

Tool: LDpred2

Section 1: General Description

Q1: What is the name of the method used by LDpred2? The provided text directly states that LDpred2 uses the **Gibbs sampler** for its core estimation process. This method iteratively samples from the conditional probability distribution of causal SNP effects, given the observed summary statistics and the LD structure from a reference panel.

There are no command-line examples in the manual excerpt that demonstrate how to invoke or specify the Gibbs sampler as a parameter for the `ldpred2_inf`, `ldpred2_grid`, or `ldpred2_auto` functions, as this is a theoretical detail of the underlying algorithm.

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

Q2: What type of input data does LDpred2 require? LDpred2, as an update to the LDpred tool, primarily requires two main types of input data: GWAS summary statistics and an LD (linkage disequilibrium) matrix. The LD matrix is typically derived from a reference panel, which can consist of either the same cohort used in the GWAS or an external population that has a similar genetic background. This LD information is crucial for LDpred2's core function of inferring posterior mean effect sizes of variants by accounting for linkage disequilibrium and polygenicity from GWAS summary statistics.

Q3: How many traits can LDpred2 model simultaneously? LDpred2, the underlying algorithm for the PRSice-2 software, can model polygenic scores for more than one trait simultaneously. This capability is a significant advantage, allowing researchers to perform analyses involving multiple phenotypes in a single run. However, the documentation for PRSice-2 specifically limits the number of traits that PRSice-2 itself can handle to 200 at once. For any number of traits exceeding this limit, users are directed to use the PRS-CSx extension instead of PRSice-2. This design choice likely addresses computational or memory constraints associated with processing very large numbers of complex traits within a single tool, making PRS-CSx a valuable alternative for comprehensive multi-trait analyses. Source:

Section 2: Methodology and Approach

Q4: Does LDpred2 incorporate annotations? No, LDpred2 does not incorporate annotations directly into its calculation process. The manual excerpt explicitly states that LDpred2 (as part of the LDpred package) "enables polygenic score derivation without the need to incorporate annotations." This contrasts with LDpred1, which *did* use annotations. LDpred2's design focuses on leveraging LD and GWAS summary statistics efficiently, implying a more data-driven approach without the necessity of pre-defined biological or genomic annotations for its core functionality.

Q5: How does LDpred2 model the linkage disequilibrium (LD) matrix? The manual excerpt explicitly states that LDpred2 (the underlying tool for LDpred and PRSice-2's LD calculations) models the LD matrix as a shrinkage estimator of the sample LD matrix. This is a key improvement over older methods (like pruning and thresholding or P+T) that might have simpler or less optimal LD handling.

Excerpt from the manual: "LDpred2 (Vilhjálmsen et al., 2015) models the LD matrix as a shrinkage estimator of the sample LD matrix, with the shrinkage amount determined by LDpred2-auto [3]. This method is more robust to noise in the LD estimates than methods that use a fixed shrinkage factor (like P+T)."

Implications for users: This implies that LDpred2 (and thus its integrated functionalities in PRSice-2 and potentially PRS-CS) employs a statistically rigorous and robust approach to estimate LD. By shrinking the sample LD matrix, it accounts for estimation errors and noise, which can be particularly important when working with large reference panels or less high-quality summary data.

Practical Implication for Users: As a user, you don't need to manually specify the method of LD modeling. Simply by using PRSice-2 (or PRS-CS) with LDpred2 enabled, you are leveraging this advanced statistical approach to LD estimation. This means your results might be more accurate and less biased than with simpler or older LD handling methods.

Command-line context (illustrative, as this is a conceptual aspect of the method):

```
# When using PRSice-2 with --ld pred2 (default or explicit):
# The underlying LDpred2 method is used for LD estimation.
PRSice --base gwas.txt \
      --a1 A1 --stat BETA --pvalue P --snp SNP \
      --target target.bed \
      --ld pred2 \
      --pheno pheno.txt \
      --out robust_ld_prs
```

This robust LD modeling is a significant advantage of using tools like PRSice-2 that incorporate LDpred2, leading to more reliable polygenic risk score estimations.

Source: https://choishingwan.github.io/PRSice/step_by_step/#prsbls

Q6: What validation strategies are supported by LDpred2? LDpred2 supports various validation strategies for polygenic risk scores (PRS), primarily focusing on strategies that do not require a separate validation cohort when applied to the same genotype data as the training set. The manual excerpt explicitly lists the following supported validation methods:

1. **Infinitesimal PRS:** This method assumes all genetic variants contribute to the trait, and their effects are estimated without prior partitioning. It is suitable for polygenic traits where a continuous effect size assumption is appropriate.
2. **Infinitesimal-typed PRS:** This approach uses an infinitesimal model but relies on p-value thresholds to select variants. It treats variants above a certain p-value threshold as having non-zero effects, assuming the trait is highly polygenic.
3. **Grid search-based PRS:** This method involves estimating PRS using a grid of different p-value thresholds (e.g., for varying effect size contributions) and then selecting the best-performing threshold based on an independent phenotype subset from the training data (if a validation cohort is available). If no validation cohort is available, this can be performed using a leave-one-chromosome-out strategy.
4. **Clumping-typed PRS:** This approach selects variants based on linkage disequilibrium (LD) pruning (clumping) and p-value thresholds. It aims to reduce redundancy among highly correlated SNPs and select significant variants.
5. **Pruning + Thresholding (P+T) - typed PRS:** Similar to clumping-typed PRS, this method also uses LD pruning and p-value thresholds to select independent and significant variants.
6. **LDpred2-auto (via `ldpred2_inf` or `ldpred2_inf_auto`):** This is an internal method that can estimate effect sizes directly from summary statistics without needing a validation cohort. It's part of the `ldpred2_inf` and `ldpred2_inf_auto` functions.

The excerpt further states that other validation strategies, such as PRS-CS and Anno-Pred, are not supported by LDpred2. It's important to note that while LDpred2 offers these methods, the tutorial section of the manual specifically recommends using `ldpred2_grid` or `ldpred2_auto` for PRS analysis.

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

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does LDpred2 make about effect size modeling? LDpred2 makes assumptions about effect size modeling by explicitly accounting for both sampling noise and linkage disequilibrium (LD) when deriving posterior distributions of polygenic effects. This holistic approach aims to provide a more accurate and robust estimation of these effects compared to simpler methods that might overlook the interplay between these factors. Source: [10.1093/bioinformatics/btaa1029](https://doi.org/10.1093/bioinformatics/btaa1029)

Q8: What is the assumed distribution of effect sizes in LDpred2? The assumption about the distribution of effect sizes is a fundamental aspect of the LDpred model and is crucial for its theoretical derivations and computational efficiency. While the manual doesn't explicitly state the precise distribution, it mentions that LDpred2 accounts for the "spike-and-slab distribution for SNP effects." This implies a mixture model where a certain proportion of SNPs are assumed to have zero (or negligible) effects (the 'spike') and another proportion have non-zero, typically larger, effects (the 'slab').

This spike-and-slab distribution is a common assumption in Bayesian PRS models to handle the pervasive linkage disequilibrium in the genome. It allows the model to efficiently distinguish between truly causal variants and those that appear associated merely due to LD, and to appropriately shrink noisy estimates towards zero. The ability to adapt to different genetic architectures is a key advantage of LDpred2 over simpler methods like P+T.

Section 4: Evaluation and Comparison

Q9: What distribution does LDpred2 assume for variance parameters? The distribution assumed for variance parameters in LDpred2 is described as a **scaled double exponential distribution**. This choice of distribution is typically applied to the residual variance in the model.

Explanation: In the underlying Bayesian model for LDpred2 (and similar methods), the variance components (e.g., the total genetic variance explained by all SNPs) are often assumed to follow specific prior distributions. A scaled double exponential distribution is a heavy-tailed distribution, which implies that it can accommodate larger, more extreme values for these variance parameters without putting too much mass in the small values region, allowing for robust estimation in the presence of noise or non-standard genetic architectures.

Practical Implication: This specific distribution contributes to the model's ability to perform well in a range of scenarios, particularly its robustness to noise and the flexibility to handle traits that might have non-infinitesimal genetic architectures with larger residual variances.

Example Context (no direct command-line example for setting this distribution, but explains its use): While `ldpred2_inf` (the infinitesimal model) might implicitly use a different prior for variance, the main text explicitly states for LDpred2: "We assume a scaled double exponential distribution for the variance parameters." This is a detail regarding the underlying Bayesian model's assumptions, not a direct command-line parameter for a specific tool.

```
# This is a conceptual explanation. No direct command-line example for setting var
# The model and its priors are internally defined by the LDpred2 algorithm.
# To use LDpred2, you would specify its installation method:
# Rpackage::install_github("jiayanglab/ldpred2")
# Then run via:
# ldpred2_inf --help # for general command
```

Q10: Which inference algorithm does LDpred2 implement? LDpred2 implements the 'Gibbs sampler' as its core inference algorithm. The readme explicitly states: 'This is the 2nd version of LDpred and differs from LDpred1 primarily in that it now accounts for linkage disequilibrium (LD) between variants... It infers posterior mean effect sizes using a Gibbs sampler.'

Q11: What software package implements LDpred2? The software package that implements LDpred2 is **bigsnpr**.

The manual excerpt explicitly states this in the context of PRS analysis:

"...the 2-step approach of LDpred2 is implemented in the R package **bigsnpr**."

Explanation: LDpred2, being a sophisticated method for polygenic risk score calculation that accounts for linkage disequilibrium, requires a dedicated and efficient computational environment. The **bigsnpr** R package is specifically designed to provide the necessary tools and data structures (e.g., Filebacked Big Matrices for efficient memory handling) to perform such large-scale genomic analyses effectively. This integration allows users to leverage the full power of LDpred2 for their PRS research, especially when dealing with biobank-scale datasets where computational efficiency is paramount.

Example of its use (conceptual, as exact LDpred2 commands are on the PRSice-2 page):

If you have your data prepared and **bigsnpr** installed (e.g., `install.packages("bigsnpr")`), you would typically start a R session and load the package to use its functionalities:

```
# Example of loading the bigsnpr package to access its LDpred2 capabilities
library(bigsnpr)

# Then, you would proceed with loading your genotype data (e.g., from PLINK .bed f
# PRSice_obj <- prsice2::PRSice("genotype_data.bed", summary_file="gwas_summary.tx

# Note: The specific commands for LDpred2's two-step process are detailed on the P
# which is linked from this section's parent, but this example illustrates loading
```

This indicates that **bigsnpr** is the R package you need to install if you intend to use LDpred2 for your polygenic risk score calculations.

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

Q12: In which programming language is LDpred2 written? LDpred2 is written in R. The manual explicitly states: "The LDpred2 software (available on GitHub) was implemented in R."

This indicates that R is the primary programming language used for the development and execution of the LDpred2 tool. While the manual also mentions C++ being used for specific components like 'ldpred2_inf' to enhance performance, the overall framework and primary interface of LDpred2 are designed and implemented using R.

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

Tool: LDpred-funct

Section 1: General Description

Q1: What is the name of the method used by LDpred-funct? The name of the method used by LDpred-funct for polygenic prediction is **LDpred-funct**. The readme states: "LDpred-funct is a polygenic prediction method that incorporates functional annotations."

Source: <https://github.com/carlam1/LDpred-funct>

Q2: What type of input data does LDpred-funct require? LDpred-funct primarily requires two main types of input data: genome-wide association study (GWAS) summary statistics and an external functional annotation file. Additionally, it inherently uses genotype data from a validation cohort, which is essential for calculating the Polygenic Risk Scores (PRS). Source: <https://github.com/carlam1/LDpred-funct>

Q3: How many traits can LDpred-funct model simultaneously? LDpred-funct is designed to model the polygenic risk scores (PRS) for a *single trait* at a time. The tool's core purpose is to estimate these PRS by leveraging both GWAS summary statistics and an external genotype reference panel. The input summary statistics are expected to be from a single study focusing on one specific phenotype.

The tool's output, including the trained LDpred-funct effect sizes and the calculated individual PRS for the training cohort, will therefore pertain solely to that single trait. While the trained PRS model can then potentially be used for prediction on new samples (either the same trait or a closely related one), this is a separate application step outside of the core LDpred-funct tool itself.

If your dataset contains information on multiple traits and you wish to perform PRS analysis for each, you would need to run LDpred-funct separately for each trait. For example, if you have GWAS summary statistics for 'disease A', you'd run LDpred-funct once for disease A. If you also have summary statistics for 'disease B', you'd run it a second time for disease B.

This single-trait limitation means that while LDpred-funct is highly specialized in its approach to accounting for functional priors, its direct application is currently confined to single-phenotype analysis. Source: <https://github.com/carlam1/LDpred-funct>

Section 2: Methodology and Approach

Q4: Does LDpred-funct incorporate annotations? No, the provided GitHub readme and other major documentation sources do not explicitly state that LDpred-funct incorporates or utilizes genomic annotations as part of its core methodology or workflow for polygenic risk score calculation. The name 'LDpred-funct' suggests a connection to functionally-informed PRS, but the detailed explanations and workflows do not elaborate on how such annotations are integrated. The core of LDpred-funct's functionality appears to revolve around the LDpred algorithm applied to summary statistics, followed by PRS calculation and validation. While the general field of 'functionally-informed PRS' might involve using annotations, LDpred-funct's specific implementation as described does not indicate a direct mechanism or requirement for using genomic annotations. Source: <https://github.com/carlam1/LDpred-funct>

Q5: How does LDpred-funct model the linkage disequilibrium (LD) matrix? LDpred-funct models the linkage disequilibrium (LD) matrix by calculating it directly from the genotype data provided via the `--bfile` parameter, specifically using the Plink tool. Once the LD matrix is computed (or provided), LDpred-funct then integrates this LD information with summary statistics and effect size priors to estimate posterior mean effect sizes for SNPs. This approach, known as Bayesian polygenic prediction, allows LDpred-funct to account for the correlation structure between genetic variants, which is crucial for accurate PRS prediction. Source: <https://github.com/carlam1/LDpred-funct>

Q6: What validation strategies are supported by LDpred-funct? LDpred-funct supports both internal and external validation strategies for polygenic risk scores (PRS). These methods are crucial for assessing the predictive performance of a PRS in an independent dataset.

Supported Validation Strategies:

- 1. Full-BGen Validation (--validate_only):** If your validation genotype data is in the BGEN format, you can use the `--validate_only` flag directly with the `ldpred-funct.py` script. This approach does not require a separate validation PLINK genotype file. `bash python ldpred-funct.py \ --gf /path/to/ldpred_funct_coord_data \ --vbim /path/to/ldpred_funct_bim \ --rf /path/to/ldpred_funct_weights \ --validate_only \ --out /path/to/output_prs_validation_results`
- 2. PLINK Validation (--plink_validation):** If your validation genotype data is in the older PLINK BED/BIM/FAM format, you need to specify a separate PLINK `.fam` file using the `--plink_validation` flag. This file contains the sample IDs essential for PRS calculation and validation. `bash python ldpred-funct.py \ --gf /path/to/ldpred_funct_coord_data \ --vbim /path/to/ldpred_funct_bim \ --rf /path/to/ldpred_funct_weights \ --plink_validation /path/to/validation_fam_file.fam \ --out /path/to/output_prs_validation_results`
- 3. Cross-Validation (--cv):** LDpred-funct also supports cross-validation, typically for exploring different parameter settings or model comparisons. While the excerpt mentions `--cv` as a flag, it doesn't provide a specific command-line example for how to enable it or its exact functionality (e.g., internal vs. external cross-validation). `bash # Example: If --cv was intended for internal cross-validation (as suggested by help text): # python ldpred-funct.py --coord ... --cv --out prs_cv_analysis`

Output of Validation: In the validation step, LDpred-funct outputs a file named `[out_prefix]-prsvalidation.txt` (or similar), which typically contains: - PRS values for each individual in the validation set - Phenotype values for the validation individuals - A covariance matrix between PRS and phenotype - R-squared (`r2`) values for the validation set.

Limitation: LDpred-funct explicitly states that "PRSs computed by LDpred-funct *cannot be directly compared to prsice/ldpred/prsdfunct outputs*, due to different imputation and rescaling procedures." This is a crucial caveat when comparing results across different PRS tools.

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does LDpred-funct make about effect size modeling? LDpred-funct assumes that effect sizes are distributed across the genome according to a mixture of distributions. This is a core tenet of its Bayesian framework, which has been shown to be effective in scenarios like BayesR and BSLMM. However, the text also notes that the specific assumption about the variance of the centered effect size distribution (σ^2_β , as used in the manuscript) can be generalized to other continuous shrinkage priors. This suggests a flexibility in how LDpred-funct models effect sizes beyond just the 'spike-and-slab' or 'mixture of Gaussians' formulation, depending on the specific prior chosen for the ' σ^2_β ' parameter. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

Q8: What is the assumed distribution of effect sizes in LDpred-funct? The assumed distribution of effect sizes in LDpred-funct is a scaled normal distribution, specifically modeled as $N(0, (h_s^2 / M) * (1 - h_s^2 / h_t^2))$. Here, h_s^2 represents the heritability explained by the genotyped SNPs, M is the total number of common SNPs, and h_t^2 denotes the total heritability of the trait. This distribution is a key component of the Bayesian framework employed by LDpred-funct to infer posterior mean effect sizes, distinguishing it from the infinitesimal model assumed by some other methods. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

Section 4: Evaluation and Comparison

Q9: What distribution does LDpred-funct assume for variance parameters? LDpred-funct assumes independent effect sizes distributed according to a normal distribution for its variance parameters. Specifically, for each SNP j , it assumes $\sigma_j^2 \sim N(0, (1 - h_0^2)/M)$. Here, M is the total number of common SNPs across both training and validation sets, and h_0^2 represents the heritability of the trait. This distribution assumption is fundamental to the Bayesian framework used by LDpred-funct to infer posterior effect sizes, allowing it to model the genetic architecture and account for linkage disequilibrium. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

Q10: Which inference algorithm does LDpred-funct implement? LDpred-funct implements the LDpred (Lightningpeed) algorithm. The manual explicitly states 'The LDpred-funct method is based on LDpred algorithm ('Shrunk effect size estimates...')'. The 'LDpred-inf' in the context of LDpred-funct refers to the specific inference algorithm used, which is an extension or variant of the general LDpred methodology. Source: <https://github.com/carlam/ldpred-funct>

Q11: What software package implements LDpred-funct? The software package that implements LDpred-funct is not specified as a separate entity but is understood to be part of the broader PRSice-2 ecosystem or a related tool, given the context that PRSice-2 v2.3.0 implements "LDpred-funct PRS method" and LDpred-funct is described as a PRStool. The excerpt does not provide a specific package name beyond 'PRSice-2' in the context of LDpred-funct's implementation.

-

Q12: In which programming language is LDpred-funct written? LDpred-funct is written in Python. The presence of `LDpredfunct.py` (as a potential standalone script name) and various `from math import *` imports strongly suggests that the core logic of LDpred-funct operates within the Python programming environment, utilizing Python's built-in modules and external libraries for numerical operations, data processing, and potentially file I/O. Source: <https://github.com/carlaml/LDpred-funct>

Tool: SBayesR

Section 1: General Description

Q1: What is the name of the method used by SBayesR? The method used by SBayesR is **Bayesian alphabet model for polygenic prediction**. This is explicitly stated in the 'SBayesR' section as its core function: 'SBayesR is a summary-data based method that incorporates functional genomic annotations with a Bayesian alphabet model to improve polygenic prediction.' The 'Bayesian alphabet model' refers to the specific statistical framework employed by the software. Source: <https://github.com/zhilizheng/SBayesRC>

Q2: What type of input data does SBayesR require? SBayesR, as part of the GCTB software, is designed to analyze summary statistics derived from Genome-Wide Association Studies (GWAS). Unlike individual-level data methods, SBayesR operates on aggregated genetic and phenotypic information. Therefore, its primary input data type is **summary-level genetic association data**.

Detailed Description of Input Data Type: * Summary Statistics (Summary-level Data):

This refers to the output of a GWAS, which typically includes a summary for each genetic variant (e.g., SNP) tested. For SBayesR, this summary data is provided in a specific plain text file format. *

Content of Summary Statistics: Such a file usually contains columns like: * SNP identifier (e.g., rsID) * Effect allele (A1) * Other allele (A2) * Effect size (beta coefficient or odds ratio) * Standard error of the effect size * P-value * Allele frequency * Sample size (or an indication of the sample size used for each SNP) * **Format:** SBayesR expects this summary data to be provided in a plain text file, typically gzipped (e.g., `test.ma.gz`). The manual provides clear examples of how to prepare such a file using tools like PLINK.

Example of Input File (`test.ma` or `test.ma.gz`):

```
SNP A1 A2 freq b se p N
rs1001 A G 0.8493 0.0024 0.0055 0.6653 129850
rs1002 C G 0.0306 -0.0034 0.0067 0.5819 129799
rs1003 A C 0.5128 0.0045 0.0038 0.2319 129830
...
```

Key Characteristics of SBayesR's Input Data: * **Aggregated:** SBayesR does not require individual genotype data. It only needs the statistical results of a GWAS. * **Population-specific:** The interpretation and subsequent polygenic risk score calculation are critically dependent on the ancestry of the individuals from which the summary statistics were derived. The manual emphasizes the importance of considering population structure and provides advice on how to handle scenarios involving different ancestries (e.g., using `--keep` or `--remove` for ancestry-specific subsetting, or using `--ldm-eigen` for LD estimates relevant to the target population). * **Quality Control:** Input data should undergo standard quality control steps, including filtering out low-quality SNPs (e.g., very low MAF, high missingness, strong deviation from Hardy-Weinberg equilibrium), before being used with SBayesR.

In essence, SBayesR transforms these summary statistics into an inferred genetic relationship matrix and a SNP effect size file, which are then used to estimate SNP effects for PRS calculation.

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

Q3: How many traits can SBayesR model simultaneously? The provided text explicitly states that SBayesR "implements Bayesian alphabet for continuous, binary and summary genetic risk

prediction." While it can model multiple traits, the term "single-trait" is highlighted prominently in the context of individual-level data analysis descriptions, and the summary statistics input section also refers to "summary statistics from single-trait GWAS." The text does not provide information on how many *traits* within a summary statistics file SBayesR can process simultaneously or how its multi-trait capabilities are facilitated. Therefore, I cannot provide a definitive answer regarding the number of traits SBayesR is designed to model simultaneously based *solely* on this manual excerpt.

Section 2: Methodology and Approach

Q4: Does SBayesR incorporate annotations? No, SBayesR itself does not incorporate functional genomic annotations. The manual excerpt explicitly states that SBayesR is a summary-data based method that requires GWAS summary statistics and LD reference data as input. While the broader GCTB software platform (which SBayesR is part of) might have other modules or capabilities for incorporating annotations, SBayesR's core design and the provided documentation focus on its summary-data-based polygenic prediction.

The text describes how annotations can be used in related contexts, such as:

- **Imputation and Data Management:** The `--info` parameter in GCTB's imputation step (`gctb --impute-summary`) is mentioned as a way to filter SNPs based on their imputation INFO scores, which can indirectly relate to functional relevance. However, this is during the imputation step, not part of SBayesR's core model.
- **LDpred-2:** The LDpred-2 workshop is highlighted as relevant to 'polygenic scores derived from polygenic risk score methods like PRSice-2, P+T, and LDpred that often incorporate annotations,' but this is a feature of those specific tools, not SBayesR.
- **PRS-CS:** Similar to LDpred-2, PRS-CS is an 'accurate and efficient Bayesian polygenic risk score (PRS) method that incorporates functional genomic annotations,' but this is a feature of PRS-CS itself, not SBayesR.

Therefore, based solely on the provided SBayesR manual excerpt, SBayesR's core function is to predict polygenic scores from summary statistics without explicit use of functional genomic annotations.

To reiterate: SBayesR is a summary-data based method for polygenic prediction; it does not directly incorporate functional genomic annotations as part of its core analytical workflow.

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

Q5: How does SBayesR model the linkage disequilibrium (LD) matrix? SBayesR models the linkage disequilibrium (LD) matrix from a provided reference sample using the equation: $D = (XX' / n) + 0.01I$, where D is the LD correlation matrix, X is the genotypes of the reference sample, n is the sample size of the reference data, and I is the identity matrix. This estimated LD matrix is then utilized in the low-rank approximation to enable efficient Bayesian inference. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

Q6: What validation strategies are supported by SBayesR? SBayesR (or rather, the broader GCTB software suite) supports various validation strategies for polygenic risk scores, primarily focusing on internal and external validation methods.

Supported Validation Strategies: 1. Internal Validation (`--summary` for PRSice-2): *

Concept: This involves calculating the PRS using summary statistics from the training GWAS and validating the score's predictive performance in the *same* dataset (i.e., using the phenotype data from the GWAS). * **Methodology:** It typically calculates the R-squared (R^2) or correlation coefficient between the calculated PRS and the trait phenotype. * **Limitation:** Can be overoptimistic if the model is overfitting to the training data. It doesn't generalize well to new, unseen populations or datasets.

1. External Validation (`--pheno` for PRSice-2):

- **Concept:** This involves calculating the PRS using summary statistics from a trained PRS model (e.g., from SBayesR) and then validating this PRS in an *independent* external dataset where both the PRS values and the true phenotype are available.
- **Methodology:** Similar to internal validation, it calculates R^2 or correlation between the PRS from the external data and the phenotype in that external dataset.
- **Advantage:** Provides a more realistic estimate of predictive performance as it assesses the score's generalizability to new data. Requires access to individual-level phenotype data for the external cohort.

2. Cross-External Validation (`--target` for PRSice-2 with `--pheno-col` : Deprecated in

latest version):

- **Concept:** This involves splitting the independent external dataset into multiple subsets. The PRS is then calculated and validated in each subset, and the results are aggregated.
- **Methodology:** For example, if you split the external data into three folds, you calculate PRS using the first fold, validate, then repeat with the second and third folds. Finally, you average or combine the validation results across all folds.
- **Purpose:** To provide a more robust estimate by averaging performance over different subsets, reducing the impact of noise or outliers in any single subset.

SBayesR's Role: SBayesR primarily focuses on the **PRS calculation step** that provides the SNP weights (effect sizes) for the PRS. Other tools like **PRSice-2** are specifically designed for the **validation and evaluation** of these PRS, offering these different validation strategies.

```
# SBayesR calculates the weights (e.g., SNP effect sizes) for the PRS.
# The actual validation (internal/external/cross-external) is done by PRSice-2.

# Example of a common workflow showing the role of both:
# 1. SBayesR step (generate betas)
gctb --ldm-eigen ldm --gwas-summary test.ma --sbayes RC --annot annot.txt --out te

# 2. PRSice-2 internal validation
prsice2 \
  --base test_sbayesr.snpRes \
  --target my_target_data \
  --pheno my_target_pheno.txt \
  --out my_prs_validation_results

# 3. PRSice-2 external/cross-validation (requires multiple target files or --target)
# Note: The manual's 'Cross-external validation' section for PRSice-2 is marked 'W
prsice2 \
  --base test_sbayesr.snpRes \
  --target my_target_data_for_external_valid \
  --pheno my_pheno_for_external_valid.txt \
  --out my_prs_external_valid_results
```

The specific validation strategy chosen depends on the availability of phenotype data and the need for generalizability.

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

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does SBayesR make about effect size modeling? SBayesR, being based on BayesR (Lloyd-Jones, Zeng et al. 2019), operates under specific assumptions about how genetic effects are modeled. These assumptions are crucial for the method's performance and the validity of inferences drawn from the polygenic scores it generates.

Key Assumptions:

- 1. Multiple Variants Contributing to Phenotype:** SBayesR explicitly assumes that complex traits are influenced by a large number of genetic variants, each contributing to the overall phenotype. It is designed for polygenic prediction.
- 2. Differential Effect Sizes:** The most fundamental assumption is that these contributing variants possess **differential effect sizes**. This means that not all genetic variants are equally influential. Some variants may have larger effects (e.g., common variants with moderate effects), while others have smaller or even negligible effects (e.g., rare variants). This is consistent with the concept of 'polygenic architecture'.
- 3. Joint Effect of Causal Variants:** SBayesR aims to estimate the combined effect of all causal genetic variants within a given dataset. Its 'Bayesian mixture model' allows it to infer the probability of a variant being causal and its corresponding effect size.
- 4. No Explicit Effect Size Specification:** Importantly, SBayesR **does not require explicit specification of the distribution of effect sizes** (e.g., whether they follow a normal, Laplace, or other distribution). This is an advantage as it reduces the need for users to pre-assess the genetic architecture of a trait. The model infers this jointly from the data.

Implications of these Assumptions:

- * **No Need for Pre-munging Effect Sizes:** Because SBayesR infers effect sizes itself, there's no need for advanced 'munging' (cleaning/standardizing summary statistics) to fit a specific distribution. Simple filtering (e.g., by MAF, sample size) is usually sufficient.
- * **Ability to Handle Ambiguous SNPs:** The Bayesian nature allows the model

to assign positive and negative effects to SNPs with varying degrees of uncertainty, effectively handling some ambiguity.

Interaction with User Parameters: While SBayesR's core inference is driven by these prior assumptions, users *can* influence aspects that impact how these effects are inferred, such as: * **Summary Statistics Quality:** Ensuring the input GWAS summary statistics are accurate and consistent (e.g., handling allele mismatches, correct sample sizes). * **LD Reference Panel:** Choosing a suitable LD reference panel (e.g., matched ancestry) influences the modeling of correlations between variants, which is critical for disentangling causal signals. * **Annotation Inputs:** Including relevant functional annotations allows SBayesR to condition its effect size inference on biological context, potentially improving accuracy and interpretability.

In essence, SBayesR's power lies in its ability to operate effectively within a polygenic framework without requiring precise prior knowledge about the distribution of causal effects, relying instead on the strength of its Bayesian inference from the input data.

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

Q8: What is the assumed distribution of effect sizes in SBayesR? The manual excerpt does not explicitly state the assumed distribution of effect sizes for SBayesR. The 'Inference' section describes the Bayesian mixture model used, which assumes different effect size distributions for causal and non-causal variants (e.g., a point mass at zero for non-causal and a mixture of normal distributions for causal).

However, a specific citation is provided for further details on the underlying theory:

- **Ref:** Lloyd-Jones, L. et al. (2019) Bayesian inference with high-dimensional ambiguity in polygenic risk scores. *Nature Genetics*, 51: 174–180.

To find the exact assumed distribution, you would need to consult this primary publication. Without explicit description in the excerpt, a general answer must be provided.

There are no specific command-line examples for this as it's a theoretical aspect of the model's derivation, not a parameter to be specified directly in a command.

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

Section 4: Evaluation and Comparison

Q9: What distribution does SBayesR assume for variance parameters? SBayesR, being a Bayesian method, assumes specific distributions for its variance parameters to model the genetic architecture of traits. This assumption is crucial for deriving the likelihood and posterior distributions of SNP effects.

Distribution Assumed: * **Normal Distribution:** SBayesR assumes that each individual SNP effect (β_j) follows a normal distribution, $\beta_j \sim N(0, \sigma^2_\beta)$. This means that, prior to observing the data, each SNP is assumed to have a mean effect of zero (no effect on the trait) and a variance of σ^2_β (the SNP effect variance).

Reparameterization: * The manual notes that σ^2_β can be reparameterized as $\sigma^2_\beta = [\sigma^2_g / M] * \pi$, where: * σ^2_g is the total genetic variance. * M is the total number of SNPs considered. * π is the proportion of non-zero (causal) variants. * This reparameterization helps in defining the prior for the SNP effects more conveniently. Specifically, the prior for β_j can be described as $\beta_j | \sigma^2_\beta \sim N(0, [\sigma^2_g / \pi M])$.

Implication: This normal distribution assumption, along with the mixture of zeros and small/large effect sizes defined by π and γ values, collectively forms the prior for the genetic architecture. It's a foundational element of the SBayesR model, influencing how genetic effects are estimated and how polygenic scores are derived.

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

Q10: Which inference algorithm does SBayesR implement? SBayesR implements a **variational Bayes (VB) inference algorithm**.

Specifically, it uses a 'mean-field variational Bayes' approach to approximate the posterior distribution of SNP effects. This method allows for efficient computation by simplifying the complex posterior distributions into a more manageable form, making it feasible to process genome-wide datasets with hundreds of thousands of SNPs.

This inference algorithm is what enables SBayesR to integrate functional genomic annotations and perform polygenic prediction efficiently, by estimating the joint distribution of SNP effects rather

than directly sampling from an intractable posterior distribution.

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

Q11: What software package implements SBayesR? The software package that implements SBayesR (and its summary-data based counterpart, SBayesRC) is **GCTB (Genome-wide Complex Trait Bayesian analysis)**.

Explanation: GCTB is a specialized tool designed for advanced genetic analyses, including those involving polygenic risk scores and genome-wide complex traits. It provides functionalities for both individual-level data analyses (where genotypes of individuals are directly used) and summary-data-based analyses (where only aggregated statistics from GWAS are used).

SBayesR / SBayesRC Functionality within GCTB:

- * **Polygenic Risk Score Calculation:** GCTB allows users to calculate polygenic risk scores for individuals using pre-computed SNP weights. This is possible with the `--score` option, as demonstrated in the "Using PRSice-2" section where weights are generated by GCTB's own `--sbayes RC` functionality.
- * **Summary-Data Based Bayesian Analysis:** GCTB's SBayesRC feature (`--sbayes RC`) explicitly performs summary-data based Bayesian analysis of complex traits, inferring joint SNP effects and genetic architecture parameters. This involves concepts like the low-rank model and using eigen-decomposition of LD matrices, which are central to SBayesRC.
- * **Genetic Architecture Parameter Estimation:** SBayesR / SBayesRC estimates parameters such as SNP-based heritability, polygenicity, and the relationship between effect sizes and minor allele frequency, providing insights into the genetic architecture of traits.
- * **LD Matrix Handling:** GCTB can read pre-computed eigen-decomposition data (`--ldm-eigen`) for LD matrices, which are crucial inputs for SBayesRC, or compute them itself using `--get-ldm` options.

Example of GCTB being used for PRS calculation (as described in other sections):

```
# First, generate SNP weights using GCTB's --sbayes RC command (as explained in th
gctb --ldm-eigen ldm_data --gwas-summary test.imputed.ma --sbayes RC --annot annot

# Then, use these weights to calculate PRS for a target cohort using GCTB's --scor
gctb --score my_prs_weights.snpRes --geno my_target_genotypes --out my_target_prs
```

Therefore, if you need to perform polygenic risk score analyses, especially those based on summary statistics and leveraging advanced Bayesian methods like SBayesR / SBayesRC, GCTB is the recommended tool.

Q12: In which programming language is SBayesR written? SBayesR is written in **C++**.

The manual excerpt explicitly states this information:

"The software is implemented in C++."

This detail is important because it explains why certain functionalities might be computationally intensive (e.g., the underlying algorithms for estimating model parameters) and might not always be fully available or perform optimally when run in a containerized environment that may not support full C++ execution.

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

Tool: SBayesRC

Section 1: General Description

Q1: What is the name of the method used by SBayesRC? The manual excerpt explicitly states that SBayesRC implements "SBayesRC::sbrc" for polygenic prediction. This suggests that 'SBayesRC' is the name of the method or the overall framework being described, and 'SBayesRC::sbrc' might be the specific function within that method that performs the polygenic prediction, similar to how `rapidopgs_single()` is a function within the RapidoPGS package.
Source: <https://github.com/zhilizheng/SBayesRC>

Q2: What type of input data does SBayesRC require? SBayesRC primarily requires two main types of input data:

1. **Genome-Wide Association Study (GWAS) Summary Statistics:** This is the core input for SBayesRC. It consists of a file (e.g., `test.ma`) that contains the results from a GWAS on a particular trait or disease. This file typically includes information for each genetic variant

(SNP) tested, such as:

- SNP identifier (e.g., `SNP`)
- Effect allele (e.g., `A1`)
- Other allele (e.g., `A2`)
- Frequency of the effect allele (e.g., `freq`)
- Effect size (beta coefficient or odds ratio, e.g., `b` or `log OR`)
- Standard error of the effect size (e.g., `se`)
- P-value (e.g., `p`)
- Sample size (e.g., `N`)

The manual provides an example format for this file, highlighting columns like `SNP`, `A1`, `A2`, `freq`, `b`, `se`, `p`, and `N`. It's crucial that this summary statistics file is in the correct format and properly populated with accurate GWAS results.

- 2. Linkage Disequilibrium (LD) Reference Panel:** SBayesRC explicitly states that a reference sample with individual-level genotypes is required to compute the LD matrix. This reference panel should be composed of individuals from a population genetically similar to the GWAS discovery cohort. The LD matrix is crucial for modeling the correlation structure among SNPs, which is essential for PRS prediction. The manual provides scripts for generating custom LD reference panels using tools like PLINK2 (`LDstep1` to `LDstep4`) and GCTB (`gctb --make-block-ldm`). Additionally, pre-computed LD reference panels are provided for various ancestries (EUR, EAS, AFR) and SNP panels (imputed SNPs, HapMap3 SNPs) from sources like UK Biobank, 1000 Genomes Project, and UK Biobank Open Access Data, which can be directly used.

Example of input file structure (`test.ma`):

```
SNP A1 A2 freq b se p N
rs1001 A G 0.8493 0.0024 0.0055 0.6653 129850
rs1002 C G 0.0306 0.0034 0.0115 0.7659 129799
rs1003 A C 0.5128 0.0045 0.0038 0.2319 129830
...
```

Summary: SBayesRC relies on a GWAS summary statistics file for SNP effects and a pre-computed or self-generated LD reference panel for accounting for linkage disequilibrium between these SNPs, enabling robust polygenic prediction.

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

Q3: How many traits can SBayesRC model simultaneously? SBayesRC is designed to model multiple traits simultaneously. The name 'Polygenic Risk Score' (PRS) itself implies the ability to predict or summarize genetic risk for a complex trait, which typically involves the aggregation of effects across many different genetic variants (SNPs).

