GWAS analysis with GLMM models using Regenie

This notebook implements pipelines for analyzing binary and quantitative trait associations using REGENIE. Updated in July 2026, the workflow replaces the previous SoS-based pipeline with a streamlined command-line implementation of REGENIE, simplifying execution while maintaining equivalent analytical functionality.

> **Kernel note:** most of this notebook runs shell (Bash) commands. The final **Plotting Results** and **Post-processing** sections run **R** code and require switching the notebook kernel to R — see the reminder before that section.

Introduction

Genome-wide association studies (GWAS) test millions of genetic variants for association with a trait, but standard regression models struggle at this scale: they are computationally expensive to run genome-wide, and they don't account for the fact that samples are often related or share fine-scale population structure, which can inflate false-positive associations. **REGENIE** (Mbatchou et al. 2021) addresses both problems with a two-step whole-genome regression approach: it first summarizes an individual's polygenic background from genome-wide markers (Step 1), then tests each variant for association while adjusting for that background using a **Leave-One-Chromosome-Out (LOCO)** offset (Step 2). This makes REGENIE fast enough for biobank-scale data while controlling for relatedness and population stratification, similar in spirit to mixed-model (GLMM) approaches but far more computationally efficient.

Method

Step 1: Whole Genome Regression Model

Two levels:

- Level 0: Ridge regression
- Level 1: Cross Validation

Step 2: Association Analysis

1. Single Variant Analysis
2. Gene Based Testing (Aggregate Analysis)

Data

Genotype data

For this exercise we will use genotype data from the 1000 Genomes Project, in PLINK binary format obtained via the PLINK 2.0 resources page. The number of individuals in this dataset is n=489.

Please note that this is only an example dataset; for real life applications a much larger sample size is necessary. We have generated principal components (PCs) for you to include in the analysis using FLASHPCA so that you can adjust for potential population substructure/admixture in the analysis. When appropriate, PCs should be included in the analysis even when a LMM/GLMM approximation is used. The PCs were created using pruned genotype data.

The genotype file has been pruned (variants in LD removed) to reduce the number of variants used in Step 1 ridge regression. Although it is not necessary to prune variants for Step 1, often variants have to be removed so the number of variants is <500,000. Please note you **DO NOT** have to run this part of the exercise, since this has already been done in advance for you. If you would like more information about the pruning procedure you can read more about this in plink

```
plink2 --bfile 1000G.EUR --indep-pairwise 100 10 0.01 --out
1000G.EUR.mwe
plink2 --bfile 1000G.EUR --extract 1000G.EUR.mwe.prune.in --
make-bed --out 1000G.EUR.mwe.pruned
```

This pruned genotype array file (`1000G.EUR.mwe.pruned.{bed,bim,fam}`) will be used for **Step 1** of regenie

```
1000G.EUR.mwe.pruned.bed
1000G.EUR.mwe.pruned.bim
1000G.EUR.mwe.pruned.fam
```

The genotype file used for **Step 2** (`1000G.EUR.mwe.chr21-22.step2.regenie.qc.{bed,bim,fam}`) of regenie had variant sites and individuals missing >10% of their data removed. Although usually you would test all chromosomes for association, the original 1000 Genomes European data contained over 22 million variants, which would increase the computation time for the analysis; therefore only chromosomes 21 and 22 will be analyzed for this exercise.

```
plink2 --bfile 1000G.EUR --indep-pairwise 100 10 0.5 --out
1000G.EUR.mwe.step2
plink2 --bfile 1000G.EUR --extract
1000G.EUR.mwe.step2.prune.in --make-bed --out
1000G.EUR.mwe.step2.regenie
```

```
plink2 --bfile 1000G.EUR.mwe.step2.regenie --geno 0.1 --mind
0.1 --make-bed --out 1000G.EUR.mwe.step2.regenie.qc
plink2 --bfile 1000G.EUR.mwe.step2.regenie.qc --chr 21-22 --
make-bed --out 1000G.EUR.mwe.chr21-22.step2.regenie.qc
```

Phenotype file

For the 1000 Genomes data there are no phenotypes available. Therefore we have created a phenotype `pheno` by randomly assigning study subjects to be either cases (1) or controls (0). We have also created the `sex` variable by randomly assigning individuals to be male (0) or female (1). Therefore the data is generated under the null of no association. Please note that a plink file format is used, so although we don't specify familial relationships, the columns which contain 0s must still be included; if families were being analyzed these would denote familial relationships. Please note that the phenotype file (1000G.EUR.regenie_phenotype) is already provided in the data folder. Code is also provided in case you are interested in understanding how this file was generated. Additionally, we have generated the first two principal components (PC1 and PC2) using FLASHPCA so that they can be added as covariates in Step 2 of regenie.

Phenotype File Format REGENIE requires a phenotype file containing sample identifiers and one or more phenotype values. The file is space- or tab-delimited and follows the format:

- **Header line:** The first line contains the sample identifiers (`FID` , `IID`) followed by the phenotype names (`Y1` , `Y2` , ..., `YP`).
- **Data lines:** Each subsequent line contains:
- Family ID (**FID**)
- Individual ID (**IID**)
- Values for each phenotype

For **binary traits**, phenotypes must be coded as:

- `0` = control
- `1` = case
- `NA` = missing (unless using the `--1` option)

Samples included in the phenotype file but not present in the genotype file (`.bed` , `.bgen` , or `.pgen`) are ignored. Conversely, genotyped samples that are not listed in the phenotype file are removed from the analysis.

Handling of Missing Values

- **Quantitative traits (QT):**
  - During Step 1, missing phenotype values are mean-imputed.
  - During Step 2, samples with missing values are excluded from association testing unless `--force-impute` is specified.

- **Binary traits (BT):**
  - During Step 1, missing values are mean-imputed when fitting the level 0 linear ridge regression model.
  - Missing values are excluded when fitting the level 1 logistic ridge regression model and during Step 2 association testing. Missing phenotype values must be coded as `NA`.

Sample code to generate phenotype and covariate files:

```
library(data.table)
library(dplyr)
# Input phenotype file
phenoFile <- "./data/1000G.EUR.pheno"
# Output paths
cwd <- "./data"
pheno_base <- tools::file_path_sans_ext(basename(phenoFile))
# Output file paths
phenotype_file <- file.path(
  cwd,
  paste0(pheno_base, ".regenie_phenotype")
)
covariate_file <- file.path(
  cwd,
  paste0(pheno_base, ".regenie_covar")
)
# Read phenotype file
pheno_dt <- fread(phenoFile)
# Phenotype file
pheno_regenie <- pheno_dt[, .(
  FID,
  IID,
  pheno
)]
fwrite(
  pheno_regenie,
  phenotype_file,
  sep = " ",
  quote = FALSE,
  na = "NA"
)
# Covariate file
covar_regenie <- pheno_dt[, .(
  FID,
  IID,
  sex,
  PC1,
  PC2
)]
fwrite(
  covar_regenie,
  covariate_file,
```

```
  sep = " ",
  quote = FALSE,
  na = "NA"
)
```

Covariate file

The covariate file contains the non-genetic variables that will be included in the association model to account for potential confounding factors. In this tutorial, we include **sex** and the first two principal components (**PC1** and **PC2**) as covariates. The principal components were generated using **FlashPCA** and capture the major axes of genetic variation, helping to adjust for population structure. The same covariate file **must** be used in both **Step 1** and **Step 2** of REGENIE. Please note that the covariate file (1000G.EUR.regenie_covar) is already provided in the data folder.

