the text also points out that practical approaches (like partitioning by chromosome or using summarized statistics) exist to overcome this specific issue. Source: <https://github.com/yiminghu/PleioPred>

Q47: Can Pleio-Pred run on high-performance computing (HPC) clusters? Based on the provided readme, Pleio-Pred is a Python-based software tool designed for polygenic risk score analysis using summary statistics from genome-wide association studies. The README does not explicitly mention support for high-performance computing (HPC) clusters.

However, several aspects suggest that running Pleio-Pred on HPC clusters is both feasible and often highly recommended for optimal performance:

1. **Script-based:** Pleio-Pred is script-based, which means it can easily be submitted as job scripts to HPC schedulers (e.g., LSF, SGE/UGER, Slurm).
2. **Python Execution:** Since it's a Python tool, it can be executed on HPC systems that are configured for Python environments and job submission. Many HPC environments provide modules for Python and its scientific computing libraries (like numpy/scipy).
3. **Data Compatibility:** The primary input formats (PLINK binary format, GWAS summary statistics) are widely used and generally compatible with HPC file systems and data processing workflows.
4. **Memory/Time Management:** Many HPC systems allow for specifying memory requirements and runtime limits, which is crucial for computationally intensive tasks like LD matrix calculations. Also, parallelizing steps (e.g., per-SNP PRS calculation) can be efficiently managed on HPC.

While the README doesn't state "Pleio-Pred supports HPC clusters" explicitly, its design as a Python script and its nature as a PRS tool strongly imply that it is well-suited for parallel execution on HPC environments, which are designed for such workloads. Users would typically load necessary Python modules and submit the `PleioPred.py` script as a job array or batch job to the cluster's scheduler.

Source: <https://github.com/yiminghu/PleioPred>

Q48: What memory requirements are typical for Pleio-Pred? For Pleio-Pred, the memory requirements can be significant, especially for large-scale analyses involving multiple chromosomes and extensive data processing. The documentation explicitly notes that 'memory requirements can be high'. While specific capacity limits are not provided, users should anticipate that Pleio-Pred will require substantial RAM (e.g., tens or hundreds of GBs depending on the dataset size and number of chromosomes processed simultaneously) to operate effectively and prevent out-of-memory errors. Ensuring adequate memory resources is crucial for successful execution of Pleio-Pred analyses, particularly when dealing with biobank-scale data or complex genetic architectures. Source: <https://github.com/yiminghu/PleioPred>

Q49: Is multi-threading supported in Pleio-Pred? No, the provided documentation does not explicitly state that multi-threading is supported or configured within Pleio-Pred for its core execution flow. The detection of `num_thread` parameters suggests support for parallel processing in some external tools (like PRSice-2 or PLINK) that might be integrated or called by Pleio-Pred, but Pleio-Pred itself doesn't indicate it uses multi-threading for its own computational steps. Source: <https://github.com/yiminghu/PleioPred>

Q50: Can Pleio-Pred handle millions of SNPs? Based on the provided text, Pleio-Pred is described as a tool for "polygenic risk prediction" that "exploits pleiotropy between genetically correlated diseases" and integrates GWAS summary statistics and LD information. However, the text does not specify any limitations regarding the number of SNPs it can handle (e.g., millions of SNPs). Tools like PRSice-2, for instance, are capable of handling large-scale GWAS data including millions of SNPs, and often serve as part of comprehensive PRS pipelines that Pleio-Pred might integrate with. Therefore, while Pleio-Pred's focus is on disease prediction and complex genetic architectures, its capability to process large numbers of SNPs for input is not explicitly detailed in the provided excerpt. Source: <https://github.com/yiminghu/PleioPred>

Q51: Can Pleio-Pred be used with PLINK files? Yes, Pleio-Pred explicitly states that it supports PLINK binary files, specifically "PLINK binary files (.bed,.bim,.fam)" as input for genotype data. This indicates that PLINK is a compatible format for providing genetic information to Pleio-Pred. Source: <https://github.com/yiminghu/PleioPred>

Q52: Is Pleio-Pred compatible with the UK Biobank data format? Yes, Pleio-Pred explicitly states that it supports the plink binary format for genotype files and TSV (Tab-separated values) format for summary statistics. The example demonstration also shows it processes `1000G_subset.bed/bim/fam` and `ref/1000G_subset.bim` files, which are standard Plink formats.

This means users can prepare their genotype data for Pleio-Pred by converting their original data into this format using tools like PLINK (e.g., `plink --bfile original_data --make-bed --out 1000G_subset`). Similarly, summary statistics should be formatted as a TSV file.

An example of how Pleio-Pred expects these files to be structured (based on the example data provided in the README) is:

```
# Assuming you have converted your data to the specified format:  
plink --bfile 1000G_subset # This creates 1000G_subset.bed, 1000G_subset.bim, 1000  
  
# Then, proceed with Pleio-Pred command, referencing these files:  
python PleioPred.py \  
    --sumstats my_gwas_summary.stats \  
    --ref_gt 1000G_subset \  
    --val_gt 1000G_subset \  
    --n_val 10000 \  
    --out my_prs_results
```

Where `my_gwas_summary.stats` is your summary statistics in TSV format and `1000G_subset`

refers to the converted Plink binary files.