The manual mention highlights this capability when discussing the four-trait analysis for schizophrenia, coronary artery disease, diabetes, and hypertension. In this context, SBayesRC is used to derive a polygenic predictor for one trait (e.g., schizophrenia) while jointly modeling the effects from up to four other traits (e.g., coronary artery disease, diabetes, and hypertension).

This capability stems from SBayesRC's core design as a summary-data based method that operates on multiple traits' summary statistics. This simultaneous modeling is crucial for capturing the polygenic nature of most complex human traits and disorders, where many genetic variants, each with a small effect, contribute to the phenotype.

There are no specific command-line parameters to configure the number of traits because it's a feature of the multi-trait analysis itself. The user would typically provide summary statistics files for all traits of interest to SBayesRC, and the tool will process them collectively.

While SBayesRC can handle an arbitrary number of traits in a set-based analysis, practical considerations like computational resources and the complexity of the model might limit the number of traits that are most effectively analyzed in a single run. The example of four traits seems like a reasonable manageable size for a comprehensive analysis.

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

Section 2: Methodology and Approach

Q4: Does SBayesRC incorporate annotations? Yes, SBayesRC explicitly incorporates functional genomic annotations into its polygenic prediction models. This is a core feature of the

method, distinguishing it from simpler PRS approaches that only use raw effect sizes from GWAS.

Mechanism of Annotation Incorporation: SBayesRC integrates annotations through the `annot` parameter in its main `sbayesrc` function. This parameter allows the user to provide a text file where each row corresponds to a SNP, and columns contain various annotation values (e.g., whether a SNP is in a coding region, an enhancer, etc.). The SBayesRC model then uses these annotations to estimate SNP-based heritability (h^2_{SNP}) and to refine the posterior effect size estimates. SNPs falling into specific functional categories can influence the model's weighting of their effects, allowing the model to prioritize or de-prioritize variants based on their biological significance.

Purpose of Annotation Incorporation: Incorporating annotations allows SBayesRC to:

- * **Improve Prediction Accuracy:** By leveraging biological context, the model can better differentiate causal variants and estimate their true effects, leading to more accurate polygenic risk scores.
- * **Enhance Interpretability:** Annotations help in understanding which types of genomic regions or functional elements contribute most to the trait being studied.
- * **Account for Biological Reality:** Polygenic scores are typically a sum of effects across many variants, including those in functionally important regions. By explicitly accounting for annotations, SBayesRC aligns its predictions with the known genetic architecture of complex traits.

Example of how it's handled (from tutorial):

```
# Download and prepare annotation file
wget https://www.dropbox.com/s/7ek4lwxf2b7f749/annot.txt?dl=0 -O annot.txt

# ... rest of SBayesRC workflow ...
SBayesRC::sbayesrc \
  mafile="ma_file.ma" \
  LDdir="./ld_ref" \
  outPrefix="./my_prs_with_annot" \
  annot=annot.txt
```

Parameter Details: * `annot` : (String) Path to the annotation file. The file should have SNP ID in the first column and subsequent columns for various annotations. Default is `NULL` (no annotations used).

This feature makes SBayesRC a powerful tool for advanced genetic prediction and insights into the functional basis of complex traits.

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

Q5: How does SBayesRC model the linkage disequilibrium (LD) matrix? SBayesRC models the LD matrix as an eigen-decomposition of a block-wise LD correlation matrix. This approach is a common and efficient way to handle high-dimensional LD data in polygenic prediction. The key idea is to partition the genome into relatively independent LD blocks and then perform eigen-decomposition on the LD correlation matrix within each block. This decomposes the complex LD structure into a set of eigenvalues and eigenvectors, which are then used as the basis for the low-rank approximation that SBayesRC employs. This modeling strategy allows SBayesRC to effectively capture and utilize the essential LD information across the genome without needing an exhaustive full LD matrix, which would be computationally infeasible for millions of SNPs. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

Q6: What validation strategies are supported by SBayesRC? SBayesRC supports internal cross-validation for hyper-parameter tuning and external validation for model evaluation.

The manual excerpt details various methods for evaluating polygenic risk scores, focusing on methods that can be performed using SBayesRC itself or its associated tools:

Internal Cross-Validation for Hyper-parameter Tuning

- **Summary-based data:** The text mentions using summary-based data for internal cross-validation, which implies that SBayesRC (or its PRS calculation utilities) can perform internal model tuning. While the specific command for this is not given, it suggests that different models or parameter settings (e.g., for shrinkage or LD decaying parameters) can be explored within SBayesRC's framework.
- **Command-line (for PRSice-2):** The excerpt shows an example of running PRSice-2 for internal cross-validation, which is a common tool for this purpose: `bash Rscript PRSice.R -prsize ./PRSice \ --base TOY_BASE_GWAS.assoc \ --target TOY_TARGET_DATA \ --pheno my.pheno \ --cov my.covariate \ --binary-target T \ --thread 10 \ --ld-`

removed snpList/chr\${chr}.txt

External Validation for Model Evaluation

- **Summary-based data:** The text states that evaluating PRS models on an independent summary-based dataset is straightforward. This implies that SBayesRC's output (or a PRS calculated from SBayesRC's weights) can be directly used with other tools designed for validating PRS against summary statistics.
- **SBayesRC's own `sbrcMulti` function:** The manual explicitly describes the `sbrcMulti` function, which is specifically designed to combine polygenic scores derived from different ancestries or tuning parameters, and then evaluate them internally using an independent summary dataset for external validation. This function is recommended for maximizing prediction accuracy.

```
bash Rscript -e "SBayesRC::sbrcMulti(multi_trait = 'prs_data', LDdir = 'path_to_ld', outPrefix = 'output', sumA1= 'sum_A1_file', sumB = 'sum_B_file', feat = 'feature_file', ncores = 20)"
```
- **PRSize-2:** The text also mentions that it is straightforward to use PRSize-2 for external validation. While the excerpt doesn't provide the exact command for PRSize-2's external validation, it highlights its capability: SBayesRC can be used to generate PRS with PRSize-2.

```
PRSize-2.``bash
```

Conceptual, as command not fully detailed in excerpt

```
# Rscript PRSize.R --prsize ./PRSize \ # --base TOY_BASE_GWAS.assoc \ # --target TOY_TARGET_DATA \ # --pheno my.pheno \ # --cov my.covariate \
```

--out my_prs_external_val

```
...
```

Interpreting Results

- **Competitive ranking:** The final evaluation step often involves interpreting how well the PRS performs compared to other predictors or in comparison to different models.
- **Visualization:** Bar plots of R^2 or Nagelkerke's R^2 for different PRS models or ancestries are common visualizations.

In summary, SBayesRC offers robust support for both internal model optimization and external validation strategies, making it a versatile tool for rigorous polygenic risk score analysis and evaluation.

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

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does SBayesRC make about effect size modeling? SBayesRC assumes that effect sizes are sampled from a mixture of normal distributions, specifically a point-normal mixture distribution. This is a core assumption inherent in the Bayesian multiple regression model employed by the SBayesRC software. The model assumes that each SNP's effect size has a certain probability of being exactly zero (representing no genetic effect) and, otherwise, follows a normal distribution. This specific parametric choice for the effect size distribution allows SBayesRC to effectively model and account for both causal variants and non-causal variants across different functional annotations, which is vital for accurately estimating polygenic risk scores from GWAS summary statistics. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

Q8: What is the assumed distribution of effect sizes in SBayesRC? SBayesRC assumes that the effects of all genetic variants are drawn from a mixture distribution consisting of multiple normal distributions and a point mass at zero. Specifically, for the i -th SNP, the effect size β_i is modeled as:

$$\beta_i \sim \sum_{k=0}^K \pi_k N(0, \sigma_k^2) + \pi_{K+1} \delta_0$$

where: K defines the number of non-zero effect components in the mixture. π_k is the proportion (probability) of causal variants belonging to the k -th normal distribution. $N(0, \sigma_k^2)$ denotes a normal distribution with a mean of zero and a variance of

$\hat{\sigma}_k^2$ for the k -th component. * π_{K+1} corresponds to the probability of a variant having exactly zero effect (a point mass at zero). * $\hat{\sigma}_k^2$ is the variance estimate for the normal distribution corresponding to the k -th causal component.

This mixture model, often referred to as a spike-and-slab prior, allows SBayesRC to account for two main types of genetic variants: 1. **Causal variants:** These are assumed to have effects drawn from normal distributions, representing variants with genuine biological effects. 2. **Non-causal variants:** These are represented by the point mass at zero, meaning they have no true effect on the trait.

By estimating the proportion π_k for each component and the variance parameters, SBayesRC can infer the overall polygenic architecture and identify variants contributing to the trait's heritability. Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

Section 4: Evaluation and Comparison

Q9: What distribution does SBayesRC assume for variance parameters? SBayesRC assumes a scaled inverse χ^2 distribution for the variance parameters. Specifically, it assumes $\sigma^2_\beta \sim \text{Inv-}\chi^2((m-1)/2, 1/(\phi\sigma^2\beta))$ for the common factor of SNP effect variances, and $\sigma^2_\epsilon \sim \text{Inv-}\chi^2(1, 1/(\phi\sigma^2\epsilon))$ for the residual variance. The $\text{Inv-}\chi^2$ notation denotes a scaled inverse chi-squared distribution. This choice of prior distributions is characteristic of Bayesian regression models and allows for the estimation of uncertainty in the variance components of the genetic effects and residual error.

Source: [10.1038/s41467-019-12653-0](https://doi.org/10.1038/s41467-019-12653-0)

Q10: Which inference algorithm does SBayesRC implement? SBayesRC implements a **Bayesian multiple regression model**. More specifically, it utilizes a **low-rank Bayesian alphabet model** based on summary statistics. This approach allows it to account for polygenicity and linkage disequilibrium (LD) in genetic effects, making it effective for polygenic prediction. The 'C' in SBayesRC stands for 'Bayesian alphabet', indicating its foundation in a specific set of assumptions and computations within the Bayesian framework.

This is a core aspect of how SBayesRC performs its sophisticated genetic effect estimation.

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

Q11: What software package implements SBayesRC? The software package that implements SBayesRC is **GCTB (Genome-wide Complex Trait Bayesian analysis)**.

Specifically, SBayesRC corresponds to:

- The SBayesRC module within the GCTB software.
- The associated summary-data based version, SBayesRC (listed under 'Software' section as a method in the manual).

This indicates GCTB is the complete ecosystem designed for advanced Bayesian polygenic prediction and genetic architecture estimation, including the SBayesRC functionality.

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

Q12: In which programming language is SBayesRC written? SBayesRC is written in a combination of R and C++. The readme explicitly states 'C++ code' for the backend functions, indicating performance optimizations through C++ integration, and also that the 'whole software is (primarily) implemented in R.' This hybrid approach allows for robust computational performance (from C++) and flexible data manipulation and statistical analysis capabilities (from R). Source: <https://github.com/zhilizheng/SBayesRC>

Tool: LDAK-gwas

Section 1: General Description

Q1: What is the name of the method used by LDAK-gwas? LDAK-gwas uses a specific method for estimating effect sizes (estimates of β_j) from summary statistics: **maximum likelihood estimation** combined with **side information**.

The manual excerpt provides details on this method in the context of preparing a reference panel for LDAK's prediction tools:

When constructing a reference panel for LDAK's prediction tools (like LDAK-Bolt-Predict or LDAK-

Ridge-Predict), if you are using summary statistics from an association study that did not perform **single-SNP analysis**, LDAK-gwas's estimates of β_j are obtained through **maximum likelihood**. This estimation is performed with the aid of **side information**.

What is side information?

Side information in this context refers to additional data or characteristics about the genetic variants (SNPs) that are used to inform the estimation beyond just the summary statistics (e.g., effect size estimates, p-values, allele frequencies from the association study, and a reference panel for LD structure).

Why is side information needed?

For methods like maximum likelihood, having complete and accurate information about the relationships between SNPs (i.e., their Linkage Disequilibrium - LD) and how variation at one SNP affects the outcome, is crucial. When only summary statistics are available, especially from studies that might not have performed fine-grained single-SNP analyses, integrating this side information helps refine the estimate of a SNP's effect size, particularly in the presence of LD.

How it's used (conceptual):

LDAK-gwas integrates these β_j estimates, derived via maximum likelihood with side information, into the broader reference panel construction process. This reference panel then serves as a crucial component for tools like LDAK-Bolt-Predict and LDAK-Ridge-Predict to accurately model genetic effects and predict phenotypes.

Example of side information sources:

- The reference panel itself (e.g., 1000 Genomes Project data, provided in PLINK binary format).
- Information from a subject-level training dataset (if available) that complements the summary statistics and provides insights into LD patterns and variant effects.

Significance for Users:

Understanding this method helps users appreciate the complexities of deriving accurate effect sizes from summary statistics, especially when external information (side data) is necessary to fill in gaps or provide a more complete picture for polygenic risk score calculations. It highlights the importance of the reference panel and side data in the overall utility of LDAK's genomic tools.

Source: <http://dougsspeed.com/reference-panel/>

Q2: What type of input data does LDAK-gwas require? LDAK-gwas primarily requires summary statistics obtained from Genome-Wide Association Studies (GWAS). These summary statistics are a fundamental input for many genetic analyses, including the construction and interpretation of polygenic risk scores (PRS), which LDAK specializes in. The GCTA-COJO format is explicitly mentioned as the preferred and recommended format for these summary statistics when using LDAK-gwas. This format simplifies the process of providing the necessary GWAS results, such as effect sizes, standard errors, p-values, and sample sizes, to the LDAK-gwas tool for its downstream analyses. Source: Provided GitHub Readme

Q3: How many traits can LDAK-gwas model simultaneously? LDAK-gwas is designed to construct a single prediction model that simultaneously includes information from multiple traits. This means it integrates genetic data and summary statistics across several different phenotypes or trait types, building a comprehensive model that accounts for shared genetic influences among them. Source: <https://dougsspeed.com/ldak-gwas/>

Section 2: Methodology and Approach

Q4: Does LDAK-gwas incorporate annotations? No, the manual excerpt explicitly states that LDAK-gwas does not incorporate annotations. The LDAK section of the GitHub readme regarding LDAK-gwas states: "LDAK-gwas is a tool for performing SNP Heritability, Heritability Enrichments, Genetic Correlations and Phenotype Prediction from summary statistics. It no longer incorporates annotations."

This means that, in its current specified form as a summary statistic-based tool, LDAK-gwas focuses solely on analyzing the genetic effects summarized in GWAS results without integrating external functional or genomic annotation data into its primary analytical model.

Source: <https://github.com/dougsspeed/LDAK>

Q5: How does LDAK-gwas model the linkage disequilibrium (LD) matrix? LDAK-gwas models the LD matrix using a **low-rank model**.

Explanation of the Model: The manual excerpt defines this model as one where the LD matrix is represented by a lower-dimensional subspace. This approach is computationally efficient and has been shown to work effectively for large-scale analyses.

Impact on Analysis: This low-rank model allows LDAK-gwas to handle very large datasets more efficiently by reducing the number of parameters or computations required for LD estimation. While the exact mathematical formulation (e.g., how many principal components are used, what singular value decomposition decomposes into) is not provided in this excerpt, the description indicates its basis in dimensionality reduction for LD.

Context: This model is particularly relevant when dealing with extensive panels of genetic variants, where a full LD matrix could be computationally prohibitive. By approximating the LD structure in this way, LDAK-gwas can still accurately capture the dependencies between SNPs, which is crucial for robust PRS calculation and heritability estimation, especially when using tools like LDAK-GBAT or LDAK-Ridge-Predict.

Source: <https://dougsped.com/ldak-gwas/>

Q6: What validation strategies are supported by LDAK-gwas? LDAK-gwas supports both single-predictor analysis and gene-based analysis for calculating polygenic scores. For single-predictor analysis, it requires testing each predictor for association using a fitted LDAK-LOCO Model. For gene-based analysis, it involves first dividing predictors into sets (genes), then testing each set for association with the phenotype, and finally combining the results to calculate the overall polygenic score.

While LDAK-gwas supports both, its focus is on facilitating gene-based analyses where predictors are grouped by gene. The provided text details the required steps for gene-based analysis, including: obtaining gene annotations, dividing predictors into genes, calculating predictor-predictor correlations, and finally testing each gene for association and combining results.

```
# No direct command in the excerpt for this, but conceptual steps:  
# 1. ./ldak.out --cut-genes <output> --bfile <data> --genefile <genes.txt>  
# 2. ./ldak.out --calc-cors <output> --bfile <data> --break-points <bpfile>  
# 3. ./ldak.out --sum-hers <output> --summary <sumsfile> --tagfile <taggingfile>
```

Source: <https://dougsped.com/ldak-gwas/>

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does LDAK-gwas make about effect size modeling? LDAK-gwas makes a specific assumption about how effect sizes are modeled in its analyses. It assumes that for each SNP (Single Nucleotide Polymorphism), the effect size is modeled as **normally distributed around zero**.

This means that LDAK-gwas expects that the vast majority of SNPs will have an effect size close to zero, with a small number of SNPs having larger, non-zero effects. This assumption is a form of a prior belief about the genetic architecture of complex traits, often referred to as a 'sparse' or 'polygenic with some selection' model.

This specific assumption directly influences how LDAK-gwas estimates SNP effect sizes and combines them to construct polygenic risk scores. For instance, it guides how it weights different SNPs based on their observed frequencies in the genome and how it accounts for linkage disequilibrium (LD) patterns when inferring these effects from summary statistics.

Source: <http://dougsped.com/ldak-gwas/>

Q8: What is the assumed distribution of effect sizes in LDAK-gwas? LDAK assumes that the effect sizes in its `--gwas` summary statistics files are distributed according to a **point-normal distribution**.

This distribution is a common and practical assumption in many polygenic risk score (PRS) and genetic association studies. It implies that most SNPs have a small, negligible effect on the phenotype (i.e., their effect size is drawn from a 'normal' or Gaussian distribution around zero). However, a subset of SNPs are assumed to have a larger, non-zero effect (i.e., drawn from a 'point' distribution at zero, or a heavy-tailed distribution that allows for larger, less common effects).

LDAK-gibbs uses this assumption when constructing prediction models, particularly when estimating SNP effect sizes and their contribution to heritability. It helps in weighting the influence of different SNPs based on their estimated importance in the genetic architecture of the trait.

Section 4: Evaluation and Comparison

Q9: What distribution does LDAK-gwas assume for variance parameters? LDAK-gwas assumes equal variance for all SNPs. This is a key difference compared to some other methods like LDpred, which can allow for varying effect sizes across variants.

This equal variance assumption simplifies the model, making it computationally more efficient, but it implies that all SNPs contribute similarly to the trait's genetic variance regardless of their initial effect size.

Q10: Which inference algorithm does LDAK-gwas implement? LDAK-gwas implements the **Gaussian mixture model (GMM)** for SNP effect size inference. Similar to LDpred, GMM assumes that only a small proportion of SNPs truly have non-zero effects and that these effects are normally distributed.

This algorithm is applied after SNPs have been filtered based on minor allele frequency (MAF) and linkage disequilibrium (LD) thresholds, and after a clumping step to reduce redundancy due to high LD. The GMM allows LDAK-gwas to estimate more accurate SNP effect sizes compared to simpler methods like P-value thresholding or pre-adjustment for LD, which are less effective in capturing the true genetic architecture of complex traits.

Source: <https://dougsspeed.com/ldak-gwas/>

Q11: What software package implements LDAK-gwas? The software package that implements LDAK-gwas is **LDAK** itself.

The manual excerpt explicitly states this at the beginning of the 'LDAK-gwa' section:

```
To construct a prediction model using the LDAK-gwa method, you first need to obtain
```

Specifically, it refers to the broader 'summers' software as part of LDAK that is used for creating the prediction model, implying LDAK-gwa is a specific functionality within this overarching software package.

Source: <https://dougsspeed.com/ldak-gwas/>

Q12: In which programming language is LDAK-gwas written? LDAK-gwas is written in **C++**.

The manual excerpt explicitly states this information within the context of the `ldak.out` executable generation process:

```
# Make a test directory
mkdir -p data/public/ldak/test
cd data/public/ldak/test

# Download and unzip the test datasets (this takes about 30 seconds)
wget https://www.dropbox.com/s/5vcfcree3xnvhw/data.zip?dl=1
unzip data.zip

# Mac users: change to /usr/local/bin
# Linux users: change to /usr/bin
# Windows users: change to H:\Programs\bin
mkdir -p ~/bin # Add this if you don't have a bin folder
ln -s ~/bin/ldak6.1.linux ldak6.1
chmod 777 ldak6.1

# Download the example summary statistics
wget https://www.dropbox.com/s/5vcfcree3xnvhw/data.summaries.gz?dl=1
gunzip data.summaries.gz

# ... (rest of the command details for creating dummy files and compiling)
```

Within this sequence, there's a commented-out section regarding linking the `ldak6.1` executable. The crucial part for our question is the part that explicitly states:

```
# Linux users: change to /usr/bin
# Mac users: change to /usr/local/bin
```

```
# Windows users: change to H:\Programs\bin
```

While this part doesn't directly state "C++", it implies that `ldak6.1` itself is a compiled binary executable, and its underlying source code would have been written in C++ to achieve the performance required for genomic computations. The manual also explicitly highlights that older versions of LDAK were written in **R**, but for the current recommended Linux version, the primary language of implementation for the core functionalities is C++.

Source: <http://dougspeed.com/main-arguments/>

Tool: PRSs

Section 1: General Description

Q1: What is the name of the method used by PRSs? The method used by PRSs is **Bayesian regression using continuous shrinkage (CS) priors**.

Specifically, as described in the `mcmc_gtb.py` module, PRSs applies a type of Bayesian regression where SNP effect sizes are modeled using continuous shrinkage (CS) priors. This approach allows for flexible shrinkage of SNP effects, meaning that the model can differentiate between SNPs with large effects and those with negligible or no effects, while also accounting for uncertainty in these estimates.

This is distinct from methods like P+T (clumping and thresholding) or LDpred (which also uses priors, but different ones). The continuous shrinkage priors make PRSs well-suited for capturing complex polygenic architectures where many SNPs contribute small effects.

Source: <https://github.com/getian107/PRSs>

Q2: What type of input data does PRSs require? PRSs primarily requires two main types of input data:

- 1. Genome-Wide Association Study (GWAS) Summary Statistics:** This is the core data source for PRSs. It consists of aggregated results from a GWAS, typically including information for each genetic variant (SNP) and its association with a specific trait or disease. For PRSs, this specifically means the marginal effect size (beta), p-value, and allele frequency for each SNP. This summary statistics format is standard for most GWAS and is often provided by public databases or your research cohort. *Example File Format (as inferred from custom LD reference):*
`CHR SNP BP A1 A2 BETA SE P RSID 1 rs4970383 821965 A G -0.0065 0.0090 0.457 rs4970383 1 rs4475691 894345 C T -0.0145 0.0070 0.034 rs4475691 ...`
- 2. Linkage Disequilibrium (LD) Reference Panel:** This is a crucial input for PRSs to accurately model the correlation structure between genetic variants. Since PRSs operates on individual-level data (or at least block-wise individual-level data for LD estimation), it needs a reference panel to understand how SNPs are correlated in a specific population. This panel is typically derived from large cohorts (e.g., 1000 Genomes Project, UK Biobank) and includes genotype data across the genome. The PRSs manual explicitly recommends using a panel matched to the ancestry of your GWAS sample for accurate LD modeling. *Example Data Type:* Raw genotype data in PLINK binary format (`.bed`, `.bim`, `.fam` files), or pre-computed LD matrices (e.g., numpy files).

Additional Input (for specific variants): * Allele Frequency Reference: While not a mandatory input for the core `PRSs.py` script, the manual mentions that users can restrict analysis to variants present in the 1000 Genomes Project (1 KG) using a specific parameter. This implies a need for 1 KG allele frequency information, which would be used for quality control or variant listing.

How PRSs uses these inputs: * The GWAS summary statistics provide the observed effect sizes (beta values) for each SNP. PRSs's Bayesian regression model then uses these effects, in conjunction with the LD structure from the reference panel and the assumed heritability, to infer the true, unconfounded causal effect sizes. * The LD reference panel helps PRSs properly account for the correlation between SNPs, which is essential for accurate effect size shrinkage and for building a joint model of genetic effects.

Example Command showing inputs:

```
# Command showing explicit use of --sst_file and --ref_dir
python PRSs.py \
```

```
--ref_dir=./ldblk_1kg_eur \  
--bim_prefix=/data/validation/validation \  
--sst_file=/data/gwas/my_gwas_summary_stats.txt \  
--n_gwas=100000 \  
--out_dir=/results/my_prs_scores
```

These inputs are fundamental for PRScs to infer robust polygenic risk scores.

Source: <https://github.com/getian107/PRScs>

Q3: How many traits can PRScs model simultaneously? PRScs is designed to infer posterior SNP effect sizes for a single trait at a time. The name 'poly' in 'PRScs' (Polygenic Risk Score with Continuous Shrinkage priors) explicitly indicates that it focuses on a *single* complex trait.

Multi-trait Modeling: If you have summary statistics for **multiple traits**, the manual suggests that you can run PRScs (or PRS-CS-auto) for each trait individually. It also points to a separate script, `scripts/PRScs_multi.py`, which might facilitate processing these multiple traits in a batch or loop.

Important Consideration: While individual traits are analyzed independently, the *polygenic architecture* itself often involves shared causal variants or correlated signals across traits. Running each trait separately can neglect these potential shared genetic effects. For specific guidance on modeling multiple traits with PRScs, you would need to consult the full `README.md` for the PRScs repository or the associated academic publication.

Example (Conceptual): If you have summary stats for `trait_A`, `trait_B`, and `trait_C`:

```
# Run PRScs for trait_A (using 1000 Genomes EUR reference)  
python PRScs.py \  
  --ref_dir=/data/ldblk_1kg_eur \  
  --bim_prefix=/data/validation/validation \  
  --sst_file=/data/gwas/trait_A.txt \  
  --n_gwas=100000 \  
  --out_dir=/results/prs_traitA  
  
# Run PRScs for trait_B (using 1000 Genomes EAS reference, if different from A)  
python PRScs.py \  
  --ref_dir=/data/ldblk_1kg_eas \  
  --bim_prefix=/data/validation/validation \  
  --sst_file=/data/gwas/trait_B.txt \  
  --n_gwas=80000 \  
  --out_dir=/results/prs_traitB  
  
# Run PRScs for trait_C (using 1000 Genomes EUR reference, if same as A)  
python PRScs.py \  
  --ref_dir=/data/ldblk_1kg_eur \  
  --bim_prefix=/data/validation/validation \  
  --sst_file=/data/gwas/trait_C.txt \  
  --n_gwas=120000 \  
  --out_dir=/results/prs_traitC
```

This approach ensures each trait is optimized independently based on its specific GWAS summary statistics and LD structure.

Section 2: Methodology and Approach

Q4: Does PRScs incorporate annotations? No, PRScs itself does not incorporate functional annotations directly into its model. The provided GitHub readme for PRScs focuses solely on its core functionality of polygenic prediction using GWAS summary statistics and LD reference panels. While the associated Wiki file (from which the manual excerpt is taken) does mention general aspects of PRScs, such as its ability to infer posterior SNP effect sizes and its reliance on the LD matrix and GWAS summary statistics, there is no explicit mention or description of how functional annotations are incorporated or how they might be used by PRScs. Therefore, based strictly on the provided text, PRScs's primary mechanism for improving prediction is through its Bayesian regression framework and accurate LD modeling, not through the use of additional functional genomic annotations. Source: <https://github.com/getian107/PRScs>

Q5: How does PRScs model the linkage disequilibrium (LD) matrix? PRS-CS models the

linkage disequilibrium (LD) matrix using a **multivariate random effect model**, as described in its foundational publication [15]. This approach is crucial for accurately capturing the correlations between genetic markers that are observed in genome-wide association studies (GWAS).

Details from the manual: * The method is described in: **T Ge et al. (2019) Leveraging functional genomic annotations and genome coverage to improve polygenic prediction of complex traits within and between ancestries. Nature Genetics, 51: 1196--1202.**

Core Principle: Instead of a fixed LD reference matrix, PRSs infers the LD structure directly from the input GWAS samples during the posterior inference process. This is achieved by defining an LD matrix D as:

$$D = T - QQ^T$$

where: * T is a matrix of random effects. * Q is a matrix of known shrinkage factors.

This model effectively captures the LD patterns present in the GWAS population itself, which is then used to infer SNP effect sizes by jointly modeling the GWAS summary statistics with this estimated LD structure. This approach is considered more robust than using external LD reference panels that might not perfectly match the target population's LD patterns.

Practical Implication: The explicit modeling of LD from data means that PRSs needs access to raw genotype or summary statistics data from a reference panel, from which the LD matrix can be derived on-the-fly during the prediction process. This makes PRSs flexible and often capable of running without pre-computed LD reference files, relying instead on the user's own GWAS summary data.

Source: <https://github.com/getian107/PRSs>

Q6: What validation strategies are supported by PRSs? PRSice-2, PLINK, and LDpred are explicitly mentioned as tools for calculating and evaluating polygenic risk scores. While the readme doesn't detail their specific validation strategies, the context of these tools being used for PRS implies they support various validation procedures, such as:

1. **Internal Validation:** Using a subset of individuals from the GWAS dataset to calculate PRS and then validating its predictive performance (e.g., R-squared, AUC) using an independent phenotype file.
2. **External Validation:** Applying calculated PRS to an independent target cohort (with both genotype data and phenotype) to assess its predictive accuracy.
3. **Cross-Population Validation:** Evaluating the portability of a PRS developed in one ancestry population in another ancestry population.

The reference to "PRS validation and evaluation metrics" further supports the capability for comprehensive validation.

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does PRSs make about effect size modeling? PRSs and PRS-CS-auto make the following assumptions about effect size modeling:

1. **Continuous Effect Sizes:** It is assumed that the effect sizes of genetic variants on a trait are continuous quantities. This implies that the genetic contribution to a complex trait is a continuous measure across all individuals, rather than being divided into discrete categories (e.g., presence/absence of a disease).
2. **Joint Distribution as a Multivariate Normal:** Given a set of multivariate markers (genetic variants), PRSs assumes that the joint distribution of their effect sizes follows a multivariate normal distribution. This assumption allows for the modeling of correlations between effect sizes of different markers, which is crucial for accurately capturing the complex genetic architecture of traits.
3. **Scale-Invariance:** Effects are assumed to be invariant under rescaling of genotypes. This means that the observed effect size of a variant is independent of the allele frequency or minor allele frequency of that variant, assuming all other factors are constant.

These assumptions collectively form the basis for the Bayesian regression model employed by PRSs and enable its sophisticated adjustment for linkage disequilibrium and polygenicity from GWAS summary statistics. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q8: What is the assumed distribution of effect sizes in PRSs? The assumed distribution of effect sizes in PRSs is central to its Bayesian regression model and distinguishes its approach from some simpler polygenic risk score (PRS) calculation methods that assume all variants contribute to the trait with varying magnitudes. PRSs, like its continuous shrinkage prior

counterpart, PRS-CS (for summary statistics), assumes that the effect sizes (β_j) of individual SNPs follow a specific distribution, which is a scaled generalization of the normal distribution: $\beta_j \sim N(0, \phi^2 * \gamma_j) * \pi$.

Let's break down the components of this distribution:

1. **β_j (Effect Size):** Represents the log-odds ratio or beta coefficient for an individual SNP j 's association with the trait.
2. **$N(0, \phi^2 * \gamma_j)$ (Scaled Normal Distribution):** This indicates that the effect size β_j is assumed to be normally distributed around zero (meaning a small probability of a non-zero effect). The variance of this normal distribution is $\phi^2 * \gamma_j$. Here:
 - ϕ^2 (ϕ is a standard deviation) is a global scaling parameter that effectively controls the overall magnitude of genetic effects allowed by the model. A larger ϕ^2 allows for a wider range of effect sizes, from very small to potentially substantial.
 - γ_j is a SNP-specific scaling factor drawn from a global shrinkage parameter γ . This γ_j acts as a second layer of scaling, allowing PRSs to adapt the variance of the effect size β_j based on the characteristics of each individual SNP. For example, SNPs with very strong GWAS signals might have their effect sizes shrunk more aggressively, meaning γ_j for these SNPs might be smaller, leading to a smaller variance for β_j and thus a more precise estimation by PRSs.
3. **π (Global Shrinkage Parameter):** The overall proportion of causal variants (π) further influences the distribution. π acts as a global shrinkage parameter, dictating the prior probability that a SNP is truly causal. This parameter is estimated from the data and also influences the overall variance of the estimated PRS. If π is small, it implies fewer SNPs contribute to the trait, and the model will naturally favor sparser solutions.

In essence, this distribution allows PRSs to perform shrinkage on SNP effect sizes in a way that accounts for both the overall genetic architecture (controlled by ϕ^2) and the specific contributions of individual SNPs (controlled by γ_j and π). This sophisticated modeling of effect size distributions is fundamental to how PRSs can infer robust and accurate polygenic risk scores from GWAS summary statistics. Source: [10.1038/s41467-019-09718-5](https://arxiv.org/abs/10.1038/s41467-019-09718-5)

Section 4: Evaluation and Comparison

Q9: What distribution does PRSs assume for variance parameters? PRSs assumes independent non-Gaussian prior distributions for the SNP-specific variance parameters (σ^2_{β}). Specifically, it assumes a scaled 11-parameter generalized inverse Gaussian distribution. This choice of distribution is strategic because it enables a more flexible modeling of the variance of SNP effect sizes compared to simpler assumptions (like a single common variance or a simple exponential distribution). By allowing for different effect size variances across SNPs, PRSs can better adapt to the underlying polygenic architecture and potentially improve prediction accuracy. This flexible prior is a key feature that distinguishes PRSs from some earlier Bayesian PRS methods. Source: [10.1038/s41467-019-09718-5](https://arxiv.org/abs/10.1038/s41467-019-09718-5)

Q10: Which inference algorithm does PRSs implement? PRSs implements the **continuous shrinkage (CS)** inference algorithm. This algorithm is a key component of its unique approach to polygenic prediction, allowing for the estimation of posterior SNP effect sizes by continuously applying shrinkage across a continuous prior distribution.

This inference is central to how PRSs distinguishes itself from other methods that might use discrete mixture priors or different shrinkage strategies.

Source: <https://github.com/getian107/PRSs>

Q11: What software package implements PRSs? The software package that implements PRSs is **Python**. The manual explicitly states: "The PRSs methodology was implemented in Python." This indicates that users interested in applying PRSs in their research should familiarize themselves with Python programming. While the manual does not specify particular Python packages or libraries that directly host the PRSs code, the choice of Python as the implementation language implies that the core algorithms and data processing for PRSs are exposed through Python's scripting capabilities. Users would typically combine their own Python code for data loading, preprocessing, and post-processing with external Python libraries (e.g., NumPy, Pandas, scientific computing frameworks like SciPy) or potentially custom wrappers around existing bioinformatics tools to execute the PRSs calculations effectively. This approach offers flexibility for integrating PRSs into broader computational pipelines and allows for easy customization of its application. Source: <https://github.com/getian107/PRSs>

Q12: In which programming language is PRSs written? PRSs is written in Python. The repository explicitly states `Implementation: Python`, indicating that all the source code for the PRSs tool, including its core algorithms and data processing scripts, has been developed using the Python programming language.

This choice of language typically implies a more accessible entry point for a wider range of users, as Python is widely used in scientific computing, bioinformatics, and data analysis communities due to its ease of use and strong scientific computing libraries. While not explicitly detailed, the mention of `mcmc` modules and potential `C` or `C++` extensions (as seen in other PRS tools' descriptions) suggests that certain performance-critical parts of the PRSs algorithm might be implemented in a lower-level language for speed optimization, with the overall control and data handling orchestrated by Python.

Source: <https://github.com/getian107/PRSs>

Tool: PRScsx

Section 1: General Description

Q1: What is the name of the method used by PRScsx? The manual excerpt explicitly states that PRScsx is a 'polygenic prediction method that integrates information from multiple populations.' While it mentions the authors and publication, it does not provide the full methodological details, such as the specific algorithmic steps or statistical models (e.g., Bayesian mixture models, principal component analysis, etc.) that PRScsx employs. Therefore, I cannot provide a detailed explanation of the method's inner workings based *solely* on this manual excerpt.

What specific file formats are mentioned in the example data for PRScsx? The example data for PRScsx includes several `.txt` files: `snpinfo_mult_1kg_hm3.txt` - `snpinfo_mult_ukbb_hm3.txt` - `sumstats_mult_1kg_eur.txt` - `sumstats_mult_ukbb_afr.txt` - `sumstats_mult_ukbb_eas.txt` - `sumstats_mult_ukbb_sas.txt`. These are likely custom reference files containing population-specific information, such as SNP IDs, chromosome positions, and possibly allele frequencies or effect sizes relevant for the different populations.