Covariate File Format REGENIE requires a covariate file containing sample identifiers and one or more covariates. The file is space- or tab-delimited and follows the format:

- **Header line:** The first line contains the sample identifiers (`FID` , `IID`) followed by the covariate names (`V1` , `V2` , ..., `VC`).
- **Data lines:** Each subsequent line contains:
  - Family ID (**FID**)
  - Individual ID (**IID**)
  - Values for each covariate

For this tutorial, the covariate file contains the following columns:

- `FID`
- `IID`
- `sex`
- `PC1`
- `PC2`

Samples included in the covariate file but not present in the genotype file (`.bed` , `.bgen` , or `.pgen`) are ignored. Conversely, genotyped samples that are not listed in the covariate file are removed from the analysis. Samples with missing values for any included covariate are also excluded. Because REGENIE uses the covariates to fit the whole-genome regression model in **Step 1**, the **same covariate file must be supplied again in Step 2** to ensure that association testing is performed using the identical adjustment variables.

References

- Chang CC, Chow CC, Tellier LCAM, Vattikuti S, Purcell SM, Lee JJ (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets. GigaScience, 4.
- Mbatchou, J., Barnard, L., Backman, J. et al. Computationally efficient whole-genome regression for quantitative and binary traits. Nat Genet 53, 1097–1103 (2021). https://doi.org/10.1038/s41588-021-00870-7
- Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. *American Journal of Human Genetics*, 81:559-575.
- R Core Team (2025), R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing

Pipeline Overview

Preprocessing: Genotype file preparation and quality control (QC)

Before running regenie you need to make sure you have the files needed as input. Step 1 of regenie only allows one genotype input file, so in real-life applications, if you have one plink binary file per chromosome you will need to merge them using plink. Quality control filters can be applied using Plink2 to filter out samples and markers in the genotype file prior to running regenie's Step 1.

Step 1: Whole Genome Model

Fits a whole-genome regression model to summarize each individual's polygenic background, producing per-chromosome (LOCO) genetic predictions that are carried forward into Step 2.

Step 2: Association Analysis

1. **Single Variant Association Analysis** Tests the association between each genetic variant and the trait while applying the **Leave-One-Chromosome-Out (LOCO)** method. This prevents "proximal contamination" by ensuring the genomic predictor for a specific chromosome does not include the variants being tested on that same chromosome.
2. **Rare Variant Aggregate Testing** Regenie offers the functionality of performing rare-variant aggregate association analysis (gene-based testing). You can combine rare variants in a gene or a region, using functional annotations to create masks that are then tested for association (as in a single-variant analysis) with a phenotype.

Single Variant Association Analysis

Single-variant association analysis evaluates the association between each genetic variant and the phenotype independently. REGENIE performs this analysis in Step 2 using the whole-genome predictions from Step 1 as an offset, applying linear regression for quantitative traits and logistic regression (with optional Firth correction) for binary traits.

The steps below follow the same **Preprocessing → Step 1 → Step 2** structure introduced in the Pipeline Overview above, applied specifically to single-variant testing.

Preprocessing: Genotype file preparation and quality control (QC)

This section will perform the preprocessing of preparing the genotype files and performing quality control prior to running regenie.

Let the parameters for the initial quality control for variants to be used in ridge regression be:

- maf_filter=0.05, we will remove variants with MAF <5%
- geno_filter=0.1, we will remove variants with > 10% genotypes missing
- mind_filter=0.1, we will remove individuals with > 10% genotypes missing
- hwe_filter=5e-08, we will remove variants exceeding a p-value for HWE > 5E-08

For example, to remove variants with minor allele frequency (MAF) lower than 5% set `maf_filter=0.05`; to remove variant sites with over 10% of genotypes missing set the parameter `geno_filter=0.1`; to remove individuals with more than 10% of their genotypes missing set the parameter `mind_filter=0.1`; and lastly to remove variants with a Hardy-Weinberg equilibrium p-value < 5e-08 set `hwe_filter=5e-08`.

Input:

- `1000G.EUR.mwe.pruned.bed/.bim/.fam`: plink files with pruned subset of the genotyped genetic markers.

Output:

- `1000G.EUR.mwe.pruned.qc_pass.id`: contains the list of individuals to keep in Step 1 of regenie.
- `1000G.EUR.mwe.pruned.qc_pass.snplist`: contains the list of variants to keep in Step 1 of regenie.

Command explanation

- `maf_filter` : the minimum allele frequency to be used to filter the `bfile`
- `geno_filter` : corresponds to the `--geno` option in plink, which removes all variants with missing call rates exceeding the provided value
- `mind_filter` : corresponds to the `--mind` option in plink, which removes all samples with missing genotypes exceeding the provided value
- `hwe_filter` : corresponds to the `--hwe` option in plink, which filters out variants which have a Hardy-Weinberg equilibrium exact test p-value below the provided threshold

In [ ]:
```
# Global parameters
data_dir="./data"
output_dir="./output"
mkdir -p "${output_dir}"
```

In [ ]:
```
# Parameters
bfile="${data_dir}/1000G.EUR.mwe.pruned"
bfile_base=$(basename "${bfile}")
# QC filters
maf_filter=0.05
geno_filter=0.1
hwe_filter=5e-08
mind_filter=0.1

# Select the SNPs and samples to be used based on maf, geno, hwe and mind op
plink2 \
    --bfile "${bfile}" \
    --mac 1 \
    --maf "${maf_filter}" \
    --geno "${geno_filter}" \
    --hwe "${hwe_filter}" \
    --mind "${mind_filter}" \
    --write-snplist \
    --write-samples \
    --no-id-header \
    --threads 2 \
    --out "${output_dir}/${bfile_base}.qc_pass"
```

Step 1: Whole-Genome Regression Model

In **Step 1**, REGENIE fits a whole-genome regression model using a subset of high-quality, common genetic variants (typically several hundred thousand genotyped markers). To reduce computational burden, variants are processed in blocks and summarized using a two-level ridge regression framework. In this step, regenie authors recommend using genotype array data (as opposed to imputed/exome data), as it provides better estimates of the phenotype variance. You also need to make sure that you don't have any Single Nucleotide Variants (SNVs) with a very low minor allele count (MAC),

otherwise regenie will give an error message in the calculation of the stacked regressions.

- **Level 0 (Ridge Regression):** Markers are analyzed in blocks, and ridge regression is applied using multiple shrinkage parameters to generate a small set of predictors for each block. This captures local linkage disequilibrium (LD) while substantially reducing the dimensionality of the genotype data.
- **Level 1 (Cross-Validated Ridge Regression):** The predictors generated in Level 0 are combined into a genome-wide prediction model. For quantitative traits, ridge regression with K-fold cross-validation is used to select the optimal shrinkage parameter. For binary traits, a logistic ridge regression model is fit instead.

The final model generates **Leave-One-Chromosome-Out (LOCO)** genetic predictions by excluding one chromosome at a time. These LOCO predictions account for the polygenic background while avoiding proximal contamination and are carried forward to **Step 2**, where they are used as offsets during single-variant association testing.