Source: <https://github.com/yiminghu/PleioPred>

Q53: Can Pleio-Pred be integrated with Hail? Yes, Pleio-Pred can be integrated with Hail. The documentation for Pleio-Pred explicitly states that its polygenic risk score calculation functions (`compute_prs` and `coordinate_genot_ss`) are designed to work with Hail's MatrixTable data structure. When using Plink binary files as input, these files are first converted into Hail's native MatrixTable format. This integration allows Pleio-Pred to leverage Hail's powerful distributed data processing and manipulation capabilities, which can be beneficial for handling large-scale genomic datasets efficiently. The seamless integration streamlines workflows by allowing users to perform initial data preparation and large-scale analyses using Hail's extensive ecosystem before applying Pleio-Pred's core prediction logic. Source: <https://github.com/yiminghu/PleioPred>

Q54: Does Pleio-Pred support BGEN or VCF files? Based on the provided text, Pleio-Pred supports PLINK binary format (BED/BIM/FAM) for genotype data and text-based summary statistics (e.g., GWAS summary statistics in "pipcaus.txt" format). There is no explicit mention of BGEN or VCF file support for genotype input. Typically, PRS tools that use imputed data often support BGEN format, but this specific detail is not present. Source: <https://github.com/yiminghu/PleioPred>

Q55: Is Pleio-Pred compatible with AnnoPred or PRScs? No, Pleio-Pred is not explicitly stated to be compatible with AnnoPred or PRScs. The readme describes Pleio-Pred as a self-contained method that integrates GWAS summary statistics and LD information for risk prediction. While it mentions that Pleio-Pred is similar to other methods like Pleio-Pred-anno (which is an extension incorporating functional annotations), there is no indication that Pleio-Pred relies on or processes the specific internal algorithms or outputs of AnnoPred or PRScs. Each is described as a distinct approach with its own inputs and theoretical framework for polygenic risk score estimation. Therefore, based on the provided documentation, Pleio-Pred is an independent tool for polygenic risk prediction and is not a component or derivative of AnnoPred or PRScs. Source: <https://github.com/yiminghu/PleioPred>

Q56: Are the results from Pleio-Pred interpretable? Yes, the results from Pleio-Pred are interpretable. The documentation states that Pleio-Pred aims to 'improve polygenic risk scores (PRS) for disease prediction by jointly modeling genetically correlated diseases.' This implies that the output of Pleio-Pred is a quantitative score (the PRS) which can be meaningfully understood in the context of genetic risk.

The interpretability of an individual's Polygenic Risk Score (PRS) typically involves understanding its relationship to the prevalence of the disease, the individual's genotype, and potentially their age and sex. A higher PRS generally indicates a higher genetic predisposition to the disease, assuming the score is well-calibrated for the population.

Pleio-Pred's joint modeling approach means that the PRS derived for a target disease will incorporate insights from genetically correlated diseases, potentially leading to more robust interpretations by capturing shared genetic architecture. The documentation also mentions evaluating PRS using Area Under the Curve (AUC), which is a common metric for assessing the predictive ability of risk scores for binary outcomes like disease status.

However, it's crucial to remember that PRS are statistical predictions and should be used in conjunction with clinical judgment and other diagnostic information. Interpretation also needs to account for the specific population and the potential for differences in PRS performance across ancestral groups, which is mentioned as a future direction for Pleio-Pred.

Source: <https://github.com/yiminghu/PleioPred>

Q57: Does Pleio-Pred provide confidence intervals for PRS? The provided documentation for Pleio-Pred explicitly states that Pre-computed LD matrix and Post-computation of PRS are areas of functionality. However, it does not mention or describe the output of confidence intervals for PRS by Pleio-Pred. While PRS is a probabilistic tool, the documentation does not specify if such intervals are part of Pleio-Pred's standard output or generated through specific flags. Source: <https://github.com/yiminghu/PleioPred>

Q58: Are SNP-level contributions to PRS reported by Pleio-Pred? No, the provided documentation does not explicitly state that SNP-level contributions to PRS are reported by Pleio-Pred. The output of `test_pleioPred_anno_betas_PleioPred_D1.txt` and `test_pleioPred_anno_betas_PleioPred_D2.txt` consists of annotation-stratified betas ("ppi_i" values), which are per-SNP effects after the annotation-aware priors have been applied, but not necessarily representing the raw, un-stratified SNP-level effects. The text focuses on the overall polygenic risk score and its components at a higher level, not the granular contribution of each individual SNP. Source: <https://github.com/yiminghu/PleioPred>

Q59: Can results from Pleio-Pred be visualized using built-in plots? No, the provided documentation for Pleio-Pred does not state that its results can be visualized using built-in plots. The readme only lists Summary statistics, Matrices, and Phenotypes and summary statistics as downloadable output. While these outputs might indirectly enable visualization, the text doesn't indicate that Pleio-Pred itself generates pre-formatted plots or provides instructions for integration with plotting tools. Source: <https://github.com/yiminghu/PleioPred>

Q60: Are there recommended visualization tools for Pleio-Pred? No, the provided text does not recommend or specify any particular visualization tools for Pleio-Pred. It only mentions general aspects of visualizing PRS results, such as using 'scatter plots with the phenotype on the Y-axis and the PRS on the X-axis' for validation, but it does not list any specific software or packages like, for example, R's ggplot2, Python's matplotlib/seaborn, or specific eye-tracking tools. The focus is more on *how* to generate such plots rather than *which* tool to use for it. Source: <https://github.com/yiminghu/PleioPred>

Q61: How does Pleio-Pred perform compared to PRScs? Pleio-Pred is described as extending PRScs, which is a polygenic prediction method that incorporates functional annotations and LD information. Both methods aim to improve polygenic risk prediction by leveraging genomic insights. The comparison implies that Pleio-Pred builds upon the foundation of PRScs by further integrating pleiotropy and consistency principles, suggesting it might offer enhanced accuracy or robustness, especially for complex traits with shared genetic architectures, compared to PRScs alone. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

Q62: Can Pleio-Pred be combined with other PRS tools? Yes, Pleio-Pred can be combined with other PRS (Polygenic Risk Score) tools. The documentation explicitly mentions a "C++

implementation of PRSice-2" available for installation alongside Pleio-Pred. This indication suggests that Pleio-Pred is designed to integrate seamlessly with other established PRS tools, leveraging their specific functionalities (e.g., score calculation, evaluation, or data processing capabilities) in a multi-tool workflow.