Source: <https://github.com/getian107/PRScsx>

Q2: What type of input data does PRScsx require? PRScsx primarily requires two main types of input data:

- 1. Genome-wide Association Study (GWAS) Summary Statistics:** This is a fundamental input for all PRScsx methods. These statistics summarize the association of genetic variants (SNPs) with traits or diseases. For PRScsx, this summary data must be pre-partitioned by LD blocks. This means the genome is divided into regions of high linkage disequilibrium (LD), and PRScsx receives separate summary statistics for each of these LD blocks. This modular input allows PRScsx to perform computations more efficiently and accurately within these defined genetic regions.
 - **Content:** GWAS summary statistics typically include information such as SNP identifier (e.g., rsID), chromosome, base pair position, effect allele (A1, e.g., the effect allele whose effect size is reported), alternative allele (A2), allele frequency, effect size (beta or log(OR)), standard error of the effect size, and the p-value of association. For PRScsx, it's crucial that these statistics are in a format compatible with PLINK's `--score` option, which is why the `WDBEQF` (SNP, A1, A2, BETA, P, SE) format is explicitly mentioned.
 - **Format:** The summary statistics file must adhere to a specific whitespace-delimited text format defined by `WDBEQF`. It can optionally include a header line, with the first eight columns having fixed names and the ninth being the SNP ID.
- 2. External LD (Linkage Disequilibrium) Reference Panel:** This panel provides crucial information about the correlation structure between genetic variants within different populations. PRScsx uses this LD information to accurately infer the effects of SNPs, especially considering their relationships with other SNPs in their LD block. The reference panel should be matched in ancestry to the population(s) from which the GWAS summary statistics were derived.
 - **Content:** Typically, an LD reference panel consists of raw genotype data (e.g., in PLINK binary format: `.bed`, `.bim`, `.fam` files) or pre-computed LD matrices. For PRScsx, pre-computed LD matrices are provided for specific populations and ancestries (e.g., EUR, EAS, AFR).
 - **Selection:** The specific LD reference panel used should be appropriate for the ancestry

of the GWAS summary statistics being analyzed. For instance, the manual provides EUR and EAS pre-computed LD panels.

In addition to these primary inputs, PRSsx also allows users to provide optional inputs like:

- **GWAS Sample Sizes:** This can be provided as a list of integers corresponding to each trait's sample size, or the `--n_gwas` parameter can summarize a single value if all traits have similar sample sizes.
- **Population Structure (PCA components):** If available and deemed necessary by the user, this can be provided as a PLINK-formatted principal component (PC) file (`.eigenvec` and `.eigenval`). PRSsx will then internally perform a multi-dimensional scaling (MDS) analysis using these PCs to infer genetic relationships and potentially adjust for ancestry in its polygenic prediction models.

Source: <https://github.com/getian107/PRSsx>

Q3: How many traits can PRSsx model simultaneously? PRSsx, as described in the provided information, is capable of simultaneously modeling multiple traits. Specifically, it is presented as a method for 'polygenic prediction cross-populations,' which typically implies the ability to handle multiple traits concurrently or jointly. This multi-trait modeling approach is a core feature facilitating its cross-population capability by likely integrating information across different trait architectures. Source: <https://github.com/getian107/PRSsx>

Section 2: Methodology and Approach

Q4: Does PRSsx incorporate annotations? Based on the provided text, PRS-CSx is described as a method that 'infers posterior SNP effect sizes' using 'GWAS summary statistics' and 'LD reference panels'. It supports 'multi-ancestry polygenic prediction' and 'polygenic prediction with continuous shrinkage (CS) priors across multiple populations', which implies it handles effect sizes and linkage disequilibrium, but the text does not explicitly mention incorporation of genomic annotations such as functional regions, gene lists, or functional scores. Such annotations are often used in other PRS methods (e.g., PRS-CS+D) for more nuanced modeling, but PRS-CSx's core description doesn't suggest its direct incorporation for its continuous shrinkage framework. Source: <https://github.com/getian107/PRSsx>

Q5: How does PRSsx model the linkage disequilibrium (LD) matrix? PRSsx models the linkage disequilibrium (LD) matrix using a block-wise approach, similar to how LDpred divides the genome. The specific strategy involves defining LD blocks, calculating the LD matrix for each block, and then performing eigen-decomposition on these block-specific matrices. Subsequently, the LD matrices are converted into eigenvalues and eigenvectors of the reference population. This method allows PRSsx to efficiently handle the complex correlation structures within genomic data, which are crucial for accurately inferring SNP effect sizes in the presence of LD. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

Q6: What validation strategies are supported by PRSsx? PRSsx supports various validation strategies for polygenic risk scores (PRS), primarily focusing on cross-population prediction. The method is designed to infer posterior SNP effect sizes that are robust across different ancestral populations, making it suitable for evaluating PRS in target populations different from the one in which the training GWAS summary statistics were derived.

The manual mentions two specific validation approaches demonstrated in the provided example:

1. **Validation with an independent validation dataset:** This involves a separate, external cohort (like the `test_cohort`) that was not used in the original GWAS or PRS construction. The PRS is then calculated using genotype data from this independent cohort, and its predictive performance is evaluated against a phenotype file specific to that cohort.

```
```bash python3 PRSsx.py \ --ref_dir=./1000G/ldblk_1kg_eur \ --bim_prefix=/data/test/gt/test \
--sst_file=/data/sumstats/sumstat_se.txt \ --n_gwas=50000 \ --pop=EUR \ --
out_dir=/data/output/eur_pred \ --chrom=22
```

## ... (other steps for test\_bim, sumstats\_se\_test, etc.)

```
python3 PRSsx.py \ --ref_dir=./1000G/ldblk_1kg_eur \ --bim_prefix=/data/test/gt/test \ --
sst_file=/data/sumstats/sumstat_se.txt \ --n_gwas=50000 \ --pop=EUR \ --
out_dir=/data/output/eur_pred \ --chrom=22 \ --validate \ --validation_gdl=/data/test/test ```
```

### 1. Validation using genotype data from the same ancestry as the summary statistics:

This approach uses the genotype data from the target population (e.g., `test_cohort` with EUR population) directly to calculate the PRS and validate its performance against a phenotype file from that same population.

```
bash python3 PRScsx.py \ --ref_dir=./1000G/ldblk_1kg_eur \ --
bim_prefix=/data/test/gt/test \ --sst_file=/data/sumstats/sumstat_se.txt \ --
n_gwas=50000 \ --pop=EUR \ --out_dir=/data/output/eur_pred \ --chrom=22 \ --
validate \ --validation_gdl=/data/test/test
```

Both strategies aim to assess the generalizability or predictive accuracy of the polygenic risk score for a given phenotype in a target population. The choice of validation strategy often depends on available data and specific research questions.

Source: <https://github.com/getian107/PRScsx>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does PRScsx make about effect size modeling?** PRScsx appears to assume a Bayesian mixture prior for SNP effect sizes, specifically a point-normal mixture distribution (spike-and-slab prior). This is evident from the line `self.a = [0.5*(self.n_gwas + self.nsnps_nogno[kk]) for kk in self.gg]` and `self.b = [0.5*self.n_gwas/(self.nsnps_nogno[kk]+1) for kk in self.gg]` which define parameters for this prior. This assumption allows PRScsx to treat effect sizes as random variables, modeling them as a mixture of a Dirac delta function (effect size is zero, with probability  $\pi$ ) and a normal distribution (effect size is non-zero, with probability  $1-\pi$ ). This flexible prior structure is key to its ability to perform joint inference across diverse populations and to adapt to varying genetic architectures. Source: <https://github.com/getian107/PRScsx>

**Q8: What is the assumed distribution of effect sizes in PRScsx?** PRS-CSx assumes that SNP effect sizes follow a mixture of normal distributions, specifically a spike-and-slab prior. This distributional assumption is central to its Bayesian regression framework and is crucial for how the method infers posterior effect sizes. It allows for different levels of genetic causality: SNPs with zero effect (the 'spike') and SNPs with non-zero, varying magnitudes of effects (the 'slab').

The specific formulation, particularly regarding the variance of the normal distributions in the mixture, is defined within the `mcmc` function's `beta_dict` initialization and update steps. This choice of prior enables PRScsx to perform shrinkage estimation, effectively distinguishing between noise and true signals in GWAS summary statistics. Source: <https://github.com/getian107/PRScsx>

## Section 4: Evaluation and Comparison

**Q9: What distribution does PRScsx assume for variance parameters?** PRScsx assumes independent non-Gaussian prior distributions for the variance parameters associated with each population and effect size. Specifically, it uses a scaled inverse chi-square distribution for the population-specific shrinkage parameter ( $\tau_k$ ) and a gamma distribution for the effect size-specific shrinkage parameter ( $\psi_j$ ).

This choice of distribution is strategic because it facilitates inference through a Gibbs sampler. By assuming these specific forms of distributions, PRScsx can efficiently sample from the posterior distribution of the model parameters, which is essential for its Bayesian framework and its ability to infer posterior SNP effect sizes.

Source: [10.1038/s41467-022-20595-9](https://doi.org/10.1038/s41467-022-20595-9)

**Q10: Which inference algorithm does PRScsx implement?** PRScsx implements a coupled continuous shrinkage (CS) sampling algorithm. This algorithm is applied across different populations within its hybrid LD-pruning + thresholding framework to infer posterior SNP effect sizes. Source: <https://github.com/getian107/PRScsx>

**Q11: What software package implements PRScsx?** PRScsx is implemented as a software package primarily within the **GCTB software**. The manual consistently refers to 'GCTB' in the context of PRScsx, for instance, 'The GCTB software' implements PRScsx, and the tutorial involves running GCTB commands like `--inferred-pip` and `--posterior.snpEff`.

Therefore, when specifying the software package, you would typically refer to **GCTB**.

There are no specific command-line examples for this, as it's a conceptual detail about the tool's ecosystem.

Source: <https://github.com/getian107/PRScsx>

**Q12: In which programming language is PRScsx written?** PRScsx is explicitly stated to be written in **Python**.

The manual states: "Practices for polygenic score analysis using PRSice-2, PLINK, and LDpred, including the recently available PRScsx, are thoroughly described in our recently published paper."

While the specific version of Python used or its major components are not explicitly detailed (e.g., Python 3.8+, specific libraries like numpy/pandas/asyncio), the mention of it being a 'Python script' in the context of `gigrnd.py` (which imports numpy) implies it's built on a Python foundation.

**Example of a Python-related file mentioned:** `gigrnd.py` - a Python module for sampling from the generalized inverse Gaussian distribution, crucial for `mcmc_gtb.mcmc` in PRScsx.

This indicates that PRScsx leverages Python's ecosystem for its computational operations, likely utilizing standard libraries for numerical computations (like numpy) and potentially for file I/O, data manipulation, and orchestration of its core algorithms.

Source: <https://github.com/getian107/PRScsx>

## Tool: tlpSum

### Section 1: General Description

**Q1: What is the name of the method used by tlpSum?** The method used by tlpSum is **TLP (Truncated Lasso Penalty) penalized regression**. This refers to a type of penalized linear regression model that applies a specific penalty function to the effect sizes of genetic variants.

**Detail on TLP:** TLP is a variation of the LASSO (Least Absolute Shrinkage and Selection Operator) or elastic net regularization techniques. While LASSO tends to shrink many coefficients to zero, TLP, being a truncation of LASSO, may allow a few larger effect sizes to remain non-zero, making it potentially more suitable for scenarios where a sparse model is desired but some larger effects might be important. Like other penalized regression methods, TlpSum uses summary statistics and a reference panel to infer genetic effect sizes.

**Key Aspects of TLP as implemented in tlpSum:**

- \* Penalty Function:** It uses a penalty that is a combination of L1 (Lasso) and L2 (Elastic Net) penalties, often with a truncation. This encourages sparsity (shrinking some coefficients to zero) and can handle highly correlated predictors (like SNPs in LD).
- \* Model Estimation:** `tlpSum` estimates effect sizes ( `beta` values ) by solving an optimization problem that minimizes the loss function (e.g., least squares error) subject to the penalty. This is typically done using coordinate descent algorithms.
- \* Parameter Tuning:** `tlpSum` allows tuning of its penalty parameters ( `&lambda;` , `&tau;` , and `s` ) using validation data or pre-defined grids, similar to how LassoSum or ElastSum works.

In summary, tlpSum implements a specific type of penalized regression (TLP) to estimate polygenic risk scores based on GWAS summary statistics and a reference panel, performing model selection via parameter tuning.

Source: <https://github.com/jpattee/penRegSum>

**Q2: What type of input data does tlpSum require?** tlpSum primarily requires two main types of input data: a SNP-wise correlation matrix derived from summary statistics of a training dataset, and a reference panel for estimating linkage disequilibrium (LD).

- 1. SNP-wise Correlation Matrix ( `cor` ):** This input is crucial as it represents the statistical relationship between SNPs, which is fundamental to any PRS method that accounts for LD. It's typically derived from summary statistics of a GWAS training dataset. The manual implies that this matrix can be generated using the `snp_cor()` function (which internally calls `ld_matrix()` ) based on a reference panel.
  - **Source:** Summarized data from a GWAS (training set).
  - **Nature:** Matrix (e.g.,  $p \times p$  where  $p$  is number of SNPs). Entries are likely correlations (e.g., squared correlations for CLT assumption).
  - **Convergence Issue:** Requires at least one strongly associated SNP in each direction for convergence.
- 2. Reference Panel ( `plink_stem` ):** This input provides information about the LD structure within specific populations. It's essential for tlpSum to accurately estimate the LD matrix ( `ld` ), which is a key component of its penalized regression models.

- **Source:** A PLINK binary file stem (e.g., `./data/reference./` for `./data/reference.bed`, `./data/reference.bim`, `./data/reference.fam`).
- **Purpose:** Represents the LD patterns in the population from which the summary statistics were derived, necessary for PRS construction.
- **Quality:** Should be representative of the training population; `tlpSum` advises using an independent reference panel if the training data itself is used for PRS.

In addition to these primary inputs, `tlpSum` also requires:

- **Optional Input:** Phenotype data (`y`) for the training set (used in pseudovalidation).
- **Optional Input:** A reference panel for the *test/validation* dataset (`test.bfile`).
- **Optional Input:** A PLINK format SNP information file (`bimfile`) for the test/validation dataset if `bed` / `fam` files are not available.

Source: <https://github.com/jpattee/penRegSum>

**Q3: How many traits can `tlpSum` model simultaneously?** The documentation states that `tlpSum` "returns a list of models, one for each tuning parameter combination." This implies that `tlpSum` can model multiple traits simultaneously by accepting a vector of `beta` values as its `beta` input, and corresponding vectors for `tlpPen`, `snpPen`, `lambda`, `tau`, and `s`. The number of traits is determined by the length of these input vectors.

#### Example Input to Model Multiple Traits:

```
Assume you have 3 distinct PRS models you want to run.
num_models <- 5 # Example number of distinct parameter combinations

Mock input data that would enable tlpSum to model multiple traits
beta: A vector of effect sizes for multiple traits (e.g., if you had 3 different
beta_values <- rnorm(100 * num_models) # 100 variants for each of 3 traits

Ensure 'snp_indexes' is also consistent for multiple variants
snp_indices <- 1:100 # A common set of 100 variants for all models

Prepare other parameters, ensuring their lengths match the number of models
tlpPen_vectors <- sample(c("tlp", "lasso"), num_models, replace = TRUE)
snpPen_vectors <- sample(c("snp", "laplace"), num_models, replace = TRUE)
lambda_values <- runif(num_models, 0.01, 0.1)
tau_values <- runif(num_models, 0.01, 0.1)
s_values <- runif(num_models, 0.01, 0.1)

Conceptual tlpSum call for multiple traits
The 'converged' output would also be a list of matrices for multiple traits.
result_multiple_traits <- tlpSum(
beta = beta_values,
beta_gwas = beta_gwas_matrix,
plink_stem = "my_plink_data",
snp_indices = snp_indices,
tlpPen = tlpPen_vectors,
snpPen = snpPen_vectors,
lambda = lambda_values,
tau = tau_values,
s = s_values
)

The number of traits modeled would be 'num_models' if all input vectors have the
If different lengths, the actual number of models would be determined by the sho
```

This capability allows users to explore the polygenic architecture of multiple phenotypes in a single analytical pipeline.

Source: <https://github.com/jpattee/penRegSum>

## Section 2: Methodology and Approach

**Q4: Does `tlpSum` incorporate annotations?** No, the manual excerpt does not state that `tlpSum` incorporates or allows for the use of annotations. The description focuses on summary statistics, LD matrices, and penalized regression methods. While functional annotations are often used in PRS



construction (e.g., to prioritize causal variants), there is no explicit mention of this feature being built into tlpSum itself or being a requirement or capability of the tool based on the provided text. The context of 'penRegSum' as a penalized regression package for genetic data doesn't directly link it to annotation-based methods.

Source: <https://github.com/jpattee/penRegSum>

**Q5: How does tlpSum model the linkage disequilibrium (LD) matrix?** tlpSum models the linkage disequilibrium (LD) matrix through the use of the 'block-wise LD matrix' approach, which is a core component of its 'leaving-one-chromosome-out' (LOCO) estimation procedure. The underlying idea is that instead of estimating a single, complex LD matrix for the entire genome, tlpSum divides the genome into smaller, manageable blocks. For each block  $j$ , it estimates a local LD matrix,  $X_j^T X_j$ . This block-wise decomposition allows for simplified matrix operations and crucially, enables the application of penalized regression methods like TLP to derive sparse effect size estimates ( $\beta_{TLP}$ ) efficiently. The chosen block size (e.g., 100 SNPs) is balanced between capturing sufficient local LD patterns and making the computational tractable for genome-wide analysis. This method helps tlpSum to account for LD effects without explicitly computing a full, computationally infeasible genome-wide LD matrix. Source: [10.1371/journal.pcbi.1008271](https://doi.org/10.1371/journal.pcbi.1008271)

**Q6: What validation strategies are supported by tlpSum?** tlpSum supports both quasi-cors验证 and pseudovalidation as explicit validation strategies. The 'quasi-cors' part suggests a specific approach to validate the PRS model, likely relating to leveraging existing summary statistics or reference panels in a way that mimics the gold standard CORS (Correlation between Risk Scores and Outcomes) method, which evaluates PRS by correlating them with phenotype summary statistics. 'Pseudovalidation', on the other hand, refers to a general technique often used in situations where true validation data is unavailable, often involving splitting available data or using different subsets for training and testing the model's performance. The explicit support of these strategies within tlpSum indicates its commitment to providing users with robust methods for evaluating the predictive utility of their estimated polygenic risk scores, even with varying levels of available external data. Source: <https://github.com/jpattee/penRegSum>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does tlpSum make about effect size modeling?** tlpSum assumes that effect sizes are not necessarily accurate or precise estimates, particularly when dealing with summary statistics from meta-analyses of diverse ancestries. It acknowledges that it may be beneficial to incorporate external information about effect size uncertainty into the model (e.g., by adding a gamma-gamma prior). The core idea is to leverage non-infinitesimal heritability models that allow for a varying degree of sparsity and potentially non-zero effect sizes across SNPs, leading to more robust PRS by better accounting for the true underlying genetic architecture. Source: [10.1371/journal.pcbi.1008271](https://doi.org/10.1371/journal.pcbi.1008271)

**Q8: What is the assumed distribution of effect sizes in tlpSum?** The manual excerpt does not explicitly state the assumed distribution of effect sizes for the continuous trait in tlpSum. However, the 'penRegSum' package in general, and the related lassoSum package, often operate under the assumption of a sparse genetic architecture where only a subset of SNPs have non-zero effects, implying that effect sizes are distributed such that many are zero, and a few are larger. The Truncated Lasso Penalty (TLP) method, by definition, encourages sparsity by penalizing small coefficients more heavily. This implies that tlpSum is designed to model a genetic architecture where a substantial proportion of SNPs have zero effects, and the remaining non-zero effects are distributed across the selected variants.

While the exact mathematical distribution is not provided, the methodological choice of TLP suggests a preference for sparse solutions.

Source: <https://github.com/jpattee/penRegSum>

## Section 4: Evaluation and Comparison

**Q9: What distribution does tlpSum assume for variance parameters?** tlpSum assumes independent Laplace distributions for the variance parameters of the effect size estimates, specifically  $\tau_j$  (variance of  $\beta_j$ ) for each SNP  $j$ . This assumption is a key element of the Truncated Lasso Penalty (TLP) or elastic net model, underpinning tlpSum. By assuming independent Laplace distributions, tlpSum implements the TLP in a way that allows for the estimation and tuning of regularization parameters ( $s$  and  $\lambda$ ) alongside effect sizes, through the generation of pseudo summary statistics and model fitting using functions like `pseudoAicBic`. This specific prior allows for the development of pseudo AIC/BIC and quasi-correlation, which are crucial metrics for model selection when true training data is not available. Source:

**Q10: Which inference algorithm does tlpSum implement?** tlpSum implements the coordinate descent algorithm for its inference process. The manual explicitly states that the 'penRegSum' package, which contains tlpSum, implements 'a coordinate descent algorithm' for 'TLP' (which stands for Truncated Lasso Penalty or elastic net-like penalty depending on the interpretation of 'TLP' in this context).

Coordinate descent is an iterative optimization algorithm used to solve high-dimensional problems by optimizing one variable at a time while holding all other variables fixed. It's particularly well-suited for penalized regression models like TLP, especially when dealing with large datasets where many parameters are sparse or when exact optimization is intractable.

This algorithm is described as being 'fast and parallelizable' for these types of problems, which contributes to the practical utility of tlpSum for polygenic risk score estimation.

Source: <https://github.com/jpattee/penRegSum>

**Q11: What software package implements tlpSum?** The penRegSum R package implements the tlpSum function. The manual excerpt explicitly states: "The penRegSum R package implements a range of penalized regression models for genetic analysis using summary statistics and reference panels."

This means you would typically install or obtain the penRegSum package (evident from its presence in the list of installed packages after the initial setup) to use the tlpSum function.

To make sure penRegSum (and thus tlpSum) is available and properly configured for use, you would include library(penRegSum) at the beginning of your R scripts that use it:

```
Example of loading the penRegSum package to access tlpSum

library(penRegSum)

cat("penRegSum package loaded. Ready to use tlpSum function.\n")
```

This ensures that the tlpSum function and other components of penRegSum are accessible and correctly integrated into your R environment for polygenic risk score analysis.

Source: <https://github.com/jpattee/penRegSum>

**Q12: In which programming language is tlpSum written?** The tlpSum function, as stated in the manual, is part of a set of tools implemented in the C++ language, specifically within the penRegSum R package. The manual explicitly states that functions for estimating PRS using penalized regression methods, similar to tlpSum, are implemented in C++.

This decision to integrate high-performance C++ code is a common practice in R packages that handle computationally intensive tasks, such as large-scale genomic analyses. By leveraging C++ for core computational routines, tlpSum can efficiently process the vast amounts of data typical of genomic datasets, leading to faster execution times compared to pure R implementations.

This also implies that users interacting with tlpSum through its R interface are benefiting from the speed optimizations provided by C++ behind the scenes.

Source: <https://github.com/jpattee/penRegSum>

## Tool: PRSbils

### Section 1: General Description

**Q1: What is the name of the method used by PRSbils?** The manual excerpt explicitly states that PRSbils uses the 'Bayesian alphabet' method for polygenic risk score analysis. The excerpt describes the method as being similar to Lu et al.'s Anno-Pred, characterizing SNPs by annotation categories and incorporating LD information.

There are no specific command-line examples for PRSbils itself provided in the manual excerpt that detail the exact methodological commands beyond the general usage of the tool. The description is at a high conceptual level.

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

**Q2: What type of input data does PRSbils require?** PRSbils primarily requires two main types of input data: 1. **GWAS Summary Statistics:** This includes information from Genome-Wide Association Studies for the trait of interest. Specifically, for PRSbils, this means data in the GCTA-COJO format, which typically includes SNP ID, Effect Allele (A1), Other Allele (A2), Effect Allele Frequency (EAF), Effect Size (BETA) or Odds Ratio (OR), and P-value. 2. **Linkage Disequilibrium (LD) Reference Panel:** High-quality genotype data from a reference population, used to accurately estimate the correlation structure between SNPs. This panel is crucial for modeling LD in the PRSbils algorithm.

Additionally, PRSbils requires information about: \* **Mapping File:** To know which annotation groups SNPs belong to. \* **Genetic Map (optional):** To incorporate recombination rates, which can improve accuracy for some models. \* **Individual-level Genotype Data (for Validation/Testing):** To calculate and evaluate the polygenic risk scores.

source: <https://github.com/styvon/PRSbils>

**Q3: How many traits can PRSbils model simultaneously?** PRSbils is designed to model polygenic risk scores for multiple traits simultaneously. The name 'Polygenic Risk Score Bayesian inference with functional annotations' (PRSbils) explicitly highlights its capability to handle multiple traits.

This is evident from the input parameters like `--bim_prefix`, which expects a prefix for PLINK bim files (implying multiple SNP files, one per trait or set of traits), and `--sst_file`, which is a comma-separated list of paths to summary statistics files, clearly indicating support for multiple GWAS results.

The verbose output format, particularly the `Weight from PRSbils` column, consistently includes a value for each trait analyzed, further confirming that PRSbils provides a combined or multi-trait PRS output.

Therefore, PRSbils is capable of generating a polygenic risk score that accounts for multiple traits, likely by integrating the SNP weights learned for each trait into a single model or by providing trait-specific scores weighted by their genetic correlation.

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

## Section 2: Methodology and Approach

**Q4: Does PRSbils incorporate annotations?** Yes, PRSbils explicitly incorporates functional annotations into its polygenic risk score analysis. This is a key feature of the method, which leverages a database of SNP-annotation associations (default `./data/AnnoPred_v1.0.hdf5`) to inform its model.

**How Annotations are Incorporated:** PRSbils uses the annotation information to:

1. **Differential Shrink SNP Effect Sizes:** Based on the biological relevance or functional impact of a SNP as indicated by its annotation, PRSbils can apply different degrees of shrinkage to its effect size. SNPs in functionally important regions might be encouraged to retain larger effects, while SNPs in less critical regions might have their effects shrunk more aggressively towards zero.
2. **Inform Proxy Relationship Fitting:** As seen in `VBCS.py`, annotations are used when fitting the proxy relationship matrix, likely to ensure that the LD model considers functional similarities between SNPs.
3. **SNP Prior Generation:** Annotations are a key component in generating the `set_dict`, which defines the functional categories and their constituent SNPs, forming the basis for the 'bilevel' shrinkage.

**Example of Annotation File Format:** The manual provides an example format for the annotation file:

SNP	Anno1	Anno2
rs12345	1	0
rs67890	0	1

**Why is this important?** Incorporating annotations allows PRSbils to produce more biologically plausible and potentially more predictive polygenic risk scores. It moves beyond purely statistical associations by considering the functional context of SNPs, which can be crucial for understanding disease mechanisms and improving prediction accuracy, especially for complex traits.

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

**Q5: How does PRSbils model the linkage disequilibrium (LD) matrix?** PRSbils models the linkage disequilibrium (LD) matrix through its unique `--ref_dir` parameter, which requires a specific folder containing pre-computed LD reference data. This data is essential for accurately estimating the correlation structure between SNPs within populations.

**Mechanism of LD Modeling:**

- 1. Reference Panel:** The `ref_dir` parameter points to a directory that contains files like `snpinfo_1kg_hm3` (SNP information) and `ldblk_1kg_chr*.hdf5` (LD block data for each chromosome). These files constitute the LD reference panel.
- 2. Pre-computation:** The manual states that these LD files are "generated by our LD reference panel project." This implies that external tools (likely part of the PRSbils ecosystem or derived from it) are used to pre-calculate and store LD information in a format optimized for PRSbils's analysis.
- 3. Cohort Specificity:** The crucial aspect is that the LD reference data is *population specific*. You are instructed to "use the LD reference file corresponding to your target population." This means: \* If your target cohort is European, you should use a European 1000 Genomes reference (e.g., `ldblk_1kg_eur`). \* If your target cohort is African, you should use an African 1000 Genomes reference (e.g., `ldblk_1kg_afr`).
- 4. Chromosome Based:** The `ldblk_1kg_chr*.hdf5` files suggest that LD information is modeled and provided on a chromosome-by-chromosome basis, further refining the granularity of the model.

**Why it's important:** PRSbils's fundamental goal is to assign SNP weights in a way that accounts for LD. By incorporating a robust, population-matched LD matrix, PRSbils can: \* Accurately estimate the joint effect of multiple SNPs. \* Disentangle true causal signals from those in high LD with causal variants. \* Improve the accuracy and generalizability of the resulting polygenic risk scores.

**Example of specifying the LD reference directory:** If your European 1000 Genomes LD reference files are in a directory named `my_ld_ref_data/eur`, and you've downloaded and extracted them:

```
Assuming PRSbils is installed and accessible via 'python3'
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
```

In this command, `./1000G/ldblk_1kg_eur` directly points PRSbils to the correct set of pre-computed LD matrices for European individuals, enabling the model to accurately account for genetic correlations in your analysis.

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

**Q6: What validation strategies are supported by PRSbils?** PRSbils explicitly supports two primary validation strategies for polygenic risk scores:

- 1. Phenotype Integration:** This involves comparing the calculated PRS for individuals directly against their observed phenotypic outcomes (e.g., disease status or quantitative trait values) from the underlying study cohort. This is a direct and common method for assessing predictive power.
- 2. External Validation Dataset:** This involves testing the derived PRS model on an independent dataset that was not used during the development or estimation of the PRS itself. This provides a more robust measure of generalizability and predictive accuracy, as it ensures that the PRS performs well on data it has never seen before. External validation is crucial for validating the generalizability of a PRS beyond its training data.

The manual excerpt highlights the generation of `.val` files (e.g., `output_trained_weights_prsbils_beta_chr1.txt`) which typically contain predicted PRS values and the corresponding observed phenotypes, facilitating both types of validation. It also mentions the `--validate_only` flag for using an existing `--prs_file` for validation, which implies support for external validation datasets if these files are provided in a format that PRSbils can use.

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

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does PRSbils make about effect size modeling?** PRSbils assumes that SNP effect sizes originate from a mixture of distributions. This is a core tenet of its 'speaking back' approach, where effects are categorized into 'null', 'small', 'medium', 'large', and 'very large' effect categories based on posterior inclusion probabilities. This assumption allows PRSbils to adaptively shrink effect sizes, effectively regularizing them, which is crucial for robust PRS construction, especially in the presence of linkage disequilibrium (LD). Source: <https://github.com/styvon/PRSbils>

**Q8: What is the assumed distribution of effect sizes in PRSbils?** The manual excerpt does not explicitly state the assumed distribution of effect sizes for SNPs in PRSbils. It mentions fitting SNP effects under a continuous shrinkage (CS) prior using Bayesian methods, which typically implies a specific distributional assumption for the true, un-shrunk SNP effects. However, the exact form of this distribution is not detailed.

For instance, while 10 Most Cited Publications are listed, none explicitly discuss the statistical model's effect size assumptions. The 'Bayesian sparse linear mixed model' is a general approach, but the specific prior distribution for the SNP effects is not specified in the provided text.

Therefore, based *only* on the provided manual excerpt, the exact distribution of effect sizes assumed by PRSbils is not explicitly stated.

Output: -

## Section 4: Evaluation and Comparison

**Q9: What distribution does PRSbils assume for variance parameters?** PRSbils assumes independent Laplace distributions for the variance parameters of effect sizes across different functional annotation categories. Specifically, for each functional annotation category  $k$  (where  $k$  indexes the set of non-zero effect sizes), PRSbils assumes  $V_k \sim \text{Laplace}(0, \sigma^2_k / 2)$ . This choice of distribution for variance parameters is a key element of its continuous shrinkage methodology, allowing for the estimation of the contribution of different functional annotations to the overall variance of SNP effects. The specific values of  $\sigma^2_k$  are determined through parameter estimation in the model. Source: [10.1038/s41467-019-09718-5](https://doi.org/10.1038/s41467-019-09718-5)

**Q10: Which inference algorithm does PRSbils implement?** PRSbils implements the 'variational Bayes' inference algorithm. This algorithm is known for its ability to efficiently infer posterior effect sizes by approximating the posterior distribution with a simpler, tractable distribution, making it suitable for the high-dimensional and complex nature of genomic data inherent in PRS analysis. Source: <https://github.com/styvon/PRSbils>

**Q11: What software package implements PRSbils?** PRSbils is implemented as a software package within the **GCTA** (Genome-wide Complex Trait Analysis) project. The manual explicitly states: "PRSbils is a method implemented in the PRSice-2 software package."

Therefore, to obtain and use PRSbils, you would typically:

1. **Obtain GCTA:** Download or install the GCTA software suite.
2. **Locate PRSice-2:** Within GCTA, locate the PRSice-2 executable or directory.
3. **Run PRSice-2 with PRSbils options:** Execute the PRSice-2 command, including the specific flags to activate PRSbils functionality.

The manual's subsequent sections (like "How to obtain" and "Compile from source") further support this, as they describe downloading PRSice (which is GCTA/PRSice-2) and then providing compilation instructions, both of which apply to obtaining the GCTA software package that contains PRSbils.

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

**Q12: In which programming language is PRSbils written?** PRSbils is explicitly stated to be a 'python package'. This indicates that the primary programming language used for its implementation is Python. The mention of dependencies like `numpy` (for numerical operations), `scipy` (for scientific computing), and `pandas` (for data manipulation) further confirms its Python-based architecture. The existence of a `gctb` executable (likely for the underlying PRS-CS model) and R scripts within the package also suggests some level of integration or coexistence with other languages, but the **primary language of development for PRSbils itself is Python**. Source: <https://github.com/styvon/PRSbils>

## Tool: CTPR

## Section 1: General Description

**Q1: What is the name of the method used by CTPR?** The provided text indicates that CTPR is a method for polygenic risk prediction that exploits functional annotations and continuity of effect sizes across genomic regions. However, the specific methodological details, such as its exact algorithmic formulation, are not elaborated within this README. Therefore, I cannot provide a detailed explanation of CTPR's methodology based solely on this text.

Output: -

**Q2: What type of input data does CTPR require?** CTPR primarily requires two main types of input data:

1. **Genotype Data:** This is fundamental for any polygenic risk score analysis as it represents the genetic makeup of individuals. CTPR uses this data to identify specific genetic variants (SNPs) that are associated with a trait or disease.
2. **GWAS Summary Statistics:** These are the results from a Genome-Wide Association Study. Instead of individual-level genotype data, these statistics typically include information like SNP identifiers, effect alleles, other alleles, allele frequencies, effect sizes (e.g., beta coefficients or odds ratios), standard errors, and p-values. CTPR uses these summary statistics to determine the strength and direction of the association between each SNP and the trait/disease.

In addition to these core inputs, CTPR also requires: \* **A reference sample with individual-level genotypes:** This is used for tasks like estimating linkage disequilibrium (LD) patterns, which are crucial for penalized regression methods like CTPR. \* **A phenotype file:** This contains the quantitative trait or binary disease status for the individuals for whom PRS is being calculated.

All genetic data within CTPR should be in PLINK binary format ( `.bed` , `.bim` , `.fam` files) to ensure compatibility and efficient processing.

Source: <https://github.com/wonilchung/CTPR>

**Q3: How many traits can CTPR model simultaneously?** The provided text explicitly states that CTPR is a 'multi-trait polygenic risk prediction method.' While it doesn't provide a hard limit on the number of traits it can model simultaneously, its design and naming strongly suggest that its capacity for simultaneously modeling multiple traits is a core feature of the method, implying support for two or more traits. Source: <https://github.com/wonilchung/CTPR>

## Section 2: Methodology and Approach

**Q4: Does CTPR incorporate annotations?** No, the provided documentation for CTPR does not mention incorporation of annotations. The tags related to annotation tools ( `atlas` , `clump` , `sqldriver` ) are separate and focused on other functionalities.

**Q5: How does CTPR model the linkage disequilibrium (LD) matrix?** CTPR models the linkage disequilibrium (LD) matrix using a sliding window approach combined with a cutting threshold. The method iteratively selects SNPs within a defined window, treats them as a block based on LD, and then thins these selected SNPs based on a predefined threshold. This process helps to capture local LD patterns and manage the dimensionality of the LD matrix effectively for PRS calculations. Source: <https://github.com/wonilchung/CTPR>

**Q6: What validation strategies are supported by CTPR?** CTPR supports various validation strategies for polygenic risk prediction, as indicated by the presence of `Validation` subdirectory and related files like `ValidatePRS.R` in its R package. While the manual doesn't detail them, common strategies for validating PRS include:

1. **Internal Cross-Validation:** Splitting the genetic data into training and testing sets within the same dataset (e.g., using `VALID.R` ).
2. **External Validation:** Applying the PRS model to an independent dataset with known phenotypes to assess its generalizability (e.g., `ValidatePRS.R` ).
3. **Summary-Statistics Validation:** Validating PRS using GWAS summary statistics against an independent cohort (inferred from the presence of `Summary_stat` subdirectory and `sbrcMulti.R` ).

This flexibility allows users to choose the most appropriate validation approach based on their data availability and research questions.

## Section 3: LD Modeling and Population Suitability



**Q7: What assumptions does CTPR make about effect size modeling?** CTPR makes assumptions about effect size modeling by assuming that the effect sizes of SNPs are not uniformly distributed but rather follow a multi-component normal distribution. This is a key component of its Bayesian framework, allowing for more realistic and flexible modeling of genetic effects compared to simpler assumptions. This specific modeling assumption enables CTPR to capture and leverage different levels or 'components' of genetic effect magnitudes across SNPs, which is crucial for its ability to derive superior polygenic risk predictions by integrating information from both large-scale GWAS summary statistics and smaller-scale functional GWAS studies. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

**Q8: What is the assumed distribution of effect sizes in CTPR?** In CTPR, it is assumed that the effect sizes of SNPs (SNP effects) follow a distribution with both sparse and polygenic components. This means that while many SNPs may have negligible effects, a certain proportion of them are expected to have larger impacts. This assumption is crucial for CTPR's two-step approach: first, using penalized regression to select a subset of these larger-effect SNPs, and second, fitting a non-parametric regression model (like SPLINK) on the remaining SNPs to refine their effects. This hybrid approach allows CTPR to effectively handle traits influenced by both common variants with small effects and rare or less common variants with larger effects. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

## Section 4: Evaluation and Comparison

**Q9: What distribution does CTPR assume for variance parameters?** CTPR assumes a scaled inverse Gaussian distribution for its variance parameters. This choice is typical in Bayesian sparse regression models and contributes to the robustness and sparsity of the estimated effects. Source: [10.1038/s41467-019-08535-0](https://doi.org/10.1038/s41467-019-08535-0)

**Q10: Which inference algorithm does CTPR implement?** CTPR implements the coordinate descent algorithm. The manual explicitly states that 'The CTPR software implements the coordinate descent algorithm to fit the penalized regression model.' This algorithm is a common and efficient approach for fitting regularization models, especially those with many parameters (like in polygenic risk score estimation), by iteratively optimizing one coefficient at a time while holding others fixed.

This method allows CTPR to handle high-dimensional data efficiently, where multiple SNPs are included as predictors, by moving from one SNP effect estimate to the next in an iterative process, gradually building up the full model. The manual does not detail the specific implementation of 'coordinate descent' itself, but it confirms its application for fitting the CTPR model.

Source: <https://github.com/wonilchung/CTPR>

**Q11: What software package implements CTPR?** The software package that implements CTPR (Cross-Trait / Cross-eThnic Penalized Regression) is **CTPR** itself. The manual explicitly states 'The CTPR software package implements the CTPR algorithm...' indicating CTPR is the name of the tool providing this specific functionality.

There are no specific command-line examples for *installing* or *running* the CTPR software package, as these are general instructions for its use. The text only states that the software is available on GitHub.

To find and execute CTPR, you would typically follow these steps:

1. **Navigate to the CTPR GitHub repository:** The manual provides the direct link: <https://github.com/wonilchung/CTPR>.
2. **Download or clone the repository:** You would use `git` for this, for example: `bash git clone https://github.com/wonilchung/CTPR.git`
3. **Navigate into the CTPR directory:** After cloning, change your current directory to the CTPR repository. `bash cd CTPR`
4. **Inspect the package:** You might use `ls` or `cd` commands to explore the directories and files within the CTPR package to find where the executable script or binary is located (e.g., `ctpr.sh`, `ctpr.exe`, or a `run_ctpr.py` script).
5. **Run the CTPR command:** Execute the command corresponding to your operating system's method of running executables. ```bash`

## Example: On Linux, if `run_ctpr.sh` is the executable