REGENIE also supports the simultaneous modeling of multiple phenotypes and allows covariates (e.g., age, sex, principal components) to be incorporated into the model. For quantitative traits, covariate effects are regressed out before model fitting, whereas for binary traits, covariates are included through a null logistic regression model.

In our case, we will use the original genotype file in the parameter `bfile` and we will tell regenie which samples (`1000G.EUR.mwe.pruned.qc_pass.id`) and variants (`1000G.EUR.mwe.pruned.qc_pass.snplist`) to keep based on the results of the previous quality control step.

Input:

- `1000G.EUR.mwe.pruned.bed/.bim/.fam` : plink files with pruned subset of the genotyped genetic markers.
- `1000G.EUR.mwe.pruned.qc_pass.id` : contains the list of individuals to keep in Step 1 of regenie.
- `1000G.EUR.mwe.pruned.qc_pass.snplist` : contains the list of variants to keep in Step 1 of regenie.
- block_size=1000, this will tell the program in how many "chunks" to divide the genotype file to make predictions.

Output:

- Per-chromosome LOCO predictions, one `.loco` file per phenotype, named `<out>_1.loco` , `<out>_2.loco` , ... in the order phenotypes appear in the phenotype file header. With a single phenotype (`pheno`), this will be `1000G.EUR_pheno.regenie_step1_1.loco` .

- `<out>_pred.list` : lists, for each phenotype, the name of the phenotype and the path to its corresponding `.loco` file. This is the file passed to `--pred` in Step 2 — in our case `1000G.EUR_pheno.regenie_step1_pred.list` .

Command explanation

- `bfile` : the genotype file to be used in Step 1 of regenie (to fit the null model). In our case `1000G.EUR.mwe.pruned.bed`
- `phenoFile` : corresponds to the phenotype file which contains FID, IID, phenotypic information and covariates to be tested for association. In this example `1000G.EUR.regenie_phenotype`
- `covarFile` : corresponds to the covariate file which contains FID, IID, sex, and covariate information (in our case `PC1 PC2`) found in `1000G.EUR.regenie_covar`
- `threads` : the number of computational threads to be used
- `bsize` : the size of genotype blocks to be used in regenie Step 1
- `trait` : the type of trait being analyzed; in this case `bt` corresponds to a binary trait, and leaving it empty tells regenie it is a continuous trait
- `keep` / `extract` : inclusion files that list the IDs of samples/variants to keep, generated by the QC step above

In [ ]:
```
# Parameters
bfile="${data_dir}/1000G.EUR.mwe.pruned"
bfile_base=$(basename "${bfile}")
# Input phenotype file containing phenotype and covariate information
phenotype_regenie="${data_dir}/1000G.EUR.regenie_phenotype"
covar_regenie="${data_dir}/1000G.EUR.regenie_covar"
# Output prefix
step1_prefix="${output_dir}/1000G.EUR_pheno.regenie_step1"
# Block size
bsize=1000
# Run REGENIE step 1
regenie \
    --step 1 \
    --bed "${bfile}" \
    --phenoFile "${phenotype_regenie}" \
    --covarFile "${covar_regenie}" \
    --keep "${output_dir}/${bfile_base}.qc_pass.id" \
    --extract "${output_dir}/${bfile_base}.qc_pass.snplist" \
    --bt \
    --bsize "${bsize}" \
    --lowmem \
    --lowmem-prefix "${step1_prefix}" \
    --threads 2 \
    --out "${step1_prefix}"
```

Step 2: Association analysis

In this step, namely step 2 of regenie, it will perform the single variant association analysis with each of the phenotypes. In real life scenarios you should test all the variants that have passed quality control, not just the pruned subset. In our case, due to time constraints (we want the exercise to finish within the available time in the course) we will use a genotype file that has already been quality controlled (`--geno 0.1` , `--mind 0.1` , and `--hwe 5e-08` — see the plink2 commands in the Data section above) to test for variant associations with the phenotype.

Note: a convenient quality of regenie is that Step 1 and Step 2 are decoupled, meaning that you can use all of the traits used in Step 1 or just a subset of them for the association analysis (Step 2). Also, you can test the association using array, exome, or imputed variants. This means that the `bfile` parameter (the genotype data used to fit the null model) and the `genoFile` parameter (the genotype data used to test for association) can be different.

For this example, we will use our phenotype with the following parameters:

- `trait='bt'` : here you define if your trait is binary (bt) or quantitative (leave empty).
- `phenoCol=pheno` : the name of the phenotype column to test, taken from the phenotype file header.
- `catCovarList=sex` : our one categorical covariate. If you have more than one, separate them with commas.
- `covarColList=PC1,PC2` : our quantitative covariates (the PCs).
- `minMAC=10` : this flag tells the program the minimum minor allele count (MAC) to use when testing variants; the default value is 5. In real data applications you may want to set this threshold depending on the power to detect associations based on your sample size.

Additionally, we will use the `--genoFile=1000G.EUR.mwe.chr21-22.step2.regenie.qc.bed` parameter for the genotype file with the variants that we want to test for association.

Regenie does not have a minMAF filter, so you would need to use the minMAC parameter and take into account the specific allele frequencies you would like to test for, to calculate the MAC based on your sample size. As an example, if you would like to test for variants with MAF ≥ 0.01 and you have 500 individuals, then you would set minMAC=10 (500 individuals × 2 alleles × 0.01 MAF).

Depending on the type of data you are using, there are other useful parameters that you can explore. For example, with imputed data you may want to set the minimum info score to use (`--minINFO`); with quantitative phenotypes you may choose to use inverse rank normalization (`--apply-rint`); for binary traits you can decide whether to use Firth (`--firth`) or Saddle Point Approximation (`--spa`) corrections. If you would like to explore any of regenie's options please take a look at their documentation.

The output produced in this step corresponds to:

- Summary statistics in a file with a `*.regenie` extension; if the `--gz` option is used, this file will be compressed to `*.regenie.gz`.

Command explanation

- `genoFile`: corresponds to the file with the genetic data for association testing (regenie Step 2). In this example `1000G.EUR.mwe.chr21-22.step2.regenie.qc.bed`
- `phenoFile`: corresponds to the phenotype file which contains FID, IID, phenotypic information and covariates to be tested for association. In this example `1000G.EUR.regenie_phenotype`
- `covarFile`: corresponds to the covariate file which contains FID, IID, sex, and covariate information (in our case `PC1 PC2`) found in `1000G.EUR.regenie_covar`
- `phenoCol`: the name of the phenotype column in the phenoFile to test
- `covarColList`: the names of the quantitative covariates to include in the analysis
- `catCovarList`: the names of the categorical covariates to include in the analysis
- `chr`: specifies which chromosome(s) to test in Step 2. In our case we will analyze chromosome 21 due to time constraints.
- `threads`: the number of computational threads to be used
- `bsize`: the size of genotype blocks to be used in regenie Step 2
- `trait`: the type of trait being analyzed; in this case `bt` corresponds to a binary trait, and leaving it empty tells regenie it is a continuous trait
- `pred`: the `_pred.list` file produced by Step 1, linking each phenotype to its LOCO predictions
- `minMAC`: the minimum minor allele count (MAC) when testing variants. In our case we set it to `10`
- `extract`: an optional inclusion file listing IDs of variants to keep — not needed here since `genoFile` was already QC-filtered upstream (see the plink2 commands in the Data section)
- `af-cc`: outputs A1FREQ in cases/controls separately in the Step 2 result file
- `firth`: use the Firth likelihood ratio test (LRT) as a fallback for p-values below a threshold
- `approx`: use an approximate Firth LRT for computational speedup (only works when `--firth` is used)