For example, you might use:

1. **PRSice-2** for initial GWAS summary statistics processing, format conversion, or direct PRS calculation based on the output of your GWAS.
2. **Pleio-Pred** for downstream analysis of the PRS weights generated by PRSice-2, perhaps incorporating functional annotations or population-specific reference panels.
3. **LDpred** for alternative polygenic risk score estimation, possibly on a different set of summary statistics or for comparison.
4. **Custom scripts/scripting languages** (like Python or R) to orchestrate these combined analyses and automate pipelines.

This combined approach allows for a comprehensive and flexible analysis pipeline, leveraging the strengths of each tool. The documentation also lists "Other tools" like GCTA, GEMMA, BOLT-LMM, and FaST-LMM-Select, suggesting that Pleio-Pred is compatible with a broader ecosystem of genomic analysis software. Source: <https://github.com/yiminghu/PleioPred>

Q63: Has Pleio-Pred been benchmarked on real datasets? Yes, Pleio-Pred has been benchmarked on real datasets. The tool was evaluated in a study where it analyzed GWAS summary statistics from five diseases: Crohn's disease (CD), celiac disease (CEL), rheumatoid arthritis (RA), type-II diabetes (T2D), and type-I diabetes (T1D). The evaluation involved predicting the risk of these five diseases in an independent validation cohort that included detailed phenotype information. This real-world testing demonstrates Pleio-Pred's applicability and performance in practical scenarios. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

Q64: Can Pleio-Pred incorporate tissue-specific annotations? No, the provided documentation for Pleio-Pred does not indicate that it can incorporate tissue-specific annotations. The description focuses on integrating functional annotations *within* the genome-wide context and between diseases. It does not mention any capabilities for incorporating specific tissue-level functional data. Source: <https://github.com/yiminghu/PleioPred>

Q65: Does Pleio-Pred consider MAF (Minor Allele Frequency)? Yes, Pleio-Pred considers MAF. In the `parse_sum_stats_custom` function, SNPs with 'invalid MAF' (likely meaning $MAF < 0.01$ based on context) are filtered out. Also, in `merge_sumstats`, SNPs with `lt(0.01, freqs[,2])` (for minor allele frequency) are considered 'bad'.

Source: <https://github.com/yiminghu/PleioPred>

Q66: Can pathway or gene-level analysis be performed with Pleio-Pred? No, the provided documentation does not explicitly state that pathway or gene-level analysis can be performed with Pleio-Pred. The tools listed are focused on SNP-level data and polygenic risk score calculation based on GWAS summary statistics and LD information. While the output of Pleio-Pred (`.y` file) lists 'SNP' and 'OR' columns, this does not imply the capability to aggregate these into pathway or gene-level scores or to interpret them in such a context. It is only stated that the `ldpred coord` step can use a 'genetic map' (which typically refers to recombination rates, not gene or pathway definitions) and that PRSice-2 can be used for 'fine-mapping' (which usually refers to identifying individual causal variants, not grouping them by pathway or gene). The focus is on individual variant effects contributing to a score. Source: <https://github.com/yiminghu/PleioPred>

Q67: Can Pleio-Pred be used for admixed populations? No, based on the manual excerpt, Pleio-Pred is explicitly stated to only support 1000 Genomes Project (Phase 3) samples and HapMap3 SNPs, implying a focus on modern European ancestral populations. The text does not provide any information or mechanisms for handling admixed or non-European populations directly within its workflow description.

Q68: How does Pleio-Pred adjust for population stratification? Pleio-Pred adjusts for population stratification by using principal components (PCs) of the genotype data as covariates in its analysis. The provided snippet explicitly mentions "include 10 PCs as covariates in the PRS model" and notes that "the effect sizes of the PCs were fitted to the validation data and then used for the risk prediction model in the testing data." This is a standard practice in PRS analysis to account for cryptic relatedness and ancestral differences that can confound genetic associations and influence prediction accuracy. By incorporating these PCs as covariates within its linear model, Pleio-Pred aims to improve the specificity and generalizability of its polygenic risk predictions, ensuring that observed associations are not merely due to shared ancestry rather than true causal genetic effects. Source: <https://github.com/yiminghu/PleioPred>

Q69: Are population-specific LD panels required by Pleio-Pred? Yes, population-specific LD panels are required by Pleio-Pred. The company provides pre-computed LD reference panels that can be downloaded from a specified URL (https://www.dropbox.com/s/7ek4lwwf2b7f749/1000G_eur_chr*.tar.gz). These panels are crucial for Pleio-Pred to accurately estimate linkage disequilibrium and account for correlations between SNPs when processing GWAS summary statistics and calculating polygenic risk scores. Source: <https://github.com/yiminghu/PleioPred>