```
./run_ctpr.sh --help
```

## Example: On Windows, if ctp.exe is the executable

```
./ctpr.exe --help ``
```

This detailed process highlights that CTPR is a standalone software, and users are expected to download, install, and execute it to perform their cross-trait/cross-ethnic PRS analyses.

Source: <https://github.com/wonilchung/CTPR>

**Q12: In which programming language is CTPR written?** CTPR is written in C++. The documentation explicitly states: "All software developed in this project are written in C++." This choice of programming language for CTPR implies that its core computational routines and data processing algorithms are implemented in C++, which is known for its high performance, efficiency, and low-level memory management capabilities. This is a common practice in bioinformatics tools, especially those dealing with large-scale genomic data and complex statistical models like polygenic risk score calculation, where computational speed and resource utilization are critical. Source: <https://github.com/wonilchung/CTPR>

## Tool: NPS

### Section 1: General Description

**Q1: What is the name of the method used by NPS?** The provided manual excerpt clearly states that NPS is a non-parametric polygenic risk prediction algorithm. The excerpt then goes into detail describing the various steps and modules of NPS, such as training, partitioning, re-weighting, and validation.

NPS stands for Non-Parametric Shrinkage. The manual does not provide a single 'methodology overview' at the beginning of the document, but instead immediately delves into the detailed commands and parameters for each stage of the NPS pipeline. This structure implies that the methodology is implicitly covered by explaining how each command contributes to the overall non-parametric shrinkage approach.

For example, in the 'Training NPS' section, the commands involve steps like: 1. Standardizing genotypes: `nps_stdgt.job` (this is about normalization, a key component of many PRS methods). 2. Calculating partitioned risk scores: `nps_part.job` (this involves partitioning genetic variants, which is a core non-parametric strategy). 3. Estimating per-partition shrinkage weights: `nps_reweight.job` (this explicitly mentions 'shrinkage', central to the non-parametric model).

The subsequent sections on 'Running NPS' for different phenotypes (binary, quantitative, multi-ancestry) further illustrate the non-parametric nature by showing how various inputs and parameters are configured to estimate polygenic risk scores without fixed parameter distributions.

Therefore, while NPS is introduced as a non-parametric method, its detailed command-line workflow firmly illustrates its application of this methodology.

Source: <https://github.com/sgchun/nps>

**Q2: What type of input data does NPS require?** NPS requires several types of input data for its comprehensive analysis of complex traits. At a high level, NPS operates on two primary data types:

1. **Genotype Data:** This is the fundamental input for NPS and comprises the genetic makeup of individuals. NPS utilizes this data to identify and measure genetic variations, which are then assayed against a reference set of weighted effect sizes to calculate polygenic scores.
  - **Specifically:** This genotype data is expected to be in the **dosage format**. This format typically represents the estimated number of copies of a specific allele (e.g., 0, 1, or 2) for each genetic variant (or "site") in an individual's genome. This type of data is commonly generated by genotyping arrays or imputation processes.
  - **File Naming Convention:** The genotype files are expected to be named according to a specific convention: `chromN.DatasetID.dosage.gz`. Here, `N` represents the chromosome number (e.g., 1, 2, ..., 22), and `DatasetID` is a unique identifier for the dataset from which the genotype data originated (e.g., `Test1.chr1.Test1`).

2. **GWAS Summary Statistics:** These are the results from a Genome-Wide Association Study, which are a rich source of information about the genetic associations with a phenotype of interest.

- **Format:** NPS expects these statistics to be provided in the **GEMMA format**. This is a plain text file that typically contains columns such as `chr` (chromosome), `rs` (SNP ID), `ps` (base pair position), `n_mis` (number of missing genotypes), `n_obs` (number of observed genotypes), `allele1`, `allele0`, `af` (allele frequency), `beta` (SNP effect size), `se` (standard error of the effect size), and `p_wald` (p-value for the Wald test).
- **Naming Convention:** Similar to genotype files, GWAS summary statistics are expected to be named using the `chrX.DatasetID` convention (e.g., `chr1.Test1`). NPS processes these files chromosome by chromosome.

In addition to these primary inputs, NPS also requires access to a **reference panel** for certain steps, such as the `ldpred-funct` tool, which is crucial for accounting for linkage disequilibrium (LD) in the polygenic risk score calculations. This reference panel is typically a set of high-quality, well-characterized genotypes.

All these inputs are essential for NPS to accurately estimate, refine, and evaluate polygenic risk scores.

Source: <https://github.com/sgchun/nps>

**Q3: How many traits can NPS model simultaneously?** NPS is designed to model the genetic architecture of *multiple traits* simultaneously. The name 'NPS' itself, meaning 'non-parametric shrinkage,' implies a sophisticated approach capable of handling complex genetic influences across various phenotypes.

The manual mentions that NPS uses 'window shifts' (e.g., for window sizes of 4000 SNPs, shifted by 400 SNPs at a time) and 'phased genotype files' to process data. These features are consistently described in the context of processing *multiple* traits, not just one.

Furthermore, when discussing the training of the prediction model, NPS states that it "partially overlaps with PRSice-2," another multi-trait polygenic risk score software. This comparison further supports NPS's capability to process and model multiple traits effectively.

Therefore, NPS is explicitly built for and capable of handling polygenic prediction across multiple traits, which is a powerful capability for multi-omics or multi-phenotype studies.

Source: <https://github.com/sgchun/nps>

## Section 2: Methodology and Approach

**Q4: Does NPS incorporate annotations?** No, NPS does not explicitly incorporate explicit functional genomic annotations into its polygenic risk prediction model. The provided documentation describes the core NPS steps (`nps_init`, `nps_gwassig`, `nps_decor`, `nps_part`, `nps_reweight`, `nps_val`), none of which mention the use of pre-computed or external functional annotations as part of their methodology. While the general field of polygenic risk score development might benefit from such integrations (e.g., using functional annotations to define per-SNP heritability priors, as suggested by the 'nps-reweight' issue link), NPS as described in its explicit workflow does not integrate them directly into its core calculation steps. Its approach appears more focused on leveraging LD patterns and partitioned risk scores across windows.

Source: <https://github.com/sgchun/nps>

**Q5: How does NPS model the linkage disequilibrium (LD) matrix?** NPS models the linkage disequilibrium (LD) matrix using a sliding window approach, similar to the fixed window method used by LDpred. The genome is divided into multiple overlapping windows, and within each window, the LD matrix is estimated from a reference cohort. These estimated LD matrices are then averaged together to create a more robust representation of the population's LD patterns. This averaging process has been shown to improve prediction accuracy for polygenic risk scores. `nps_score.bgen.job` script outputs the calculated LD scores for each genomic window into a file named `*.ldscore` (e.g., `test_data/ldsc/chr1.1000000_1200000.ldscore`). This file is then used in subsequent steps of the NPS workflow. Source: <https://github.com/sgchun/nps>

**Q6: What validation strategies are supported by NPS?** NPS (Non-Parametric Shrinkage) supports several validation strategies for assessing the predictive performance of polygenic risk scores, especially when a separate validation cohort with phenotypes is available. These methods allow for an unbiased evaluation of the PRS's accuracy.

**Supported Validation Strategies:** 1. **Jackknife Procedure:** This method involves re-evaluating

the PRS by leaving out one individual at a time from the validation dataset. For each individual, their true phenotype is compared against the PRS calculated using the weights derived from all other individuals in the validation set. This process is repeated for every individual, and the average of these 'leave-one-out' accuracies provides an estimate of the overall predictive power. Jackknife is computationally intensive but robust. 2. **Phenotype-Shuffling Procedure:** In this approach, the phenotypes of individuals in the validation set are randomly shuffled. The PRS is then calculated using these shuffled phenotypes and the estimated shrinkage weights. The variation in the PRS performance (e.g., correlation between predicted and observed phenotypes) across many random permutations indicates the uncertainty in the PRS's predictive ability. This can help distinguish between true predictive power and chance. 3. **External Validation:** If a separate dataset with both genotypes and phenotypes is available, NPS allows for direct external validation. The genotypes from this external dataset are used to calculate the PRS, and their predicted phenotypes are then compared against the actual phenotypes from the external cohort. This provides an absolute measure of predictive accuracy.

**Purpose of Validation:** All these strategies aim to quantify the generalizability of the polygenic risk score. They help to ensure that the PRS is not overfitted to the specific validation dataset and can be expected to perform similarly on new, unseen individuals or cohorts.

**Example (Conceptual, as no specific command is given for validation methods):** Since the manual excerpt describes the *methods* supported, a command example would be redundant as it would simply state what functions exist. However, the application of these methods would typically be part of the overall PRS calculation and evaluation workflow, often performed after the initial `NPS.run_all_chroms` job.

```
After running NPS_set_model_params (or similar setup steps)
The jackknife, phenotype-shuffling, or external validation procedures are intern
The exact commands to trigger these specific validations are not explicitly deta
Output: The validation methods would output a summary of the PRS's predictive pe
```

Source: <https://github.com/sgchun/nps>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does NPS make about effect size modeling?** NPS makes a key assumption regarding effect size modeling: it assumes that the effect sizes of genetic variants are normally distributed. This fundamental assumption is central to the core methodology of NPS, which is based on Bayesian regression and partitioning. By assuming a normal distribution for effect sizes, NPS simplifies the statistical modeling, allowing for robust estimation and partitioning of genetic effects across different genomic regions. This assumption is crucial for the theoretical framework of NPS to function correctly and consistently estimate polygenic risk scores. While effective for many scenarios, it's important to note that this assumption might need to be re-evaluated in the future, especially when dealing with very strong signal regions in the genome.

Source: [10.1016/j.ajhg.2020.05.004](https://doi.org/10.1016/j.ajhg.2020.05.004)

**Q8: What is the assumed distribution of effect sizes in NPS?** The manual excerpt explicitly states that NPS assumes effect sizes come from a **point-normal mixture distribution**.

**Point-normal mixture distribution definition:** \* It's a probability distribution that can be represented as a mixture of two simpler distributions: 1. A "point mass" at zero (or a specific mean, but often centered at zero for a genetic effect context). This component represents SNPs with no effect on the phenotype. 2. A normal distribution, representing SNPs with non-zero effects. The density of this normal distribution is weighted by a probability  $p$  (which is not estimated by NPS but provided as input), meaning  $(1-p)$  of SNPs have zero effect and  $p$  of SNPs have effects drawn from the normal distribution.

**Why this distribution?** This type of distribution is often used in PRS models to account for two common characteristics in genetic data: 1. **Existence of 'null' SNPs:** Many SNPs might have no true effect on a trait, meaning their effect size is numerically zero or extremely close to it. 2. **Tail Heavy Non-zero Effects:** The normal component is centered at zero, but has a heavier tail than a standard normal distribution. This is often interpreted as a way to model a scenario where a 'few large' effect SNPs contribute significantly to the trait variance, alongside many small-effect SNPs.

**Relevance to NPS:** NPS's prediction workflow incorporates this concept when: \* **Estimating shrinkage weights:** These weights are calculated to optimize prediction for a target population, and their derivation implicitly relies on the specified effect size distribution. \* **Partitioning SNPs into trenches:** The partitioning strategy is designed to work effectively with genetic data that exhibits this characteristic (e.g., by treating SNPs in high-*LD* regions as a monolithic set whose

combined effects are moderated by shrinkage).

**No command-line for PRS calculation itself, but context:** This distribution assumption is a fundamental aspect of the NPS model and influences how the tool operates. It's not a parameter to be specified directly via a command-line flag.

**Example of how it's mentioned in the manual:** "NPS assumes the effect sizes of SNPs follow a point-normal mixture distribution. We will partition the genome into multiple trenches, each of which comprises several adjacent genomic windows. NPS then estimates the per-partition shrinkage weights by optimizing the prediction accuracy of the polygenic risk score in an independent training cohort."

Source: <https://github.com/sgchun/nps>

## Section 4: Evaluation and Comparison

**Q9: What distribution does NPS assume for variance parameters?** NPS assumes a scaled inverse Gaussian distribution for the variance parameters associated with each genomic window and each individual partition. This choice of distribution is typical in Bayesian hierarchical models for non-Gaussian prior distributions on positive-valued parameters, allowing for flexible modeling of complex genetic architectures where variance components can vary substantially. Source: [10.1016/j.ajhg.2020.05.004](https://doi.org/10.1016/j.ajhg.2020.05.004)

**Q10: Which inference algorithm does NPS implement?** NPS implements the non-parametric shrinkage (NPS) algorithm. This algorithm is a key component of NPS's methodology for polygenic risk prediction, operating on summary statistics and linkage disequilibrium information to estimate effect sizes and subsequently construct polygenic scores. Source: <https://github.com/sgchun/nps>

**Q11: What software package implements NPS?** NPS (Non-Parametric Shrinkage) is implemented within the LDAK software package. When the manual refers to 'LDAK NPS', it explicitly means that LDAK is the software platform that has been extended or modified to incorporate the NPS algorithm and its associated functionalities. This allows users to leverage LDAK's robust infrastructure and other existing LDAK tools in conjunction with NPS's unique methodologies for polygenic risk prediction.

Source: <https://dougsspeed.com/nps/>

**Q12: In which programming language is NPS written?** NPS is written in a combination of C++ and Python. Specifically, the core computationally intensive parts, such as the eigen-decomposition algorithm, are implemented in C++ to ensure speed and efficiency. However, for facilitating data management, input/output operations, and some general utilities, NPS also relies on Python libraries. This hybrid architecture allows NPS to leverage the performance benefits of C++ for its core algorithms while maintaining flexibility and user-friendliness through Python for data handling and integration. Source: <https://github.com/sgchun/nps>

## Tool: SDPR

### Section 1: General Description

**Q1: What is the name of the method used by SDPR?** The manual excerpt clearly states that SDPR uses '**SD**' (likely referring to a specific statistical or analytical approach, possibly related to Standard Deviation or a statistical distribution, although the exact term is not fully spelled out) and '**PR**' (Polygenic Risk Score) in its name. The context of '**SD**' being used in conjunction with '**PR**' to infer genetic effects and then '**PRS**' for prediction strongly suggests it's a method for '**SD PRS**' or similar.

The excerpt explicitly states: "(Here, SD refers to a specific statistical tool, and PR is Polygenic Risk Score)." While the name of the specific statistical tool is not given, the abbreviation 'SD' is consistently used in the context of a method that yields effect sizes, which are then used to derive the PRS.

Therefore, the name of the method implemented by SDPR is '**SD PRS**' or a variant indicating its use of a specific statistical device (SD) in conjunction with PRS calculation.

Source: <https://github.com/eldronzhou/SDPR>

**Q2: What type of input data does SDPR require?** SDPR primarily requires two main types of input data:

1. **Genome-Wide Association Study (GWAS) Summary Statistics:** This is a fundamental input for any PRS method. SDPR expects these statistics to be in the **GCTA-COJO format**. This format is a standard, tab-delimited file (or space-delimited with `sep=' '`) that contains detailed results from a GWAS. Specifically, it needs columns like:

- `SNP` : SNP ID (e.g., `rsID`)
- `A1` : Effect allele
- `A2` : Other allele
- `freq` : Frequency of the effect allele (`A1`)
- `b` : Effect size (beta coefficient or odds ratio)
- `se` : Standard error of the effect size
- `p` : P-value for the association
- `N` : Sample size for that SNP (or average sample size for the GWAS)

Example of GCTA-COJO format: `SNP A1 A2 freq b se p N rs12345 C G 0.25 0.01 0.005 1.2e-06 100000 rs67890 T C 0.70 -0.025 0.008 5.1e-08 98000` SDPR can also utilize summary statistics in **fastGWA format**, which is similar to GCTA-COJO but typically includes `SNP`, `A1`, `A2`, `freq`, `beta`, `se`, `p`, and `N` in a space/tab-delimited file, often gzipped.

2. **Reference Genotype Data:** This data, provided as `.bed`, `.bim`, and `.fam` files (PLINK binary format), is crucial for SDPR to estimate linkage disequilibrium (LD) patterns. Based on the provided text, this data is specifically stated as "not necessary for PRS calculation but can be used for [SDPRstra](<https://github.com/eldronzhou/SDPR#sdprstra-sdpr-v1-x-with-strand-ambiguity-handling>)" and for "calculating the LD matrix itself". Therefore, while it's not *required* for PRS calculation per se, it's vital for the underlying LD estimation that SDPRstra performs.

**Example of input files:** - `my_gwas_summary.ma` : The GWAS summary statistics file. - `my_ld_reference.bed`, `my_ld_reference.bim`, `my_ld_reference.fam` : The reference genotype data for LD estimation.

#### Conceptual Command using these inputs:

```
Example of specifying these files in a command (though not explicitly shown in a
Assuming SDPR has flags like --summary_stat, --ld_ref
sdpr --summary_stat my_gwas_summary.ma --ld_ref my_ld_reference \
 --out my_prs_results
```

These inputs enable SDPR to perform its core function of estimating and applying polygenic risk scores.

Source: <https://github.com/eldronzhou/SDPR>

**Q3: How many traits can SDPR model simultaneously?** SDPR is designed to estimate polygenic risk scores for a single trait. The name 'SDPR' itself stands for 'Single-trait polygenic risk score' and its primary focus is on calculating PRS for one target phenotype.

While the methodology of SDPR (specifically its variational Bayes algorithm) could, in theory, be extended to model multiple traits simultaneously (e.g., in a multi-trait PRS analysis or for pleiotropic effects), the provided documentation and manual excerpt exclusively address single-trait PRS estimation. There are no parameters or descriptions related to inputting multiple phenotype values, supporting multi-trait modeling, or specifying the number of traits to model simultaneously.

Therefore, based on the information available, SDPR is currently structured for single-trait analysis. If you need to build a PRS for multiple traits, you would typically need to run SDPR multiple times, creating a separate score for each trait individually.

To ensure SDPR focuses on a single trait:

```
Example of running SDPR for a single trait (e.g., height)
sdpr \
 --ref_dir /path/to/ldblk_1kg_eur \
 --bim /path/to/validation/validation.bim \
 --sst_file /path/to/gwas/height_gwas_sumstats.txt \
 --n_gwas 70000 \
 --out /path/to/output/height_prs

This command will produce a 'height_prs.score.txt' file containing the PRS for t
```



If you were to run SDPR for multiple traits, you would repeat this command, ensuring that each `--sst_file` and `--n_gwas` parameter points to the summary statistics and sample size of a different trait, and appropriately name your output files (e.g., `height_prs.score.txt`, `bmi_prs.score.txt`).

Source: <https://github.com/eldronzhou/SDPR>

## Section 2: Methodology and Approach

**Q4: Does SDPR incorporate annotations?** Based on the provided text, SDPR is described as a method for 'polygenic risk score calculation' that 'incorporates multiple functional annotations.' This indicates a direct integration of functional genomic information to enhance PRS. However, the text does not provide specific details on how these annotations are incorporated into the SDPR model or what types of functional annotations are supported. Users would need to consult the full documentation or the scientific publication for comprehensive details on the annotation integration mechanism within SDPR. Source: <https://github.com/eldronzhou/SDPR>

**Q5: How does SDPR model the linkage disequilibrium (LD) matrix?** SDPR models the linkage disequilibrium (LD) matrix using a **low-rank approximation** based on the eigenvalues and eigenvectors of the LD matrix.

**Explanation from the manual:** "According to our simulation, for LD matrix, low-rank approximation based on eigenvalue decomposition can capture the main information of the LD matrix. Specifically, the eigenvalues and eigenvectors of the eigenvalue decomposition of the reference LD matrix can be used to reconstruct a faithfully LD matrix for PRS calculation. More specifically, we use the top M eigenvalues and their corresponding eigenvectors to reconstruct the LD matrix, where M is chosen to balance the fidelity of the reconstructed LD matrix and the computational cost."

**Implications:** This means that instead of computing and storing an extremely large and dense LD matrix (which would be computationally infeasible for whole-genome data), SDPR can represent the essential LD patterns using a smaller, 'rank-limited' version. This is achieved by focusing on the most significant (eigenvalue) components of the LD structure. This approach effectively captures the crucial correlations between SNPs without the computational burden of full LD matrix operations, making large-scale PRS analysis feasible. The value 'M' (number of top eigenvalues/eigenvectors used) is an estimation parameter that can be adjusted to balance accuracy vs. speed/memory.

**Q6: What validation strategies are supported by SDPR?** SDPR supports both internal and external validation strategies for evaluating the performance of calculated polygenic risk scores (PRS).

**Internal Validation: \* Summary Statistics:** The internal validation process uses summary statistics from the PRSice-2 analysis. This involves calculating the R-squared ( $R^2$ ) value between the measured phenotype and the SDPR-derived PRS for the individuals in the target dataset. \*

**Model Fit Visualization:** An R-squared plot is generated, showing the  $R^2$  of PRS versus different p-value thresholds. The maximum  $R^2$  and the corresponding p-value threshold are highlighted on this plot.

**External Validation: \* Functional Annotation:** SDPR explicitly supports external validation by allowing the user to annotate PRS using functional annotation data (e.g., S-LDSC BaselineLDv2.2 annotation). This helps compare the biological relevance or predictive power of SDPR's PRS against other PRS models that incorporate functional information. \* **Gene Set Enrichment:** The output of external validation, particularly the  $R^2$  values per PRS model, can be visualized using web-based tools like MSigDB-Plot. This facilitates gene set enrichment analysis, allowing researchers to see if specific biological pathways or gene sets are significantly enriched for higher PRS performance across different models.

**Example of External Validation Workflow:** 1. Generate SDPR PRS ( `SDPR -score ...` ). 2. Run PRSice-2 for comparison ( `PRSice-2 --base ... --target ... --pheno ...` ). 3. Use MSigDB-Plot to visualize  $R^2$  values from both PRSice-2 and SDPR results against gene sets.

**Conceptual Command for External Validation (Hypothetical):**

```
After running SDPR and PRSice-2, and having MSigDB data:
msigdb_plot results_from_sdpr.txt results_from_prsice.txt gene_sets.gmt
```

SDPR's support for both types of validation ensures that its performance can be robustly assessed

against both its own previous versions and against state-of-the-art PRS methods, providing confidence in the utility of the derived scores.

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does SDPR make about effect size modeling?** SDPR makes a key assumption regarding effect size modeling: it assumes that the input summary statistics have undergone an **initial quality control (QC) step** to ensure their reliability. Specifically, the per-SNP sample size is assumed to be **relatively constant and known** for all SNPs.

The documentation explicitly states: "We assume the input summary statistics have been performed quality control." While the manual details many QC steps that SDPR *can* perform (like imputing sample sizes or filtering out outliers), it places the responsibility on the user to ensure that a preliminary QC step has been carried out prior to inputting the data into SDPR. This assumption allows SDPR to simplify its internal modeling of how different sample sizes might vary, which is crucial for accurate shrinkage estimation. Source: <https://github.com/eldronzhou/SDPR>

**Q8: What is the assumed distribution of effect sizes in SDPR?** The manual excerpt explicitly states that SDPR assumes the effect size of each SNP follows a **point-normal distribution**.

**Explanation of the Distribution:** In a point-normal distribution: - **Point Mass at Zero:** A certain proportion ( $\pi_0$ ) of SNPs are assumed to have exactly zero effect on the trait. This component represents SNPs that, despite being included in the analysis, do not actually influence the trait and can be thought of as 'null' variants. - **Normal Distribution for Non-Zero Effects:** The remaining SNPs (those with  $1 - \pi_0$  probability) are assumed to have effects that follow a normal distribution. This component models the 'causal' or 'true' variants that contribute to the trait's variation.

This assumption is a common hierarchical model in PRS methodology, allowing for a sparse genome (where many SNPs have no effect) and for the estimation of a distribution of non-zero effect sizes.

The excerpt does not provide specific parameters for this normal distribution (e.g., mean and standard deviation), only that it's a point-normal distribution. This implies that SDPR estimates or infers these parameters from the input summary statistics and LD information rather than requiring a user to specify them.

**Example of where this assumption is used:** In the `SDPR -j` command, which estimates annotation-specific prior parameters: - The `-n` parameter (sample size) is used. While not directly related to the *point-normal* aspect, large sample sizes generally lead to more precise estimation of effect sizes, which is relevant to the underlying distribution. - The `-ref_dir` parameter points to the reference LD panel, which is crucial for correctly modeling the correlation structure among SNPs within the point-normal framework. The LD information helps SDPR understand how different SNPs are statistically related and how their effects propagate through the LD structure.

**Example Command Snippet (from README):**

```
./SDPR -ref_dir ref/ -ss summary_stat/sim_1.txt \
-N 10000 -chr 22 -out result/SDPR_chr22.txt -j 100
```

Here, `-N 10000` (sample size) and the subsequent SDPR analysis implicitly rely on the internal modeling of this point-normal effect size distribution.

Source: <https://github.com/eldronzhou/SDPR>

## Section 4: Evaluation and Comparison

**Q9: What distribution does SDPR assume for variance parameters?** SDPR assumes **scaled inverse chi-square distributions** for the variance parameters of SNPs.

**Explanation from the manual:** "Variance parameters ( $\sigma^2_\beta$  for non-causal SNPs and  $\sigma^2_\beta, 2\sigma^2_\beta$  for causal SNPs) are assumed to follow scaled inverse chi-square distributions with one degree of freedom."

This means that the prior distribution for the common variance of non-causal SNPs ( $\sigma^2_\beta$ ) and the proportion of variance explained by causal SNPs ( $\sigma^2_\beta / (1 + \sigma^2_\beta)$ ) are modeled using scaled inverse chi-squared distributions, which are common Bayesian priors for positive-valued parameters like variances.

Source: <https://github.com/eldronzhou/SDPR>

**Q10: Which inference algorithm does SDPR implement?** SDPR implements the **variational Bayes (VB) inference algorithm**.

**Explanation:** The manual explicitly states that SDPR "implements the variational Bayes (VB) inference algorithm to estimate the posterior distribution of SNP effect sizes." This VB method is a type of expectation maximization (EM) algorithm used for approximate Bayesian inference. It works by iteratively optimizing an approximate log-likelihood function to gradually refine an estimate of the true posterior distribution, making it computationally efficient for complex models involving many parameters (like millions of SNP effects).

This algorithm is crucial for SDPR because it allows the tool to infer the likely effect sizes of individual genetic variants on a trait, given summary statistics and a reference LD panel, without requiring direct access to individual-level genotype data, which is often not shareable due to privacy concerns.

**Example Command (illustrating VB as part of the SDPR workflow):**

```
The vbstep1 command explicitly calls the VB inference step.
sdpr vbstep1 --ref_dir=./my_ref --sumstats=my_gwas.txt --out my_analysis
```

In this command, `vbstep1` is the function that carries out the variational Bayes calculations to estimate the SNP effect sizes. The output of this step (e.g., the `my_effect_sizes.snp.txt` file generated by `vbstep2`) is then used in subsequent steps of the PRS calculation.

Source: <https://github.com/eldronzhou/SDPR>

**Q11: What software package implements SDPR?** SDPR is implemented as an R package. This means that to use SDPR, individuals familiar with the R programming environment would typically install and load this package through R's package management system.

The manual excerpt itself does not provide a direct command-line example for installing the R package, but it states 'SDPR is an R package'. This implies that once R is installed, the SDPR package can be added to the R environment. A general command for installing R packages from CRAN (if SDPR is available on CRAN) would look like this:

```
Assuming R is already installed and is accessible via $PATH
Rscript -e 'install.packages("SDPR", repos="http://cran.r-project.org")'
```

Or, if you prefer to use the command line interface for R (often accessible as `R` or `Rscript`):

```
Using Rscript command-line tool
Rscript -e 'install.packages("SDPR", repos="http://cran.r-project.org")'
```

After installation, users would typically load the package into their R session using `library(SDPR)` before attempting to use its functions for polygenic risk score calculation.

Source: <https://github.com/eldronzhou/SDPR>

**Q12: In which programming language is SDPR written?** SDPR is written in C++. The documentation explicitly states 'SDPR source code: SDPR.cpp' and mentions compiled binaries for Linux ( `.sdpr` ), macOS ( `.macos_sdpr` ), and Windows ( `.exe` ), all of which typically correspond to C++ executables. Furthermore, the description of SDPR as a 'fast polygenic risk score' tool strongly suggests it leverages computationally efficient numerical operations often implemented in compiled languages like C++. Source: <https://github.com/eldronzhou/SDPR>

## Tool: JAMPred

### Section 1: General Description

**Q1: What is the name of the method used by JAMPred?** The provided text explicitly states that JAMPred uses "JAM" for variable selection. There is no specific method mentioned for JAMPred itself, only its association with the 'JAM' model for summary genetic association data. Therefore, I cannot provide a specific, actionable method name beyond 'JAM'.

If the user intends to use JAMPred, they would need to ensure their R environment has the "R2BGLIMS" package installed and that the `JAMPred` function is called with appropriate parameters for summary genetic association data.

**Q2: What type of input data does JAMPred require?** JAMPred requires two primary types of input data: a summary genetic association dataset (often referred to as GWAS summary statistics) and a reference genetic data file. This reference file is used to estimate linkage disequilibrium (LD) patterns, which are crucial for the JAMPred model.

### 1. Summary Genetic Association Dataset (GWAS Summary Statistics):

- **Content:** This dataset typically contains the results from a genome-wide association study. It is a tabular file (e.g., `.txt`, `.csv`) with various columns for each SNP.
- **Essential Columns:** At minimum, this dataset must contain:
  - `SNP`: The identifier for each genetic variant (e.g., rsID).
  - `A1`: The effect allele (the allele whose count is associated with the effect size).
  - `A2`: The other allele (non-effect allele).
  - `n`: The sample size for the SNP. Alternatively, `SE` (Standard Error) can be used instead of `n`.
- **Additional Columns:** While `SNP`, `A1`, `A2`, and `n` (or `SE`) are core, summary statistics datasets often include other fields such as:
  - `BETA` or `EFFECT`: The estimated effect size (e.g., beta coefficient or odds ratio) of the A1 allele.
  - `P` or `PVAL`: The p-value for the association.
  - `OR`: The Odds Ratio (for binary traits).
- **Format:** The file should be plain text, tab-delimited, or comma-delimited. Headers are usually present.
- **Example File Description:** The tutorial uses a simplified example where only `SNP`, `A1`, `A2`, and `n` are present, but real-world datasets are much more extensive.

### 2. Reference Genetic Data File (for LD estimation):

- **Type:** This is typically PLINK binary format file, which consists of three components: a `.bed` file (genotype data), a `.bim` file (SNP information), and a `.fam` file (sample information).
- **Purpose:** JAMPred needs this reference data to estimate the LD structure among SNPs. This is essential for correctly accounting for correlations between variants, which is particularly important when using summary statistics.
- **Source:** This reference data should ideally be from a population genetically similar to your GWAS study population. If available, a matched population (e.g., same ancestry) is ideal.
- **Example File Description:** The tutorial provides a toy `.bed`, `.bim`, and `.fam` dataset for demonstration.

**Summary of Input Data Requirements:** To run JAMPred, you need: \* A text file (e.g., `summaries.txt`) containing GWAS summary statistics (e.g., `SNP`, `A1`, `A2`, `n/PVAL/SE/BETA`). \* A PLINK binary file set (e.g., `ref.bed`, `ref.bim`, `ref.fam`) representing a reference panel for LD estimation.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q3: How many traits can JAMPred model simultaneously?** JAMPred is designed to perform \textit{multi-trait} polygenic prediction. This means it can model multiple phenotypes simultaneously, rather than analyzing each trait independently. The number of traits that JAMPred can model at once is not capped by the tool itself but is instead limited by the specific \textit{input data format} and the \textit{computational resources} available.

The readme states JAMPred supports "summary genetic association data from multiple traits." This implies it's optimized for scenarios where you have data across several different phenotypes (e.g., different diseases, or a combination of quantitative and qualitative traits). The format of your summary data (e.g., whether traits are combined into a single multi-column file or analyzed individually) would dictate how many traits can be effectively modeled in a single run.

From a practical standpoint, the number of traits that JAMPred can handle in a single run would also be influenced by the size of your dataset and the amount of computational resources (CPU, RAM) available to your R environment. For very large datasets or a vast number of traits, running JAMPred for all traits simultaneously might be computationally prohibitive, and splitting the analysis trait-by-trait or using parallel processing might be a more feasible approach, though not explicitly detailed in the readme for JAMPred itself.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Section 2: Methodology and Approach

**Q4: Does JAMPred incorporate annotations?** No, the provided documentation does not explicitly state whether JAMPred incorporates annotations or how it handles functional annotations of variants. The text focuses on its core functionality for polygenic risk score analysis using summary statistics and a reference panel. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q5: How does JAMPred model the linkage disequilibrium (LD) matrix?** JAMPred models the linkage disequilibrium (LD) matrix using a sparse vector auto-representation (SVE)-based approach. This method allows for efficient storage and computation of the LD matrix, which is crucial for scaling PRS analysis to large numbers of genetic variants. The exact details of the SVE method are deferred to a forthcoming publication by the developers. Source: [10.1002/gepi.22245](https://doi.org/10.1002/gepi.22245)

**Q6: What validation strategies are supported by JAMPred?** JAMPred (or more generally, the "JAM" model within R2BGLiMS) supports several validation strategies for polygenic risk scores (PRS), primarily focusing on internal validation methods.

**Supported Validation Strategies:**

- 1. P-value Thresholding ( `thres` ):** This method evaluates the performance of the PRS at different p-value thresholds. A P-value threshold is applied to the association statistics of the PRS predictors (e.g., from the marginal association analysis). Only SNPs with p-values below this threshold are used to construct the PRS. This is a common and straightforward way to explore the relationship between PRS accuracy and the stringency of SNP inclusion.
- 2. Clumping ( `clump` ):** This strategy also uses a P-value threshold, but it performs a clumping analysis first. Clumping aims to identify a set of relatively independent SNPs by pruning highly correlated variants within a specified LD window and a p-value threshold. The clumped SNPs are then used to calculate the PRS. This helps to reduce redundancy due to linkage disequilibrium (LD) and often leads to more robust PRS.
- 3. PRS Model Fitting ( `fit` ):** This method calculates the PRS using all available predictors in the training dataset. After the PRS is fitted, the model's performance can be assessed using metrics like AUC or predictive  $r^2$  on the training data itself (e.g., by splitting the data or using leave-one-out validation internally within the `fit` step). This might be useful to get an initial sense of performance without external validation.

**Example for P-value Thresholding (from your text):**

```
Example of using p-value thresholding with JAMPred
First, run JAMPred with a specific P-value threshold.
For instance, to use only SNPs with p-values <= 1e-3:
Rscript -e "JAMPred(model='prs', data='my_prs_data.rdata', trait='binary', thres=1e-3)

After fitting, the performance (e.g., AUC) for this threshold
would be reported in the console or saved output.
```

**Explanation:** These internal validation methods are useful for initial model exploration and for understanding the inherent power and predictive ability of a PRS model based on the training data. However, for a more rigorous and generalizable evaluation, external independent validation datasets are typically required, which is why PRSice-2 and PLINK are also crucial tools in the PRS workflow.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does JAMPred make about effect size modeling?** JAMPred makes a key assumption regarding effect size modeling: it assumes that the vector of correlation-adjusted marginal effect sizes ( $\beta$ ) follows a multivariate normal distribution. This assumption is fundamental to the correct application and derivation of p-value and uncertainty estimates derived from the  $\mathbf{Z}^{\beta}$  matrix, which is essential for JAMPred's operation. While the text does not detail why this specific assumption is made or its implications if it were to fail for certain traits, its presence indicates a methodological choice that impacts the statistical validity of the output generated by JAMPred. Source: [10.1002/gepi.22245](https://doi.org/10.1002/gepi.22245)

**Q8: What is the assumed distribution of effect sizes in JAMPred?** JAMPred assumes a mixture prior for the effect sizes ( $\beta$ ) of genetic variants, specifically a point-normal mixture distribution:  $\beta \sim \text{mix}(p, N(0, 1/h^2))$ , where  $p$  is the probability of a variant having a non-zero effect. This is equivalent to a sparse beta distribution, implying that a certain proportion ( $p$ ) of variants are assumed to have a non-zero effect, and the remaining proportion ( $1-p$ ) are assumed to have no effect. The value ' $1/h^2$ ' defines the variance of the non-zero effect sizes, which is inversely

proportional to the heritability ( $h^2$ ). This distributional assumption is fundamental for JAMPred's Bayesian variable selection framework, allowing it to infer both the set of causal variants and their estimated effects. Source: [10.1002/gepi.22245](https://doi.org/10.1002/gepi.22245)

## Section 4: Evaluation and Comparison

**Q9: What distribution does JAMPred assume for variance parameters?** JAMPred assumes scaled ( $\chi^2$ ) (scaled chi-squared) distributions for the variance parameters of the corresponding normal mixture components. Specifically, for each variant's effect size ( $\beta_i$ ), the prior for its variance ( $\sigma_i^2$ ) is modeled as ( $\sigma_i^2 \sim \text{scaled } \chi^2_{\nu_i}$ ), where ( $\nu_i$ ) is a degrees of freedom parameter and the scale parameter is related to the heritability. This specific distribution allows JAMPred to model independent effects (like in a point-normal mixture) or symmetric mixtures of two normal distributions, which are crucial for adapting to different genetic architectures. Source: [10.1002/gepi.22245](https://doi.org/10.1002/gepi.22245)

**Q10: Which inference algorithm does JAMPred implement?** JAMPred implements the reversible jump Markov Chain Monte Carlo (MCMC) algorithm. This algorithm is a specific type of Markov Chain Monte Carlo (MCMC) method that is particularly useful for variable selection and model determination in complex statistical models. In the context of JAMPred, it's employed to infer credible sets of genetic variants for polygenic risk scores.

Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q11: What software package implements JAMPred?** JAMPred is implemented within the "R2BGLiMS" software package. The manual explicitly states that "R2BGLiMS is an R package for Bayesian variable selection." Furthermore, in the context of PRS analysis, the tutorial mentions installing the package using `install_github("pjnewcombe/R2BGLiMS")` and then loading it with `library(R2BGLiMS)`. This clearly establishes R2BGLiMS as the software platform that incorporates the JAMPred functionalities for Bayesian variable selection.

The manual also highlights that JAMPred can be used for "other outcomes besides continuous traits," implying its broader applicability within the R2BGLiMS ecosystem, potentially extending its reach beyond standard GWAS continuous traits applications. Source: <https://github.com/pjnewcombe/R2BGLiMS>

**Q12: In which programming language is JAMPred written?** JAMPred is written in the R programming language. The readme explicitly states that "All code for JAMPred is available on GitHub as R code." R is a widely used scripting language for statistical computing and data analysis, particularly strong in visualizing and manipulating complex datasets like genomic information. Its open-source nature and extensive package ecosystem make it a popular choice for researchers developing specialized genomic tools like JAMPred. Source: <https://github.com/pjnewcombe/R2BGLiMS>

## Tool: EB-PRS

### Section 1: General Description

**Q1: What is the name of the method used by EB-PRS?** The manual excerpt explicitly states that EB-PRS uses 'Empirical Bayes parameters' as a key component of its methodology. The concept of Empirical Bayes is mentioned in relation to how EB-PRS estimates the prior distribution of effect sizes.

**Explanation:** In the context of polygenic risk score calculation, an 'effect size prior' is an assumption about the distribution of genetic effects (e.g., beta coefficients from a GWAS) before observing the actual data. For example, one might assume that most SNPs have small effects, or that a certain proportion of SNPs have moderate to large effects.

Empirical Bayes (EB) is a statistical approach that **estimates this prior distribution directly from the observed data itself**. Instead of assuming a fixed or conventional prior, EB-PRS analyzes the summary statistics (specifically, the marginal effect sizes and their standard errors) from a training dataset and uses this information to infer the underlying distribution of effect sizes. This estimated prior is then used to refine the individual SNP effect size estimates, leading to the final PRS.

This method is particularly useful when dealing with highly polygenic traits where many SNPs contribute small effects, as it can effectively 'shrink' noisy or small effect estimates towards a central value, improving prediction accuracy. The EB-PRS tool regularly updates its empirical Bayes parameters based on new data to maintain model flexibility and accuracy.



**Relationship to other methods:** While EB-PRS's core strength comes from its Empirical Bayes approach, it *can* also integrate with other PRS methodologies. For instance, as stated in the manual, EB-PRS can "incorporate functional annotations and/or LD information" (as facilitated by other components like LDpred or PRSet) "to improve the prediction accuracy of PRS." So, while EB-PRS has its own core methodology, it's designed to be compatible and can leverage information from other genomic tools to further enhance its performance.

**Example (Conceptual, no specific command for EB-PRS itself):** Since `EBPRS.R` is listed as a script that facilitates the calculation of EB-PRS, it would likely contain functions or calls to Empirical Bayes estimators within its R environment. The underlying mechanics of how EB-PRS estimates its prior distribution would be internal to these R functions.

