GARSA was develped using Python3 and aims to semi-automate the main steps for Genome Wide Association Study and Polygenic Risk Score analysis

> For now, GARSA only works with Linear Mixed Models (LMM) for populations with continuous traits and related samples.
> We aim to improve GARSA functionalities over time, requests and bug reports should be made in the *Issues* session on this page.

```
Dependencies:

1. Python3
    -- pandas 1.4.3+
    -- os
    -- matplotlib 3.5.2+
    -- subprocess
    -- numpy 1.22.3+
    -- sys
    -- argparse
    -- textwrap
    -- shutil
    -- time
    -- gzip
    -- seaborn 0.11.2
    -- assocplots 0.0.2 --> pip3 install
https://github.com/khramts/assocplots/archive/master.zip
    -- scipy 1.7.3

2. R 4.1.2
    -- SeqArray
    -- SNPRelate
    -- ggplot2
    -- GENESIS
    -- dplyr
    -- SeqVarTools
    -- rgl
    -- tidyr
    -- reshape
    -- genio
    -- tidyverse
    -- tibble
    -- optparse
    -- bigsnpr

3. Pre-installed in the GARSA-env
    -- Plink
    -- Plink2
    -- BCFTools 1.15.1
4. Tools precompiled and available with the pipeline
    -- AdMixture
    -- FlashPCA
    -- GCTA
    -- Bolt-lmm
    For those tools it is only necessary to provide a path if the provided
precompiled tool is not working
```

Instalation

1. Download or clone this repository: `git clone https://github.com/LGCM-OpenSource/GARSA.git`

2. Install GARSA dependencies with the provided conda environments (.yml) located at ~/GARSA/modules/scripts/

```
#GARSA environment
conda env create -f GARSA-env.yml

#bolt-lmm environment
conda env create -f boltlmm.yml
```

> **Note**
> Remember to activate the environment before running the GWAS module with the bolt-lmm flag

4. Install R and some python dependencies in the GARSA environment

```
conda activate GARSA-env

pip3 install https://github.com/khramts/assocplots/archive/master.zip

R -q -e 'install.packages("genio", repos="http://cran.us.r-project.org",
dependencies=T)'

R -q -e 'remotes::install_github("privefl/bigsnpr")'
```

Databases

We provide the necessaire datasets for Bolt-lmm, LDPred2 execution, alongside the long-range LD prunning data for both hg37 and hg38.

Also, for large database downloads we provide the script `download_db`
This script will download the dbSNP database (version b151) in VCF format and the corresponding indexing file (.tbi)

The downloaded data is placed in the `~/GARSA/databases` folder

Database	Size

Database	Size
dbSNP Hg37	15.7 Gb
dbSNP Hg38	16.3 Gb
Bolt-lmm data	101.2 Mb
LDPred2 data	45,9 Mb
long-range regions Hg37	732 b
long-range regions Hg38	888 b

Usage

Once all the dependencies and envrionments are in place, running the GARSA pipeline should be relatively simple.
The main GARSA module (GARSA.py) wraps all the available modules, which can be called independently.

> **Note** After GARSA instalation run `chmod +x GARSA.py` from the main GARSA folder to make it executable
> GARSA can be added to your path by adding `export PATH=path/to/GARSA:$PATH` to your .bashrc or .bash_profile file in your *home* folder

```
usage: GARSA.py [-h]

This script integrates each analysis of the GARSA pipeline
______________________________________________________________
desdup          -- Runs the desduplication analysis, removing duplicated
SNPs or multiallelic variants
update_rsID     -- Runs the update of all (possible) rsIDs using hg19 or
hg38 references
rename_sample_id    -- Runs an update of samples ID
quality_control     -- Runs the quality control script for SNPs
quality_ind     -- Runs Quality control for individuals with missing data
or high heterozygosity
kinship         -- Runs Kinship analysis and correction for admixed
populations
PCA         -- Runs PCA and population analysis
GWAS            -- Runs GWAS analysis using GCTA or BOLT-LMM software
PRS             -- Runs PRS analysis using LDPred2

optional arguments:
  -h, --help  show this help message and exit
```

The pipeline can be executed in any order and the inputs do not need to be generated by the pipeline. The user only need to be carfull with the correct formating for inputs.