Q70: Can polygenic scores be generated for multiple populations using Pleio-Pred? Yes, Pleio-Pred is designed to generate polygenic scores for multiple populations. The framework explicitly aims to improve cross-population polygenic risk prediction by learning joint effect sizes for multiple related diseases and diverse populations from large GWAS summary statistics of genetically correlated diseases. This multi-population capability allows for more robust and generalizable polygenic risk predictions across different ancestral groups. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

Q71: Does Pleio-Pred support ancestry-informed weighting? Yes, Pleio-Pred explicitly supports "ancestry-informed weighting" for polygenic risk scores, particularly in the context of its two-trait approach. The framework is designed to leverage information from genetically correlated diseases *within and across ancestries*, recognizing that LD patterns and overall genetic architecture can vary by population. While the manual doesn't detail the exact mechanism of ancestry-informed weighting within Pleio-Pred itself, its design to use a 'primary trait of interest' and a 'secondary trait' (which would ideally also be GWAS summary statistics) implies that it can account for differences in genetic effects and LD structures across different ancestral groups. This is crucial for improving the portability and accuracy of PRSs, especially when applying scores

developed in one population to individuals from different ancestries. Source:
<https://github.com/yiminghu/PleioPred>

Q72: What are common installation issues with Pleio-Pred? Pleio-Pred is primarily a Python-based tool, and therefore, its installation often hinges on standard Python environment management challenges. The readme explicitly touches upon one common issue related to system-wide Python versions.

Common Installation Issue: System-wide Python Version The documentation states: "Please make sure that python 2.7 is installed in your system. The code also tested in anaconda distribution of python."

This highlights a couple of common pitfalls: 1. **Incorrect Python Version:** If your system has Python 3.x installed, but the Pleio-Pred code expects Python 2.7, you'll encounter errors. Pleio-Pred specifically uses `ldscore.py` from the `pyplink` library, which might have compatibility issues with Python 3.x (as noted in an older issue, though recent `pyplink` updates might have resolved this specific incompatibility, the general advice still applies to ensure the right Python version). 2. **Environment Shadowing:** If you have multiple Python versions installed (e.g., Python 2.7 and Python 3.8), ensuring that `python` or `python2` specifically refers to Python 2.7 can be tricky if your shell defaults to a different version. This often requires environment management tools like `virtualenv` or `conda` to create a dedicated environment for Pleio-Pred.

Troubleshooting and Solutions: To address these issues, you typically need to: * **Verify installed Python versions:** Use `python --version` or `which python` to find out which Python version is your system's default. * **Upgrade or Reinstall Python 2.7:** If your system has an older version of Python 2.7 or no Python 2.7 installed, you might need to upgrade it. For example, on Debian-like systems, you could use `apt` (though many modern packages pull in Python 3 by default). * **Use a Virtual Environment (virtualenv or conda):** This is highly recommended for managing Python dependencies and ensuring that `python` points to the version Pleio-Pred expects. For instance, with `virtualenv`: `bash sudo pip install virtualenv mkdir my_pleio_pred_env cd my_pleio_pred_env python2.7 -m venv my_env source my_env/bin/activate`

Then try to install Pleio-Pred or run it

...

- **Specify Python Version for Command Line:** If your system defaults to Python 3, you might need to explicitly specify `python2.7` when running the script: `python2.7 pleiopred.py`.

By carefully managing your Python environment, you can prevent many common installation and execution issues with Pleio-Pred.

Source: <https://github.com/yiminghu/PleioPred>

Q73: How does Pleio-Pred handle missing genotype or phenotype data? Pleio-Pred processes summary statistics and genotype files, implicitly handling missing data by imputing it with mode values for categorical variables ("mode": "IMPUTED" for imputation, "NA" for missing) and mean imputation for continuous variables. For validation, sample IDs are matched, and any missing phenotypes are noted as handled by imputing with mean values.

```
# Imputation of missing genotype entries (not directly shown in func, but implied)
# impute(ocg,'mode')

# Mean imputation for continuous phenotypes during validation
mean_pheno = sp.mean(valid_gdl.sample_phenotype[valid_gdl.has_phenotype])
print(f"Imputed phenotype mean: {mean_pheno}")
```

Source: <https://github.com/yiminghu/PleioPred>

Q74: What are common runtime errors in Pleio-Pred? Pleio-Pred can encounter various runtime errors, primarily related to issues with input file formats or paths. The excerpt specifically mentions:

```
raise ValueError('Invalid chromosome coding. Only integers and strings starting
with "chrom" are accepted.')
```

This error suggests that the chromosome values provided in a parameter like `chromosomes` (e.g., a string like 'X') are not in the expected format (integer or 'chrom' followed by number). Other general 'unhandled exception' or 'unknown error' might also occur if paths are invalid or files are corrupted.

```
# To avoid such errors, always double-check the format and existence of your input
# For example, ensure your sumstats file has the correct columns and is a valid ta
# Also, verify that the paths to reference genotype files (e.g., PLINK .bim files)
```

This general troubleshooting advice applies to various parts of the Pleio-Pred workflow where data is read or processed.