In [ ]:
```
# Parameters
genoFile="${data_dir}/1000G.EUR.mwe.chr21-22.step2.regenie.qc"
geno_base=$(basename "${genoFile}")
# Step 1 output prefix
step1_prefix="${output_dir}/1000G.EUR_pheno.regenie_step1"
# Input phenotype file containing phenotype and covariate information
phenotype_regenie="${data_dir}/1000G.EUR.regenie_phenotype"
```

```
covar_regenie="${data_dir}/1000G.EUR.regenie_covar"
# Chromosome to analyze
chr="21"
# Set minimum minor allele count (MAC)
minMAC="10"
# Set block size
bsize="400"
# Run REGENIE Step 2: Chromosome 21 only
regenie \
    --step 2 \
    --bed "${genoFile}" \
    --phenoFile "${phenotype_regenie}" \
    --covarFile "${covar_regenie}" \
    --phenoCol pheno \
    --catCovarList sex \
    --covarColList PC1,PC2 \
    --bt \
    --chr "${chr}" \
    --firth \
    --approx \
    --af-cc \
    --pred "${step1_prefix}_pred.list" \
    --bsize "${bsize}" \
    --minMAC "${minMAC}" \
    --threads 2 \
    --out "${output_dir}/1000G.EUR.mwe.chr${chr}.step2"
```

Rare Variant Aggregate Tests

The advancement and availability of feasible sequencing technologies provide a rich opportunity to study the association between rare/low frequency variants and complex traits. Rare variants [minor allele frequencies (MAFs) e.g. <0.01] might play an important role in the etiology of complex traits and account for missing heritability unexplained by common variants (Bodmer W et.al 2008, Schork N.J et.al 2009 and Manolio TA et.al 2009).

Gene-based or aggregate tests are recommended since individual rare variants have very low frequencies and limited statistical power when analyzed individually. Single-variant tests are typically conducted to investigate associations of low-frequency and common variants (e.g. MAF>0.01); however, the same approach has little power when testing for rare-variant effects because of low frequencies. This led to the development of rare-variant aggregate analyses, which focus on testing the cumulative effects of rare variants in genetic regions. It should be noted that when performing rare variant aggregate tests, there should be at least two rare variants in each gene/region tested.

In recent years, significant efforts have been devoted to developing powerful and computationally efficient statistical methods for testing associations between rare variants and complex traits.

1. Burden Tests

Burden tests are fixed-effect tests that combine the effects of rare variants within a gene into a single test statistic. A variety of different coding schemes can be used for fixed-effect tests, including binary and allele counting. Regenie provides several masks to perform fixed-effect tests (see below). These tests collapse variants into a single variable, which is then tested for association with the phenotype. Hence, they are more powerful when variants have effects in the same direction and of similar magnitude. In regenie, multiple options are available to aggregate variants together into a burden mask beyond a simple linear combination. For example, the burden tests employed in Backman et al. (2021) use the default strategy in regenie of collapsing variants by taking the maximum number of rare alleles across the sites.

2. Variance component tests

- **SKAT (Sequence Kernel Association Test)** Wu et.al 2011

Instead of aggregating variants, SKAT aggregates individual variant-score test statistics with weights when SNV effects are modeled linearly.

- **SKAT-O (Sequence Kernel Association Combined Test)** Lee et.al 2012

SKAT-O is an extension of SKAT that combines SKAT and the burden test. It allows for the flexibility of both kernel-based and collapsing approaches and provides a combined test statistic.

Regenie also allows for testing other types of aggregate tests, such as SKATO-ACAT, ACATV, ACATO, and ACATO-FULL. Please refer to regenie's tutorial for more details about these tests.

Packages/Software This particular gene-based analysis pipeline is based on **regenie**. Regenie can apply various set-based tests on the variants, as well as collapse them into a single combined 'mask' genotype that can be tested for association just like a single (mega) variant. For example, the option `--vc-tests` is used to specify the SKAT and ACAT tests, and the p-values for SKAT, SKATO, ACATV, and ACATO will be output. For details, see the regenie documentation.

NOTE: Quality control and variant annotation are essential steps before conducting rare-variant aggregate tests.

Preprocessing: Genotype file preparation and quality control (QC)

This section will perform the preprocessing of preparing the genotype files and performing quality control prior to running regenie.

Let the parameters for the initial quality control for variants to be used in ridge regression be:

- maf_filter=0.01, we will remove variants with MAF <1%
- geno_filter=0.1, we will remove variants with > 10% genotypes missing
- mind_filter=0.1, we will remove individuals with > 10% genotypes missing
- hwe_filter=5e-08, we will remove variants exceeding a p-value for HWE > 5E-08

For example, to remove variants with minor allele frequency (MAF) lower than 1% set `maf_filter=0.01`; to remove variant sites with over 10% of genotypes missing set the parameter `geno_filter=0.1`; to remove individuals with more than 10% of their genotypes missing set the parameter `mind_filter=0.1`; and lastly to remove variants with a Hardy-Weinberg equilibrium p-value ≤ 5e-08 set `hwe_filter=5e-08`.

Note: this MAF cutoff (1%) is lower than the one used for the single-variant Step 1 model (5%, see above). Step 1 and Step 2 are decoupled in regenie, so in practice you could reuse the same LOCO predictions for both the single-variant and rare-variant analyses. Here we re-fit Step 1 with a different MAF threshold purely to illustrate that the Step 1 QC parameters can be tuned independently of the downstream Step 2 test — feel free to reuse the single-variant Step 1 output instead to save compute.

Input:

- `1000G.EUR.mwe.pruned.bed/.bim/.fam`: plink files with pruned subset of the genotyped genetic markers.

Output:

- `1000G.EUR.mwe.pruned_rarevariant.qc_pass.id`: contains the list of individuals to keep in Step 1 of regenie.
- `1000G.EUR.mwe.pruned_rarevariant.qc_pass.snplist`: contains the list of variants to keep in Step 1 of regenie.

Command explanation

- `maf_filter`: the minimum allele frequency to be used to filter the `bfile`
- `geno_filter`: corresponds to the `--geno` option in plink, which removes all variants with missing call rates exceeding the provided value
- `mind_filter`: corresponds to the `--mind` option in plink, which removes all samples with missing genotypes exceeding the provided value
- `hwe_filter`: corresponds to the `--hwe` option in plink, which filters out variants which have a Hardy-Weinberg equilibrium exact test p-value below the provided threshold

In [ ]:
```
# Global parameters
# Directories:
```

```
data_dir="./data"
output_dir="./output_rarevariant"
mkdir -p "${output_dir}"
```

In [ ]:
```
# Parameters
bfile="${data_dir}/1000G.EUR.mwe.pruned"
bfile_base=$(basename "${bfile}")
# QC filters
maf_filter=0.01
geno_filter=0.1
hwe_filter=5e-08
mind_filter=0.1

# Select the SNPs and samples to be used based on maf, geno, hwe and mind op
plink2 \
    --bfile "${bfile}" \
    --mac 1 \
    --maf "${maf_filter}" \
    --geno "${geno_filter}" \
    --hwe "${hwe_filter}" \
    --mind "${mind_filter}" \
    --write-snplist \
    --write-samples \
    --no-id-header \
    --threads 2 \
    --out "${output_dir}/${bfile_base}_rarevariant.qc_pass"
```

Step 1: Whole-Genome Regression Model

In **Step 1**, REGENIE fits a whole-genome regression model using a subset of high-quality, common genetic variants (typically several hundred thousand genotyped markers). To reduce computational burden, variants are processed in blocks and summarized using a two-level ridge regression framework. In this step, regenie authors recommend using genotype array data (as opposed to imputed/exome data), as it provides better estimates of the phenotype variance. You also need to make sure that you don't have any Single Nucleotide Variants (SNVs) with a very low minor allele count (MAC), otherwise regenie will give an error message in the calculation of the stacked regressions.