Each GARSA module is run separately. For exemple, a simple SNP quality control run goes as follows:

```
GARSA.py quality_control -vcf path/to/data.vcf.gz --score_type r2 -o
output/folder/path --threads 4
```

Detailed usage for each module

Advices:

Before the analysis starts, we recommend to check which individuals have the desired phenotype (to be used in the GWAS) and filter the dataset -- e.g. keep genotype data only for the samples with phenotype data.

This will prevent errors in the GWAS and PRS modules. Both those modules require the phenotype data for all individulas in the analysis for correct execution.

This step requires a list of all samples containing the desired phenotype formatted as *FID IID (or IID IID)*. Where FID and/or IID are sample ID.

After all preparaionts simply run: `plink1.9 --vcf file.vcf --keep list_of_samples_with_phenotype.txt --recode vcf bgz --out selected_dataset`

IMPORTANT:

GARSA uses plink for many of the pipeline steps, and so it is necessary that your VCF is correctly formatted as FID_IID (or IID_IID).

For this we provide the module **rename_sample_id**, this module should be used for corret formatting of your VCF.

For this module to work, the user need to provide a table formatted as OLD SAMPLE ID NEW SAMPLE ID as follows:

OLD	NEW
sample1	FID1_IID1 or IID1_IID1
sample2	FID2_IID2 or IID2_IID2
...	...
sampleN	FIDN_IIDN or IIDN_IIDN

From this simply run `GARSA.py rename_sample_id -vcf path/to/data.vcf.gz -table path/for/formatted_table.csv -o output/folder/path`

The output VCF will be written to the provided output path with the added suffix *.Nsamples*

Module options

```
usage: rename_sample_id.py [-h] -vcf VCF_FILE -table SAMPLE_TABLE [-
bcftools BCFTOOLS_PATH] [-o OUTPUT_FOLDER] [--threads THREADS]

This script updates the Sample IDs of a VCF file, for this the user must
provide a tab or comma separated file with Old sample ID on the first
column and the New sample ID in the seconda column

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -table SAMPLE_TABLE, --sample_table SAMPLE_TABLE
                        File with OLD_SAMPLE_ID<tab>NEW_SAMPLE_ID
  -bcftools BCFTOOLS_PATH, --bcftools_path BCFTOOLS_PATH
                        Path for the bcftools executable, requierd for
script execution -- default is to look for the variable on path
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  --threads THREADS     Number of computer threads -- default = 1
```

Deduplication Module (desdup)

This module will remove all non-biallelic and duplicated variants. This step ensures that there are no multiallelic variants, that are not supported in GWAS and PRS analysis.

In this step the user should provide only a VCF file, correctly formatted like described before.

`GARSA.py dedup -vcf path/to/data.vcf.gz -o output/folder/path`

Module options

```
usage: deduplication.py [-h] -vcf VCF_FILE [-bcftools BCFTOOLS_PATH] [-o
OUTPUT_FOLDER] [-plink2 PLINK2_PATH] [--threads THREADS]

This is a script to identify and remove duplicated SNPs

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -bcftools BCFTOOLS_PATH, --bcftools_path BCFTOOLS_PATH
                        Path for the bcftools executable, requierd for
script execution -- default is to look for the variable on path
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  -plink2 PLINK2_PATH, --plink2_path PLINK2_PATH
                        Path for the plink2 executable, requierd for script
```

```
execution -- default is to look for the variable on path
  --threads THREADS     Number of computer threads -- default = 1
```

Update rsIDs module (update_rsID)

This module will annotate the users variants using the dbSNP dataset. The user should provide a VCF file and the correct human assembly (hg37 or hg38) version. In this step GARSA will attempt to flip and swap variants in relation to the reference to guarantee correct annotations.