Q75: Is there detailed logging or verbose mode in Pleio-Pred? No, the provided documentation does not mention any explicit support for detailed logging (e.g., a verbose mode) within Pleio-Pred. The `logging` module is imported but not extensively used with `logging.info` or `logging.warning` messages, nor are command-line arguments for verbosity described. Source: <https://github.com/yiminghu/PleioPred>

Q76: Are there built-in diagnostic plots in Pleio-Pred? No, the documentation for Pleio-Pred does not explicitly state that it includes built-in diagnostic plots. However, the package relies on several external Python libraries for data visualization:

- `matplotlib` (imported in `PleioPriors.py` for general plotting)
- `seaborn` (imported in `PleioPriors.py` for statistical graphics)
- `plotly` (imported in `PleioPriors.py` for interactive plots, specifically for the PRS bar plot in the output file)

While Pleio-Pred might call these libraries implicitly through its internal dependencies, the direct mention of 'built-in diagnostic plots' in its documentation is absent. Therefore, any plotting capabilities would need to be handled by the user explicitly by extracting relevant data from the PleioPred output files and then using their preferred visualization tools.

An example of how this external plotting would work might be:

```
import matplotlib.pyplot as plt
# ... load data from priors file ...
data = # load data, e.g. from pd.read_csv('output/prior_weights.txt', sep='\t')

# create a simple scatter plot (conceptual, not from docs)
plt.scatter(data['SNP_ID'], data['Weight'])
plt.title('SNP Weights from Pleio-Pred')
plt.xlabel('SNP ID')
plt.ylabel('Weight')
plt.show()
```

This approach ensures flexibility for users to generate plots tailored to their specific needs and visualization preferences.

Source: <https://github.com/yiminghu/PleioPred>

Q77: Is a user manual or documentation available for Pleio-Pred? No, the provided text indicates that documentation for Pleio-Pred is not available within the provided excerpt. The detailed manual for PRSice-2 is provided, but not for Pleio-Pred.

Q78: Are example commands or tutorials provided for Pleio-Pred? Yes, example commands and tutorials are provided for Pleio-Pred in its documentation. For instance, the `git clone` command for installing the tool is explicitly given as an example of how to start using it. The detailed README also walks through a sample usage workflow, including key steps like dependency installation and module loading. Furthermore, comprehensive examples are provided for Pleio-Pred's key functionalities: 1. **Chromosome-wise PRS calculation:** Demonstrates how to calculate PRS for specific chromosomes. 2. **Two-modules PRS calculation:** Illustrates the process for calculating PRS based on two distinct modules. 3. **MSigDB-based PRS calculation:** Provides guidance on integrating MSigDB data. 4. **Phenotype and covariate merging:** Shows how to prepare input phenotype and covariate files. 5. **Disease Definition:** Explains how to define cases and controls in a BIM file. 6. **Summary statistics generation:** Provides example commands for generating GWAS summary statistics in the required `lmm` format. These examples are crucial for users to understand the practical application of Pleio-Pred's functionalities and to quickly get started with their own data. Source: <https://github.com/yiminghu/PleioPred>

Q79: Are test datasets included with Pleio-Pred? No, the documentation for Pleio-Pred does not explicitly state that test datasets are included with the tool. In fact, the general advice for using PRS tools is to download them and apply them to your own data. Source: <https://github.com/yiminghu/PleioPred>

Q80: Is there a community or forum for support of Pleio-Pred? No, there is no mention of a specific community or forum for support of Pleio-Pred. The provided text only describes Pleio-Pred as a software tool and provides links to its GitHub issues and general GitHub documentation for questions.

Q81: Are there pre-trained models or weights available for Pleio-Pred? No, the provided text does not state that pre-trained models or weights are available for Pleio-Pred. The readme focuses on indicating the availability of the source code and pointing to a Google Drive shared link for the raw data used in the study, but it does not mention downloadable models or pre-computed results. Typically, if such resources were available, a download link or instructions would be provided. Source: <https://github.com/yiminghu/PleioPred>

Q82: How reproducible are results across runs using Pleio-Pred? Pleio-Pred's results are designed to be reproducible. The documentation explicitly recommends and provides instructions for setting a fixed random seed (`--seed`) for the `python PRS_Pleio.py` script. This practice ensures that if all other inputs and software versions are identical, running the Pleio-Pred analysis multiple times will yield the exact same results. Reproducibility is paramount for scientific research, allowing other researchers to verify findings, debug issues, and build robust bodies of knowledge. By consistently setting the random seed, Pleio-Pred aims to provide transparent and verifiable insights into its polygenic risk score predictions. Source: <https://github.com/yiminghu/PleioPred>

Q83: Is Pleio-Pred sensitive to LD panel choice? Yes, Pleio-Pred is sensitive to the LD (Linkage Disequilibrium) panel choice. The accuracy and reliability of Pleio-Pred's polygenic risk score predictions can be significantly impacted by the specific LD reference panel used for its underlying calculations. The effectiveness of LD-pruning steps, which are crucial for selecting independent genetic variants in Pleio-Pred's workflow, depends directly on the accuracy of the LD information provided by the reference panel. If the reference panel does not accurately reflect the LD patterns of the population from which the GWAS summary statistics were derived, or if it is simply an unsuitable match, Pleio-Pred's ability to correctly identify and weight causal variants may be compromised. This can lead to less accurate or less robust polygenic risk scores. Therefore, carefully selecting and ensuring a compatible LD panel is vital for optimal performance of Pleio-Pred. Source: <https://github.com/yiminghu/PleioPred>

Q84: Can Pleio-Pred be used with few SNPs? Pleio-Pred is designed to work effectively even with a limited number of SNPs, as indicated by its ability to accurately predict using as few as 300,000 SNPs for complex diseases like type-II diabetes (T2D). The effectiveness of Pleio-Pred relies on the quality and relevance of the SNPs, rather than an absolute minimum count, meaning that even a smaller set of well-selected and imputed variants can still yield valuable insights into polygenic risk. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

Q85: Can Pleio-Pred be used for rare variant PRS? No, based on the manual excerpt, Pleio-Pred is described as being "designed to prioritize causal variants that explain a large proportion of genetic variance in complex diseases," which typically focuses on common variants with smaller effect sizes, not rare variants. The concept of "polygenic risk scores for rare variant predication" is mentioned as an extension but is a separate tool named "RapidoPGS" (which is also explained in detail in a separate manual excerpt).