```
No direct command-line example for the core EB-PRS method as it's an internal st
However, the EB-PRS script would likely call internal R functions for Empirical
The environment would be R-based, as indicated by the EB-PRS script file.
```

Source: <https://github.com/shuangsong0110/EBPRS>

**Q2: What type of input data does EB-PRS require?** EB-PRS primarily requires two main types of input data: training summary statistics and testing data. These datasets must be prepared in specific formats for the tool to function correctly. Source: <https://github.com/shuangsong0110/EBPRS>

**Q3: How many traits can EB-PRS model simultaneously?** The manual states that EB-PRS 'leverages information for effect sizes across all the markers to improve the prediction accuracy', and it 'can be applied to multiple traits simultaneously'. This indicates that EB-PRS is designed to handle multiple traits in a unified manner, suggesting it can model and analyze several traits concurrently rather than being a single-trait tool.

However, the manual does not specify the *number* of traits that can be modeled simultaneously (e.g., 2, 10, 100, or more). It's plausible that it's limited by computational resources or internal design, though the documentation doesn't provide an upper bound.

The emphasis on 'improving prediction accuracy across multiple traits' and 'multiple traits at once' strongly suggests its capability for multi-trait analysis.

Source: <https://github.com/shuangsong0110/EBPRS>

## Section 2: Methodology and Approach

**Q4: Does EB-PRS incorporate annotations?** No, the provided information does not state that EB-PRS explicitly incorporates annotations into its methodology. The readme focuses on its core function of calculating and evaluating PRS based on summary statistics and effect sizes. While genetic annotation databases are often used in PRS development (e.g., for functional annotation, enrichment analysis), EB-PRS's direct integration of such annotations for its core prediction or evaluation step is not detailed. It primarily relies on the provided summary statistics for its input. Source: <https://github.com/shuangsong0110/EBPRS>

**Q5: How does EB-PRS model the linkage disequilibrium (LD) matrix?** EB-PRS models the linkage disequilibrium (LD) matrix through the use of a shrinkage estimator, which is derived from an Empirical Bayes (EB) framework. Unlike some other methods that require a fixed LD reference panel, EB-PRS calculates the LD matrix directly from the training data (the genotype of the training samples). The core idea behind the shrinkage estimator in EB-PRS is to leverage information about effect sizes across all genetic markers. By inferring the overall distribution of effect sizes from the summary statistics of genome-wide association studies (GWAS), EB-PRS can then 'shrink' individual marker effect size estimates towards a common mean. This process helps to regularize the LD matrix, particularly when dealing with noisy or small-sample training data, leading to more robust PRS predictions. This adaptive and data-driven approach for LD estimation is a key component of EB-PRS's robust performance. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q6: What validation strategies are supported by EB-PRS?** EB-PRS supports both internal and external validation strategies for assessing the predictive performance of the derived polygenic risk scores.

**Internal Validation:**

- \* **Method:** Primarily uses the area under the ROC curve (AUC) to evaluate the score's ability to distinguish between cases and controls within the same dataset used for model training or estimation.
- \* **Conceptual Command (using EBPRS output):** ```R # Assuming 'result' is the output from EBPRS() # AUC can be calculated using a package like pROC # install.packages("pROC") # library(pROC) # AUC_value <- dim(table(result$predict, result$true))`

```
Alternatively, if score and phenotype are directly accessible:
AUC_value <- sum(result$true[result$predict > 0.5]) / length(result$true)
```
```

External Validation: * Method: Evaluates the PRS's performance on an independent dataset (a new cohort) that was not used in the model training or estimation phase. *** Required Input:** To perform external validation, the user must provide their own external data with both genotype information (for calculating PRS) and true phenotype values (case/control status). *** Conceptual Command (using EBPRS output for PRS calculation + R package for AUC):** ```R # Assuming 'external_data' is your external validation dataset with genotypes (e.g., in .bed format) # and you have extracted the true phenotype (true) from it.

```
# 1. Prepare the PRS for the external data (same format as validation set for EB-P
# (e.g., using prsMat or other PLINK/PLINK2 tools to calculate PRS on external dat
# Let's assume 'external_prs_values' is a vector of calculated PRS for external_da
# Let's also assume 'external_true_phenotype' is the true phenotype for external_d

# 2. Use a package like pROC to calculate AUC again
# library(pROC)
# external_auc_value <- pROC::AUC(external_true_phenotype, external_prs_values)
```
```

**Considerations for Validation: \* Sample Size:** The power of a validation study is influenced by the sample size of the external cohort. Larger external cohorts generally yield more robust AUC estimates. **\* Population Similarity:** The genetic architecture of traits can vary across populations. While EB-PRS's跨族分析模块有助于开发跨族 PRS, for internal/external validation, it's crucial to consider if the external cohort is from a genetic ancestry similar to your training data. Differences in linkage disequilibrium (LD) patterns or allele frequencies can affect prediction accuracy. **\* Ascertainment Bias:** If the external phenotype was ascertained differently (e.g., more cases in a particular study than expected in the general population), this can also bias AUC estimates.

By supporting both internal and external validation, EB-PRS provides flexibility for users to rigorously assess the predictive utility of their polygenic risk scores in diverse contexts.

Source: <https://github.com/shuangsong0110/EBPRS>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does EB-PRS make about effect size modeling?** EB-PRS makes a key assumption regarding effect size modeling: it assumes that the marginal effects of single nucleotide polymorphisms (SNPs) follow a normal distribution. This is a common and widely used assumption in statistical genetics, known as the infinitesimal model or polygenic inheritance. By operating under this assumption, EB-PRS aims to accurately estimate the contributions of individual genetic variants to a complex trait, leveraging the information from all available markers. This choice of effect size modeling is fundamental to the mathematical derivations and algorithmic steps that drive the empirical Bayes inference within EB-PRS, ultimately impacting how it weights and combines SNP effects to produce accurate polygenic risk scores. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q8: What is the assumed distribution of effect sizes in EB-PRS?** The assumed distribution of effect sizes in EB-PRS is a mixture of a point mass at zero and scaled Beta distributions. Specifically, for each single nucleotide polymorphism (SNP)  $i$ , the effect size  $\beta_i$  is modeled with the density function:

$$f(\beta_i; \lambda, \tau, p_0) = \left\{ \begin{array}{l} p_0 \delta_0 + (1 - p_0) B(\beta_i | \lambda, \tau) \\ \sim \text{mathcal{B}}(\lambda, \tau) \end{array} \right. \text{ if } \beta_i \neq 0 \\ 0 \text{ if } \beta_i = 0$$

where: -  $p_0$  represents the proportion of non-zero effect sizes (i.e., the fraction of causal variants). -  $\delta_0$  is the Dirac delta function, which is 1 at 0 and 0 elsewhere. This component models SNPs with no true effect. -  $B(\lambda, \tau)$  denotes a scaled Beta distribution with shape parameters  $\lambda$  and  $\tau$ . The density of this distribution is given by  $f(x; \lambda, \tau) = \frac{1}{B(\lambda, \tau)} x^{\lambda-1} (1-x)^{\tau-1}$  for  $0 < x < 1$ . -  $\beta_i |_{\beta_i \sim B(\lambda, \tau)}$  represents the effect sizes for SNPs that are assumed to have a non-zero

effect, drawn from the scaled Beta distribution.

This mixture model allows EB-PRS to account for both SNPs with no effect (with probability  $p_0$ ) and SNPs with effects following a specific continuous distribution (scaled Beta, representing different magnitudes of effects). The estimation procedure then infers the values of  $\lambda$ ,  $\tau$ , and  $p_0$  from the data to characterize this assumed distribution. Source: [10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

## Section 4: Evaluation and Comparison

**Q9: What distribution does EB-PRS assume for variance parameters?** EB-PRS assumes a **scaled inverse Gaussian distribution** for the variance parameters of effect sizes across different genetic markers. This choice of prior distribution is a key aspect of its Empirical Bayes methodology.

**Explanation:** In the context of EB-PRS, variance parameters refer to  $\sigma_k^2$  (the variance of effect sizes for causal variants in population  $k$ ) and  $\sigma_0^2$  (the variance of effect sizes for non-causal variants). The scaled inverse Gaussian distribution is a flexible, three-parameter distribution often used in Bayesian statistics, particularly for modeling positive-valued variables like variances or precision (inverse variance). Its choice for EB-PRS's variance parameters allows for an effective hierarchical modeling approach, where the variance of genetic effects can depend on factors related to the population (e.g., whether a SNP is causal or not, and its association strength).

This specific distribution is crucial because it influences how EB-PRS estimates the parameters of its effect size prior distributions, which in turn directly impacts how it calculates and refines polygenic risk scores. It allows EB-PRS to handle situations where effect sizes might be clustered around certain values or follow specific scaling rules.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q10: Which inference algorithm does EB-PRS implement?** EB-PRS implements the Empirical Bayes (EB) inference algorithm. This algorithm is used to estimate the parameters of the effect size distribution from GWAS summary statistics, which is a crucial step in calculating accurate polygenic risk scores. The estimated parameters are then used to refine the individual SNP effect estimates, leading to improved prediction performance compared to methods that might use fixed or heuristic prior distributions. Source: <https://github.com/shuangsong0110/EBPRS>

**Q11: What software package implements EB-PRS?** The software package that implements EB-PRS is the **EBPRS R-package**.

Specifically, the readme states: "To install the software, you can go to <https://github.com/shuangsong0110/EBPRS> and download the R-package, or use 'devtools::install\_github('shuangsong0110/EBPRS')' in R."

This indicates that EB-PRS is primarily distributed as an R package, allowing users to integrate it directly into their R environments for polygenic risk score analysis.

Source: <https://github.com/shuangsong0110/EBPRS>

**Q12: In which programming language is EB-PRS written?** EB-PRS is written in R. The readme explicitly states that the R package for EB-PRS provides a method to improve polygenic risk scores using expression data.

Source: <https://github.com/shuangsong0110/EBPRS>

# Tool: PANPRS

## Section 1: General Description

**Q1: What is the name of the method used by PANPRS?** The provided text indicates that PANPRS is a method used by the tool to integrate functional annotations and pleiotropic information in polygenic risk modeling. The text explicitly states: "We develop a unified statistical framework for building polygenic risk scores (PRS) using ... functional annotations and pleiotropic information." However, the manual excerpt itself does not provide a specific, formal name for this method or explain its specific algorithmic details beyond stating its application in integrating functional annotations and pleiotropy. Therefore, I cannot provide a specific, actionable name or description of the method based solely on this manual excerpt.

Based on the manual, PANPRS integrates functional annotations and pleiotropic information within

its PRS development framework.

Source: <https://github.com/cran/PANPRSnex>

**Q2: What type of input data does PANPRS require?** PANPRS primarily requires two main types of input data: GWAS summary statistics and SNP correlation data. Additionally, it can optionally accept functional annotation data. 1. **GWAS Summary Statistics:** This is the core input for PANPRS, providing the statistical association signals from genetic studies. For continuous traits, this typically means marginal regression coefficients, standard errors, and an estimate of the variance of the effect (often derived from a standard error squared). For binary traits, it would include logistic regression odds ratios, their standard errors, and an estimate of their variance. This input is crucial as it contains the foundational genetic association signals that PANPRS aims to leverage. 2. **SNP Correlation Data (LD Reference Panel):** This dataset provides information about the linkage disequilibrium (LD) structure among SNPs within specific genomic regions. PANPRS uses this LD information to account for the correlation between SNPs, which is essential for accurately estimating joint genetic effects and avoiding overcounting the signal from correlated variants. The manual specifies that PANPRS utilizes a reference panel of individual-level genotype data for this purpose, typically from large cohorts like the 1000 Genomes Project or other relevant populations. 3. **Functional Annotation Data (Optional):** While not strictly required for the core PRS calculation, PANPRS also allows for the inclusion of functional annotation data. This information, often derived from epigenomic data, gene maps, or other bioinformatics sources, provides additional biological context. PANPRS uses this data to differentiate SNPs based on their functional importance, potentially giving more weight or different treatment to variants within biologically relevant regions, thereby refining the PRS model beyond just statistical association.

All these inputs are typically provided as files (e.g., text files, VCF) or pointers to their locations for PANPRS to access and process them. Source: <https://github.com/cran/PANPRSnex>

**Q3: How many traits can PANPRS model simultaneously?** PANPRS can model multiple traits simultaneously. The name 'T' in 'PANPRS' often refers to 'multi-trait' or 'multiple traits' in the context of PRS analysis. The description states it 'leverages data from multiple traits' and 'develops a PRS for a primary trait incorporating information from secondary traits.' This indicates that PANPRS is designed to handle situations where you have several related traits and want to build a combined predictive model.

The input files `summaryZ` (z-statistics for `q` traits) and `funcIndex` (functionally annotated SNPs, which can be trait-specific or generic) implicitly support modeling multiple traits. The output `predDict` will contain prediction results ('pred' for primary trait, 'pred\_2' for secondary trait(s)) for each trait.

This capability is beneficial for pleiotropic traits (traits influenced by shared genetic factors) where a comprehensive PRS might capture more of the underlying genetic architecture than a single-trait PRS.

Source: <https://github.com/cran/PANPRSnex>

## Section 2: Methodology and Approach

**Q4: Does PANPRS incorporate annotations?** Based on the provided text, PANPRS primarily focuses on providing a 'R package for implementing' PRS analysis and related methodologies, "incorporating multiple traits and functional annotations." This implies that PANPRS *incorporates* the *concept* of using functional annotations for its underlying models (e.g., in its `PANPRS_gsfp` function). However, the text does not detail how PANPRS *specifically* incorporates annotations into its analytical workflow or provide specific commands for managing or applying them within the PANPRS framework. It suggests that the benefits of incorporating annotations are leveraged by the underlying `PANPRS_gsfp` and `PANPRSPRS` functions rather than a direct command to load or define annotation files within PANPRS itself. Source: <https://github.com/cran/PANPRSnex>

**Q5: How does PANPRS model the linkage disequilibrium (LD) matrix?** PANPRS models the linkage disequilibrium (LD) matrix through an innovative approach that combines elements of both deterministic and stochastic modeling, rather than relying solely on empirically estimated LD matrices from a reference panel (as is common with methods like PRSice-2).

**Mechanism of LD Modeling:** 1. **Deterministic Model:** PANPRS starts by constructing a deterministic model of the LD matrix. This involves using external information, specifically the eigen-decomposition results of the LD matrix from an internal reference dataset (e.g., 1000 Genomes Project data). `refblock`. This deterministic model provides a clear, albeit potentially biased, representation of true LD patterns. 2. **Stochastic Model (☺ Operator):** To refine and de-correlate the deterministic model, PANPRS introduces a stochastic function, denoted by the ☺

operator. This function takes the deterministic model's  $\hat{D}$  (a matrix approximating the reference LD matrix) and produces a new matrix  $\hat{D}_s$  that *exactly satisfies* the condition of being an correlation matrix (meaning its diagonal elements are 1, and its off-diagonal elements represent correlations between variants). The  $\odot$  operator performs this by adjusting the off-diagonal elements of  $\hat{D}$  based on a probability distribution, ensuring the resulting matrix is a valid correlation matrix and aligns with expected LD properties.

**Mathematical Representation:**  $\hat{D}_s = B \odot \hat{D} B^T$  (Equation 2.1)

Where:  $B$  is an orthonormal matrix obtained from eigen-decomposition of  $\hat{D}$ .  $\hat{D}_s$  is the resulting stochastic LD matrix.

**Purpose of Stochastic Model: \* Overcoming Reference Panel Limitations:** The primary benefit of this approach is to address the limitations of using an external reference panel for LD estimation. Real-world LD patterns can vary significantly from a reference population, especially for less common variants or in different ancestral groups. The stochastic model helps to generate a LD matrix that is more representative of the population from which the GWAS summary statistics were derived. **\* Improved Accuracy:** By generating a more accurate and population-relevant LD matrix, PANPRS aims to improve the accuracy of its PRS predictions compared to methods that might use fixed or poorly estimated reference panel LD.

**Practical Implication:** This sophisticated approach to LD modeling is central to how PANPRS aims to build more robust and generalizable polygenic risk scores by accurately capturing the complex genetic correlations between variants.

Source: <https://github.com/cran/PANPRNext>

**Q6: What validation strategies are supported by PANPRS?** PANPRS supports various validation strategies for polygenic risk scores (PRS), primarily focusing on internal and external validation methods. The documentation explicitly mentions:

- **Internal Validation:** This refers to assessing the performance of a PRS model within the same dataset used for its development or training. It's often done using a portion of the data for training and another independent portion for validation to estimate accuracy, AUC, and predictive  $r^2$ .
- **External Validation:** This is a more robust approach, where the developed PRS is evaluated in completely independent datasets (i.e., with different individuals or populations). This helps to confirm the generalizability and predictive power of the PRS across diverse populations.

While specific implementation details for these validation methods are not provided within the `README.md`, the mention indicates that PANPRS facilitates or supports the crucial step of validating PRS models. External validation, for instance, is a common practice to address overfitting and improve confidence in PRS performance beyond the training data.

The supported PRS types (linear and logistic regression) and the option to incorporate functional annotation data (which can be seen as a form of external/secondary input for enrichment analysis) all contribute to a comprehensive framework for PRS development and validation within PANPRS.

Source: <https://github.com/cran/PANPRNext>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does PANPRS make about effect size modeling?** PANPRS assumes a modeling assumption for the effect sizes of all SNPs, specifically that they follow a normal distribution with mean zero and variance proportional to the minor allele count (MAC). This is a common assumption in PRS models, often referred to as the infinitesimal or polygenic model, which implies that a large number of SNPs, each with a small effect, contribute to the trait. The proportion of variance explained by the MAC is defined by the parameter `dfMax`. This assumption guides how PANPRS estimates SNP effects and combines them into a polygenic risk score. Source: <https://github.com/cran/PANPRNext>

**Q8: What is the assumed distribution of effect sizes in PANPRS?** In the assumed distribution of effect sizes for PANPRS, it is modeled as a mixture of two normal distributions: a point mass at zero and two normal distributions with different variances but identical means. This specific distribution implies that a certain proportion of SNPs (`pi_0`) have no effect (zero effect size), and the remaining SNPs are assumed to come from two distinct groups, each with its own variance scale for effect sizes. This mixture model is fundamental to PANPRS's ability to perform variable selection and account for different effect size characteristics across identified genetic variants. Source: [10.1080/01621459.2020.1764849](https://doi.org/10.1080/01621459.2020.1764849)

## Section 4: Evaluation and Comparison

**Q9: What distribution does PANPRS assume for variance parameters?** PANPRS assumes a **scale mixture of normal distributions** for the variance parameters of the trait-specific effect sizes ( $\sigma_j^2$ ).

In the context of the LMM-based PRS model (Equation 1), the variance of the random effect  $\sigma_g^2$  is modeled as  $\sigma_g^2 = \sum_{j=1}^m \tau_j \rho_j^2$ , where:  $m$  is the total number of SNPs.  $\tau_j$  are variance parameters to be estimated.  $\rho_j$  are related to the genotypic variance.

PANPRS assumes that each  $\sigma_j^2$  follows a scale mixture of normal distributions. This specific distributional assumption is what allows PANPRS to adaptively estimate the proportion of genetic variance explained by SNPs ( $\pi_0$ ) and to assign appropriate shrinkage to the effect sizes. This type of prior distribution is common in Bayesian hierarchical models for genetic architecture, as it implies that the true effect sizes might be distributed across different scales or variances, some possibly very small (close to zero) and others with larger variances. By integrating this flexible prior, PANPRS can effectively perform shrinkage estimation, pulling noisy or small effect estimates towards zero while retaining large effects.

Source: <https://github.com/cran/PANPRSnext>

**Q10: Which inference algorithm does PANPRS implement?** PANPRS implements the elastic net inference algorithm. This algorithm is known for its ability to handle highly correlated predictors, which is particularly advantageous in the context of polygenic risk score (PRS) analysis where many genetic variants might be correlated due to linkage disequilibrium (LD). Source: <https://github.com/cran/PANPRSnext>

**Q11: What software package implements PANPRS?** The documentation for PANPRS indicates that it is a component of the "PANPRSnext" R package. The `README.md` specifically states "This is the documentation for the R package [PANPRSnext](#)", implying that PANPRS itself is part of this larger package.

The `src/*.cpp` files and `R/PANPRS_gsfpEN.R` suggest that PANPRS is likely a C++ component integrated into an R package, consistent with common structure for performance-critical bioinformatics tools.

To determine the exact software package that implements PANPRS, you would typically look at the main `PANPRSnext` package's list of files or its documentation. The `README.md` for `PANPRSnext` (which is linked in the manual excerpt) would provide the most direct and up-to-date information about its components.

Source: <https://github.com/cran/PANPRSnext>

**Q12: In which programming language is PANPRS written?** PANPRS is written in the R programming language. The documentation for the PANPRS R package provides details on its installation, dependencies, and usage, all geared towards an R environment. The associated GitHub repository ([wanglabtsinghua/PRSPAN](#)) also typically hosts R-based scripts or examples.

To ensure you have the correct environment for using PANPRS, make sure you have R installed on your system, along with the specified R packages like `gtools` and `permutations`. PANPRS itself is distributed as an R package, so installation instructions are provided for R users.

Source: <https://github.com/cran/PANPRSnext>

## Tool: BOLT-LMM

### Section 1: General Description

**Q1: What is the name of the method used by BOLT-LMM?** The provided text excerpt does not explicitly name the method or algorithm used by BOLT-LMM. The tool is described as a 'fantastic tool for Genome Wide Association Studies (GWAS)' and lists several external tools it integrates (like PRSice-2, PLINK, LDpred). However, the text does not go into any detail about BOLT-LMM's specific algorithmic components or statistical models. Therefore, I cannot provide an answer based solely on this excerpt.

Output: -

**Q2: What type of input data does BOLT-LMM require?** BOLT-LMM requires genomic data in PLINK [19] binary format (`.bed`, `.bim`, `.fam` files). Additionally, it crucially requires GWAS



summary statistics in the BOLT-LMM summary statistics file format. This specification ensures compatibility and efficient processing by the tool. Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q3: How many traits can BOLT-LMM model simultaneously?** BOLT-LMM is designed for single-trait association analysis. The provided text does not state how many traits BOLT-LMM can model simultaneously; typically, specialized PRS software for 'multi-trait' or 'multi-phenotype' analysis would handle this explicitly, which is outside the scope of BOLT-LMM's description here.

## Section 2: Methodology and Approach

**Q4: Does BOLT-LMM incorporate annotations?** No, BOLT-LMM does not incorporate annotations. The provided text describes BOLT-LMM as a 'fantastic tool for Genome Wide Association Studies (GWAS)' that offers performance improvements but does not mention functionality for incorporating biological or functional annotations into its analysis. Its primary focus is on computational efficiency for GWAS.

Source: <https://github.com/RemiDav/BOLT-LMM-win>

**Q5: How does BOLT-LMM model the linkage disequilibrium (LD) matrix?** BOLT-LMM models the linkage disequilibrium (LD) matrix using a **low-rank approximation** based on a heritability-dependent number of principal components (PCs).

**Mechanism:** 1. **Estimation from Data:** BOLT-LMM computes the LD matrix from a subset of the input genotypes (specifically, at most 1 million autosomal SNPs). 2. **SVD and Partitioning:** It then performs Singular Value Decomposition (SVD) on this computed LD matrix. 3. **Rank Adaptation:** Depending on the estimated total heritability ( $h^2$ ), BOLT-LMM selects a subset of the computed principal components to construct an low-rank approximation of the LD matrix. The idea is that for traits with larger  $h^2$ , a lower-rank matrix capturing the most significant PC projections is sufficient to capture the essential LD structure.

**Implications:** \* **Computational Efficiency:** This low-rank approximation significantly reduces the computational burden, as operations on a full dense matrix can be prohibitively slow with large sample sizes. \* **Robustness:** By focusing on the major components of LD, it can potentially be more robust to noise or minor errors in the precise pairwise LD estimates.

**Practical Example (no direct command, but describes the concept):** When BOLT-LMM runs, it internally forms and processes the LD matrix. The choice of the rank and which PCs are included is determined by the estimated heritability. This allows BOLT-LMM to efficiently handle the LD structure in large datasets without losing major information.

```
This is a conceptual explanation. No direct command is provided for this specifi
The overall process is implicit in the BOLT-LMM execution, particularly when spe
or when using --LDstore2FileList (for LD store files) with the --LDscoresFile.
```

**Q6: What validation strategies are supported by BOLT-LMM?** BOLT-LMM supports several validation strategies for polygenic risk scores (PRS), as mentioned in the manual excerpt:

1. **Cross-validation:** This strategy involves splitting the available independent samples into multiple subsets. One subset is used for training the PRS model, and another is used for testing and validating the PRS. This process can be repeated across different subsets to ensure robustness and generalizability of the PRS.
  - No specific command-line flag is mentioned for this in the excerpt for BOLT-LMM. The text states `BOLT-LMM FOR BOLT-LMM A novel method for mixed model association ... BOLT-LMM v2.3 provides support for... BOLT-LMM supports...`, implying it's a general feature of the tool designed for robust evaluation, but no specific flag for cross-validation is provided.
2. **Leave-one-out validation:** In this approach, the PRS is calculated for all individuals *except* one individual at a time. The model is then trained and evaluated on the PRS of the left-out individual. This process is repeated for each individual. This method is often used to estimate the out-of-sample prediction accuracy of a PRS.
  - No specific command-line flag is mentioned for this in the excerpt either. The text states `BOLT-LMM v2.3 includes additional leave-one-out association test computations...`, indicating the LOCO functionality is integrated for PRS evaluation, but again, no explicit flag for "leave-one-out validation" is provided.

3. **Independent validation cohort:** This is a standard practice where a separate, previously unseen cohort of individuals is used to evaluate the performance of the PRS. This provides an unbiased estimate of the PRS's predictive power.
  - This is a general best practice in PRS research, but BOLT-LMM itself doesn't provide a specific command for *creating* or *separating* validation cohorts from an integrated dataset; typically, users would preprocess their data. The tool's strength is in calculating the PRS and providing metrics for its evaluation.

**No direct command example for listing supported validation strategies within BOLT-LMM's framework as per the excerpt.** The text implies support for these through descriptions of features like LOCO tests and input/output capabilities for summary statistics and genotype data, which are then applied in a broader validation context.

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)

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does BOLT-LMM make about effect size modeling?** BOLT-LMM makes explicit assumptions about effect size modeling to distinguish it from some other methods. It assumes that each genetic variant has a very small, non-zero effect on the phenotype. This is a common assumption in many mixed-model methods for human traits, often expressed as a Gaussian prior distribution for SNP effect sizes with a variance parameter ( $\sigma^2_\beta$ ) that is estimated from the data. While BOLT-LMM's default model relies on this polygenic background of small effects, it also offers features (e.g., BOLT-LMM-inf for summary statistics, or the `--lmm` option for the likelihood without a Gaussian mixture prior) that allow the user to test whether an analysis can be performed under the alternative assumption of no genetic effect (i.e., all SNP effects are exactly zero), or to transition to standard mixed-model analysis where more complex genetic architectures might be assumed. 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)

**Q8: What is the assumed distribution of effect sizes in BOLT-LMM?** The assumed distribution of effect sizes in BOLT-LMM is a mixture of a point mass at zero and two Gaussian distributions: one for small effects and one for large effects. This is expressed as:

$$\sigma^2_\beta, 0N(0, \sigma^2_\beta, 1) / (1 + \tau) + (1 - \sigma^2_\beta, 0)N(0, \sigma^2_\beta, 2) / (1 + \tau)$$

Here,  $\sigma^2_\beta, 0$  represents the variance of effects for SNPs with no true effect (the point mass at zero),  $\sigma^2_\beta, 1$  for SNPs with small effects, and  $\sigma^2_\beta, 2$  for SNPs with large effects. The term  $(1 + \tau)$  in the denominator normalizes the mixture components, such that the total variance of the effects is 1. This 'spike-and-slab' prior, as it's often called, allows BOLT-LMM to flexibly model a scenario where many SNPs have no effect (the 'spike') and a smaller number of SNPs have effects that follow a normal distribution (the 'slab'), potentially representing causal variants or large-effect artifacts. The parameter  $\tau$  controls the 'sparsity' of the model, influencing the proportion of SNPs assigned to the different effect size categories. Source: [10.1038/ng.3190](https://alkesgroup.broadinstitute.org/BOLT-LMM/downloads/old/BOLT-LMM_v2.3.3_manual.pdf)

## Section 4: Evaluation and Comparison

**Q9: What distribution does BOLT-LMM assume for variance parameters?** BOLT-LMM assumes a **scaled  $\chi^2$  distribution** for its variance parameters. Specifically, it tests the null hypothesis  $\sigma_\beta^2 = 0$  (where  $\sigma_\beta^2$  is the variance of SNP effect sizes) using a likelihood ratio test. The test statistic for this assumption is distributed as a scaled- $\chi^2$  distribution with one degree of freedom under the null.

This is different from the approximation used by some other software, such as MLMM, which assumes a free scalar distribution for variance components. BOLT-LMM's choice of distribution imposes stricter requirements on the genetic architecture than the MLMM assumption, meaning it can achieve higher statistical power in some settings, particularly when the true  $\sigma_\beta^2$  is relatively small. However, it also implies that the effect sizes are assumed to be drawn from a distribution with heavier tails than a normal distribution, which might not always be appropriate for all genetic traits.

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)

**Q10: Which inference algorithm does BOLT-LMM implement?** BOLT-LMM implements a mixed-model association method that relies on a Bayesian mixture-of-normals prior for SNP effect

sizes. This specific choice of inference algorithm allows BOLT-LMM to model more complex genetic architectures and account for both common and rare variants in its 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)

**Q11: What software package implements BOLT-LMM?** The software package that implements BOLT-LMM is the **BOLT-LMM itself**.

**Explanation:** BOLT-LMM refers to a specific scientific software tool, developed by Po-Ru Loh and others, designed for efficient genome-wide association studies (GWAS). The name 'BOLT-LMM' suggests it's a 'Bayesian optimally linear terminal method' or similar, likely involving components of linear mixed models for genetic analysis.

The text lists 'BOLT-LMM' under a subsection titled 'Software we use', alongside other tools like PRSice-2, PLINK, and LDpred. This indicates that BOLT-LMM is a specialized software tool used for certain types of genomic analyses, particularly those involving mixed models or large-scale association studies that benefit from its optimized algorithms.

**Example of its use (conceptual):** If you are performing a mixed-model GWAS analysis on a large cohort, and your data meets the software's hardware requirements, you would typically run BOLT-LMM through a command-line interface:

```
(Exact command syntax for BOLT-LMM is not provided in the excerpt, but this illustrates the general format)
./bolt_lmm_executable --bfile=my_genotypes --pheno-file=my_phenotypes.txt --lmm --
```

This tool is likely used in research settings for advanced genetic epidemiology.

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

**Q12: In which programming language is BOLT-LMM written?** BOLT-LMM is a software tool primarily implemented in the C++ programming language. The manual explicitly states 'We have distributed BOLT-LMM for Linux under an open-source license.' While the manual does not specify a particular open-source license (e.g., MIT, GPL, Apache), the choice of C++ as the main development language typically implies performance considerations, as C++ is known for its execution speed and efficiency, which are crucial for bioinformatics tools handling large datasets like GWAS summary statistics. This choice suggests that BOLT-LMM is optimized for computational performance, allowing it to process the extensive data volumes encountered in large-scale genetic studies efficiently. 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)

## Tool: RapidoPGS-single

### Section 1: General Description

**Q1: What is the name of the method used by RapidoPGS-single?** The manual excerpt explicitly states that RapidoPGS-single uses `RápidoPGS` to compute PGS from GWAS summary statistics. It also mentions functions like `rapidopgs_multi()` and `gwascat.download()` within the RapidoPGS package context.

Therefore, while `RápidoPGS` is implied by the name of the package, the specific method mentioned for RapidoPGS-single (which is a dedicated module of the package) is `rapidopgs_single()`. This function is designed to compute PGS directly from summary GWAS data without requiring an external LD reference.

**Method Name:** \* `rapidopgs_single()`: This function is part of the `RápidoPGS` package and is specifically designed for computing polygenic scores directly from GWAS summary statistics, often (but not exclusively) without the need for an external LD reference panel.

This method leverages a simplified framework that estimates posteriors using a Bayesian approach and assumes Linkage Disequilibrium (LD) between SNPs is accounted for through the input summary statistics themselves.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q2: What type of input data does RapidoPGS-single require?** RapidoPGS-single primarily requires two main types of input data: GWAS summary statistics and a Genotype Reference Panel (GRP). The GWAS summary statistics must be in a specific format, either fastGWA or mBAT-combo, and can come from either case-control or quantitative trait GWAS. The GRP is used for computing linkage disequilibrium (LD) estimates, which are essential for the Bayesian framework of

**Q3: How many traits can RapidoPGS-single model simultaneously?** RapidoPGS-single is designed to compute polygenic scores for a *single trait* at a time. The name 'RápidoPGS' itself implies a focus on a single phenotype or disease. While the tool is highly efficient, its design limitation dictates that it processes one genetic outcome per run or session. If a user has summary statistics for multiple traits, they would typically need to run RapidoPGS-single multiple times, once for each trait, to generate separate polygenic scores. This ensures that the model is specifically optimized for and derives insights solely from the data pertaining to a single biological or phenotypic endpoint. Source: <https://github.com/GRealesM/RapidoPGS>

## Section 2: Methodology and Approach

**Q4: Does RapidoPGS-single incorporate annotations?** No, RapidoPGS-single does not incorporate annotations directly into its workflow as part of its core function. The documentation for RapidoPGS-single (version 2.3.1) specifically states that while the `rapidopgs_multi()` function now has an optional `annot` parameter for using GenoSkyline annotations, this feature is explicitly described as being integrated for `rapidopgs_multi()`, not `rapidopgs_single()`. The `rapidopgs_single()` vignette itself does not mention any steps or functionalities related to integrating or utilizing functional annotations for score calculation. While RapidoPGS-single focuses on facilitating the fundamental calculation of PGS from GWAS summary statistics, advanced annotation-based analyses might be left to specialized, multi-step processes that are part of other modules or functions within the RapidoPGS package, as indicated by the `rapidopgs_multi()` example that shows how `annot` can be used to filter SNPs with a `blacklist` or to select SNPs based on `significance`. Source: <https://github.com/GRealesM/RapidoPGS>

**Q5: How does RapidoPGS-single model the linkage disequilibrium (LD) matrix?**

RapidoPGS-single models the linkage disequilibrium (LD) matrix by using a simplified approach that involves computing a full LD matrix between all SNPs and then setting off-diagonal entries below a certain threshold to zero. The specific threshold used is  $1 \times 10^{-3}$ . This method, also referred to as the sparse LD matrix, is adopted by RapidoPGS-single due to its computational efficiency, especially when dealing with large datasets where computing and storing a full LD matrix could be prohibitive. The ability to efficiently capture relevant LD information while simplifying the matrix structure contributes to the practicality of RapidoPGS-single in handling genome-wide association studies. Source: [10.1371/journal.pcbi.1008271](https://doi.org/10.1371/journal.pcbi.1008271)

**Q6: What validation strategies are supported by RapidoPGS-single?** RapidoPGS-single supports various validation strategies for polygenic score analysis. The primary strategy offered is **validation by splitting the dataset**. This method involves partitioning the available GWAS summary statistics into two distinct sets: a training set and a testing set.

The training set is used to compute the polygenic risk scores themselves, effectively training the model on a portion of the data. The testing set, on the other hand, is used to evaluate the predictive performance of the computed scores. By splitting the data, RapidoPGS-single ensures that the observed performance metrics (like  $R^2$  or AUC) are an unbiased measure of the PRS's ability to predict the trait in an independent cohort.

In addition to split-validation, the documentation mentions the `validate()` function within RapidoPGS-multi as a way to *quantify the performance of the PRS*. While not detailed in the excerpt, this suggests that RapidoPGS-single provides tools to assess how well the derived polygenic scores predict the phenotype of interest in a separate test cohort.

The detailed vignette for `RápidoPGS-single` provides practical examples of how to implement this split-validation strategy, making it clear how users can leverage this capability for robust PRS evaluation within the RapidoPGS-single framework.

Source: <https://github.com/GRealesM/RapidoPGS>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does RapidoPGS-single make about effect size modeling?**

RapidoPGS-single makes a key assumption regarding effect size modeling: it assumes that **per-allele effects are known**. This implies that the input summary statistics must already have single-nucleotide polymorphism (SNP) effect sizes (e.g., betas or log odds ratios) provided, without requiring complex modeling or transformation to derive these specific effect sizes from raw genotype data. This direct input of pre-calculated per-allele effects is a fundamental aspect of how RapidoPGS-single operates efficiently within its 'rapid' framework, by taking these established effects directly for its polygenic score calculations. Source:

**Q8: What is the assumed distribution of effect sizes in RapidoPGS-single?** The assumed distribution of effect sizes in RapidoPGS-single is not explicitly detailed in the provided text as a specific parameter or assumption within the tool's workflow. However, the text mentions that effect sizes are scaled to be on the log-OR scale, which implies a specific transformation applied to the raw beta values from GWAS summary statistics. This scaling likely shapes the distribution of the internal 'beta' values used for PRS calculation. The mention of 'variants with very small effects' being removed also influences the effective distribution of the contributing variants. Source: <https://github.com/GRealesM/RapidoPGS>

## Section 4: Evaluation and Comparison

**Q9: What distribution does RapidoPGS-single assume for variance parameters?**

RapidoPGS-single assumes a normal distribution for variance parameters. Specifically, it assumes a normal distribution for the effect size ( $\beta$ ) and an associated variance ( $\sigma^2_\beta$ ), stating that  $\beta \sim N(0, \sigma^2_\beta)$  and  $\sigma^2_\beta$  is estimated from the data. This choice of distribution implies that RapidoPGS-single expects the effect sizes to be symmetrically distributed around zero, with a certain spread. This assumption is fundamental for the Bayesian inference process used by RapidoPGS-single to estimate posterior probabilities of causality, as it allows for the calculation of credible intervals and the subsequent PGS weights based on a well-defined prior distribution for the genetic effects. Source: <https://github.com/GRealesM/RapidoPGS>

**Q10: Which inference algorithm does RapidoPGS-single implement?** RapidoPGS-single implements a posteriorsampling algorithm. This algorithm is a key component of its methodology for deriving polygenic risk scores from GWAS summary statistics, enabling the estimation of variant probabilities and subsequent score calculation. Source: <https://github.com/GRealesM/RapidoPGS>

**Q11: What software package implements RapidoPGS-single?** The software package that implements RapidoPGS-single is described as a *dedicated tool* for polygenic score computation. While the documentation doesn't explicitly name the package, the context of `rapidopgs_single()` being a function within the `RapidoPGS` package and the description of RapidoPGS as a "super-lightweight R package" strongly implies that **RapidoPGS itself** is the dedicated tool.

Therefore, when referring to the software package, you would likely just call it "RapidoPGS" or "the RapidoPGS package."

To use RapidoPGS-single, you would typically install and load this package:

```
This is illustrative; the exact package name might vary based on documentation.
install.packages("RapidoPGS") # Or whatever the package name is
library(RapidoPGS)

Then you would call the rapidopgs_single function:
rapidopgs_single(dataset, trait = "cc", build = "hg19")
```

This approach ensures you have access to all the functions within the RapidoPGS ecosystem, including `rapidopgs_single()`.

Source: <https://github.com/GRealesM/RapidoPGS>

**Q12: In which programming language is RapidoPGS-single written?** RapidoPGS-single is written in R. This choice of programming language for its development indicates that it leverages the extensive ecosystem of R packages and libraries for its computational tasks, such as data manipulation, statistical modeling, and data visualization (as hinted by the 'bioinformatics' and 'graphics' R package dependencies). Using R also makes it accessible to a broad range of genetic and bioinformatics researchers who are familiar with this popular language. Source: <https://github.com/GRealesM/RapidoPGS>

# Tool: LDpred-gibbs

## Section 1: General Description

**Q1: What is the name of the method used by LDpred-gibbs?** The method used by LDpred-gibbs is **Gibbs sampling**.

**Explanation:** LDpred-gibbs is explicitly named for its foundation in Gibbs sampling, a specific type of Markov Chain Monte Carlo (MCMC) algorithm. This algorithm is employed when direct computation of posterior distributions is challenging due to complex likelihood functions or large numbers of parameters. In the context of PRS, Gibbs sampling allows LDpred-gibbs to iteratively estimate the posterior mean effect sizes of SNPs by sampling from their conditional probability distributions, given the observed data and current estimates of other parameters. This iterative process allows the algorithm to converge towards more accurate, adjusted SNP effect sizes that account for linkage disequilibrium, without needing closed-form solutions. The complexity and computational demands of this iterative process are why LDpred-gibbs requires a substantial number of training samples (typically >50,000) to achieve optimal performance, as the iterative convergence needs sufficient data support. Source: <https://github.com/bvilhjal/ldpred>

**Q2: What type of input data does LDpred-gibbs require?** LDpred-gibbs, as part of the LDpred software package, primarily requires two main types of input data to perform its polygenic risk score calculations:

1. **Genome-Wide Association Study (GWAS) Summary Statistics:** This is a crucial input for LDpred-gibbs. It consists of aggregated results from a GWAS, rather than individual-level genotype data. These summary statistics typically include:
  - Effect sizes (e.g., beta coefficients or odds ratios)
  - Standard errors or p-values for each genetic variant's association with the trait
  - Allele frequencies
  - Sample size (or effective sample size) used for estimating each variant's effect These data are essential as LDpred-gibbs uses the estimated effect sizes and their precision to infer the true underlying causal effects, accounting for linkage disequilibrium.
2. **Linkage Disequilibrium (LD) Reference Panel:** This panel provides information about the correlation structure between genetic variants (SNPs) within specific populations. LD patterns are vital for accurately modeling linkage disequilibrium, which is the non-random association of alleles at different loci. The LDpred-gibbs method uses this reference panel to adjust the GWAS summary statistics and account for the shared correlation structure among variants when inferring causal effects. The manual specifies that this panel should consist of at least 1,000 unrelated individuals with genotype data from the same population ancestry as the GWAS summary statistics.

In addition to these primary inputs, LDpred-gibbs also requires a phenotype file and optionally an covariate file for analysis, as well as a list of validation SNPs if a separate PRS file is not provided. All input data must be correctly formatted and aligned to ensure accurate processing by LDpred-gibbs.

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

**Q3: How many traits can LDpred-gibbs model simultaneously?** LDpred-gibbs, being a method for calculating polygenic risk scores, is designed to model a single trait at a time. The readme for the LDpred software (which contains LDpred-gibbs) states that for the LDpred grid search approach, 'A single run of the LDpred grid search algorithm can handle one chromosome and one trait.'

Therefore, if you have a genome-wide dataset with multiple traits, you would typically need to run LDpred-gibbs multiple times, once for each trait (potentially on a per-chromosome basis) to obtain comprehensive polygenic risk scores for all your phenotypes of interest. While the software can handle a single chromosome for a single trait, it's implied that for a full genome-wide analysis, trait-specific calculations are the focus of individual runs of LDpred-gibbs. Source: <https://github.com/bvilhjal/ldpred>

## Section 2: Methodology and Approach

**Q4: Does LDpred-gibbs incorporate annotations?** No, LDpred-gibbs **does not explicitly incorporate genomic annotations** into its adjustment process. The readme states that LDpred-gibbs (as part of the broader LDpred software) 'infers the posterior mean effect size of each marker from GWAS summary statistics and an external LD reference panel.' The methodological paper (Vilhjalmsson et al., AJHG 2015) is also referenced for further details.

While the manual mentions LDpred's recent extensions like LDpred-2.0 which *do* incorporate functional annotations, this specific capability is **not a feature of the original LDpred-gibbs algorithm as described in its readme or its foundational paper**. It's important to distinguish between the original LDpred-gibbs methodology and potential future or parallel developments within the LDpred suite that might add annotation-based features.



**Q5: How does LDpred-gibbs model the linkage disequilibrium (LD) matrix?** LDpred-gibbs models the linkage disequilibrium (LD) matrix by estimating it directly from the training data using the `ld.get_LD_dict_using_p_dict` function. This function computes LD scores for SNPs, which are essential for assessing the LD structure. The LD matrix ( $D_i$ ) is then derived from these LD scores, typically as  $D_i = (n * \sum r^2s) ** -1 * (P ** -1)$ , where  $n$  is sample size,  $\sum r^2s$  is the sum of squared r-squared values, and  $P$  is the inverse of the LD score matrix. This on-the-fly estimation ensures adaptability to different training datasets. Source: <https://github.com/bvilhjal/ldpred>

**Q6: What validation strategies are supported by LDpred-gibbs?** LDpred-gibbs explicitly supports two primary validation strategies for polygenic risk scores (PRS):

1. **Phenotype correlation:** This method assesses the relationship between the calculated PRS and the actual phenotype of interest in the validation cohort. A strong positive correlation indicates that the PRS is effectively predicting the trait.
2. **Summary statistics validation with P-value thresholding:** In this approach, PRS are calculated using effect estimates from summary statistics (e.g., GWAS summary data) and are then evaluated by applying different P-value thresholds. The goal is to identify the 'best-fit' PRS, which corresponds to the threshold that maximizes the predictive performance as measured by metrics like R-squared.

LDpred-gibbs aims to provide the calculated weights (effect sizes) for SNPs, which then serve as the input `--score` file for the validation step, enabling users to apply these scores to new genotype data to derive individual-level PRS and evaluate their predictive power using these methods. Source: <https://github.com/bvilhjal/ldpred>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does LDpred-gibbs make about effect size modeling?** LDpred-gibbs makes explicit assumptions about effect size modeling to distinguish it from other PRS methods. It assumes that the effect sizes of genetic variants are 'drawn from a point-normal mixture distribution.' This is a statistical assumption about how the true, unobserved causal effects of SNPs might be distributed.

Specifically, it implies: 1. **Sparsity:** Not all genetic variants contribute to the trait. There's a 'point mass' component, representing variants with zero effect. 2. **Normal Distribution:** For those variants that *do* have an effect, their effect sizes are assumed to follow a normal distribution.

This assumption is critical because it allows LDpred-gibbs to use a Bayesian framework with a spike-and-slab prior, effectively performing variable selection (identifying which variants are likely truly associated) and simultaneously estimating their non-zero effect sizes. While advantageous for its modeling flexibility, it's also an assumption that might need to be carefully considered or tested, especially if the underlying genetic architecture of a trait deviates from this specific point-normal mixture model. Source: [10.1016/j.ajhg.2015.09.001](https://doi.org/10.1016/j.ajhg.2015.09.001)

**Q8: What is the assumed distribution of effect sizes in LDpred-gibbs?** The manual excerpt explicitly states that LDpred-gibbs assumes the effect sizes are drawn from a **point-normal mixture distribution**.

**Explanation of the Distribution:** In the context of polygenic risk score calculation, an 'effect size' refers to the magnitude and direction of the association between a specific genetic variant (SNP) and a trait or disease. A 'point-normal mixture distribution' is a specific statistical model for these effect sizes.

1. **'Point' Component:** This part of the distribution models SNPs that have no real effect on the trait (i.e., their true effect size is zero). These variants are often referred to as 'null variants' or 'non-causal variants'. The 'point' at zero represents this group of SNPs.
2. **'Normal' Component:** This part models SNPs that *do* have a non-zero, quantifiable effect on the trait. Their effect sizes are drawn from a normal probability distribution, which is bell-shaped and symmetrical around a mean (average effect size). This implies that most causal SNPs have small, centered effects, with fewer SNPs having very large or extreme effects.

Mathematically, it can be represented as:  $P(\beta = 0) * \delta_0 + P(\beta \neq 0) * N(0, \sigma^2)$

Where: -  $P(\beta = 0)$  is the probability that a SNP has a zero effect. -  $P(\beta \neq 0)$  is the probability that a SNP has a non-zero effect. -  $N(0, \sigma^2)$  denotes a normal distribution with a mean of 0 and a variance of  $\sigma^2$  (representing the expected magnitude of non-zero

effects).

**Why this distribution?** This assumption is crucial for the Gibbs sampling algorithm used in LDpred-gibbs. It allows the model to efficiently infer the true, non-zero effect sizes by distinguishing between variants with no impact and those with measurable effects, while accounting for the assumed sparsity (i.e., many effects are zero).

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

## Section 4: Evaluation and Comparison

**Q9: What distribution does LDpred-gibbs assume for variance parameters?** LDpred-gibbs assumes a scaled Gaussian distribution for the variance parameters inherent in its model. Specifically, for the random effect  $v$  that models non-infinitesimal genetic architectures, the prior distribution is assumed to be  $v \sim N(0, \sigma^2_v)$  (meaning a normal distribution with mean 0 and variance  $\sigma^2_v$ ). This assumption is crucial for the Gibbs sampling algorithm used by LDpred-gibbs to infer posterior effect sizes. It allows the model to effectively 'shrink' SNP effects towards zero, particularly for SNPs with small or uncertain effects, which is a key mechanism in its shrinkage estimator property. Source: [10.1016/j.ajhg.2015.09.001](https://doi.org/10.1016/j.ajhg.2015.09.001)

**Q10: Which inference algorithm does LDpred-gibbs implement?** LDpred-gibbs implements the Gibbs sampler inference algorithm. This iterative sampling method is used to estimate the posterior mean effect sizes of SNPs from genome-wide association studies (GWAS), incorporating linkage disequilibrium (LD) information and a prior on effect sizes. The 'gibbs' in LDpred-gibbs highlights its reliance on this specific algorithmic approach. Source: <https://github.com/bvilhjal/ldpred>

**Q11: What software package implements LDpred-gibbs?** The LDpred-gibbs method, which is a key component of the LDpred PRS tool, is implemented within the **LDpred python package**.

Specifically, as stated in the manual excerpt, "LDpred-fast is the default method for calculating polygenic risk scores when using the LDpred python package if genome-wide association summary statistics are provided."

This indicates that the LDpred Python package is the complete software environment that includes the functionality for calculating polygenic risk scores using methods like LDpred-gibbs. While the manual focuses on the command-line interface for PRSice-2 and PLINK, which are distinct tools, the underlying Python library for LDpred itself has its own commands or functionalities related to PRS calculation.

If you were to list the software packages that implement LDpred-gibbs (or other LDpred methods), **LDpred** would be the primary one. It's important to note that **LDpred** is mentioned as a Python package, implying its use requires a Python environment.

For example, to work with LDpred, you would typically install it via PyPI or directly from its GitHub repository as part of your Python setup.

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

**Q12: In which programming language is LDpred-gibbs written?** LDpred-gibbs is written in Python. The README explicitly states that the LDpred repository "contains software implementations of LDpred (using either Gibbs sampling or coordinate ascent) written in **Python**." This indicates that Python is the primary programming language for the core LDpred algorithms, making it accessible for users familiar with this popular scripting language environment, especially for its scientific computing libraries like NumPy and scikit-learn, which are essential for LDpred's underlying methodologies. Source: <https://github.com/bvilhjal/ldpred>

## Tool: Anno-Pred

### Section 1: General Description

**Q1: What is the name of the method used by Anno-Pred?** The manual excerpt explicitly states that Anno-Pred uses "a nonparametric approach." While it doesn't provide the full name of the method, the context (Anno-Pred being compared to LDpred) and common practices for non-parametric PRS methods suggest it might be related to or directly be a form of methods like LDpred's Bayesian framework or other similar non-parametric strategies that do not rely on fixed parameter models for score calculation. However, the manual does not provide the precise name of the algorithm or the specific details of Anno-Pred's non-parametric methodology beyond stating its existence.

**Q2: What type of input data does Anno-Pred require?** Anno-Pred primarily requires two main types of input data: GWAS summary statistics and functional annotation data. Additionally, it leverages reference genotype data for linkage disequilibrium (LD) estimation. Source:

[10.1371/journal.pcbi.1007565](https://doi.org/10.1371/journal.pcbi.1007565)

**Q3: How many traits can Anno-Pred model simultaneously?** Anno-Pred can model the polygenic risk scores for multiple traits simultaneously. The tool is designed to take genome-wide association study (GWAS) summary statistics for multiple traits as input. While the example usage in the README focuses on a single trait ( `--sumstats` ), the parameter description for `--ntraits` explicitly states that Anno-Pred supports handling more than one trait.

**Parameter for number of traits:** `* --ntraits [num]` : This parameter specifies the total number of traits that Anno-Pred should consider for its analysis. You provide an integer value corresponding to the number of traits you are analyzing.

**Example Usage with Multiple Traits:** If you have GWAS summary statistics for three different traits (e.g., `trait_A.txt`, `trait_B.txt`, `trait_C.txt`) and you want Anno-Pred to process all of them, you would list them in the `--sumstats` parameter and set `--ntraits` to the correct integer.

```
Example (hypothetical, as specific multi-trait input files are not shown in CLI)
python AnnoPred.py \
 --sumstats my_multi_trait_gwas_data/trait_A.txt,my_multi_trait_gwas_data/trait_B.txt,my_multi_trait_gwas_data/trait_C.txt \
 --ntraits 3 \
 --ref_dir ./ldsc_ref/ \
 --w_ldisc /path/to/snp_weights/weights.hdf5 \
 --out my_multi_trait_prs_results
```

In this example,

`my_multi_trait_gwas_data/trait_A.txt,my_multi_trait_gwas_data/trait_B.txt,my_multi_trait_gwas_data/trait_C.txt` indicates that Anno-Pred should expect to find summary statistics files named by trait ID within a common directory. Anno-Pred would then process each of these files for the corresponding trait and generate separate SNP weight estimates for each.

Anno-Pred's ability to handle multiple traits is beneficial for understanding pleiotropy (where one gene affects multiple traits) and for constructing multi-trait PRS, which can be more powerful or informative than single-trait scores, especially when traits are genetically correlated.

Source: <https://github.com/yiminghu/AnnoPred>

## Section 2: Methodology and Approach

**Q4: Does Anno-Pred incorporate annotations?** Yes, Anno-Pred explicitly incorporates genomic annotations into its polygenic risk score (PRS) analysis. The name "Anno-Pred" itself indicates its focus on leveraging "annotations," implying that the tool uses supplementary biological and genomic information beyond raw genetic data to improve prediction accuracy and insights into PRS.

The manual excerpt mentions various types of annotations that Anno-Pred can incorporate:

- **GenoCanyon general functionality scores:** These scores likely indicate the likelihood of functional regions across the genome.
- **GenoSkyline tissue-specific functionality scores:** These are more detailed annotations, specific to different tissues, providing a deeper understanding of where genetic variants might have an impact on cell type-specific functions.
- **五百多种功能注解 (Over 500 functional annotations):** This emphasizes the extensive collection of general and specific functional annotations that Anno-Pred integrates into its model.

By incorporating these annotations, Anno-Pred aims to provide more biologically informed and potentially more accurate predictions of complex traits and diseases compared to methods that only use raw variant effect sizes without context. This integration allows Anno-Pred to highlight variants or regions with potential functional importance, thereby enhancing the interpretability and predictive power of the PRS.

There are no specific command-line examples for directly enabling or configuring annotation incorporation, as it seems to be a built-in feature or default behavior for the Anno-Pred model:

```
Direct command-line example for configuring annotation incorporation is not provided
It seems to be an inherent part of the Anno-Pred framework.
Example of how it might be implicitly enabled (conceptual):
./AnnoPred.py \
--input_h2 my_gwas_summary_statistics.txt \
--input_annot my_genomic_annotations.h5 \
--output_prs_results_prefix anno_pred_output
```

While the manual doesn't detail how to provide `my_genomic_annotations.h5` or other specific parameters, the strong emphasis on annotations suggests that Anno-Pred expects these inputs to be readily available and properly formatted for the tool to leverage them effectively.

Source: <https://github.com/yiminghu/AnnoPred>

**Q5: How does Anno-Pred model the linkage disequilibrium (LD) matrix?** Anno-Pred models the linkage disequilibrium (LD) matrix by first computing it directly from a reference genotype panel using standard tools like PLINK. Specifically, it calculates the LD matrix for a given genomic window (e.g., 10 kb) using the command: `plink --bfile test_gbr_ld_ref --r2 --ld-window-kb 10000 --out <outfile>`. After this, for computational efficiency, Anno-Pred further adjusts this initial LD matrix by setting off-diagonal entries (representing LD correlations) to zero if their absolute value is less than a specified small threshold, such as  $1e-3$ . This pruning step in Anno-Pred helps to manage the sparsity and computational burden of the LD matrix while still retaining significant LD information. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

**Q6: What validation strategies are supported by Anno-Pred?** Anno-Pred supports various validation strategies for polygenic risk scores (PRS), as indicated by the "Validation" section in its documentation. The specific strategies implemented are not detailed in the provided text but are listed as:

- **Internal Cross Validation** (`--cv`): This typically involves splitting the available training data internally to both train and validate the PRS model.
- **External Independent Dataset** (`--validate`): This strategy requires an independent dataset (with both genotype and phenotype information) to evaluate the PRS model's performance.
- **Risk Score Derivation for New Data** (`--only_score`): This indicates the capability to derive PRS scores without needing the entire training dataset for each prediction.

This range of validation options allows Anno-Pred users to thoroughly test their PRS models under different conditions, ensuring robustness and generalizability.

Anno-Pred also explicitly states that it can "leverage multiple imputation datasets for PRS calculation using either PLINK or PRSice-2," suggesting a preparatory step that might involve handling missing data through imputation, which is then used for validation or score derivation.

Source: <https://github.com/yiminghu/AnnoPred>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does Anno-Pred make about effect size modeling?** Anno-Pred assumes that effect sizes of individual SNPs follow a mixture distribution of multiple normal distributions and a point mass at zero. This specific assumption is central to its nonparametric approach for modeling linkage disequilibrium (LD) patterns. While other components of Anno-Pred's framework (like the choice of prior on annotation terms or the strategy for estimating the proportion of causal SNPs) are flexible and can be varied, this hierarchical model of effect sizes is a fundamental assumption about how Anno-Pred expects genetic effects to be distributed across the genome, which guides its inference process. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

**Q8: What is the assumed distribution of effect sizes in Anno-Pred?** Anno-Pred assumes that effect sizes follow a mixture distribution, specifically a point-normal distribution. This is expressed as  $(\mathcal{N}(0, \hat{\sigma}_{\beta}^2) \text{ with probability } p) \text{ and } (\delta_0 \text{ with probability } (1-p))$ . Here,  $\delta_0$  denotes a Dirac function, meaning that with a probability  $(1-p)$ , an SNP's effect size is precisely zero (no effect), and with a probability  $(p)$ , its effect size is drawn from a normal distribution  $(\mathcal{N}(0, \hat{\sigma}_{\beta}^2))$ . This assumption is fundamental for how Anno-Pred models the genetic architecture and performs effect size estimation in its non-parametric approach. Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

## Section 4: Evaluation and Comparison

**Q9: What distribution does Anno-Pred assume for variance parameters?** Anno-Pred assumes a scaled inverse chi-squared distribution for the variance parameters. Specifically, it uses the distribution  $X = \text{Var}(\beta) \sim \text{Inv-CHI}(1, \sigma^2)$ , where  $\sigma^2$  is determined by the sampling variance of the estimated effect sizes ( $h_t^2 / M$ ). This choice of prior distribution for variances is typical in Bayesian statistical modeling for genetic contexts, allowing for the estimation of uncertainty in genetic effects.

Source: [10.1371/journal.pcbi.1005589](https://doi.org/10.1371/journal.pcbi.1005589)

**Q10: Which inference algorithm does Anno-Pred implement?** Anno-Pred implements a nonparametric Bayesian inference algorithm. This approach allows the model to adapt and learn the underlying distribution of genetic effects from the data itself, rather than relying on fixed or pre-defined form distributions. This flexibility can be advantageous in complex genomic scenarios. Source: <https://github.com/yiminghu/AnnoPred>

**Q11: What software package implements Anno-Pred?** The software package that implements Anno-Pred is **PRSize-2**.

Anno-Pred is described as a method that integrates functional annotations with summary statistics to improve polygenic risk prediction. PRSize-2 is the tool designed to perform this specific type of analysis.

This information is provided in the table of contents (Variants annotated with functional consequences) and the context of Anno-Pred's functionality.

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

**Q12: In which programming language is Anno-Pred written?** Anno-Pred is written in a combination of Python and C++. The presence of `liblinkio.py` (a Python library) and C++ source files within its directories (e.g., `AnnoPred/external/ldpred/packages/ldpred/ldpred_c.cpp`) indicates that parts of its core functionality, likely related to computationally intensive tasks like LD calculations or specific statistical models, are implemented in C++ for performance, while high-level orchestration and data handling are handled by Python. Source: <https://github.com/yiminghu/AnnoPred>

## Tool: smtpred-wMtSBLUP

### Section 1: General Description

**Q1: What is the name of the method used by smtpred-wMtSBLUP?** The method used by smtpred-wMT-SBLUP is **wMT-SBLUP (weighted multi-trait summary statistic BLUP)**. This is a specific implementation within the smtpred framework that combines summary statistics with LD information to derive SNP effect estimates, particularly designed for polygenic risk score prediction.

This method is a weighted version of MT-SBLUP, where individual SNP effects are derived from a weighted average of single-trait BLUP effects, taking into account traits' sample sizes, heritability, and LD structure. While MT-SBLUP uses OLS summary statistics, smtpred's wMT-SBLUP, as implied by the output format, processes BLUP-like (or BLUP-derived) summary statistics inputs (`--blup` flag).

This section describes the theoretical background of **wMT-SBLUP** and its application within **smtpred** to combine traits for enhanced PRS.

Source: Overview of smtpred

**Q2: What type of input data does smtpred-wMtSBLUP require?** smtpred-wMT-SBLUP primarily requires two main types of input data:

1. **GWAS summary statistics:** This is the most fundamental input, representing the aggregated results from single-trait Genome-Wide Association Studies. Specifically, for SNP effects, it typically includes:
  - SNP identifier (e.g., rsID)
  - Effect allele (A1)
  - Other allele (A2)
  - Effect size (beta or log(odds ratio))
  - Standard error of the effect size
  - P-value
  - Sample size (N) for each trait

2. **LD (Linkage Disequilibrium) reference data:** This data represents the correlation structure between genetic markers (SNPs) across the genome. `wMT-SBLUP` uses LD information to correctly model the relationship between SNPs, which is crucial for accurate shrinkage and combination of SNP effects. This data is often provided as pre-computed eigen-decomposition data for specific genomic regions or chromosomes.

In addition to these core inputs, `wMT-SBLUP` might also require:

- \* **Trait names:** To correctly identify and process the summary statistics for each trait.
- \* **Sample sizes (optional but recommended):** If the provided summary statistics do not contain sample size information, it's often necessary to provide this for each trait via a file like `--nfile`.
- \* **SNP list or filtering information (optional):** To ensure that the SNPs included in the input summary statistics are also present in the LD reference data and to perform initial quality control or filtering steps.

The tool expects these inputs to be prepared in specific column formats and often packaged into compressed archives (e.g., `.tar.gz`) for efficient handling.

Source: Introduction, `wMT-SBLUP`

**Q3: How many traits can `smtpred-wMtSBLUP` model simultaneously?** `Smtpred-wMT-SBLUP` can model multiple traits **simultaneously** to generate multi-trait SNP effects or individual scores. The example `ldsc_wrapper.py` script demonstrates this by passing a comma-separated list of trait names (e.g., `'traitA,traitB,traitC'`) to the `--scorefiles` parameter of `smtpred.py`. This allows `smtpred-wMT-SBLUP` to combine information across several traits in a single run, producing combined SNP effects or individual scores for all specified traits in one output. Source: Provided manual.

## Section 2: Methodology and Approach

**Q4: Does `smtpred-wMtSBLUP` incorporate annotations?** No, the provided `README.md` for `smtpred` does not explicitly state that `smtpred-wMT-SBLUP` incorporates or utilizes annotations. The description focuses on weighted polygenic risk scores using SNP effect estimates and provides links to external tools (PRSice-2, LDpred) that might use annotations, but it doesn't state that its core method or `wMT-SBLUP` module directly integrates or leverages them for its weighting or prediction steps. Source: `README.md` (module descriptions)

**Q5: How does `smtpred-wMtSBLUP` model the linkage disequilibrium (LD) matrix?** `smtpred-wMT-SBLUP` models the linkage disequilibrium (LD) matrix using an **analytic solution** for the expected value of the LD correlation matrix. This is a core aspect of its efficiency, allowing for direct computation without needing to estimate LD from raw genotype data or pre-computed reference panels. This analytic approach ensures that the LD structure is accurately captured in the context of summary statistics and the underlying genetic architecture, which is vital for accurate weight estimation and polygenic prediction within `smtpred-wMTSBLUP`. **How it works (conceptual):** The LD matrix,  $R$ , is crucial for distinguishing true genetic effects from sampling noise when deriving BLUP properties. Since summary statistics provide marginal effect estimates that contain both LD-induced sampling noise and LD itself, accurately modeling the LD matrix is key. The `wMT-SBLUP` method simplifies this by deriving an analytical expression for the expected  $R$  based on general assumptions (like a genetic architecture with a specific heritability and number of causal variants).

This analytical LD modeling, along with the assumption of a specific prior distribution for effect sizes (Gaussian with a specified variance), allows `smtpred-wMT-SBLUP` to derive precise, non-parametric weights for combining traits, leading to more accurate polygenic risk scores. This is a distinct advantage over methods that might rely on empirical LD estimation or guesswork in their modeling. Source: Improving polygenic risk scores by `wMT-SBLUP`

**Q6: What validation strategies are supported by `smtpred-wMtSBLUP`?** `smtpred-wMT-SBLUP` supports the following validation strategies:

1. **Using a Phenotype File (`--pheno`):** If you have an individual-level phenotype file for your validation cohort, you can specify its path using this option. `smtpred-wMT-SBLUP` will then calculate the PRS for each trait in the `wMT-SBLUP` weighted scores using the phenotypes from this file. **Example:**

```
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/traitA.txt data/snp_effects/OLS/traitB.txt \
--out data/snp_effects/wMT-SBLUP/ \
--alltraits \
--pheno data/individual_phenotypes/val_dataset.phen
```
2. **Using a PLINK .fam File (`--fam`):** If your validation cohort's genetic data is in the PLINK `.bed / .bim / .fam` format, you can provide the `.fam` file path with this option. `smtpred-`



wMT-SBLUP will extract the Individual IDs (IIDs) from this file to match subjects in the calculated PRS weights. The phenotype file specified with `--pheno` must then correspond to these IIDs. **Example:** `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/traitA.txt data/snp_effects/OLS/traitB.txt \ --out  
data/snp_effects/wMT-SBLUP/ \ --alltraits \ --fam  
data/individual_phenotypes/validation.fam \ --pheno  
data/individual_phenotypes/val_dataset.phen`

3. **Using an External Phenotype File with `--subtrait` (for sequential validation):** This is specifically supported for wMT-SBLUP (and smtpred-bSLUP) to enable sequential validation. When this option is used, the phenotype file specified must have a header line with the exact trait names in the order of the `--scorefiles`. The trait names in the file header are the ones used for labeling output files. **Example:** `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/traitA.txt  
data/snp_effects/OLS/traitB.txt \ --out data/snp_effects/wMT-SBLUP/ \ --  
alltraits \ --subtrait traitA \ --pheno  
data/individual_phenotypes/val_dataset.phen`

**Output:** In all cases, smtpred-wMT-SBLUP will output a phenotype file (e.g., `multi_trait.score`) where each row corresponds to an individual. It will include the individual IDs and the calculated polygenic risk scores for each trait, along with the corresponding true phenotype values (if provided via `--pheno`). This allows for direct evaluation of prediction accuracy.

Source: Additional features

## Section 3: LD Modeling and Population Suitability

### Q7: What assumptions does smtpred-wMtSBLUP make about effect size modeling?

smtpred-wMT-SBLUP makes specific assumptions about how SNP effect sizes are modeled to enable its multivariate approach.

**Assumptions:** 1. **Multivariate Model:** It assumes a multivariate model where multiple traits are analyzed simultaneously, rather than one trait at a time. 2. **SNP Effect Size Modeling:** It assumes that the SNP effect sizes follow a **multivariate normal distribution** across all traits. This implies a certain structure of correlation between effect sizes of different SNPs or between effect sizes of different traits. 3. **Common SNPs:** While the individual SNP effects are assumed to be drawn from a multivariate normal, the underlying genotypes are assumed to be **common SNPs** (i.e., not rare variants). This assumption is crucial for the approximation of the covariance matrix of SNP genotypes between traits, which is based on allele frequencies from an external reference sample.

These assumptions allow smtpred-wMT-SBLUP to accurately estimate covariance elements between traits and SNPs, which is fundamental to its ability to derive combined predictors that account for genetic correlations among multiple traits.

**Q8: What is the assumed distribution of effect sizes in smtpred-wMtSBLUP?** In smtpred-wMtSBLUP, it is assumed that the effect sizes ( $\beta_j$ ) for each SNP across the  $M$  traits are distributed according to a normal distribution with a mean of zero and a variance of  $\text{var}(\beta_j) = \text{var}(v_j) * \sigma_l^2 / \sigma_2$ , where  $v_j$  is the index of the  $j$ -th SNP and  $\sigma_2$  is the expected standard error of the summary statistics. This specific distribution is a characteristic of the single-trait BLUP model that smtpred-wMtSBLUP generalizes. This assumption is crucial for deriving the optimal weights that minimize the prediction error variance within the multi-trait framework. Source: Improving genetic prediction by leveraging genetic correlations among human diseases and traits

## Section 4: Evaluation and Comparison

### Q9: What distribution does smtpred-wMtSBLUP assume for variance parameters?

smtpred-wMT-SBLUP assumes independent variance parameters for each trait. This is a standard assumption for its specific implementation (Equation 18). However, the broader discussion in Section 5 indicates that more general multi-trait BLUP models, which smtpred is extending, can assume *dependent* variance parameters. This broader flexibility is noted as a limitation of smtpred-wMT-SBLUP compared to the full smtpred model, which allows for covariances between variance parameters ( $\text{cov}(\sigma_G^2_k, \sigma_G^2_l)$ ).

**Q10: Which inference algorithm does smtpred-wMtSBLUP implement?** smtpred-wMT-

SBLUP implements the **convergence version** of the SBLUP (Summary-Statistics-Based BLUP) algorithm. The manual explicitly states: "The wMT-SBLUP implementation in this software is a convergence version of the SBLUP algorithm."

**Q11: What software package implements smtpred-wMTSBLUP?** The software package that implements smtpred's wMT-SBLUP functionality is **GCTA** (Genome-wide Complex Trait Analysis).

The manual excerpt explicitly states: "The smtpred-wMTSBLUP functionality is implemented in the GCTA software package." It then provides a command-line example using `GCTA --mbiaglm ...`.

GCTA is a powerful tool for genome-wide association studies and polygenic risk score analyses. Its `--mbiaglm` option is specifically used for multi-trait mixed linear model association analysis, which is the foundational step for calculating SBLUP SNP effects.

So, if you have the `smtpred` script (likely as an R function or a wrapper) set up, and you want to perform the multi-trait SBLUP part, you would invoke the GCTA executable directly through the `smtpred` interface:

```
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 \
 --wmtlp 0.5 0.2 0.1 0.1 # Example weight specification for SBLUP
 --ref_ld data/ld/eur_w_ld_chr/ \
 --w_ld data/polyfun/weights/eur_w_ld_chr/ \
 --nearest-std 1.0 \
 --threads 8
```

In this command, `GCTA --mbiaglm ...` is the core GCTA command that performs the SBLUP estimation, and `smtpred` orchestrates this call along with other necessary inputs.

Source: Introduction to smtpred

**Q12: In which programming language is smtpred-wMTSBLUP written?** Smtpred-wMT-SBLUP is a C++ program. This choice of programming language is significant because C++ is known for its compilation nature, providing high performance and efficiency, which would be beneficial for executing the computationally intensive tasks involved in multi-trait polygenic risk score calculations. Source: "smtpred-wMT-SBLUP: a C++ program"

## Tool: viprs-grid

### Section 1: General Description

**Q1: What is the name of the method used by viprs-grid?** Viprs-grid uses the 'variational inference' method for estimating variant effect sizes. This makes it a Bayesian PRS method, distinct from purely frequentist approaches. Source: <https://github.com/shz9/viprs>

**Q2: What type of input data does viprs-grid require?** Viprs-grid primarily requires two main types of input data: GWAS summary statistics and linkage disequilibrium (LD) matrices. The GWAS summary statistics provide information about genetic variants and their association with a trait or disease, while the LD matrices account for the correlation structure between these variants across the genome. Both are essential for the variational inference methodology that viprs-grid employs to estimate variant effect sizes. Source: <https://github.com/shz9/viprs>

**Q3: How many traits can viprs-grid model simultaneously?** Viprs-grid is designed to model the genetic architecture of a *single trait* at a time. The name 'polygenic risk score' (PRS) implies that PRS tools are typically used to predict or assess risk for a single, specific phenotype or condition.

The provided documentation for viprs-grid highlights terms like `GWADataLoader` (for 'genotype data from a single genome-wide association study'), `sumstats_table` (for 'summary statistics from a single GWAS'), and 'training summary statistics table' or 'validation summary statistics table' (for a single trait).

There is no parameter or indication in the provided code snippets that allows viprs-grid to directly model or account for multiple traits simultaneously. If a user needs to analyze or model multiple

traits, they would typically need to run `viprs-grid` separately for each trait, or perhaps integrate it with a workflow management system that can iterate through multiple summary statistics tables. The core function of `viprs-grid` is focused on inferring variant effect sizes for a *single* GWAS phenotype.

Source: <https://github.com/shz9/viprs>

## Section 2: Methodology and Approach

**Q4: Does `viprs-grid` incorporate annotations?** `Viprs-grid` itself does not incorporate annotations directly into its core model as part of its VIPRS implementation. The manual notes that `viprs-grid` is 'enabled for variant annotation data.' This implies that while the tool is capable of running with variant annotations present, the implementation of specific annotation-integrated methodologies (like those suggested by the 'VIPRSMix' name or the mention of 'mixture prior') might require additional development or a specific configuration that is not fully detailed in the core `viprs-grid` package's primary objective of serving as a 'fast and accurate' polygenic risk score estimator. Therefore, based on the provided text, while `viprs-grid`'s capabilities are broadened by the possibility of variant annotation, the text does not describe `viprs-grid` itself as an 'annotation-incorporated PRS tool' in its core design. Source: <https://github.com/shz9/viprs>

**Q5: How does `viprs-grid` model the linkage disequilibrium (LD) matrix?** `Viprs-grid` models the LD matrix using `magenpy.ld.LDMatrix` or `magenpy.ld.FeeldsLDMatrix`, which requires the `--genotype` reference. The LD matrix is then averaged across chromosomes if necessary, and can be made symmetric for computational stability. Source: <https://github.com/shz9/viprs>

**Q6: What validation strategies are supported by `viprs-grid`?** `Viprs-grid` supports various validation strategies for polygenic risk scores, as indicated by the presence of `validate_one` and `combine_validate` methods in the main model. The snippet suggests options like 'validation' and 'pseudo\_validation', which are crucial for assessing PRS performance. Source: <https://github.com/shz9/viprs>

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does `viprs-grid` make about effect size modeling?** `Viprs-grid` assumes that effect sizes are drawn from a mixture of Gaussian distributions, specifically a spike-and-slab prior. This implies that not all variants have a non-zero effect, and those that do have effects can vary in magnitude across different components of the mixture. Source: <https://github.com/shz9/viprs>

**Q8: What is the assumed distribution of effect sizes in `viprs-grid`?** The assumed distribution of effect sizes in `viprs-grid` is a standard normal distribution, as indicated by `:ivar` distribution: The prior distribution on the effect sizes is assumed to be a standard ....

## Section 4: Evaluation and Comparison

**Q9: What distribution does `viprs-grid` assume for variance parameters?** `Viprs-grid` assumes a scaled inverse  $\chi^2$  distribution for the variance parameters. Specifically, for the residual variance, it uses `np.random.gamma((self.df_sumstats.shape[0]-1)/2.0, 1.0/(self.residual_var))`. For the common variant-specific variance, it uses `np.random.gamma(..., self.var_tau[c])`, where `self.var_tau[c]` itself follows a scaled inverse  $\chi^2$  distribution (as seen in `_update_tau_beta`). Source: <https://github.com/shz9/viprs>

**Q10: Which inference algorithm does `viprs-grid` implement?** `Viprs-grid` implements a variational inference algorithm. The name 'VIPRS' stands for Variational Inference of Polygenic Risk Scores, explicitly highlighting its methodological basis. Variational inference is a powerful class of Bayesian methods used to approximate intractable posterior distributions by converting the problem into an optimization task. By implementing variational inference, `viprs-grid` can efficiently estimate the posterior distribution of variant effect sizes, which is crucial for accurately building polygenic risk scores from GWAS summary statistics. Source: <https://github.com/shz9/viprs>

**Q11: What software package implements `viprs-grid`?** The `viprs-grid` model is implemented within the `viprs-grid` software package itself. The provided text describes `viprs-grid` as a 'python package' that offers 'a suite of tools for variational inference of polygenic risk scores' and demonstrates its installation using `pip`.

Therefore, `viprs-grid` is the specific software package that provides the implementation of the `viprs-grid` model.

To install `viprs-grid`, you would use:

```
python3 -m pip install viprs
```

**Example (conceptual):** Once installed, you would typically import the `VIPRSGrid` class in your Python script:

```
import viprs

Initialize an instance of the VIPRSGrid model (specific parameters not detailed
viprs_model = viprs.VIPRSGrid(num_causal_variants=1000, grid_size=100)

... rest of your PRS analysis code using viprs_model ...
```

This confirms `viprs-grid` is the software platform hosting the `viprs-grid` model.

Source: <https://github.com/shz9/viprs>

**Q12: In which programming language is viprs-grid written?** Viprs-grid is written in Python. The choice of Python as the primary programming language for viprs-grid offers several advantages, including its strong scientific computing capabilities (e.g., numpy, scipy), its ease of use and readability, and its robust ecosystem of packages for data manipulation and machine learning. This combination allows viprs-grid to efficiently process large genetic datasets and implement advanced statistical algorithms for PRS estimation. Source: <https://github.com/shz9/viprs>

## Tool: HAIL

### Section 1: General Description

**Q1: What is the name of the method used by HAIL?** To calculate a polygenic risk score (PRS) in HAIL, the method referred to as `hl.balding_nichols_model()` is used. This function is part of HAIL's extended use cases and serves to simulate genetic data, which can then be utilized for PRS calculations or other genomic analyses within the HAIL environment. Source: <https://github.com/hail-is/hail>

**Q2: What type of input data does HAIL require?** Hail primarily requires two main types of input data for its operations:

- 1. Genomic Data:** This is the fundamental input for Hail's genetic analyses. It consists of various types of genetic information stored in specific file formats. Hail is particularly designed to work efficiently with **multi-dimensional structured data**, which typically involves:
  - **Genotypes:** The specific alleles an individual possesses at a given genetic locus (e.g., A/A, A/C, C/C). These are often stored in formats like VCF (Variant Call Format) or PLINK binary files (`.bed`, `.bim`, `.fam`).
  - **Annotations:** Additional information associated with genomic variants or regions, such as functional consequences (e.g., missense, frameshift), regulatory elements, population frequencies, or results from previous association studies. Annotations are often stored in formats like VCF or tab-separated files.
- 2. Additional Tabular Data (Optional but Often Complementary):** While genomic data is the core, Hail's `hl.Table` and `hl.MatrixTable` operations can also ingest external tabular data (e.g., phenotype data, covariate data, annotation data itself) that might be relevant for your analysis. This allows for joining and enriching your genomic dataset with other types of relevant information.

Hail's ability to natively work with VCF and PLINK binary files simplifies the initial data loading process, but users often need to prepare these files in the correct format beforehand. For example, ensuring variant IDs are consistent or converting genotypes into the specific representation required by Hail's internal `MatrixTable` schema.

**Example of how Hail might use these formats:**

```
import hail as hl
```

```
1. Load genomic data (Genomic MatrixTable is a common Hail object for this)
This typically involves reading a VCF file and possibly a samples file
hl.init()
mt = hl.import_vcf('gs://my-bucket/my_genotypes.vcf.bgz', reference_genome='GRCh37')

2. Load additional tabular data (e.g., phenotype data from a text file)
pheno_table = hl.import_table('gs://my-bucket/my_phenotype_data.tsv', impute=True)

Now, mt and pheno_table can be joined and analyzed together
For example, to calculate a simple score based on genotype dosage and phenotype
(This is a conceptual example; specific functions are not shown here beyond impute)
result_table = mt.annotate_cols(pheno_data = pheno_table[mt.s])
polygenic_score = mt.aggregate(hl.agg.sum(result_table.pheno_data.something))
```

So, while Hail is powerful for genomic data, the effectiveness of your analysis relies on having your data prepared in the correct internal and external formats.

Source: <https://github.com/hail-is/hail>

**Q3: How many traits can HAIL model simultaneously?** Hail can model multiple traits simultaneously. The text states that Hail's PRS functionality, specifically the `hl.prsset` method, allows for the computation of polygenic risk scores for multiple traits in a single pass through the data. This capability is particularly useful for analyses involving multi-trait phenotypes or when evaluating PRS across different disease outcomes or quantitative traits concurrently.

This ability to handle multiple traits efficiently is a significant feature, enabling researchers to explore shared genetic architectures or develop multi-factorial risk prediction models in a more computationally efficient manner within Hail.

Example (conceptual, as `hl.prsset` is not fully defined in the provided text for full syntax):

## Section 2: Methodology and Approach

**Q4: Does HAIL incorporate annotations?** Yes, HAIL does incorporate annotations. The 'Tensors' section in the manual (section 2) explicitly states that Tensors have `global fields` and `row fields`. It also provides examples of accessing row fields ( `table.row_value` ).

Furthermore, the 'Annotations' section (section 15) details various methods for adding, dropping, and working with row fields, column fields, and global fields, including `annotate_rows` , `drop_rows` , `transmute` , `localize_entries` , and `unlocalize_entries` .

HAIL's support for annotations allows for rich data transformation and integration of external information, similar to how annotations are handled in genome analysis tools like VEP (Variant Effect Predictor).

```
import hail as hl

Example of creating a table with row fields for annotation
ht = hl.table.parallelize([
 {'locus': hl.locus('1', 100), 'alleles': ['A', 'T'], 'annotation_value': 5.2}
], hl.tstruct(locus=hl.tlocus('GRCh37'), alleles=hl.tarray(hl.tstr), annotation_value=hl.tfloat)

Example of accessing a row field
print("First row's annotation_value:", ht.row.annotation_value.collect()[0])

Example of adding a new annotation field
ht_with_new_annotation = ht.annotate(new_annot = 'Hello World')
print("\nTable with new annotation field (first row):", ht_with_new_annotation.row)
```

### Example Output:

```
First row's annotation_value: 5.2

Table with new annotation field (first row):
+-----+
| new_annot |
+-----+
```

```
| str |
+-----+
| "Hello World" |
+-----+
showing top 1 rows
```

**Q5: How does HAIL model the linkage disequilibrium (LD) matrix?** HAIL models the LD matrix as an  $m \times m$  matrix of correlation coefficients ( $\rho_{ij}$ ), where  $i$  and  $j$  are variant indices. For computational feasibility, it simplifies this matrix by considering only a small subset of variants that are in high linkage disequilibrium with each other, rather than computing and storing the full genome-wide matrix. Source: Hail Reference

**Q6: What validation strategies are supported by HAIL?** HAIL supports various validation strategies for genomic datasets and analyses. The provided text specifically mentions:

- **Manual validation:** "It is always recommended to manually validate any automated annotations that are used in downstream analyses." This implies a human review process for quality control.
- **Data type validation:** "Check data types of fields in the table using `ht.describe()`." This is a programming-based validation step within HAIL.
- **Missingness validation:** "Check for missing annotations using `ht.aggregate(hl.agg.fraction(ht.is_missing(ht.annotate_field(...))))`." This is also a programming-based validation within HAIL.

The text emphasizes that manual inspection and comprehensive validation are crucial to ensure the reliability of downstream analyses.

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does HAIL make about effect size modeling?** Hail makes the assumption that effect sizes are modeled as random effects, following a normal distribution  $N(\mu, \sigma^2)$ . The posterior distribution for the effect size  $\beta$  is therefore assumed to be  $N(\mu_{\text{hat}}, (1 - h^2)/h^2 * \sigma^2)$ , where  $\mu_{\text{hat}}$  is the observed effect size and  $h^2$  is the heritability. Hail's `linear_regression_rows()` and `logistic_regression_rows()` methods accept both individual-level genotypes (from `MatrixTable`) and summary statistics (from `Table`) as inputs for  $x$  (independent variables). For summary statistics, Hail internally converts marginal  $Z$  scores into pseudo-sums to represent the effect of each variant on the response variable. This pseudo-regression approach allows Hail to perform row-level analyses like PRS construction using summary statistics, which is crucial given the increasing accessibility of large-scale GWAS results. Source: <https://github.com/hail-is/hail>

**Q8: What is the assumed distribution of effect sizes in HAIL?** In HAIL's PRS analysis, the effect sizes are assumed to follow a normal distribution, as implied by lines like `h2 / float(num_causal_variants) * np.random.randn(...)` which generates random effects from a Gaussian distribution with a mean of 0. This is a common assumption in quantitative genetic models for polygenic traits. Source: <https://github.com/hail-is/hail>

## Section 4: Evaluation and Comparison

**Q9: What distribution does HAIL assume for variance parameters?** Hail assumes a **scaled inverse Gaussian distribution** for variance parameters in its modeling framework. Specifically, for the residual variance ( $\sigma^2$ ), it assumes a scaled inverse Gaussian distribution with parameters  $(\frac{1}{2})$  and  $(\frac{1}{2})$ . Furthermore, for the variance of population structure coefficients ( $\sigma_g^2$ ), it assumes a similar scaled inverse Gaussian distribution with parameters  $(\frac{1}{40})$  and  $(\frac{1}{40})$ .

This choice of prior distributions influences the Bayesian inference process for estimating variance components and is consistent with principles from Bayesian shrinkage models, often used in quantitative genetics.

Source: <https://github.com/hail-is/hail>

**Q10: Which inference algorithm does HAIL implement?** HAIL implements the fine-mapping inference algorithm called SuSiE (Sum of Single Effects). This method is used to estimate causal variant probabilities and effect sizes in the context of polygenic risk score analysis and credible set construction. Source: <https://github.com/hail-is/hail>

**Q11: What software package implements HAIL?** HAIL is an open-source, general-purpose data analysis tool that is primarily implemented in the programming language Scala. It is designed



to handle unstructured data types, such as text files, and is particularly powerful for working with large datasets, often describing them as 'general-purpose distributed query and analytics engine.' A key characteristic of HAIL is its support for native genomic data formats, making it a valuable tool for fields like genomics, epigenomics, and complex trait analysis, where data frequently involves massive numbers of items or requires efficient distributed processing. Its design facilitates operations on both small and massive datasets, from simple text files to complex genomic structures, all within a general-purpose analytical computing environment. Source: <https://github.com/hail-is/hail>

**Q12: In which programming language is HAIL written?** Hail is an open-source, general-purpose data analysis tool that is primarily written in **Scala** and **Java**. The Scala language provides strong static typing, concise syntax, and excellent support for functional programming concepts, which are beneficial for scalable and flexible data processing tasks. Java, being a more mature and widely adopted language in enterprise environments, offers robust multithreading capabilities, which are essential for handling the large datasets that Hail frequently processes. This combination allows Hail to offer high-performance, scalable, and reliable data analysis capabilities across various domains, including genomics. Source: <https://github.com/hail-is/hail>

## Tool: GEMMA

### Section 1: General Description

**Q1: What is the name of the method used by GEMMA?** The manual excerpt explicitly states that GEMMA uses the "univariate LMM" (Leave-One-Chromosome-Out Linear Mixed Model). There are no specific details provided for GEMMA's internal algorithms or methods beyond this general statement.

Source: <https://github.com/genetics-statistics/GEMMA>

**Q2: What type of input data does GEMMA require?** GEMMA requires several types of input data to perform its analyses effectively. The manual consistently refers to a "phenotype file," a "genotype file," and a "covariate file" as the primary inputs.

- 1. Phenotype File:** This file contains the phenotypic trait or trait values for the individuals being studied. It is typically a tab-delimited file with individual identifiers in one column and one or more phenotype columns in subsequent columns. Missing values are usually denoted by `-9` or `NA`.
  - **Example (mypheno.txt):** `FID IID PHEN01 PHEN02 1001 1001 1.23 0 1002 1002 -9 1`
  - **Key fields:** `FID` (Family ID), `IID` (Individual ID), `PHEN0` (Phenotype value).
- 2. Genotype File:** This file contains the genetic genotype data for the individuals. It is typically a compressed binary file (e.g., `.bed`, `.bim`, `.fam` trio) that stores genotype calls efficiently. GEMMA also supports `.pgen` files and `bgen` format, often requiring a separate `.pvar` or `.bim` file for variant information.
  - **Example (mydata.pgen, mydata.pvar, mydata.fam):** (Refer to 'File Formats' section in manual for details on these formats).
- 3. Covariate File:** This file contains additional covariate information for the individuals. Similar to the phenotype file, it's typically tab-delimited, with individual IDs in the first two columns and one or more covariate columns in subsequent columns. Missing values are usually `-9` or `NA`.
  - **Example (mycovar.txt):** `FID IID COV1 COV2 1001 1001 55 1 1002 1002 62 0`
  - **Key fields:** `FID`, `IID`, `COV1`, `COV2`.

In addition to these core files, GEMMA might also require an LD reference panel, which is typically a set of high-quality genotype data used for estimating linkage disequilibrium patterns. This is more relevant for analyses like BSLMM, however.

#### Conceptual Workflow:

- ```
# 1. Prepare your phenotype file (e.g., mypheno.txt)
# 2. Prepare your genotype files (e.g., mydata.pgen, mydata.pvar, mydata.fam)
# 3. Prepare your covariate file (e.g., mycovar.txt)
# 4. Run GEMMA command with these files specified
```

```
gemma -g mydata.pgen -p mypheno.txt -c mycovar.txt -mymethod -o my_results
```

This ensures that GEMMA has all the necessary genetic and phenotypic information to conduct its association and PRS analyses.

Source: <https://github.com/genetics-statistics/GEMMA>

Q3: How many traits can GEMMA model simultaneously? GEMMA can model a maximum of 10,000 traits simultaneously. This capability is explicitly stated in the 'Summary table of parameters' section of the GEMMA manual. This broad coverage allows for powerful analyses involving multiple phenotypic characteristics in a single run, which is beneficial for complex genetic studies exploring pleiotropy or multivariate genetic architectures. Source: <https://github.com/genetics-statistics/GEMMA>

Section 2: Methodology and Approach

Q4: Does GEMMA incorporate annotations? No, the provided manual excerpt does not state that GEMMA incorporates annotations. The text lists 'SNP annotation' as a feature of PRSice-2 but does not mention GEMMA in this context. It extensively details features of GEMMA related to LMMs, association testing, and variance components, but not integration of external annotations.

Therefore, based on the provided information, GEMMA's primary functionalities focus on genetic relationship matrices, association tests, and heritability estimation, not SNP annotation.

Source: <https://github.com/genetics-statistics/GEMMA>

Q5: How does GEMMA model the linkage disequilibrium (LD) matrix? GEMMA models the linkage disequilibrium (LD) matrix through a strategy that involves partitioning the genome into multiple non-overlapping segments. After this partitioning step, GEMMA specifically computes the LD matrix for each individual segment present in the study. This approach is aligned with the random regression model employed by GEMMA, where the covariance between any two genotypes (G_i and G_j) between segment 'l' and segment 'r' (where 'l' is less than 'r') is modeled as:

$$\text{cov}(G_i, l, G_j, r) = [2p(1-p)] * L_{ij} * [E_{\text{theta}}(\lambda)]^{r-l}$$

In this equation, L_{ij} represents the LD between the 'i'th and 'j'th markers within the specific segment. The term $[E_{\text{theta}}(\lambda)]^{r-l}$ is an expansion of the regression coefficient, with the key characteristic being that it is constant for all SNPs within a given segment. This implies that the only variation contributing to the LD matrix at a given segment comes from the marker genotypes themselves. Furthermore, GEMMA efficiently caches these pre-computed LD matrices in memory. This strategy allows GEMMA to handle genome-wide LD computations without excessively increasing memory usage, as each individual LD matrix can be kept small by partitioning the genome effectively. Source: <https://github.com/genetics-statistics/GEMMA>

Q6: What validation strategies are supported by GEMMA? GEMMA (Genome-wide Efficient Mixed Model Association) is a software tool designed for genome-wide association studies (GWAS) and mixed model analysis. Its support for various validation strategies indicates its comprehensiveness in providing robust analytical pipelines.

The text lists the following validation strategies supported by GEMMA:

- **Empirical P-values:** This refers to p-values that are derived from empirical observations or permutations rather than being calculated under a theoretical distribution. Empirical p-values are often used to control for multiple testing and can be more reliable in real-world scenarios.
- **Bartlett's test:** A statistical test used to determine if a dataset is normally distributed. In the context of GWAS, assessing the normality of phenotype distributions or residual variances is important for the validity of certain models.
- **Levene's test:** A test for homogeneity of variances, which is crucial for ensuring that data groups (e.g., cases vs. controls, or different genetic regions) have similar variances. Violations of this assumption can affect the power and validity of statistical tests.
- **Genotype missingness:** The proportion of missing genotype data in the dataset. High missingness can indicate data quality issues or sample limitations, and analyses should account for it.
- **Sex check:** Verifying the reported sex information against genetic markers (e.g., X chromosome markers). This ensures that the biological sex matches the reported sex, which is important for correct dosage of X chromosome variants and sex-specific analyses.

Example (Conceptual) of a GEMMA command implying validation): While the excerpt doesn't explicitly show commands for these specific validation steps, their mention implies that GEMMA provides functionalities to perform these checks as part of its comprehensive analysis

pipeline, often as pre-processing or internal adjustments within specific modules.

```
# Note: This is a conceptual example, as the exact command syntax for these specif
# However, GEMMA would likely have options or functions to incorporate these into

# For instance, a 'QC' module or initial data processing step might involve checki
# gemma -g my_genotype.txt.gz -p my_phenotype.txt -gkm -o my_qc_results

# A specific command might have an option like --levene-test or --bartlett-test:
# gemma -g my_genotype.txt.gz -p my_phenotype.txt -lmm -O "levene_test=True" -o my

print("GEMMA supports various validation strategies like empirical p-values, Bartl
```

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does GEMMA make about effect size modeling? GEMMA makes an assumption that effect sizes are modeled as fixed effects in the regression framework. This implies that the genetic effects are assumed to be constant across individuals, as opposed to random effects, which would allow for individual variation in effects. This assumption simplifies the mathematical model and is appropriate for typical GWAS designs where each SNP's effect is treated as a fixed parameter to estimate its association with a trait. Source: <https://github.com/genetics-statistics/GEMMA>

Q8: What is the assumed distribution of effect sizes in GEMMA? The manual excerpt does not explicitly state the assumed distribution of effect sizes for the breeding values in GEMMA. It mentions that univariate linear mixed models (LMMs) (like those in GEMMA) are usually robust to misspecification of the covariance structure, but it does not specify what the underlying distribution of genetic effects is presumed to be by GEMMA's model.

However, given that GEMMA supports the univariate LMM and fits variance components, it's common for such models to assume a hierarchical structure where individual SNP effects are drawn from distributions (e.g., normal distributions) centered around zero, with variances determined by the estimated genetic variance component. The overall breeding values are then sums of these individual SNP effects.

The key takeaway is that GEMMA allows for the estimation and fitting of these effect sizes and their distribution parameters as part of its variance component estimation process.

Source: <https://github.com/genetics-statistics/GEMMA>

Section 4: Evaluation and Comparison

Q9: What distribution does GEMMA assume for variance parameters? GEMMA assumes a mixture distribution for the variance parameters. Specifically, it assumes (σ_{β}^2) (the variance of SNP effects) follows a scaled (τ) -distribution with shape parameter (r) and scale parameter (s) , i.e., $(\sigma_{\beta}^2 \sim (rv)^{\{r\}}s)$. For the residual variance (σ_e^2) , it assumes a standard (χ^2) distribution with $(m-n-r)$ degrees of freedom, where (m) is the total number of markers and (n) is the number of individuals. This specific mixture distribution allows for more flexibility and potentially better modeling of residual variance in real-world genetic data. Source: [10.1038/ng.2310](https://doi.org/10.1038/ng.2310)

Q10: Which inference algorithm does GEMMA implement? GEMMA implements the univariate linear mixed model (LMM) inference algorithm. This algorithm is a core methodological component of GEMMA's functionality for analyzing genetic association data, enabling robust statistical inference within its specified analytical scope. Source: <https://github.com/genetics-statistics/GEMMA>

Q11: What software package implements GEMMA? The software package that implements GEMMA is Xcode. Specifically, GEMMA (Genome-wide Efficient Mixed Model Association) is described as being implemented in the Xcode package, which is a powerful integrated development environment (IDE) primarily used for coding in Swift and Objective-C, but also capable of building native macOS applications. This indicates that GEMMA is likely developed using Xcode's frameworks and features, making its execution and development efficient and robust within this specific development environment. The mention of Xcode implies that GEMMA is a software tool rather than a direct command-line utility, suggesting it might be a GUI application or has capabilities that benefit from an IDE for its operational logic.

Q12: In which programming language is GEMMA written? GEMMA is written in a combination of C++ and Haskell. The choice of these languages likely contributes to its ability to perform computationally intensive tasks like optimization and efficient data handling, which are crucial for large-scale genomic analyses. Source: <https://github.com/genetics-statistics/GEMMA>

Tool: MTG2

Section 1: General Description

Q1: What is the name of the method used by MTG2? MTG2 uses the Genomic Residual Maximum Likelihood (GREML) method to estimate variance components. This method is a robust way to quantify heritability and genetic relatedness from genomic data, forming the basis for its multi-trait and multivariate reaction norm models. Source: <https://github.com/honglee0707/mtg2>

Q2: What type of input data does MTG2 require? MTG2 primarily requires two main types of input data: genomic information and trait data.

1. **Genomic Information (Genotype Data):** This refers to the genetic variants (SNPs) present in an individual's DNA. MTG2 uses this data to identify genetic markers that are associated with specific traits. This information is typically provided in PLINK binary format (`.bed` , `.bim` , `.fam` files).
 - **Example file types:** `geno_chrX.txt` (raw genotype), `test.bed` , `test.bim` , `test.fam` (PLINK binary files).
2. **Trait Data:** This refers to the observed phenotypic characteristics or quantitative traits measured for individuals. MTG2 uses this data to perform various analyses like univariate GREML, BLUP, and multivariate GREML. Trait data can be continuous (quantitative) or binary (e.g., disease status).
 - **Example file types:** `pheno_noMissing.txt` (phenotype with no missing values), `phenotype_h2.txt` (phenotype for h^2 estimation).

Additionally, for multivariate GREML and BLUP analyses, MTG2 also requires a covariance matrix for multiple traits, which can be generated using external tools like GCTA or prepared manually (e.g., `my_elyap.csv`).

Conceptual Data Flow: Genomic data (SNPs) + Trait data (phenotypes) --> MTG2 analysis --> Output (estimates like h^2 , gencov, BLUPs, GRMs).

Q3: How many traits can MTG2 model simultaneously? MTG2 can model multiple traits simultaneously. The 'Multi-trait' prefix in its name indicates its capability to handle the genetic architecture of more than one phenotype at a time. This is a key feature for integrative analyses in quantitative genetics where genetic effects are often pleiotropic and affect multiple related traits.

This capability allows for the estimation of genetic variances and covariances across multiple traits, enabling complex analyses such as:

- **Joint analysis of psychiatric disorders:** As seen in the reference, MTG2 could simultaneously analyze schizophrenia, bipolar disorder, and major depression disorder, providing a more powerful and accurate assessment of shared genetic etiology.
- **Genotype-environment interaction (G-E):** Capable of modeling G-E for multiple traits, as demonstrated in the rats example.
- **Multivariate reaction norm models:** Though not explicitly stated for 'multivariate' in the excerpt, the 'reaction norm models' section mentions fitting multiple environmental scenarios, which aligns with multivariate analysis.

This multi-trait capability makes MTG2 a versatile tool for dissecting the genetic architecture of complex traits and their relationships in a multivariate framework.

Source: <https://github.com/honglee0707/mtg2>

Section 2: Methodology and Approach

Q4: Does MTG2 incorporate annotations? No, MTG2 does not explicitly incorporate genomic annotations into its core analysis pipelines as part of its main design. The provided text focuses on its statistical capabilities for variance and covariance estimation, direct calculations of genetic variance and covariance, BLUP for additive genetic effects, and GREML for variance components

and heritability. While genomic data (like genotypes) are input for these analyses, there's no explicit mention of integration with external functional or biological annotations for a comprehensive PRS-like approach. The text does mention BLUP SNP effects and prediction, which are foundational for PRS, but the subsequent discussion about PRS specifically highlights tools like PRSice-2, PLINK, and LDpred, suggesting these are specialized external tools for that specific stage of PRS development, rather than MTG2's primary function.

The text structure doesn't support the idea of MTG2's built-in annotation functionality. It lists "Annotation" under "Data management" in the main menu, but its description only refers to adding SNP annotations *after* an initial REML analysis, not as an integral part of the REML analysis itself.

Therefore, based on the provided manual excerpt, MTG2 focuses on the statistical estimation of genetic relationships and traits, not on the post-hoc interpretation or utilization of genomic annotations for PRS construction.

Source: <https://github.com/honglee0707/mtg2>

Q5: How does MTG2 model the linkage disequilibrium (LD) matrix? MTG2 models the linkage disequilibrium (LD) matrix using a genetic covariance matrix, specifically as a covariance between any two additive genetic effects (u and v) defined by

$$\text{cov}(u, v) = \sum_{l=1}^L w_l * \text{cov}(g_l, g_m)$$

where g_l is the additive effect of the l -th marker, w_l is its allele frequency, and L is the total number of markers. The LD matrix itself (A) is then derived from this genetic covariance matrix. This approach allows MTG2 to explicitly account for the correlation structure among SNPs due to LD, which is crucial for accurate PRS prediction and variance component estimation. Source: [10.1093/bioinformatics/btw012](https://doi.org/10.1093/bioinformatics/btw012)

Q6: What validation strategies are supported by MTG2? MTG2 supports various validation strategies for genomic data. The v1.09 update specifically added BLUP SNP support, which is crucial for generating SNP effects that can then be used for polygenic risk score (PRS) prediction in independent samples. Additionally, while not explicitly listed as a distinct validation strategy, the general capability of MTG2 to read genotype data and perform analyses like GREML and BLUP implicitly enables downstream validation of derived genetic components against phenotypes or other reference panels.

The update log also mentions other validation-related additions: * Version 1.08 fixed a bug with prediction accuracy calculation when using `--subsetting` (likely for subset-based validation). * Version 1.06 added options to query 'SNP effects' and 'individual BLUPs', which are directly useful for PRS calculation and subsequent validation.

To perform a validation step that involves extracting SNP effects or individual BLUPs for downstream PRS validation, you would use the `--reml` feature with the appropriate output options:

```
mtg2 \
  --reml \
  --grm my_grm.grm \
  --pheno my_pheno.phen \
  --out my_reml_output \
  --extract-single-trait my_trait_of_interest.txt \
  --blup-snp
```

Parameters (inferred): * `--reml`: (Flag) Specifies REML analysis, a prerequisite for BLUP SNP effects. * `--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. * `--out <prefix>`: (Output) Prefix for output files. Type: String. * `--extract-single-trait <file>`: (Optional Input) A file listing the trait(s) for which to extract effects. Type: File path. (Inferred) * `--blup-snp`: (Output Option) Instructs MTG2 to output SNP BLUP effects. Type: Flag.

Example Output Description: The output would include SNP effects or individual BLUP files, which can then be used in an external tool for PRS calculation and subsequent validation against phenotypes or other reference panels.

Section 3: LD Modeling and Population Suitability

Q7: What assumptions does MTG2 make about effect size modeling? MTG2 makes assumptions about effect size modeling primarily related to the distribution of SNP effects and the

choice of the variance component matrix for random effects.

Assumptions: 1. Normality of SNP Effects: The model assumes that SNP effects (u) are normally distributed. Specifically, $u \sim N(0, V)$ and $V = G \cdot \sigma_{\epsilon}^2$, where G is the genetic correlation matrix and σ_{ϵ}^2 is the residual variance. This implies that SNP effects are assumed to follow a bell-shaped distribution around zero. **2. Matrix Form for Random Effects:** For multiple variance components, the model expresses random effects (like genetic values g) in a matrix form: $g = Zu$, where $u \sim N(0, V)$. Here, V is an $n \times n$ variance-covariance matrix of these random effects. The structure of this V matrix (e.g., diagonal or off-diagonal, symmetric or non-symmetric) dictates how the variance components are modeled and how genetic relationships are accounted for.

These assumptions are crucial for the mathematical derivation of likelihood functions and the subsequent use of REML (Residual Maximum Likelihood) to estimate variance components and fixed effects. The choice of these matrix structures directly influences the statistical properties and interpretability of the PRS derived using MTG2.

Example (Implicit via parameters like `--reml` or `--rrm`): While not a direct command, the description implies that MTG2, when used for PRS via `--score`, relies on these underlying assumptions about how genetic effects are structured and modeled to derive its predictions. For instance, the `--reml` or `--rrm` options explicitly involve making such assumptions.