`python3 GARSA.py update_rsID -vcf path/to/data.vcf.gz -ref_hg hg38 -o output/folder/path`

Module options

```
usage: update_rsID.py [-h] [-vcf VCF_FILE] [-ref_hg REF_BUILD] [-bcftools
BCFTOOLS_PATH] [-plink2 PLINK2_PATH] [-plink PLINK_PATH] [-o OUTPUT_FOLDER]
[-rm_tmp] [--threads THREADS]

This is a script to update SNP rsIDs (for hg19). This script assumes that
your file name have the pattern chr[1-22], e.g
project_name_chr12.extensions

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -ref_hg REF_BUILD, --ref_build REF_BUILD
                        Select the human genome build version -- hg37 or
hg38, default=hg37
  -bcftools BCFTOOLS_PATH, --bcftools_path BCFTOOLS_PATH
                        Path for the bcftools executable, requierd for
script execution -- default is to look for the variable on path
  -plink2 PLINK2_PATH, --plink2_path PLINK2_PATH
                        Path for the Plink2 executable, requierd for script
execution -- default is to look for the variable on path
  -plink PLINK_PATH, --plink_path PLINK_PATH
                        Path for the Plink1.9 executable, requierd for
script execution -- default is to look for the variable on path
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  -rm_tmp, --rm_temp_files
                        Force keeping temporary files (Files may be quite
large) -- default: Delete temporary files
  --threads THREADS     Number of computer threads -- default = 1
```

The outputed VCF will have the suffix ***.RSIDs.vcf.gz**

Variant quality control

In this module GARSA will perform variant quality control, using user provided parameters. It is possible to filter imputed data based on both INFO scores, and R2 scores (usually the output of the Michigan Imputation Server - MIS). Also, we provide a flag for the usage o Hanrdy-Weinberg equilibrium, depending on the dataset it should be applied, but varies from study to study (see more here).

Main flags for variant quality filtering

-geno --> Controls the amount of missing data allowed per variant (default is 5% (e.g. 0.05)) -maf --> Controls the Minor Allel Frequency threshold for the population (default is 1% (e.g. 0.01)) -HWE (combined with --use_hardy) --> Controls the threshold of significance for Hardy-Weinberg desiquilibrium (default is 1e-6) -R2 or -INFO (combined with --score_type) --> Controls the thersold for imputation score

`python3 GARSA.py quality_control -vcf path/to/data.vcf.gz -geno 0.05 -maf 0.01 --use_hardy -HWE 1e-6 --score_type r2 -R2 0.8`

```
usage: SNP_QC.py [-h] -vcf VCF_FILE [-bcftools BCFTOOLS_PATH] [-plink2
PLINK2_PATH] [-o OUTPUT_FOLDER] [-geno GENO_PLINK] [-maf MAF_PLINK] [-HWE
HARDY] [-use_HWE] [-R2 R_SQUARED] [-INFO INFO_SCORE]
                 [--score_type SCORE_TYPE] [--no_score] [--threads THREADS]

This is a script runs standard QC process for imputed datasets

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -bcftools BCFTOOLS_PATH, --bcftools_path BCFTOOLS_PATH
                        Path for the bcftools executable, requierd for
script execution -- default is to look for the variable on path
  -plink2 PLINK2_PATH, --plink2_path PLINK2_PATH
                        Path for the plink2 executable, requierd for script
execution -- default is to look for the variable on path
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  -geno GENO_PLINK, --geno_plink GENO_PLINK
                        Threshold value for SNP with missing genotype data
-- default=0.05
  -maf MAF_PLINK, --maf_plink MAF_PLINK
                        Threshold value for minor allele frequency (MAF) --
default=0.01
  -HWE HARDY, --hardy HARDY
                        Check for SNPs which are not in Hardy-Weinberg
equilibrium (HWE) -- default=1e-6
  -use_HWE, --use_hardy
                        Define if the HWE analysis will be run --
default:False
  -R2 R_SQUARED, --r_squared R_SQUARED
                        Imputation r-squared threshold value -- default >=
0.8 (Use this flag when dataset was imputed using MIS (Michigan Imputation
Server))
  -INFO INFO_SCORE, --INFO_SCORE INFO_SCORE
                        Imputation INFO score threshold value -- default >=
0.5 (Use this flag when dataset was imputed using IMPUTE5)
  --score_type SCORE_TYPE
                        Select r2 or info for imputation score filter --
default: r2
  --no_score            Dataset with no imputation score -- default: False
  --threads THREADS     Number of computer threads -- default = 1
```