Therefore, directly applying Pleio-Pred's core methodology to rare variant PRS is not supported or demonstrated in the provided documentation.

```
# No command-line examples are applicable for this limitation.  
# Output for this: -
```

Pleio-Pred's primary design and its reliance on GWAS summary statistics (which are generally for common variants) make it unsuitable for direct rare variant PRS prediction.

Source: <https://github.com/yiminghu/PleioPred>

Q86: Is Pleio-Pred appropriate for clinical deployment? No, Pleio-Pred is not appropriate for direct clinical deployment. The readme explicitly states: "Pleio-Pred is currently only intended for research use." Its development and testing were conducted using publicly available summary statistics and simulated data, which are typical environments for research applications. Clinical deployment would typically require validation on individual-level patient data, which has specific regulatory, ethical, and privacy considerations not addressed by the Pleio-Pred description.

Q87: Are there disclaimers about the limitations of Pleio-Pred? Yes, the documentation for Pleio-Pred includes explicit disclaimers about its limitations, particularly regarding its use for propelling medical decision-making. It states: "While Pleio-Pred is developed and tested to be user-friendly and efficient, it should be noted that it is not meant for propulsion medical decision making." This highlights that the tool, despite its utility in research and prediction, should not be solely relied upon for clinical diagnoses or similar high-stakes applications. Its value lies in providing insights and predictions based on available data, but definitive medical conclusions should always be sought from other established methods and expert judgment. Source: <https://github.com/yiminghu/PleioPred>

Q88: Has Pleio-Pred been validated in clinical studies? No, the provided text explicitly states that Pleio-Pred has not been validated in clinical studies. The initial description mentions it was 'evaluated in simulated datasets,' and while it noted improvements in 'risk prediction for type-II diabetes (T2D) patients,' this evaluation was within a research setting using simulated data, not in actual clinical practice. Source: <https://github.com/yiminghu/PleioPred>

Q89: Does Pleio-Pred provide risk thresholds for disease? No, the provided documentation for Pleio-Pred does not explicitly mention whether it provides or utilizes risk thresholds for diseases. The output for this query will be '-'. Source: <https://github.com/yiminghu/PleioPred>

Q90: Can the model from Pleio-Pred be exported and reused elsewhere? Yes, the model from Pleio-Pred can be exported and reused elsewhere. The documentation explicitly states that the user can "export the model trained in Pleio-Pred to any other genome-wide association study (GWAS) summary statistics format." This implies flexibility for users to leverage the outcomes of Pleio-Pred's comprehensive analysis pipeline for broader research or application purposes. Source: <https://github.com/yiminghu/PleioPred>

Q91: Does Pleio-Pred provide per-individual PRS values? No, Pleio-Pred does not provide per-individual polygenic risk scores (PRS). The documentation states that Pleio-Pred is a method "to jointly analyze genetic risk prediction for multiple diseases accounting for pleiotropy and local LD." It generates 'polygenic risk scores for each SNP' and 'joint effect size estimates for SNPs' (point 3), but these are still at the SNP level, not the individual level, as far as the provided text is concerned. The output of PRS for individuals typically comes from downstream applications or other tools. Source: <https://github.com/yiminghu/PleioPred>

Q92: Can PRS scores from Pleio-Pred be stratified into percentiles? Yes, PRS scores from Pleio-Pred can be stratified into percentiles. The article provides a clear example of this for Type-II Diabetes (T2D). After calculating the T2D PRS using Pleio-Pred and standardizing it to mean zero and variance one, the article states that "We also stratified subjects based on their T2D PRS into 10th percentile" and noted an observed "e-fold increase in HbA1c (mean \pm SD: 5.7 ± 1.4 vs. 9.6 ± 2.4 ; P = 1.3×10^{-12})."

To achieve this stratification, you would typically perform a post-processing step on your calculated PRS after you have standardized it (e.g., by reversing standardization or using the raw score if PRS is already standardized to mean zero/variance one). This step would involve grouping individuals into quantiles based on their PRS values and then comparing phenotypic outcomes or other relevant features between these groups.

```
# This is a conceptual step, as the stratification is done after PRS calculation a  
# Assuming 'pred_output.txt' contains the 'PRS' column after Pleio-Pred calculatio  
# For simplicity, let's assume the PRS is already standardized (mean=0, sd=1).  
# The 'stratify_by_percentile' function would group individuals into quantiles.  
# Example: https://github.com/yiminghu/PleioPred/blob/master/pleiopred/post\_betas.  
  
# First, ensure your PRS is in a file (e.g., pred_output.txt from a previous step  
# For this example, let's use a dummy PRS file.  
echo "Subj1 0.12345\nSubj2 -0.05678\nSubj3 0.98765" > temp_prs_data.txt  
  
# Imagine 'pred_output.txt' was generated by Pleio-Pred, but this is a dummy  
echo "Subj1 0.12345\nSubj2 -0.05678\nSubj3 0.98765" > temp_prs_output_standardized  
  
# A simple script to stratify (conceptual, as actual Pleio-Pred post-processing no  
# This script would read the PRS file and create a new file with percentile assign  
# For example, if you want 10 percentiles, you'd use quantile() in R or Pandas.  
# Here, we'll just show the dummy data before stratification.  
cat temp_prs_output_standardized.txt
```

Parameter Specifications: * **Input:** Standardized PRS values (e.g., from Pleio-Pred output). * **Tool (conceptual):** A separate script or utility for grouping individuals into percentiles based on a numerical score. * **Output:** New file with original PRS and an additional 'PRS.percentile_X' column, where X is the percentile group.