- **Level 0 (Ridge Regression):** Markers are analyzed in blocks, and ridge regression is applied using multiple shrinkage parameters to generate a small set of predictors for each block. This captures local linkage disequilibrium (LD) while substantially reducing the dimensionality of the genotype data.
- **Level 1 (Cross-Validated Ridge Regression):** The predictors generated in Level 0 are combined into a genome-wide prediction model. For quantitative traits, ridge regression with K-fold cross-validation is used to select the optimal shrinkage parameter. For binary traits, a logistic ridge regression model is fit instead.

The final model generates **Leave-One-Chromosome-Out (LOCO)** genetic predictions by excluding one chromosome at a time. These LOCO predictions account for the polygenic background while avoiding proximal contamination and are carried forward to **Step 2**, where they are used as offsets during rare-variant aggregate testing.

REGENIE also supports the simultaneous modeling of multiple phenotypes and allows covariates (e.g., age, sex, principal components) to be incorporated into the model. For quantitative traits, covariate effects are regressed out before model fitting, whereas for binary traits, covariates are included through a null logistic regression model.

In our case, we will use the original genotype file in the parameter `bfile` and we will tell regenie which samples (`1000G.EUR.mwe.pruned_rarevariant.qc_pass.id`) and variants (`1000G.EUR.mwe.pruned_rarevariant.qc_pass.snplist`) to keep based on the results of the previous quality control step.

Input:

- `1000G.EUR.mwe.pruned.bed/.bim/.fam` : plink files with pruned subset of the genotyped genetic markers.
- `1000G.EUR.mwe.pruned_rarevariant.qc_pass.id` : contains the list of individuals to keep in Step 1 of regenie.
- `1000G.EUR.mwe.pruned_rarevariant.qc_pass.snplist` : contains the list of variants to keep in Step 1 of regenie.
- block_size=1000, this will tell the program in how many "chunks" to divide the genotype file to make predictions.

Output:

- Per-chromosome LOCO predictions, one `.loco` file per phenotype, named `<out>_1.loco` , `<out>_2.loco` , ... in the order phenotypes appear in the phenotype file header. With a single phenotype (`pheno`), this will be `1000G.EUR_pheno.regenie_step1_rarevariant_1.loco` .
- `<out>_pred.list` : lists, for each phenotype, the name of the phenotype and the path to its corresponding `.loco` file. This is the file passed to `--pred` in Step 2 — in our case `1000G.EUR_pheno.regenie_step1_rarevariant_pred.list` .

Command explanation

- `bfile` : the genotype file to be used in Step 1 of regenie (to fit the null model). In our case `1000G.EUR.mwe.pruned.bed`
- `phenoFile` : corresponds to the phenotype file which contains FID, IID, phenotypic information and covariates to be tested for association. In this example `1000G.EUR.regenie_phenotype`
- `covarFile` : corresponds to the covariate file which contains FID, IID, sex, and covariate information (in our case `PC1 PC2`) found in

`1000G.EUR.regenie_covar`

- `threads` : the number of computational threads to be used
- `bsize` : the size of genotype blocks to be used in regenie Step 1
- `trait` : the type of trait being analyzed; in this case `bt` corresponds to a binary trait, and leaving it empty tells regenie it is a continuous trait
- `keep` / `extract` : inclusion files that list the IDs of samples/variants to keep, generated by the QC step above

In [ ]:
```
# Parameters
bfile="${data_dir}/1000G.EUR.mwe.pruned"
bfile_base=$(basename "${bfile}")
# Input phenotype file containing phenotype and covariate information
phenotype_regenie="${data_dir}/1000G.EUR.regenie_phenotype"
covar_regenie="${data_dir}/1000G.EUR.regenie_covar"
# Output prefix
step1_prefix="${output_dir}/1000G.EUR_pheno.regenie_step1_rarevariant"
# Block size
bsize=1000
# Run REGENIE step 1
regenie \
    --step 1 \
    --bed "${bfile}" \
    --phenoFile "${phenotype_regenie}" \
    --covarFile "${covar_regenie}" \
    --keep "${output_dir}/${bfile_base}_rarevariant.qc_pass.id" \
    --extract "${output_dir}/${bfile_base}_rarevariant.qc_pass.snplist" \
    --bt \
    --bsize "${bsize}" \
    --lowmem \
    --lowmem-prefix "${step1_prefix}" \
    --threads 2 \
    --out "${step1_prefix}"
```

Step 2: Rare Variant Aggregate Testing

Instead of performing single-variant association tests, multiple variants can be aggregated in a given region, such as a gene. Regenie offers the functionality of performing rare-variant aggregate association analysis. You can combine rare variants in a gene or a region, using functional annotations to create masks that are then tested for association (as in a single-variant analysis) with a phenotype. Two related approaches are covered below: **A. Burden test** and **B. SKAT-O test**; both share the input files and parameters described here.

Input files

1. An `annotation file` formatted as: variant_id, gene/region, functional annotation (e.g. LoF, missense). You can use VEP or ANNOVAR to generate this information and then format it accordingly.

2. A `set-list-file` : this file contains a list of variants within each gene/region that's used when building the masks. The format is: gene/region name, chromosome, start_position, list of variants in the gene/region separated by commas.
3. Optional: a file with genes/regions that you want to include or exclude from your analysis.
4. Optional: an `allele-frequency` file to use when creating the masks. By default the allele frequency is computed from the sample, which is not a good idea to do. In our case we are providing an allele-frequency file obtained from gnomAD exome frequencies. We will use the AF_nfe field, which contains the allele frequencies from exome data for non-Finnish Europeans available in gnomAD.
5. A mask file: this is a text file that contains the name of the mask and the type of annotations to use when building it (one mask per line). E.g.

   ```
   mask1 LoF,missense
   ```

6. The `--aaf-bins` cut-off in the parameters. This refers to the AAF upper bounds to use when creating the masks. By default regenie_burden produces results for singletons, and if you set `--aaf-bins` to, for example, 0.01, it will create masks from [0, 0.01] and singletons.

Additionally, you can choose the way the masks are built. These options are:

- using the maximum number of ALT alleles across sites ('max'; the default)
- using the sum of ALT alleles ('sum')
- using a maximum threshold of 2 ('comphet')

		Mask		
Single variant sites		max	sum	comphet
00000000000000	→	0	0	0
00000100010000	→	1	2	2
00201011010100	→	2	7	2

For this MWE we have already generated the annotation, set list, allele frequency, and mask files, which are provided in the `data` folder.