The output from this module recieves the suffix *.R2andQC

Befor the next steps we recommend that the user concatenate all chromosomes into one VCF file. Sugestion --> `bcftools concat -Oz -o concatenated_file.vcf.gz chr{1..22}.vcf.gz`

This suggestion for concatenation aims to guarantee correct sample quality control, and avoid generating chromosome files with different samples filtered

Sample quality control

This module of GARSA will filter samples based on missingness and heterozygosity rates. Missingness will check for individuals with high missing genotype rates. High heterozygosity rates indicates low sample quality, and low rates might be related to inbreeding. With this, the heterozygosity is calcualted for all samples and the samples that deviates 3 standard deviations from the mean are filtered out.

`python3 GARSA.py quality_ind -vcf path/to/data.vcf.gz -mind 0.1`

```
usage: sample_QC.py [-h] -vcf VCF_FILE [-plink PLINK_PATH] [-mind
MIND_PLINK] [--threads THREADS] [-o OUTPUT_FOLDER]

This is a script runs standard QC process for imputed datasets

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -plink PLINK_PATH, --plink_path PLINK_PATH
                        Path for the plink(1.9) executable, requierd for
script execution -- default is to look for the variable on path
  -mind MIND_PLINK, --mind_plink MIND_PLINK
                        Threshold value for individuals with missing
genotype data -- default=0.1
  --threads THREADS     Number of computer threads -- default = 1
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
```

The output of this module recieves the suffix *.MIND.HET

Kinship with correction for admixed populations

This is a module that is suitable only for users with admixed populations (like the Brazilian population), other populations (like UKBB) can use the regular kinship calculations in the GWAS step using the flag --*make_king* when using GCTA. Otherwise, BOLT-LMM will automatically calculate the kinship matrix.
The input VCF is LD prunned before the start of the kinship calculations.
In this analysis the kinship values are calculated by the *SNPRelate* library in R and then corrected using PCA for population structure corretion.
For more details check Conomos et. al (2016).

`python3 GARSA.py kinship -vcf path/to/data.vcf.gz --degree 2`

```
usage: Kinship_and_correction.py [-h] -vcf VCF_FILE [-plink PLINK_PATH] [-o
OUTPUT_FOLDER] [--window_size WINDOW_SIZE] [--sliding_window_step
SLIDING_WINDOW_STEP] [--prune_r2 PRUNE_R2] [--degree DEGREE]
                                 [--threads THREADS]

This is a script to run kinship analysis and correct the values using
population stratification

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -plink PLINK_PATH, --plink_path PLINK_PATH
                        Path for the plink(1.9) executable, requierd for
script execution -- default is to look for the variable on path
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  --window_size WINDOW_SIZE
                        Window size for prunning step -- default = 1000
  --sliding_window_step SLIDING_WINDOW_STEP
                        Sliding Window step -- default = 50
  --prune_r2 PRUNE_R2   R2 value for prunning-- default = 0.03
  --degree DEGREE       Degree for relatedeness (INT --> 1, 2 or 3) --
default = 2nd degree [2]
  --threads THREADS     Number of computer threads -- default = 1
```