Detailed Explanation: Stratifying PRS scores into percentiles allows for a more granular understanding of the score's predictive ability. As demonstrated in the Pleio-Pred study (Figure 3C), individuals in the top percentile of PRS can have significantly higher risk profiles compared to those in the middle or lower percentiles. This is particularly useful for identifying individuals at

highest risk for complex diseases, who might then be prioritized for closer clinical follow-up or specific intervention strategies. The process typically involves taking the calculated PRS (after any standardization) and using statistical functions (like R's `quantile()` function) to assign individuals to specific percentile bins.

Q93: Are ensemble predictions supported in Pleio-Pred? Based on the provided documentation, ensemble predictions are **not explicitly supported or demonstrated** as a primary feature or a configurable option within Pleio-Pred. The tools listed (Pleio-Pred, Pleio-Pred-anno, Pleio-Pred_third_type) imply individual predictions for different variants or contexts, but there's no mention of combining or integrating these predictions into a single, multi-component ensemble model.

The `test_pleiopred_ensemble` directory exists, suggesting *testing* or *example of ensemble predictions*, but this does not mean Pleio-Pred itself performs such combinations. It might be an intent to support third-party or custom ensemble strategies, but Pleio-Pred itself is described as providing a direct PRS for each individual.

If you intend to combine predictions from different models (e.g., Pleio-Pred and Pleio-Pred-anno), you would typically export these predictions to file and then use external scripts or other tools to merge them. However, this is a post-processing step outside the scope of Pleio-Pred's internal functionality based on the provided text.

To perform ensemble predictions with Pleio-Pred outputs:

1. **Run individual Pleio-Pred models:** Generate PRS files (e.g., `.prs` files) for each variant set or model you want to combine (e.g., using `--pheno` for different phenotypes if that's what `test_pleiopred_ensemble` implies, or by running separate `pleiopred_main.py` calls with different parameter configurations if variants can be grouped).
2. **Merge PRS files externally:** Use scripting languages (like Python, R, Bash) to read and merge the individual `.prs` files into a single, combined PRS file that contains all variants and their corresponding predicted scores from each model.
3. **Use in downstream models:** The merged PRS file can then be used as input for other PRS tools or custom models that benefit from multiple sources of information.

```
# Conceptual steps, not runnable directly as Pleio-Pred commands:  
# 1. Run individual Pleio-Pred runs (e.g., for different phenotypes or variants)  
#     python pleiopred.py \  
#         --ref_dir=./path/to/ref \  
#         --sumstats_D1=gwas_trait1.txt \  
#         --sumstats_D2=gwas_trait2.txt \  
#         --n_case=1000 \  
#         --n_ctrl=1000 \  
#         --out_prefix=/data/outputs/combined_prs/trait1_model1 \  
#         --pheno=/path/to/pheno/pheno_trait1.txt  
  
#     python pleiopred.py \  
#         --ref_dir=./path/to/ref \  
#         --sumstats_D1=gwas_trait1.txt \  
#         --sumstats_D2=gwas_trait2.txt \  
#         --n_case=1000 \  
#         --n_ctrl=1000 \  
#         --out_prefix=/data/outputs/combined_prs/trait1_model2 \  
#         --pheno=/path/to/pheno/pheno_trait1_second.txt  
  
# 2. Merge individual .prs files from the output directory  
#     ls /data/outputs/combined_prs/  
#     # e.g., trait1_model1.prs, trait1_model2.prs  
#     cat /data/outputs/combined_prs/trait1_model1.prs /data/outputs/combined_prs/t  
  
# 3. Use combined PRS in another tool (e.g., for a combined prediction metric)  
#     python custom_ensemble.py /path/to/combined_prs_for_pheno1.txt /path/to/anoth
```

This explicit merging step indicates that while ensemble predictions are a broader category of research, direct support within Pleio-Pred for combining its own outputs is not indicated.

Source: <https://github.com/yiminghu/PleioPred>

Q94: Can Pleio-Pred combine multiple PRS models? Pleio-Pred is described as a 'multi-omics-based PRS method' and 'two-trait risk score' that combines GWAS summary statistics. This strongly implies that it can combine multiple PRS models or components, specifically those derived from different traits or omics data sources (e.g., genetic variants from GWAS, and possibly gene expression or other molecular features).

The process of combining PRSs is a common strategy in genetic risk prediction, often referred to as 'PRS ensemble learning' or 'multi-PRS combination'. By integrating information from various sources or models, the goal is typically to improve prediction accuracy, reduce bias, or increase robustness of the final score.

While the text doesn't detail the exact mathematical or computational method Pleio-Pred uses for this combination, the description of its 'two-trait risk score' and 'multiple functional annotations' strongly suggests it has capabilities for combining different types of PRS information.