Command explanation

In addition to the commands used for single-variant testing, some extra parameters need to be set for the rare-variant aggregate tests:

- `anno_file` : path to the annotation file, required by regenie to run the aggregate tests. See the regenie tutorial to learn more
- `aaf_file` : path to the allele frequency file, required by regenie to run the aggregate tests. See the regenie tutorial to learn more
- `set_list` : path to the set list file, required by regenie to run the aggregate tests. See the regenie tutorial to learn more
- `mask_file` : path to the mask definition file. See the regenie tutorial to learn more
- `build_mask` : indicates how the mask should be created (max, sum, or comphet). See the regenie tutorial to learn more
- `aaf_bins` : the alternative allele frequency (AAF) upper bounds to use when creating the masks. In this example, `0.01` will be used

A. Burden Test

The burden test collapses the rare variants in each gene/region into a single mask, using the rules described above, and tests that mask for association with the phenotype exactly as it would a single variant.

In [ ]:
```
########################################
# Rare Variant Aggregate Burden Tests
########################################
# Parameters
genoFile="${data_dir}/1000G.EUR.mwe.chr21-22.step2.regenie.qc"
geno_base=$(basename "${genoFile}")
# Path to the annotation file defining functional annotations for variants
anno_file="${data_dir}/1000G.EUR.hg19.hg19_multianno.csv.anno_file"
# Path to mask file specifying which annotation categories should be combine
mask_file="${data_dir}/1000G.EUR.mask_file"
# Path to set list file containing variants within each set/gene used when b
set_list="${data_dir}/1000G.EUR.hg19.hg19_multianno.csv.set_list_file"
# Path to aaf file containing alternative allele frequency (AAF) cutoffs use
aaf_file="${data_dir}/1000G.EUR.hg19.hg19_multianno.csv.aff_file"
# Input phenotype file containing phenotype and covariate information
phenotype_regenie="${data_dir}/1000G.EUR.regenie_phenotype"
covar_regenie="${data_dir}/1000G.EUR.regenie_covar"
# Output prefix of step1
step1_prefix="${output_dir}/1000G.EUR_pheno.regenie_step1_rarevariant"
# Chromosome to analyze
chr="21"
# Set block size
bsize="400"
# Set rule to build masks
build_mask="max"
# Set minimum minor allele count (MAC)
minMAC="10"
# Set AAF upper bounds to use when building masks
aaf_bins="0.01"
# Run REGENIE rare variant aggregate analysis
regenie \
    --step 2 \
```

```
    --bed "${genoFile}" \
    --phenoFile "${phenotype_regenie}" \
    --covarFile "${covar_regenie}" \
    --bt \
    --pred "${step1_prefix}_pred.list" \
    --set-list "${set_list}" \
    --anno-file "${anno_file}" \
    --mask-def "${mask_file}" \
    --chr "${chr}" \
    --aaf-bins "${aaf_bins}" \
    --build-mask "${build_mask}" \
    --aaf-file "${aaf_file}" \
    --minMAC "${minMAC}" \
    --bsize "${bsize}" \
    --out "${output_dir}/1000G.EUR.mwe.chr${chr}.step2_rarevariant_burden_re
```

B. SKAT-O Test

In Regenie version >3.0, new methods to perform rare variant aggregate tests have been added. More specifically, for a given set of variants (e.g. within a gene), which can be defined using functional annotations, regenie can apply various set-based tests:

- SKAT: Variance component test
- SKAT-O: Omnibus test combining features of SKAT and Burden
- SKATO-ACAT: Same as SKAT-O but using the Cauchy combination method to maximize power across SKAT-O models
- ACATV: Test using the Cauchy combination method to combine single-variant p-values
- ACATO: Omnibus test combining features of ACATV, SKAT, and Burden
- ACATO-FULL: Same as ACATO but using the larger set of SKAT-O models used in the SKAT-O test

If you would like to include variants only under a given allele frequency threshold (rare or ultra-rare variants), you can define the parameter `--vc-maxAAF` or `--vc-MACthr` to set an AAF/MAC threshold under which variants will be considered in the analysis. These are not used in the example below, since we are already restricting variants via `--aaf-bins`, but are worth knowing about for your own analyses.

Command explanation

In addition to the parameters described above, one more parameter needs to be set for the SKAT-O test:

- `vc_tests`: specifies the gene-based test(s) to run (e.g. skat, skato, acato, etc.)

In [ ]:
```
########################################
# Rare Variant Aggregate SKAT-O Tests
########################################
```

```
# Parameters
genoFile="${data_dir}/1000G.EUR.mwe.chr21-22.step2.regenie.qc"
geno_base=$(basename "${genoFile}")
# Path to the annotation file defining functional annotations for variants
anno_file="${data_dir}/1000G.EUR.hg19.hg19_multianno.csv.anno_file"
# Path to mask file specifying which annotation categories should be combine
mask_file="${data_dir}/1000G.EUR.mask_file"
# Path to set list file containing variants within each set/gene used when b
set_list="${data_dir}/1000G.EUR.hg19.hg19_multianno.csv.set_list_file"
# Path to aaf file containing alternative allele frequency (AAF) cutoffs use
aaf_file="${data_dir}/1000G.EUR.hg19.hg19_multianno.csv.aff_file"
# Input phenotype file containing phenotype and covariate information
phenotype_regenie="${data_dir}/1000G.EUR.regenie_phenotype"
covar_regenie="${data_dir}/1000G.EUR.regenie_covar"
# Output prefix of step1
step1_prefix="${output_dir}/1000G.EUR_pheno.regenie_step1_rarevariant"
# Chromosome to analyze
chr="21"
# Set block size
bsize="400"
# Set rule to build masks
build_mask="max"
# Set minimum minor allele count (MAC)
minMAC="10"
# Set AAF upper bounds to use when building masks
aaf_bins="0.01"
# Specify the gene-based tests to run
vc_tests="skato"
# Specify AAF upper bound to use for SKAT/ACAT-type tests
vc_maxAAF="0.01"
# Run REGENIE rare variant aggregate analysis
regenie \
    --step 2 \
    --bed "${genoFile}" \
    --phenoFile "${phenotype_regenie}" \
    --covarFile "${covar_regenie}" \
    --bt \
    --af-cc --firth --approx \
    --pred "${step1_prefix}_pred.list" \
    --set-list "${set_list}" \
    --anno-file "${anno_file}" \
    --mask-def "${mask_file}" \
    --chr "${chr}" \
    --aaf-bins "${aaf_bins}" \
    --build-mask "${build_mask}" \
    --aaf-file "${aaf_file}" \
    --minMAC "${minMAC}" \
    --bsize "${bsize}" \
    --vc-tests "${vc_tests}" \
    --vc-maxAAF "${vc_maxAAF}" \
    --check-burden-files \
    --out "${output_dir}/1000G.EUR.mwe.chr${chr}.step2_rarevariant_skato_res
```

Annotating with ANNOVAR

Merge the final results if output is sharded across chromosomes, and annotate the results using ANNOVAR.

Merge the Results

Combine the REGENIE outputs (per chromosome) into a single consolidated file. Since we only ran a minimal working example for chromosome 21, there is no need to merge. The code below is a sample of how to merge the final results.