Here 3 main outputs are generated and can be analyzed by the user:

1. Kinship_corrected.tsv --> Kinship table (all against all) with corrections for admixed population
2. RKinship_for_grm.grm.id e RKinship_for_grm.grm.bin --> Required input files for GWAS module, if using the *-gcta* flag
3. Related_at_degree[1,2 or 3].txt --> File with all related individuals, necessary for the *PCA module*

PCA module

This module runs PCA analysis in 5 main steps:

1. Uses FlashPCA on the unrelated dataset (generated in the kinship module) and outputs, alongside the PCs for each sample, laoding values for each SNP
2. From the loadings, search for outlier SNPs that might introduce bias to the analysis and remove those
3. After that, run a new PCA analysis without the outlier SNPs in the same unrelated population --> In this step, GARSA tries to ensure PCA values with reduced bias. For more information refer to Florian Privé, 2017
4. Project the PCs on the related dataset
5. Run a "DeNovo" admixed analysis for identification of best N of populations --> this generates a colored graphical output that the user can check for the number of informative PCs

In this step, it is necessary to provide the file ***Related_at_degree[1,2 or 3].txt***. If the user opted not to run the *kinship module*, the necessary file is formated as a tabular file (separeted by "\t") with no header. The first column contains the FID and the second the IID (or IID and IID) of the realted samples found in the kinship analysis of choice.

Example:

```
Sample1<\t>Sample1
Sample2<\t>Sample2
Sample2<\t>Sample2
...
SampleN<\t>SampleN

or

Family1<\t>Sample1
Family1<\t>Sample2
Family2<\t>Sample1
...
FamilyN<\t>SampleN
```

The ***Related_at_degree[1,2 or 3].txt*** or the user provided file explained above should be passed in the flag *-related*

The flag *--garsa_path* should be ignored. It is a workaround some limitations of the ***Admixture tool***

`python3 GARSA.py PCA -vcf path/to/data.vcf.gz -realted Related_at_degree2.txt`

```
usage: PCA_analysis.py [-h] -vcf VCF_FILE [-plink PLINK_PATH] [-o
OUTPUT_FOLDER] -related RELATED_FILE [--window_size WINDOW_SIZE] [--
sliding_window_step SLIDING_WINDOW_STEP] [--prune_r2 PRUNE_R2]
                       [--threads THREADS] [--garsa_path GARSA_PATH]

This script runs PCA for non-related individualas and projects to related
individuals

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for processing, requierd for script execution
  -plink PLINK_PATH, --plink_path PLINK_PATH
                        Path for the plink(1.9) executable, requierd for
script execution -- default is to look for the variable on path
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  -related RELATED_FILE, --related_file RELATED_FILE
                        File from the kinship module with all related
individuals
  --window_size WINDOW_SIZE
                        Window size for prunning step -- default = 1000
  --sliding_window_step SLIDING_WINDOW_STEP
                        Sliding Window step -- default = 50
  --prune_r2 PRUNE_R2   R2 value for prunning-- default = 0.03
  --threads THREADS     Number of computer threads -- default = 1
  --garsa_path GARSA_PATH
                        Path to main script GARSA -- always provided by
default
```

For this step there are 4 main outputs:

1. table_for_plot.tsv --> Table with PC information, predicted population and Sample ID for all samples
2. <file_name>_PCA_total.txt --> Output with all the information about the PCs for related and unralted samples
3. PC_plots_PCA1.pdf --> File with all PCA plots for user visualization
4. R_kinship_for_GRM.grm.bin and R_kinship_for_GRM.grm.id --> The user can provide this file in the GWAS module if using gcta

GWAS module

In this step the user must provide a phenotype, covariates (covar), quantitative covariates (qcovar) and kinship files for correct execution.
It is possible to provide only covar *OR* qcovar files, but the phenotype file is mandatory.