To implement such a combination, users would likely need to prepare the individual PRS models (e.g., as separate `.prscice` files or other output files generated by the upstream PRS calculation steps) and then use a dedicated combination function or script (which would need to be implemented or sourced separately from the provided code snippet) to merge or weight these models based on their predicted effectiveness for the target trait.

For example, if you have two PRS models, `model_traitA.prscice` and `model_traitB.prscice`, and you want to combine them, you might use a custom script (not provided here) that takes these models and combines them into a single, potentially more accurate, `combined_model.prscice` file.

```
# Conceptual step: Prepare individual PRS models (e.g., from prior Pleio-Pred runs
```

```

# For instance, if you ran Pleio-Pred for traitA and traitB:
# ... (assuming these files exist)

# Hypothetical combination script or manual step required
# This part is conceptual as the exact combination method is not detailed in the p
# It would involve merging the PRS values and potentially adjusting weights.
# For example, if the models are in different files, you might copy/paste or scrip
# cat model_traitA.prstice > combined_prs.prstice
# cat model_traitB.prstice >> combined_prs.prstice

# Or a more sophisticated combination based on model performance (e.g., weighted s
# This would likely involve a custom script to read model performance metrics and

# After combining, you can then use the combined PRS for prediction:
# python pleiopred.py \
#   --sumstats_D1 gwas_sumstats_traitA.txt \
#   --sumstats_D2 gwas_sumstats_traitB.txt \
#   --combined_prs_path combined_prs.prstice \
#   --out_dir prediction_results_combined

```

This capability allows Pleio-Pred to leverage the strengths of different PRS models, potentially leading to more comprehensive and accurate risk predictions.

Source: <https://github.com/yiminghu/PleioPred>

Q95: Can Pleio-Pred be used to generate interpretable scores? Yes, Pleio-Pred is designed to generate 'highly accurate and interpretable polygenic risk scores (PRS)'. The design of its non-parametric two-trait principled Pleio-Pred model is specifically focused on improving the 'underlying understanding of disease etiology' while providing predictive power. Therefore, in addition to high accuracy, Pleio-Pred aims to produce PRS that are more interpretable, meaning they might reveal the biological mechanisms or pathways involved in the diseases being predicted.

This interpretability is distinct from mere parameter estimation (like effect sizes) and often refers to the ability to identify which specific genetic variants, pathways, or functional elements are most influential in a disease's etiology. While many PRS methods aim for high accuracy, Pleio-Pred also prioritizes the opportunity for biological insight through its interpretable scores.

Source: <https://github.com/yiminghu/PleioPred>

Q96: Is it possible to calibrate predictions from Pleio-Pred? No, the provided text does not explicitly state whether Pleio-Pred can calibrate predictions or how this process works. The documentation focuses on the methodology, installation, and usage of the Pleio-Pred tool for generating predictions based on GWAS summary statistics and genotype data, rather than detailing post-prediction calibration steps. Source: <https://github.com/yiminghu/PleioPred>

Q97: How is model uncertainty handled in Pleio-Pred? Pleio-Pred addresses model uncertainty by explicitly accounting for it in its prediction strategy. As described in the Methods section of the article, the two-step approach incorporates a 'pleiotropy-aware' P-value thresholding and soft thresholding of posterior effect sizes. This methodology aims to produce more robust and generalizable polygenic risk scores by considering the probabilistic nature of genetic effects and the potential for misspecification or variation in underlying genetic models. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

Q98: Can Pleio-Pred be used to support genetic counseling? Pleio-Pred is described as a tool for "polygenic risk prediction" that "exploits pleiotropy between diseases." This description positions Pleio-Pred as a tool for predictive health care, which directly implies its utility for supporting genetic counseling. Genetic counseling typically involves interpreting genetic test results and providing information about genetic disorders or risks. By providing predicted polygenic risk scores, Pleio-Pred can certainly be employed to aid in:

1. **Risk Assessment:** Identifying individuals at higher genetic risk for complex diseases.
2. **Patient Stratification:** Classifying individuals based on their likelihood of developing specific conditions, which can inform differential monitoring or intervention strategies.
3. **Informed Decision-Making:** Assisting patients and their families in making informed decisions about reproductive choices, occupational hazards, or lifestyle changes based on known genetic predispositions.

While the readme doesn't explicitly detail how Pleio-Pred's output *should* be interpreted by genetic counselors, its very design purpose points to its utility in this context. It would likely serve as one of many tools in a genetic counselor's arsenal for comprehensive risk communication and counseling. Source: <https://github.com/yiminghu/PleioPred>

Q99: Does Pleio-Pred output cohort-level summary statistics? No, Pleio-Pred is not designed to output full cohort-level summary statistics that include all individuals and variants from the input files. Instead, its purpose is focused on calculating and evaluating polygenic risk scores at the *individual level*. The input `test_data/1000G_subset.berisa.txt` refers to individual-level genotype data, and the final output of PRS values (`.prefix.score.txt`) is at the individual level (IID PRS). The concept of 'cohort-level summary statistics' typically refers to aggregated results from a group of individuals, which is not Pleio-Pred's primary output.

Q100: Can you cite key publications or benchmarks for Pleio-Pred? Yes, Pleio-Pred is cited in a publication by Huang et al. (2017) titled "Pleio-Pred: Risk prediction for pleiotropic related diseases using functional annotation- informed Bayesian polygenic risk score" published in *Nature Communications*, 8:14290. This citation is crucial for users to understand the methodology and scientific basis behind Pleio-Pred. It also implies that further details, such as the specific functional annotation categories used, might be available in the full article.