Example of how to merge all results if a full analysis ran across all chromosomes:

```
# Merge REGENIE chromosome outputs
cwd="./output"  # set to wherever your per-chromosome regenie outputs live
regenie_output_prefix=(
  "1000G.EUR.mwe.step2_results"
  "1000G.EUR.mwe.step2_burden_results"
  "1000G.EUR.mwe.step2_skato_results"
)
for prefix in "${regenie_output_prefix[@]}"; do
    merged_output="${cwd}/${prefix}.all_autosomes.regenie"
    count_output="${cwd}/${prefix}.snp_counts.txt"
    log_output="${cwd}/${prefix}.snp_stats.log"
    rm -f "$merged_output" "$count_output" "$log_output"
    first=true
    total=0
    # Merge REGENIE results
    for chr in {1..22}; do
        chr_file="${cwd}/${prefix}.chr${chr}.regenie"
        [[ ! -f "$chr_file" ]] && continue
        if $first; then
            cat "$chr_file" > "$merged_output"
            first=false
        else
            tail -n +2 "$chr_file" >> "$merged_output"
        fi
    done
    # Count variants/tests
    echo -e "Chromosome\tTests" > "$count_output"
    for chr in {1..22}; do
        chr_file="${cwd}/${prefix}.chr${chr}.regenie"
        [[ ! -f "$chr_file" ]] && continue
        n=$(tail -n +2 "$chr_file" | wc -l)
        echo -e "${chr}\t${n}" >> "$count_output"
        total=$((total+n))
    done
    echo -e "Total\t${total}" >> "$count_output"
    # Merge log/stdout/stderr files
    for chr in {1..22}; do
        base="${cwd}/${prefix}.chr${chr}.regenie"
```

```
        for ext in stdout stderr log; do
            if [[ -f "${base}.${ext}" ]]; then
                {
                    echo
"================================================="
                    echo "$(basename "${base}.${ext}")"
                    echo
"================================================="
                    cat "${base}.${ext}"
                    echo
                } >> "$log_output"

                rm -f "${base}.${ext}"
            fi
        done
    done
    # Compress merged output
    if [[ -f "$merged_output" ]]; then
        gzip -f "$merged_output"
    fi
    echo "========================================="
    echo "Finished: ${prefix}"
    echo "Merged results : ${merged_output}.gz"
    echo "Variant counts : ${count_output}"
    echo "Merged logs    : ${log_output}"
    echo "========================================="
done
```

Since the analysis was only ran on chromosome 21, we will preview the output from the single variant analysis and rare variant burden/skato analysis.

In [ ]:
```
# REGENIE chr21 output statistics
# Parameters
regenie_single_variant_file="./output/1000G.EUR.mwe.chr21.step2_pheno.regeni
regenie_burden_file="./output_rarevariant/1000G.EUR.mwe.chr21.step2_rarevari
regenie_skato_file="./output_rarevariant/1000G.EUR.mwe.chr21.step2_rarevaria
echo "======================================="
echo "REGENIE chr21 Summary"
echo "======================================="
for result in \
    "Single Variant:${regenie_single_variant_file}" \
    "Burden:${regenie_burden_file}" \
    "SKAT-O:${regenie_skato_file}"
do
    label="${result%%:*}"
    file="${result#*:}"

    if [[ -f "${file}" ]]; then
        count=$(tail -n +2 "${file}" | wc -l)
        printf "%-18s %10d tests\n" "${label}:" "${count}"
    else
        printf "%-18s %10s\n" "${label}:" "Missing"
    fi
```

```
done
echo "========================================"
```

Gene-Based Annotation

ANNOVAR is an efficient software tool that uses up-to-date information to functionally annotate genetic variants detected from diverse genomes (including human genome hg18, hg19, hg38, hs1 (T2T-CHM13), as well as mouse, worm, fly, yeast, SARS-CoV-2, and many others). Given a list of variants with chromosome, start position, end position, reference nucleotide, and observed nucleotide,

ANNOVAR can perform:

- **Gene-based annotation**: identify whether SNPs or CNVs cause protein-coding changes and the amino acids that are affected. Users can flexibly use RefSeq genes, UCSC genes, Ensembl/Gencode genes, or a custom gene definition system.
- **Region-based annotation**: identify variants in specific genomic regions, for example, conserved regions among multiple species, predicted transcription factor binding sites, segmental duplication regions, database of genomic variants, DNase I hypersensitivity sites, ENCODE H3K4Me1/H3K4Me3/H3K27Ac/CTCF sites, ChIP-Seq peaks, RNA-Seq peaks, ENCODE cCRE sites, or many other annotations on genomic intervals.
- **Filter-based annotation**: identify variants that are documented in specific databases, for example, whether a variant is reported in dbSNP, what is the allele frequency in the 1000 Genomes Project, NHLBI-ESP 6500 exomes, or Exome Aggregation Consortium (ExAC) or Genome Aggregation Database (gnomAD); calculate the SIFT/PolyPhen/LRT/MutationTaster/MutationAssessor/FATHMM/MetaSVM/MetaLR/MetaR scores; find pathogenic mutations from ClinVar; identify recurrent somatic mutations in COSMIC in specific cancer types; or many other annotations on specific mutations.
- **Other functionalities**: retrieve the nucleotide sequence at any user-specified genomic positions in batch, identify a candidate gene list for Mendelian diseases from exome data, and other utilities.

ANNOVAR has a variety of additional databases (gnomAD, ClinVar, dbNSFP, etc.) ranging from hundreds of MB to 100+ GB; however, in this minimal example we will be using the refGene database to populate gene-based annotations. More information regarding ANNOVAR can be found in the documentation.

In this section, ANNOVAR will be run on the single-variant analysis results, as the burden/SKAT-O tests already contain annotation information since they are gene-based tests.

In [ ]:
```
# Parameters:
input_file="./output/1000G.EUR.mwe.chr21.step2_pheno.regenie"
av_input="./output/1000G.EUR.mwe.chr21.step2_pheno.avinput"
output_file="./output/1000G.EUR.mwe.chr21.step2_pheno"
humandb="./annovar/humandb/"
# 1. Create ANNOVAR Input
awk 'NR > 1 {
    if (length($4) > 1) {
        print $1, $2, $2 + length($4) - 1, $4, $5, $3
    } else {
        print $1, $2, $2, $4, $5, $3
    }
}' OFS="\t" "${input_file}" > "${av_input}"
# 2. Run annovar
table_annovar.pl \
    "${av_input}" \
    ${humandb} \
    -buildver hg19 \
    -out "${output_file}" \
    -otherinfo \
    -remove \
    -polish \
    -nastring . \
    -protocol refGene \
    -operation g \
    -arg "-splicing 12" \
    -csvout
```

Plotting Results

Plots are generated from the full summary statistics generating Manhattan and Q-Q Plots

> **Kernel:** run the code cells in this section with the **R** kernel.