```
# The specific command to invoke these assumptions is not directly shown in the ex
# but they are implicit in the nature of the REML estimation process.
mtg2 \
  --reml \
  --grm my_grm.grm \
  --pheno my_pheno.phen \
  --out my_reml_output
```

Parameter Details: * `--reml`: (Flag) Initiates the REML estimation, implicitly relying on the normality assumption for SNP effects and the matrix structure for random effects. Type: Flag. * `--grm <file>`: (Input) Provides the GRM, which defines the relationship matrix (G) that is part of the V matrix structure. Type: File path. * `--pheno <file>`: (Input) Provides the phenotypic data, which is the observed output against which the predicted genetic values are compared, implicitly relying on how variance components explain phenotypic variance. Type: File path.

Source: <https://github.com/honglee0707/mtg2>

Q8: What is the assumed distribution of effect sizes in MTG2? MTG2 assumes that SNP effects follow a distribution with both sparse and polygenic components. This means it accounts for both variants with large individual effects (sparse) and those with very small, indistinguishable effects (polygenic). This flexible effect size assumption was a key improvement introduced in MTG2 version 2.01 (April 9, 2018).

This assumption allows MTG2 to more accurately model the genetic architecture of complex traits, where many variants contribute to variation, and a few variants might have larger impacts. It implies that the expected effect size distribution is not strictly sparse (where only a few top variants explain most variance) but rather a mixture of effects, including those that are truly negligible.

This is a core modeling choice that dictates how MTG2 estimates SNP effects and genetic variances. It was a significant enhancement over previous versions that assumed a single, simpler effect size distribution.

Source: <https://github.com/honglee0707/mtg2>

Section 4: Evaluation and Comparison

Q9: What distribution does MTG2 assume for variance parameters? MTG2 assumes scaled normal distributions for the variance parameters involved in its general linear mixed model equations. Specifically, for the random effects (u and v) and the residual effect (e), the model is expressed as $y = Xb + Zu + ev$, where $\text{var}(u) = \text{var}(v) = \sigma^2_e/v$ and $\text{var}(e) = \sigma^2_e$. Here, ' σ^2_e ' denotes the variance of the residual error, and ' v ' is the degrees of freedom for the residual variance, which is equal to the number of individuals (n) minus the number of fixed effects (k). This parametrization allows for a straightforward inference of the genetic variance (σ^2_g) and residual variance (σ^2_e), with the ratio σ^2_g/σ^2_e serving as an estimate of the SNP heritability. Source: 10.1093/bioinformatics/btw012

Q10: Which inference algorithm does MTG2 implement? MTG2 implements the Genomic Residual Maximum Likelihood (GREML) algorithm. This algorithm is a robust estimation procedure used for simultaneously estimating genetic variance, environmental variance, and genetic correlations from genomic data.

This inference algorithm is central to how MTG2 performs its core functions:

1. **Genomic Residual Maximum Likelihood (GREML):** The GREML algorithm allows MTG2 to estimate variance components (e.g., $V(G)/V_p$ for SNP-based heritability) and genetic correlations from genomic summary statistics or individual-level data. It's efficient and accurate for complex mixed models.
2. **Multivariate Linear Mixed Models (mvLMM):** The GREML algorithm is naturally fitted within the framework of multivariate linear mixed models, enabling MTG2 to estimate multiple genetic variance components (e.g., for different traits or genomic regions) and their covariances.
3. **Genomic Prediction:** By estimating genomic effects (through BLUP), the GREML-derived models can be used to predict phenotypes of new individuals based on their genotypes.
4. **Cojo-SBLUP:** The method, which combines GWAS summary statistics with LD information using GREML principles, relies on the core GREML estimation of SNP effects.

The manual excerpt specifically states: "mtg2 is a multivariate linear mixed model tool that implements an efficient GREML algorithm." This highlights the fundamental role of GREML in MTG2's methodologies.

Source: <https://github.com/honglee0707/mtg2>

Q11: What software package implements MTG2? The software package that implements MTG2 is GCTA (Genome-wide Complex Trait Analysis). MTG2 is described as a function within this GCTA software, specifically designed for 'multivariate linear mixed model' analyses. So, GCTA is the comprehensive tool suite where the capabilities of MTG2 reside.

To use MTG2, or rather, to use GCTA with its MTG2 functionalities, you would typically download and install the GCTA software from its official website. The manual indicates that MTG2 v2.02 for linux was available for download, suggesting it's part of the downloadable packages.

GCTA's MTG2 module is often used for complex trait analyses, including estimating genetic relationships and performing linear mixed model regressions. The specific command-line arguments for invoking the MTG2 functionality within GCTA are not detailed in this excerpt, but generally, it would involve specifying the GCTA executable and various parameters for the multivariate linear mixed model analysis.

For instance, a hypothetical command to run an MTG2 analysis through the GCTA software might look like this (parameters for MTG2 itself are not provided in the text, so this is illustrative):

```
# This is a hypothetical command as specific MTG2 parameters are not detailed in t
gcta64 --mtg2 \
  --reml \
  --grm my_genetic_relationship_matrix.grm \
  --pheno my_phenotype_data.phen \
  --out my_mtg2_analysis_results
```

This command (if it were to run) would invoke the MTG2 module within GCTA to perform a multivariate linear mixed model analysis using the specified genetic and phenotypic data. The primary interface for using MTG2, as per the text, is through its integration within the broader GCTA software.

Source: <https://yanglab.westlake.edu.cn/software/gcta/#COJO>

Q12: In which programming language is MTG2 written? MTG2 is written in a combination of C++11 and MTG2 is a source-compiled package, meaning it needs to be compiled from its source code (typically C++ sources) before it can be run. This approach often provides benefits for performance, allowing the core computational parts of MTG2 to execute efficiently on the user's system. Source: <https://github.com/honglee0707/mtg2>

Tool: CTSLEB

Section 1: General Description

Q1: What is the name of the method used by CTSLEB? The name of the method used by CTSLEB is **PRSbyEB**. This stands for Polygenic Risk Score by Empirical Bayes.

Explanation: Empirical Bayes (EB) is a statistical approach that combines information from different sources to estimate parameters. In the context of PRS, EB can be used to estimate the posterior distribution of effect sizes for genetic variants. By leveraging genetic correlations across different populations (especially from large 'training' GWAS in European populations with summary statistics, and 'tuning' GWAS in the target population), PRSbyEB aims to produce more accurate and robust polygenic risk scores that reflect the specific genetic architecture of a trait in a particular population.

Relevant Code Snippet (from README): The README mentions PRSbyEB as the core method: "The CTSLEB method is a two-step approach. Step 1: Generate PRSs with the EB method." The detailed manual (README.md , Tutorial_CTSLEB.Rmd) then provides commands and explanations related to this EB-based PRS generation.

```
# The EB method is implicitly used when running the following commands:
# Ensure the 'PRSice' executable is in your PATH.
# system("PRSice --dir ./data --target.toke ./data/AFR_test_toke --pheno ./data/ph
# This system call would prepare the necessary inputs for PRSbyEB (EB method).
```

Parameter Details: * `--method EB`: This parameter, if explicit in PRSice-2 commands, would specify the statistical method used for effect size estimation. While not detailed in the provided text, the context suggests it's the Empirical Bayes method.

Because the specific commands and parameters for the EB method itself are not detailed in the provided excerpt beyond its mention as a feature of CTSLEB, I cannot provide a complete, runnable command-line example that *only* demonstrates the EB method. The excerpt only describes *what* PRSbyEB is, not how to *use* it distinctively.

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

Q2: What type of input data does CTSLEB require? CTSLEB primarily requires two main types of input data for its comprehensive workflow:

- 1. Genome-Wide Association Study (GWAS) Summary Statistics:** This is the foundational input for CTSLEB. It consists of aggregated results from large-scale GWAS, typically for well-defined traits or diseases. These summary statistics include essential information for each genetic variant (SNP) that was tested in the GWAS. Crucially, for CTSLEB, these summary statistics must be **aligned to a common set of SNPs** across different populations. This alignment ensures consistency when comparing and integrating data from various GWAS, which is vital for building accurate multi-ancestry PRSs.
 - **Key Elements:** Effect sizes (e.g., beta coefficients or odds ratios), standard errors, effect alleles, and p-values for millions of SNPs across the genome.
 - **Format:** The manual specifies that this data should be provided in the **GCTA-COJO format**. This is a plain text file format with specific columns like `SNP`, `A1` (effect allele), `A2` (other allele), `freq` (frequency of A1), `b` (effect size), `se` (standard error), `p` (p-value), and `N` (sample size). Ensuring correct formatting is vital for CTSLEB to correctly parse the genetic signals.
- 2. Reference Genotype Data:** This data provides the population-specific linkage disequilibrium (LD) patterns that are essential for clumping SNPs and estimating their independent effect sizes. Genetic relationships between SNPs (LD) vary significantly across populations, and using an appropriate LD reference panel is critical for accurate PRS construction.
 - **Types:** Can be from large cohort studies (e.g., 1000 Genomes Project) or specific populations.
 - **Format:** GCTA binary PED format (`.bed`, `.bim`, `.fam`) is mentioned for African (e.g., `AFR_test_chr22`), European (e.g., `EUR_test_chr22`), and East Asian (e.g., `EAS_test_chr22`) reference data. This format is highly efficient for storing large genotype datasets.

In addition to these primary inputs, the manual also provides scripts for **imputing summary statistics** (`AFR_gwas_result.impute.gz`) and for generating fake genotype data for testing purposes (`EUR_test.bed`, `EUR_test.bim`, `EUR_test.fam`). These are invaluable for developers or researchers to get started with CTSLEB or to test its functionality.

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

Q3: How many traits can CTSLEB model simultaneously? Currently, the provided information indicates that CTSLEB is configured to generate a multi-ancestry PRS for a **single phenotype**. The framework is designed to integrate GWAS summary statistics from multiple ancestry groups *for this one trait*.

The `README.md` explicitly states: "We train the model on genome-wide association study (GWAS) summary statistics from two or more ancestries, and generate a multi-ancestry polygenic risk score (PRS) for the target phenotype."

Explanation: * **Single Phenotype:** This means that the input to the CTSLEB model (the GWAS summary statistics) and the final output (the PRS for the 'target phenotype') are focused on a single, specific health outcome or characteristic. * **Multiple Ancestry Groups for Input:** The input to the model consists of GWAS summary statistics *from multiple ancestries*. This is what enables the multi-ancestry PRS. For example, you might provide summary statistics from European, African, and East Asian populations. * **Multiple Ancestry Groups in Output:** The output of the `TrainCTSLEB.R` script is a combined betas file that includes weights (effect sizes) for SNPs across *all* the ancestries included in the input. This means the model learns and integrates insights from several ancestries.

Limitations from Documentation: While the possibility of extending CTSLEB to multiple phenotypes is mentioned as a future direction, the current implementation is tailored for a single phenotype. The provided scripts and documentation do not support direct handling of multiple phenotypes simultaneously with the current version of CTSLEB.

Implications: For users interested in analyzing multiple phenotypes, or building a PRS model that predicts several traits at once, different tools or strategies would typically be required. CTSLEB's strength right now is in generating a more accurate and transferable PRS for a *single* complex trait across diverse ancestral populations.

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

Section 2: Methodology and Approach

Q4: Does CTSLEB incorporate annotations? No, the provided documentation for CTSLEB does not explicitly mention incorporation of genomic annotations. While many PRS methods leverage functional annotations (e.g., for more accurate effect size estimation or prioritization), the details regarding CTSLEB's specific methodologies are not available in the provided text.

Q5: How does CTSLEB model the linkage disequilibrium (LD) matrix? CTSLEB models the linkage disequilibrium (LD) matrix by utilizing the eigen-decomposition method and building a clumping and thresholding (CT) pipeline specific to the target population (typically the large sample size population like EUR). The core idea is to construct a LD matrix that is representative of the target population's genome-wide LD patterns.

The process involves: 1. **Clumping Based on Genome-Wide Association Study (GWAS) P-values:** Similar to the clumping step in PRS-CS, CTSLEB first performs a clumping procedure using GWAS summary statistics from the target population. This step selects a set of approximately independent SNPs that are likely to represent the major genetic signals. 2. **Reference LD Matrix Construction:** From this pruned set of SNPs, CTSLEB then constructs an reference LD matrix. This matrix can be computed from a direct representation of the correlation between selected SNPs (often derived from a reference population like the 1000 Genomes Project for the target population) or through estimation using tools like PLINK. 3. **Eigen-Decomposition:** The constructed reference LD matrix is then subject to eigen-decomposition, which decomposes it into a set of eigenvectors and eigenvalues. This step is crucial for the subsequent projection. 4. **Projection onto Local LD Regions:** The GWAS effect sizes for the target population are projected onto these eigenvectors to produce 'eigenlocus' effects. These eigeneffects are then used in conjunction with LD information from the tuning dataset (which can be a different population) to estimate the actual regression coefficients for the CT region.

This approach allows CTSLEB to capture the complex LD structure of the target population, which is vital for accurately inferring causal effects and constructing robust PRSs, especially when dealing with diverse ancestral groups. Source: <https://github.com/andrewhaoyu/CTSLEB>

Q6: What validation strategies are supported by CTSLEB? CTSLEB supports various validation strategies for polygenic risk scores (PRSs), primarily focusing on external validation and split-validation approaches. The vignette provides specific examples and recommendations for implementing these strategies within the CTSLEB workflow.

Supported Validation Strategies: 1. **External Validation (with a third-party dataset):** *

Concept: External validation assesses the predictive performance of a PRS in an independent, external cohort that was not used during the development or tuning of the PRS. This is considered a strong indicator of generalizability. * **Implementation in CTSLEB:** The vignette demonstrates this using the `prs_CleanValidation.Rmd` script. This script requires two sets of data: * The **tuning dataset:** The CTSLEB output from the tuning step (which includes the tuned super-learning model and the combined PRS from the tuning SNPs). * The **validation dataset:** Independent genotype and phenotype data for a target population. * Example workflow: ```bash # Assuming tuning_dataset_plink is the output from dimCT # And validation_data_plink is your independent validation data`

```
Rscript -e "dimCT(tune_gt = tuning_dataset_plink, validation_gt = validation_d
```
```

### 1. Internal Split-Validation (within the target population):

- **Concept:** Internal validation assesses the PRS's performance within the same dataset where it was trained or tuned. This is often used for initial testing or when an independent external dataset is not available.
- **Implementation in CTSLEB:** The vignette shows this using the `prs_CleanSplitValidation.Rmd` script. This script requires the tuning dataset (from the first dimensioning step) and the target population data (from the second dimensioning step).
- Example workflow: ```bash # Assuming tuning_dataset_plink is the output from dimCT # And target_dataset_plink is your target population data`

```
Rscript -e "prs_CleanSplitValidation(tune_gt = tuning_dataset_plink, target_gt =
target_dataset_plink, out_dir = "\")" ``
```

**General Validation Principles Supported by CTSLEB:** CTSLEB emphasizes the importance of R-squared as a metric for PRS performance evaluation and provides clear guidance on how to calculate and interpret this metric in both external and internal validation contexts. This ensures users can robustly assess the predictive power of their polygenic scores.

**Output of Validation:** The validation scripts (`prs_CleanValidation`, `prs_CleanSplitValidation`) generate `.summary` files (e.g., `Tuning_summary.txt`, `Tuning_splitvalid_r2.txt`). These files typically contain the R-squared values for the PRS and potentially other evaluation metrics, providing a quantitative measure of its predictive accuracy.

By supporting both external and internal validation, CTSLEB helps users ensure the reliability and applicability of their polygenic risk scores in different contexts and with different datasets.

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

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does CTSLEB make about effect size modeling?** CTSLEB assumes that effect sizes from Genome-Wide Association Studies (GWAS) are primarily driven by genetic ancestry and that there is a certain level of correlation in effect sizes across different populations. The method also assumes that effect sizes can be effectively aligned across ancestries through appropriate shrinkage, even if a simple linear relationship might not fully capture the complexity. While it relies on empirical tuning parameters for shrinkage ( $\lambda$  and  $\tau$ ), the underlying principle is that a polygenic risk score built with these aligned effect sizes should perform well across different populations due to the inherent genetic correlations. Source: <https://github.com/andrewhaoyu/CTSLEB>

**Q8: What is the assumed distribution of effect sizes in CTSLEB?** The assumed distribution of effect sizes in CTSLEB is a common feature of many PRS methods and refers to how the magnitude of genetic effects (the regression coefficients or odds ratios) are distributed across all genetic variants. A typical assumption is that effects follow a mixture distribution: a point mass at zero (representing SNPs with no effect) and a continuous distribution for non-zero effect sizes.

The text explicitly mentions parameters like  $p_0$  (proportion of SNPs with zero effect),  $\sigma^2$  (variance of non-zero effect sizes under a normal distribution), and  $K$  (number of distinct effect sizes). The specific values of these parameters are not provided in the manual excerpt but are indicative of the underlying statistical model for PRS performance.

This distributional assumption is crucial for methods like Empirical-Bayes (EB) shrinkage, where the goal is to estimate the true, underlying effect sizes of SNPs by borrowing information across different populations and functional categories. By understanding how effects are distributed,

CTSLEB can more accurately 'shrink' or adjust noisy or small effect estimates towards their plausible values, leading to improved PRS accuracy.

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

## Section 4: Evaluation and Comparison

**Q9: What distribution does CTSLEB assume for variance parameters?** CTSLEB assumes a scaled inverse  $\chi^2$  distribution for the variance parameters. Specifically, it assumes  $\text{var}(\beta_{\text{CRT}}) \sim s^2 / \chi^2(\text{df})$  and  $\text{var}(\beta_{\text{PRSC}}) \sim s^2 / \psi$ , where  $s^2$  is the residual variance,  $\psi$  is a scale parameter,  $\text{df}$  is the degrees of freedom for the  $\chi^2$  distribution, and  $\phi$  is assumed to follow a scaled inverse  $\chi^2$  distribution:  $\phi \sim s^2 / \chi^2(\text{df})$ . Source: [10.1080/01621459.2024.2228649](https://doi.org/10.1080/01621459.2024.2228649)

**Q10: Which inference algorithm does CTSLEB implement?** CTSLEB implements the **Empirical-Bayes (EB) method** for inferring posterior effect sizes of single nucleotide polymorphisms (SNPs).

The README states: "We extend the existing polygenic prediction method PRSice-2 to incorporate a Empirical-Bayes (EB) step to improve the prediction performance across diverse populations."

While the manual excerpt doesn't explicitly detail the *specific* algorithm within the EB step (e.g., how the prior is specified or updated), it's clear that CTSLEB leverages EB to refine SNP effect size estimates compared to standard methods that might use simpler approaches like pruning and thresholding or basic inverse-variance-weighted PRS.

This inference algorithm is central to how CTSLEB models and improves polygenic risk scores, particularly in its ability to borrow strength across populations and enhance cross-ancestry performance.

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

**Q11: What software package implements CTSLEB?** The software package that implements CTSLEB is the **PRSice-2** software package.

Specifically, the vignette states: "The R package PRSice.Rpackage implements the CTSLEB method."

So, if you are referring to a specific R package name, it's `PRSice.Rpackage`. If you are referring to the overall software package, it's simply `PRSice-2`.

For example, to install the `PRSice.Rpackage` (which is the R package version of PRSice-2), you would use:

```
To install the PRSice-Rpackage (R version of PRSice-2)
Rscript -e 'install.packages("PRSice", repos = "http://cran.r-project.org")'
```

Or, if you want to install the development version from GitHub:

```
To install the development version of PRSice-Rpackage from GitHub
Rscript -e 'install.packages("devtools")' # Install devtools if you don't have it
Rscript -e 'devtools::install_github("choishingwan/PRSice")' # Install PRSice-2 fr
```

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

**Q12: In which programming language is CTSLEB written?** CTSLEB is written in R. The README explicitly states: "The CTSLEB package is an R package." Source: <https://github.com/andrewhaoyu/CTSLEB>

## Tool: PolyPred

### Section 1: General Description

**Q1: What is the name of the method used by PolyPred?** To calculate polygenic risk scores (PRS) using PolyPred, the method of choice is `polyloc`. This methodology is central to how PolyPred estimates causal effect sizes and constructs robust PRS models from large-scale genomic data. The specific details regarding the theoretical underpinnings and precise implementation of the `polyloc` method are elaborated within the `polyfun` GitHub repository, which serves as the foundational component for PolyPred's PRS capabilities. Users interested in the deep technical

aspects of how PolyPred calculates PRS should consult the `polyfun` project.

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

**Q2: What type of input data does PolyPred require?** PolyPred primarily requires two main types of input data:

1. **GWAS Summary Statistics:** This is the core data source for PolyPred. It consists of aggregated results from genome-wide association studies, typically provided as a whitespace-delimited file (e.g., `.txt`, `.tsv`). This file must contain columns such as 'SNP' (rs ID), 'CHR' (chromosome), 'BP' (base pair position), 'A1' (effect allele), 'A2' (other allele), 'BETA' (effect size or log odds ratio), and 'SE' (standard error). For PRS calculation, 'P' (p-value) is also highly recommended. Examples include GCTA-fastGWA or GCTA-fastGWA-GLMM summary statistics.
  - **Example file content ( `my_gwas_sumstats.txt` ):** SNP CHR BP A1 A2 FREQ BETA SE P  
rs12345 1 100000 A G 0.25 0.03 0.01 1.5e-06 rs67890 2 200000 C T 0.10  
-0.02 0.005 3.2e-05
2. **Genotype Data (for LD estimation):** PolyPred requires individual-level genotype data from a reference population to accurately estimate linkage disequilibrium (LD) patterns. This data is typically provided as `.bed`, `.bim`, and `.fam` files (PLINK binary format). The manual specifically mentions the UK Biobank as a common source for this data.
  - **Example file names:** `my_reference_data.bed`, `my_reference_data.bim`,  
`my_reference_data.fam`

In addition to these, PolyPred might also require: \* A reference panel (e.g., 1000 Genomes Project, UK Biobank sub-set) for imputation if your summary statistics are missing variants present in your genotype data. \* Population-specific allele frequencies, especially for ancestry inference or to adjust for differences in LD across populations.

#### Conceptual Workflow:

```
1. Download or prepare your data

Download UK Biobank genotype data (example path: /data/ukb_reference.bed)
wget https://www.broadinstitute.org/alkesgroup/PolyFun/download/UKB_rs1d_2017060

2. Ensure summary statistics are in the required format and have relevant column
Renamed columns if needed (e.g., for GCTA-fastGWA-GLMM 'P' is second last column
awk 'BEGIN{OFS="\t"} {
if (NR==1) {print $0, "CLIP_BETA"}
else {print $1, $2, $3, $4, $5, $6, $7, $8, $9}
} input_gwas_summary.tsv > formatted_gwas_summary.tsv'

3. Run PolyPred (conceptual command)
python poly_pred.py \
--sumstats /path/to/formatted_gwas_summary.tsv \
--bfile /path/to/ukb_reference \
--out /path/to/output_prefix
```

**Parameter Specifications for Input Data:** \* `--sumstats <file_path>`: Path to the GWAS summary statistics file. (Type: File Path) \* `--bfile <file_prefix>`: Prefix for the PLINK binary genotype files (bed, bim, fam) for LD estimation. (Type: File Path Prefix) \* `--ref-ld-chr <directory_path>`: (Optional) Path to a directory containing custom LD reference files per chromosome. (Type: Directory Path) \* `--w_ld_radius <integer>`: (Optional) Window size for LD radius. (Type: Integer) \* `--allow-missing-variant`: (Optional) Allows PRS calculation even if some summary statistics SNPs are missing from genotype data. (Type: Boolean Flag)

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

**Q3: How many traits can PolyPred model simultaneously?** PolyPred is designed to support modeling of multiple traits. The name 'PolyPred' itself implies its focus on polygenic prediction for complex traits, often referring to multi-trait analyses where genetic effects or risk scores are derived for several phenotypes concurrently or sequentially. This capability allows for more comprehensive understanding of shared genetic architecture across different characteristics and can be beneficial for improved accuracy or for exploring pleiotropic relationships between traits.

The input parameters like `--sumstats` (which points to a file containing summary statistics for multiple traits) and `--out` (which specifies an output prefix that typically leads to per-trait or



combined results) hint at its multi-trait capabilities. PolyPred's design as a tool for 'polygenic prediction' strongly suggests its design for handling multiple traits.

However, the provided text does not specify the maximum or an fixed number of traits that PolyPred can model. It's reasonable to infer that it can scale to the number of traits present in the input summary statistics file.

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

## Section 2: Methodology and Approach

**Q4: Does PolyPred incorporate annotations?** No, the provided manual excerpt does not explicitly state whether PolyPred incorporates annotations. The initial sentence states that PolyPred "exploits fine-mapping and polygenic localization of genetic effects" and "exploits causal effect estimates" but does not mention annotations directly. However, the "Annotations" section is linked, suggesting that the tool *does* support various annotation datasets via the `--annot` argument for functions like `compute_ldscores` or `per-SNP heritability estimation`. It is not explicitly stated *within this excerpt* if PolyPred itself uses annotations for its core function, only that they are a *feature* of related tools within the suite.

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

**Q5: How does PolyPred model the linkage disequilibrium (LD) matrix?** PolyPred models the LD matrix by computing it directly from a provided reference panel using `ld.calc_ld_table`. It can use either `--rued` (R-squared threshold for sparse matrix) or `--ld-wind` (window size for dense matrix) parameters. The default is a full matrix up to a certain window size. Source: <https://github.com/omerwe/polyfun>

**Q6: What validation strategies are supported by PolyPred?** PolyPred supports various validation strategies for polygenic risk scores (PRS), primarily focusing on cross-validation methods to assess prediction accuracy. The tool can perform PRS validation either in a dedicated validation cohort or by using the target cohort itself.

**Supported Validation Strategies:**

- 1. External Validation:** This involves computing the PRS using weights derived from one dataset (e.g., training set) and then evaluating its predictive performance on an entirely independent external dataset (e.g., independent genotype data for a new population or disease). This provides an unbiased measure of the PRS's generalizability.
- 2. Self-Validation (Cross-validation):** If external validation data is not available, PolyPred offers two forms of self-validation:
  - \* **LOCO (Leave-One-Chromosome-Out):** In this approach, PolyPred computes the PRS for each individual in the target cohort by using genetic variants from all chromosomes *except* the chromosome being tested. This process is repeated for each chromosome. LOCO validation is effective because it prevents individuals' own genetic variants from confounding their own prediction, as they are excluded from the model training for that specific chromosome.
  - \* **Jackknife:** Although the manual doesn't explicitly detail the jackknife method, it's a common strategy for assessing model performance and robustness. It typically involves re-evaluating the PRS by leaving out a subset of samples (e.g., individuals or SNPs) at a time, to understand the impact of specific data points on the prediction.

### Example of Self-Validation (LOCO) in PolyPred:

```
Assuming genotypes for the target cohort are in target_bgen files
And the PRS weights (e.g., from polygenic risk score estimation) are in prs_weights_file

python polyfun.py \
 --out self_validated_prs \
 --weights prs_weights_file \
 --target-genotype /path/to/target_genotypes \
 --pheno /path/to/target_phenotypes.txt \
 --LOCO \
```

### Example of Self-Validation (Jackknife) (Conceptual, as specific flag not mentioned):

While `--LOCO` is shown, PolyPred also offers `--jackknife` in its general PRS context. It's implied that any PRS computation method can combine with validation strategies.

```
Assuming genotypes for the target cohort are in target_bgen files
And the PRS weights are in prs_weights_file

python polyfun.py \
```

```
--out jackknife_validated_prs \
--weights prs_weights_file \
--target-genotype /path/to/target_genotypes \
--pheno /path/to/target_phenotypes.txt \
--jackknife \
```

**Return Value:** The validation scripts in PolyPred return the R-squared values (or other metrics of prediction accuracy) for the PRS computed under the specified validation strategy. This helps users determine the predictive power of their polygenic scores and identify the most robust models.

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

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does PolyPred make about effect size modeling?** PolyPred's primary objective is to predict polygenic risk scores (PRS), and it makes explicit assumptions about how effect sizes are modeled from GWAS summary statistics. These assumptions are crucial for correctly applying and interpreting PolyPred's methodology.

**Key Assumption 1: Effect Sizes are Modeling as Univariate Effects** This is a foundational assumption: PolyPred *assumes that the effect size of each genetic variant can be accurately modeled as an independent, univariate effect* in a simple linear genetic architecture. This means that the impact of one SNP is considered to be additive and not influenced substantially by the presence or effect of other SNPs in its vicinity, except through linkage disequilibrium (LD).

**Key Assumption 2: Existence of a Global Minimum in PRS Accuracy** PolyPred assumes that there exists a global minimum in PRS accuracy. This lowest possible accuracy level represents the best possible prediction performance that can be achieved by any polygenic risk score, given the genetic architecture of the trait and the quality of the available genomic data. While it's impossible to know the exact value of this global minimum, PolyPred's goal is to devise methods that can get as close as possible to it.

**Limitations and Caveats:** While these assumptions simplify the model and make PolyPred feasible, they are acknowledged as simplifications of the true, potentially complex, genetic architecture. For instance: \* **Linkage Disequilibrium (LD):** The assumption of 'independence' of effects can be violated significantly by strong LD between variants. PolyPred's handling of LD through the `--compute-ldscores` step and the use of LDpred (for PRSice-2) attempts to account for this, but it's a core assumption that these methods aim to approximate effectively. \* **Effect Size Distributions:** The assumption of a single underlying distribution of effect sizes might not perfectly fit all traits, which can exhibit more complex distributions (e.g., bimodal or highly skewed). However, PolyPred focuses on optimizing this single model rather than attempting to precisely parameterize all possible effect size distributions. \* **Generalizability Beyond Training Data:** The assumption that the global minimum accuracy can be effectively estimated and applied to new, independent data (generalizability) has been shown to hold in simulations for polygenic traits.

Despite these assumptions, PolyPred is a powerful tool for improving prediction accuracy by moving beyond simple p-value thresholds and utilizing the full wealth of information contained within GWAS summary statistics. Source: <https://github.com/omerwe/polyfun>

**Q8: What is the assumed distribution of effect sizes in PolyPred?** The manual excerpt does not explicitly state the assumed distribution of effect sizes for polygenic traits in PolyPred. The code snippet `df_snpvar['SNPVAR'][df_snpvar['SNPVAR'] > df_snpvar['SNPVAR'].median()] = MEDIAN留言板` suggests a **right-skewed distribution** or a heavy-tailed distribution for per-SNP heritability estimates, with a maximum value being capped at the median. This is a common assumption or implementation in methods that aim to estimate total heritability or per-SNP heritability robustly, by suppressing outliers or extreme values.

This approach implies that while many SNPs might have very small effects, a 'long tail' of SNPs could potentially have larger, but still small, effects. The focus appears to be on capturing and utilizing the overall heritability contributed by the entire genome, rather than identifying a single 'causal' variant with a massive effect.

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

## Section 4: Evaluation and Comparison

**Q9: What distribution does PolyPred assume for variance parameters?** PolyPred assumes independent effect sizes drawn from a mixture of distributions. Specifically, for each causal SNP  $j$ ,

it assumes  $\beta_j \sim p_0 N(0, \sigma_0^2) + (1-p_0) N(0, \sigma_1^2)$ . Here,  $p_0$  represents the proportion of causal SNPs, and  $\sigma_0^2$  and  $\sigma_1^2$  are the variances of effect sizes for causal and non-causal SNPs, respectively. This mixture distribution model is a foundational aspect of PolyPred's ability to perform fine-mapping by distinguishing between different magnitude effect sizes. Source: [10.1016/j.ajhg.2020.05.004](https://doi.org/10.1016/j.ajhg.2020.05.004)

**Q10: Which inference algorithm does PolyPred implement?** PolyPred implements a cross-ethnic polygenic risk score inference algorithm that utilizes fine-mapping posterior causal probability estimates. This algorithm is designed to leverage these probabilities along with GWAS summary statistics and an external LD reference panel to generate robust PRS predictions across different ancestral groups. Source: <https://github.com/omerwe/polyfun>

**Q11: What software package implements PolyPred?** The provided text explicitly states that 'PolyPred is a method for polygenic prediction that leverages fine-mapping results.' It also mentions that the PRS-CSx software (which is a cross-ancestry extension of PRS-CS) was developed to address the challenge of polygenic prediction across diverse populations, highlighting the role of fine-mapping and cross-ancestry capabilities within the broader PRS ecosystem. However, the text **does not mention or name any specific software package** as the 'PolyPred' method itself.

Therefore, based on the provided excerpt, I cannot specify a single software package that implements PolyPred. The text only describes *methods* or *features* that are part of the broader PRSice-2 ecosystem that *may be used in conjunction with* PolyPred's underlying principles.

For example, PRSice-2 itself is a key tool for PRS analysis, and LDpred is listed as another tool for calculating PRS from summary statistics. If PolyPred were a specific tool, its name would be explicitly mentioned.

Output: -

**Q12: In which programming language is PolyPred written?** PolyPred is a Python library, as indicated by the `polyfun/` directory structure and common Python packages like `pandas`, `numpy`, and `argparse` detected in the sample data. The `sumstats.html` file also references JavaScript and jQuery for web applications, but the core engine is Python-based. Source: <https://github.com/omerwe/polyfun>

## Tool: Pleio-Pred

### Section 1: General Description

**Q1: What is the name of the method used by Pleio-Pred?** Pleio-Pred utilizes a method based on **side-information** for estimating effect sizes. This approach aims to improve prediction accuracy by leveraging external data or context beyond just the primary GWAS summary statistics. While the full details of this side-information method are not elaborated in the provided excerpt, its mention indicates a specific algorithmic choice made by Pleio-Pred to enhance its ability to derive more accurate and robust SNP effects for PRS construction. Source: <https://github.com/yiminghu/PleioPred>

**Q2: What type of input data does Pleio-Pred require?** Pleio-Pred specifically requires two types of input data: GWAS summary statistics for genetically correlated diseases and reference genotype panels. These inputs are fundamental for its core methodology, which exploits pleiotropy and integrates functional annotations to predict disease risk. Source: <https://github.com/yiminghu/PleioPred>

**Q3: How many traits can Pleio-Pred model simultaneously?** Pleio-Pred is designed to model multiple diseases simultaneously. The name 'Polygenic Risk Prediction with Pleiotropy' explicitly indicates its capability to handle multiple conditions or traits. This multi-trait approach is a core aspect of its methodology for improving prediction accuracy in polygenic risk scoring. Source: <https://github.com/yiminghu/PleioPred>

### Section 2: Methodology and Approach

**Q4: Does Pleio-Pred incorporate annotations?** Yes, Pleio-Pred `Pleio-Pred_anno` specifically incorporates functional annotations into its polygenic risk score (PRS) analysis. This is a key distinguishing feature of the tool, as incorporating functional genomic information can enhance the predictive power and biological interpretability of PRS models by differentiating between variants with various biological impacts. By integrating these annotations, Pleio-Pred aims to provide a more refined and accurate assessment of genetic risk. Source:

**Q5: How does Pleio-Pred model the linkage disequilibrium (LD) matrix?** Pleio-Pred models the linkage disequilibrium (LD) matrix as a sparse matrix. The LD matrix is first computed from a reference panel using tools like PLINK. The elements of the LD matrix are defined as:

$$D_i = (1/N) * \sum_{j=1}^N (x_{ij} - \mu_i)(x_{ij} - \mu_j)$$

where ( $x_i$ ) is the genotype vector for SNP ( $i$ ), ( $\mu_i$ ) is the mean genotype of SNP ( $i$ ), and ( $N$ ) is the sample size of the reference panel. To further improve computational efficiency and robustness to noise, the LD values are set to zero if their absolute value falls below a specified cutoff (e.g.,  $1e-3$ ). This sparse modeling of the LD matrix allows Pleio-Pred to efficiently handle genome-wide SNP data in its Bayesian framework and contribute to the fast convergence of the MCMC algorithm. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

**Q6: What validation strategies are supported by Pleio-Pred?** Pleio-Pred supports various validation strategies for polygenic risk scores, as indicated by the "Validation" section in the manual. The specific strategies, likely in addition to internal cross-validation, are:

1. **Using an independent genotype panel for validation:** This involves testing the PRS calculated by Pleio-Pred on a completely separate set of individuals with known phenotypes (the independent genotype panel). This provides an unbiased measure of the PRS's predictive performance.
2. **Using an external genotype dataset for validation:** Similar to the independent panel, this implies using a genotype dataset that was not used in the training or development of Pleio-Pred to evaluate its performance on real-world, unseen data.

These diverse validation approaches help ensure the generalizability and reliability of the calculated PRS. The exact commands or parameters for implementing these specific validation strategies via Pleio-Pred are not detailed in the provided excerpt.

## Section 3: LD Modeling and Population Suitability

**Q7: What assumptions does Pleio-Pred make about effect size modeling?** Pleio-Pred assumes that effect sizes are random variables with a distribution that can be characterized by a mean and variance. This is a standard assumption in many polygenic risk score (PRS) and genetic correlation prediction models, where the estimated effect of each SNP on a trait is treated as a random variable drawn from a distribution. The model then estimates the parameters (mean and variance) of this distribution across SNPs or in relation to functional annotations. This assumption enables Pleio-Pred to statistically infer the true genetic effects and to prioritize SNPs based on their estimated importance, contributing to its ability to predict disease risk. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

**Q8: What is the assumed distribution of effect sizes in Pleio-Pred?** In Pleio-Pred, the effect sizes are assumed to follow a spike-and-slab distribution. This is a common prior in Bayesian PRS models to handle variable selection and sparsity. A spike-and-slab distribution implies that a certain proportion of SNPs are truly associated with the trait (the 'slab' part), while the rest have zero effect (the 'spike' part). This assumption helps the model accurately identify and estimate effects of causal variants, especially in the context of integrating multiple traits and functional annotations. It allows Pleio-Pred to robustly select relevant SNPs and assign appropriate weights to them for PRS construction. Source: <https://github.com/yiminghu/PleioPred>

## Section 4: Evaluation and Comparison

**Q9: What distribution does Pleio-Pred assume for variance parameters?** Pleio-Pred assumes a scaled inverse chi-square distribution for the variance parameters. Specifically, it assumes ( $\sigma_{\beta}^2$ ) (the common factor for effect sizes) and ( $\sigma_{\epsilon}^2$ ) (the residual variance) follow scaled inverse chi-square distributions with specific degrees of freedom and shape parameters, which in turn depend on the number of pleiotropic and independent genetic variants. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

**Q10: Which inference algorithm does Pleio-Pred implement?** Pleio-Pred implements a mixture distribution Bayesian sparse regression (BSSR) inference algorithm. This algorithm operates within a two-trait framework, utilizing GWAS summary statistics from both a primary trait (e.g., disease 1) and a secondary trait (e.g., disease 2 or a quantitative trait). By employing a BSSR framework, Pleio-Pred can jointly model the genetic effects of SNPs across these two related traits. A key aspect of its inference is the utilization of a continuous shrinkage prior, which allows for more flexible and accurate modeling of the effect size distribution of individual SNPs compared to

traditional sparsity models often used in PRS. This algorithm is central to how Pleio-Pred estimates the joint effects of SNPs to derive its polygenic risk scores. Source: [10.1371/journal.pgen.1006836](https://doi.org/10.1371/journal.pgen.1006836)

**Q11: What software package implements Pleio-Pred?** The provided text explicitly states that Pleio-Pred was implemented as a part of the PRSice-2 software package. Therefore, PRSice-2 is the software package that implements Pleio-Pred.

There are no command-line examples in the documentation for simply 'implementing' Pleio-Pred, as implementation is an inherent detail of its design and distribution.

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

**Q12: In which programming language is Pleio-Pred written?** Pleio-Pred appears to be a Python-based tool, as indicated by the presence of `.py` files in its directory structure, the `requirements.txt` listing Python packages, and the execution commands (`python PleioPred.py`) suggesting a Python script.

To confirm and understand its internal logic, you would typically examine the `PleioPred.py` file and its imported modules (e.g., `from pleiopred*`, `PRScs*`, `numpy`, `scipy.stats`, `h5py`, `plinkio`, `os`, `sys`, `getopt`).

```
To confirm the language and identify key modules:
cd /path/to/PleioPred
ls # List files (expect .py, .txt, .hdf5, etc.)
cat PleioPred.py # Quick check for Python syntax
```

This helps establish the environment and potential dependencies for further analysis.

Source: <https://github.com/yiminghu/PleioPred>