Those files must be formated as shown below, and can be used in both GCTA and BOLT-LMM strategies.

We recommend the use of GCTA for populations with sample size lower than 5000 individuals and BOLT-LMM for populations with sample size higher than 5000 individuals

The flag *--bh_correction* uses Benjamini-Hochberg p-value correction for multiple tests, even if some work uses this FDR correction it may overcorrect the p-values. So we recommend *NOT* using this falg, and by default GARSA will perform p-value correction using ***Genomic Inflation***.

For the BOLT-LMM run, GARSA will generate a list of *Independent SNPS* using LD-Prunning from the plink1.9 tool with the following parameters `--indep-pairwise 50 5 0.08`

If the user wishes to provide its own list of prunned SNPs, plesase do so using the flag *--independent_snps*. The accepted file uses the same format as the plink *.prune.in output file. This is, one SNP ID per line in a text file. *If using plink simply provide the .prune.in file* This allows the user to control what they consider as being Independent SNPs.

> **Note Important to notice that BOLT-LMM calculates it's own kinship matrix and do not provide a way to input the corrected one calculated above.**
> **For heritability estimates check the log from either BOLT or GCTA runs**

Phenotype file:

FID	IID	phenotype
10001	10001	117
10002	10002	83

Just like the files from Plink, we use as input the FID on the first column and IID on the second. That is the reason for the FID_IID format mentioned above. This pattern is kept during the whole analysis.

covariable file:

In this example only "sex" is used as qualitative covariable. Important: For this analysis, if the user whish to use more covariables, order them after the "sex" covariable on the file (keeping it on the third columns as showed below).

FID	IID	Sex
10001	10001	1
10002	10002	2

qcovar file (quantitative covariable): On this file, we use the PCs generated above, file *<file_name>_PCA_total.txt*. This file is already correctly foramted.
If the user wishes, more quantitative covariables can be added to this file.

FID	IID	PC1	PC2	PC3
10001	10001	0.06	-0.07	0.01
10002	10002	0.009	-0.1	0.008

Example using both covar and qcovar files

> **Note Important to reamember is that the user can select either '-gcta' or '-BoltLmm'**
> *Also the gcta mode for GWAS accepts the corrected kinship matix using the flag -kinship and providing the path for the generated .grm. files. The user should provide only the prefix of the files, without the extension .grm.*.*

> **Note**
> Remember to activate the *boltlmm* environment before running the GWAS module with the bolt-lmm flag --> `conda activate boltlmm`

`python3 GARSA.py GWAS -vcf path/to/data.vcf.gz -pheno phenotype_file.tab -qcovar data_PCA_total.txt -covar covar_file.tab -gcta -kinship R_kinship_for_GRM`

Example using only the qcovar file

`python3 GARSA.py GWAS -vcf path/to/data.vcf.gz -pheno phenotype_file.tab -qcovar data_PCA_total.txt -gcta -kinship R_kinship_for_GRM`

Example using only the covar file

> **Warning Not using the qcovar file migth introduce bias related to popualtion structure in the analysis!**

`python3 GARSA.py GWAS -vcf path/to/data.vcf.gz -pheno phenotype_file.tab -covar covar_file.tab -gcta -kinship R_kinship_for_GRM`

Example running the GCTA kinship calculation (no correction applied)

