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

```
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

3. Tools needed in path
    -- 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
```

```
For BOLT-LMM, we provide boltlmm.yml, an anconda recipe for the environment
instalation -- Most of the time, the pre-compiled binary works fine. If it
doesn't, please run
conda env create -f boltlmm.yml.
remember to activate the environment before running the GWAS module with
the bolt-lmm flag.
```

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.

** Before the analysis starts, we recommend --> 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. With that, no further adjusts in the dataset will be necesseray on the following analysis

This filter can be done using Plink1.9 --> `plink --vcf file.vcf --remove list_of_samples_with_no_phenotype.txt --recode vcf bgz --out filtered_dataset`
IMPORTANT: Plink uses the pattern FID IID (or IID IID) to identify samples**

IMPORTANT: For the execution of the pipeline the pattern FID_IID (or IID_IID) to identify samples is necessary

Main script usage

`python3 GARSA.py`

```
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
```

Preprocessing modules

Deduplication Module (desdup)

`python3 GARSA.py desdup`

```
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
```

This module searches and removes duplicated SNPs and multi-allelic variants

The flag `-vcf` is required for execution

Usage: `python3 GARSA.py desdup -vcf chr22_pop1.vcf.gz`

Update rsIDs module (SNP annotation)

`python3 GARSA.py update_rsID`

```
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
```

This module uses dbSNP for rsID annotation, also a SNP swap and flip analysis is performed to guarantee correct annotations

The flag `-vcf` is required for execution

Exemplo de uso: `python3 GARSA.py update_rsID -vcf chr22_pop1.vcf.gz`

Rename Sample ID

This is an optional module, it was created to facilitate the update/correction of sample IDs to match the required format listed above

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

OLD	NEW
sample1	FID1_IID1 or IID1_IID1
sample2	FID2_IID2 or IID2_IID2

`python3 GARSA.py rename_sample_id`

```
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
```

The falgs `-vcf and -table` are required

Variant quality control

This module executes variant quality controls

`python3 GARSA.py quality_control`

```
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 flag `-vcf` is required, and also the flags `-geno -maf -R2 --score-type and -use_HWE` are important for the user to pay attention

Usage: `python3 GARSA.py quality_control -vcf chr22_pop1.vcf.gz -o path/to/ouput_folder --score_type info`

Before continuing with 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 executes quality control on samples, including heterozigosity rates

`python3 GARSA.py quality_ind`

```
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 flag `-vcf` is required for execution and the flag `-mind` is important for the user to pay attention

Usage: `python3 GARSA.py quality_ind -vcf output_file_merged.vcf.gz`

Kinship with correction for admixed populations

This is an important module, aiming the kinship analysis with corrections for admixed populations. As proposed by Conomos et. al (2016)

`python3 GARSA.py kinship`

```
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
```

The flag `-vcf` is required for execution and the flag `---window_size --sliding_window --prune_r2 --degree` are important for the user to pay attention

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 the GCTA GWAS analysis
3. Related_at_degree[1,2 or 3].txt --> File with all related individuals, necessary for the PCA analysis

PCA module

This module runs the PCA analysis in 4 main steps.

1. Uses FlashPCA on the unrelated dataset for PCA analysis and generates, alongside the PCs, laoding values for each point (SNP) used for the analysis
2. From the loadings found, performe a search for outliers that might introduce bias to the analysis and remove those SNPs -- after that, run a new PCA analysis without the outlier SNPs
3. Project the PCs for the related dataset
4. 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

`python3 GARSA.py PCA`

```
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
```

The flag `-vcf and -related` are required for execution and the flags `---window_size --sliding_window and --prune_r2` are important for the user to pay attention

For this step there are 3 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

GWAS module

`python3 GARSA.py GWAS`

```
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] [--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
  --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
```

In this step the user must provide a phenotype, covariates (covar), quantitative covariates (qcovar) and kinship files

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

We recommend the use of GCTA for populations with sample size < 5000 individuals and BOLT-LMM for populations with sample size > 5000 individuals

We recommend the use of BH correction (with the flag --bh_correction) only for big (like UKBiobank) sample sizes, if not selected GARSA will apply the Genomic Inflation correction

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.

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 and add to the file containing other quantitative traits that might be importante for the association analysis.

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

Usage:

1. When running the GWAS module, the user is free to choose between the corrected kinship analysis (files *.grm.id e *.grm.bin) or the GCTA kinship analysis using the flag `--make_king`
2. The input for this module can be a .vcf file (`--vcf`) or an already converted plink binary file (provide the prefix with no extensions -- `--bfile`)

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. Manhattam plot
3. QQ plot

PRS

Implemented in R, using LDPred2

`python3 GARSA.py PRS`

```
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 contaning 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
```

On this step the user must provide all the files (covar, qcovar and phenotype) used on the GWAS step -- Important: use the flag `-n_pcs` to provide the number of PCs used on the step above (GWAS).

IMPORTANT: The qcovar file *MUST* be provided with the PCA information. Also, the generated outputs on the GWAS step (.mlma or .stats) must be provided aswell.

As a result, a table containing the PRS values and different graphs associated with the distribution of the PRS and the distribution of risk deciles are given, being able to identify which samples fall into each risk decile

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

Module	Time	Peak Mem (Gb)	Threads
GWAS	00:02:20	1.3	4
PRS	05:20:00	14.8	4