`python3 GARSA.py GWAS -vcf path/to/data.vcf.gz -pheno phenotype_file.tab -qcovar data_PCA_total.txt -covar covar_file.tab -gcta --make-king`

```
usage: GWAS.py [-h] [-vcf VCF_FILE] [-plink PLINK_PATH] [-bfile
PLINK_BINARY_PREFIX] [-pheno PHENOTYPE_FILE] [-qcovar QUANTITATIVE_COVAR]
[-covar COVAR] [-kinship KINSHIP_GRM] [--make_king] [-o OUTPUT_FOLDER]
               [-gcta] [-ind_snps INDEPENDENT_SNPS] [--bh_correction] [-
BoltLmm] [-BoltLD BOLTLD_FILE] [--threads THREADS]

This is a script to GWAS analysis and plot the results with Manhattam plot

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for GWAS analysis, required if user dont have
Plink binary files
  -plink PLINK_PATH, --plink_path PLINK_PATH
                        Path for the plink(1.9) executable, requierd with -
vcf flag -- default is to look for the variable on path
  -bfile PLINK_BINARY_PREFIX, --plink_binary_prefix PLINK_BINARY_PREFIX
                        Path for the plink(1.9) binary file, provide only
the prefix (no extensions)
  -pheno PHENOTYPE_FILE, --phenotype_file PHENOTYPE_FILE
                        Path for the phenotype file, this file must have
FID and IID (like the .fam file) and must be separated by tab or space.
Header is not mandatory
  -qcovar QUANTITATIVE_COVAR, --quantitative_covar QUANTITATIVE_COVAR
                        Path for the quantitative covariables, e.g. PCs,
age, and other continuous variables. The file must have FID and IID (like
the phenotype file and .fam. The file must be separated by tab
                        or space. Header is not mandatory
  -covar COVAR, --covar COVAR
                        Path for the covariables, e.g. Sex and other
```

```
qualitative variables. The file must have FID and IID (like the phenotype
file and .fam. The file must be separated by tab or space. Header
                        is not mandatory
  -kinship KINSHIP_GRM, --kinship_grm KINSHIP_GRM
                        Path for the kinship grm file generated by the
kinship script, if user wishes the kinship analysis can be generated with
the flag --make-king
  --make_king           Make the kinship analysis (no correction by
admixture)
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  -gcta, --gcta_run     Select gcta analysis for GWAS -- recomended for N
sample < 5000
  -ind_snps INDEPENDENT_SNPS, --independent_snps INDEPENDENT_SNPS
                        File made by plink --indep-pairwise with extension
*.prunne.in containing only independent SNPs for analysis -- Default is to
run --indep-pairwise 50 5 0.08
  --bh_correction       Select the p-value correction by Benjamini-
Hochberg. Used for big populations (> 100.000) -- Use this flag to select
to use BH correction, the default is to correct by Genomic Inflation
  -BoltLmm, --BoltLmm_run
                        Select Bolt-lmm for GWAS -- recomended for N
samples > 5000
  -BoltLD BOLTLD_FILE, --BoltLD_file BOLTLD_FILE
                        Path for the Bolt-lmm LD file -- default: File
provided by the BOLT-LMM distribution
  --threads THREADS     Number of computer threads -- default = 1
```

The output from the GWAS analysis goes through a p-value corretion using Genomic Inflation (λgc)

Generated outputs:

1. GWAS_summary_adjusted_pvalues.csv --> Sumarry statistics with corrected p-value using λgc
2. A file with **.mlma** extension with the summary statistics from the GCTA run / A file with **.stats** extension with the summary statistics from the BOLT-LMM run
3. Manhattam plot
4. QQ plot

The file *GWAS_summary_adjusted_pvalues.csv* can be passed to FUMA web application for variant annotation, and other functional analysis.

Polygenic Risk Score - PRS

> **Warning This step requires an independent dataset from the one used in the GWAS step**
> To generate the Principal component analysis to use in this step, please refer to the PCA module

This module performs the reestimation of betas calculated in the GWAS step (present in the summary statistics file, either *.mlma* or *.stats*).
This file is provided using the flag *-mlma*, if the output is from BOLT-LMM the user should also provide the

flag *--BOLT*
This module was mostly implemented in R using the library ***bigsnpr***

On this step the user should also provide all the files (covar, qcovar and phenotype) similar to the ones used in the GWAS module. Besides that, the user must provide the column name (header) of the desired phenotype on the phenotype file.

Example usage:

`python3 GARSA.py PRS -vcf path/to/independent_data.vcf.gz -mlma GCTA_summary_statistics.mlma -pheno phenotype_file.tab -qcovar data_PCA_total.txt -covar covar_file.tab --pheno_col SBP`

```
usage: LDPred_PRS.py [-h] [-vcf VCF_FILE] [-plink PLINK_PATH] [-plink2
PLINK2_PATH] [-bfile PLINK_BINARY_PREFIX] -mlma GWAS_MLMA [--BOLT] [-pheno
PHENOTYPE_FILE] [--pheno_col PHENO_COL]
                     [-qcovar QUANTITATIVE_COVAR] [-n_pcs NUMBER_OF_PCS] [-
covar COVAR_FILE] [-o OUTPUT_FOLDER] [--threads THREADS]

This is a script to GWAS analysis and plot the results with Manhattam plot

optional arguments:
  -h, --help            show this help message and exit
  -vcf VCF_FILE, --vcf_file VCF_FILE
                        File for PRS analysis, required if user dont have
Plink binary files (Same file as used for GWAS)
  -plink PLINK_PATH, --plink_path PLINK_PATH
                        Path for the plink(1.9) executable -- default is to
look for the variable on path
  -plink2 PLINK2_PATH, --plink2_path PLINK2_PATH
                        Path for the Plink2 executable, requierd for script
execution -- default is to look for the variable on path
  -bfile PLINK_BINARY_PREFIX, --plink_binary_prefix PLINK_BINARY_PREFIX
                        Path for the plink(1.9) binary file, provide only
the prefix (no extensions) -- Same used in the GWAS setp
  -mlma GWAS_MLMA, --GWAS_mlma GWAS_MLMA
                        Output file from de GWAS step -- the extension of
this file is .mlma for GCTA and .stats for BOLT-LMM
  --BOLT                Use this flag if the BOLT-LMM output (.stats) was
provided
  -pheno PHENOTYPE_FILE, --phenotype_file PHENOTYPE_FILE
                        Path for the phenotype file, this file must have
FID and IID (like the .fam file) and must be separated by tab or space.
Same used on the GWAS setp
  --pheno_col PHENO_COL
                        Name of the columns containg the Phenotype data --
Default is to look for 'Phenotype' as the column name
  -qcovar QUANTITATIVE_COVAR, --quantitative_covar QUANTITATIVE_COVAR
                        Path for the quantitative covariables, e.g. PCs,
age, and other continuous variables. The same used on the GWAS step
  -n_pcs NUMBER_OF_PCS, --number_of_pcs NUMBER_OF_PCS
                        Number of PCs to use on model evaluation -- default
```

```
= 4
  -covar COVAR_FILE, --covar_file COVAR_FILE
                        Path for the covariables file, e.g. Sex. The same
used on the GWAS step
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Wanted output folder (default: current output
folder)
  --threads THREADS     Number of computer threads -- default = 1
```

The main result from this analysis is the table ***weights_LDPred2.tsv***, which contains 3 columns:

```
rsID    A1  LDPRED2_betas
rs3131972   A   2,35E+07
rs3131969   A   2,12E+06
rs1048488   C   7,32E+07
```

1. The first column is the rsID of the variant
2. The second is the effect allele
3. And the third is the recalculated beta

Resources used for a population with n=49 samples with around one milion variants:

Module	**Time**	**Peak Mem (Gb)**	**Threads**
desdup	00:00:04	0.2	4
update_rsID	00:00:22	0.12	4
quality_control	00:00:03	0.2	4
quality_ind	00:00:05	0.2	4
kinship	00:00:08	0.7	4
PCA	00:00:15	0.3	4
GWAS	00:02:20	1.3	4
PRS	05:20:00	14.8	4