In [ ]:
```
rm(list=ls())
library("CMplot")
#library(fastman)
library(qqman)
```

In [ ]:
```
# Helper scripts for plots
source("data/basic_packages.R")
source("data/script.R")
```

Single Variant Analysis

In [ ]:
```
# Single Variant association analysis
single_variant_file <- "./output/1000G.EUR.mwe.chr21.step2_pheno.regenie"
single_variant_data <- format_single_variant(single_variant_file)
```

```
# Manhattan Plot
time_1 <- Sys.time()
plot_manhattan(
  single_variant_data,
  "./output/single_variant_manhattan",
  display = TRUE,
  title = "Single-variant analysis"
)
time_2 <- Sys.time()
# Plotting time
time_2 - time_1
```

In [ ]:
```
# Q-Q Plot
time_1 <- Sys.time()
plot_qq(
  single_variant_data,
  "./output/single_variant_qq",
  display = TRUE,
  title = "Single-variant analysis"
)
time_2 <- Sys.time()
# Plotting time
time_2 - time_1
```

Burden Analysis

In [ ]:
```
# Burden analysis
burden_file <- "./output_rarevariant/1000G.EUR.mwe.chr21.step2_rarevariant_b
burden_data <- format_gene_test(burden_file, test = "ADD")

# Manhattan Plot
time_1 <- Sys.time()
plot_manhattan(
  burden_data,
  "./output_rarevariant/burden_manhattan",
  display = TRUE,
  title = "Burden analysis"
)
time_2 <- Sys.time()
# Plotting time
time_2 - time_1
```

In [ ]:
```
# Q-Q Plot
time_1 <- Sys.time()
plot_qq(
  burden_data,
  "./output_rarevariant/burden_qq",
  display = TRUE,
  title = "Burden analysis"
)
time_2 <- Sys.time()
# Plotting time
time_2 - time_1
```

SKAT-O analysis

In [ ]:
```
# SKAT-O analysis
skato_file <- "./output_rarevariant/1000G.EUR.mwe.chr21.step2_rarevariant_sk
skato_data <- format_gene_test(skato_file, test = "ADD-SKATO")

# Manhattan Plot
time_1 <- Sys.time()
plot_manhattan(
  skato_data,
  "./output_rarevariant/skato_manhattan",
  display = TRUE,
  title = "SKAT-O analysis"
)
time_2 <- Sys.time()
# Plotting time
time_2 - time_1
```

In [ ]:
```
# Q-Q Plot
time_1 <- Sys.time()
plot_qq(
  skato_data,
  "./output_rarevariant/skato_qq",
  display = TRUE,
  title = "SKAT-O analysis"
)
time_2 <- Sys.time()
# Plotting time
time_2 - time_1
```

Post-processing

Create a final clean table, merging the annotation results back in, and visualize the results.

In [ ]:
```
library(data.table)
library(dplyr)
```

Helper Functions

The three helper functions below are shared across all three analyses (single-variant, burden, and SKAT-O):

- `read_regenie_results()` : parses a raw `.regenie` output file and computes a `P` column from `LOG10P` .
- `read_annovar_results()` : parses the ANNOVAR `_multianno.csv` annotation file.

- `print_result_summary()`: prints a quick overview of a results table, including `Total_rows` (the number of rows in the table — one row per tested variant for the single-variant results, or one row per gene/mask for the aggregate tests) and `Unique_results` (the number of distinct values in whichever `id_column` you pass in — e.g. the number of unique genes actually tested, when `id_column = "Gene"`).

In [ ]:
```r
# Read REGENIE results and create the P-value column
read_regenie_results <- function(input_file) {

  fread(input_file, data.table = FALSE) %>%
    filter(CHROM != "CHROM") %>%
    mutate(
      CHROM = as.numeric(CHROM),
      GENPOS = as.numeric(GENPOS),
      LOG10P = as.numeric(LOG10P),
      P = 10^(-LOG10P)
    ) %>%
    filter(
      !is.na(CHROM),
      !is.na(GENPOS),
      !is.na(P),
      P > 0,
      P <= 1
    )
}
```

In [ ]:
```r
# Read and clean the ANNOVAR multianno file
read_annovar_results <- function(annotation_file) {

  fread(annotation_file, data.table = FALSE) %>%
    mutate(
      Chr = as.numeric(Chr),
      Start = as.numeric(Start),
      Ref = as.character(Ref),
      Alt = as.character(Alt)
    )
}
```

In [ ]:
```r
# Print a concise result summary
print_result_summary <- function(
    df,
    analysis_name,
    id_column
) {

  summary_table <- data.frame(
    Analysis = analysis_name,
    Total_rows = nrow(df),
    Unique_results = n_distinct(df[[id_column]]),
    Chromosomes = n_distinct(df$CHROM),
    Minimum_P = min(df$P, na.rm = TRUE),
    Median_P = median(df$P, na.rm = TRUE),
```

```
    Maximum_P = max(df$P, na.rm = TRUE)
  )

  print(summary_table)

  cat("\nFirst results ordered by P value:\n")

  print(
    df %>%
      arrange(P) %>%
      head(10)
  )
}
```

Single Variant Analysis

Merges the single-variant REGENIE output with the ANNOVAR gene annotation, so each tested SNP is labeled with its overlapping/nearest gene.

In [ ]:
```
# Merge and create finalized result
prepare_single_variant_results <- function(
    regenie_file,
    annotation_file
) {
  regenie <- read_regenie_results(regenie_file)
  annotation <- read_annovar_results(annotation_file)
  regenie %>%
    left_join(
      annotation,
      by = c(
        "CHROM" = "Chr",
        "GENPOS" = "Start",
        "ALLELE0" = "Ref",
        "ALLELE1" = "Alt"
      )
    ) %>%
    transmute(
      SNP = ID,
      Chromosome = CHROM,
      Position = GENPOS,
      Reference = ALLELE0,
      Alternate = ALLELE1,
      Allele_frequency = A1FREQ,
      Test = TEST,
      Beta = BETA,
      SE = SE,
      Chi_square = CHISQ,
      LOG10P = LOG10P,
      P = P,
      Function = Func.refGene,
      Gene = Gene.refGene,
      Exonic_function = ExonicFunc.refGene,
      Amino_acid_change = AAChange.refGene
    ) %>%
```

```
    arrange(P)
}
single_variant_results <- prepare_single_variant_results(
  regenie_file = "./output/1000G.EUR.mwe.chr21.step2_pheno.regenie",
  annotation_file = "./output/1000G.EUR.mwe.chr21.step2_pheno.hg19_multianno
)
print_result_summary(
  df = single_variant_results,
  analysis_name = "Single-variant analysis",
  id_column = "SNP"
)
head(single_variant_results)
```

Burden Analysis

Filters the burden test output down to the `mask1` mask (dropping the singleton-only rows) and extracts the gene name from each mask's `ID` (e.g. `GENE.mask1.0.01` → gene `GENE`).

In [ ]:
```
# Merge and create finalized result
prepare_burden_results <- function(input_file) {

  read_regenie_results(input_file) %>%
    filter(
      ALLELE1 == "mask1.0.01",
      TEST == "ADD"
    ) %>%
    mutate(
      Gene = sub("\\..*$", "", ID),
      Mask = sub("^[^.]+\\.", "", ID)
    ) %>%
    transmute(
      Gene,
      Mask,
      ID,
      Chromosome = CHROM,
      Position = GENPOS,
      Test = TEST,
      Beta = BETA,
      SE = SE,
      Chi_square = CHISQ,
      LOG10P = LOG10P,
      P = P
    ) %>%
    arrange(P)
}
burden_results <- prepare_burden_results(
  "./output_rarevariant/1000G.EUR.mwe.chr21.step2_rarevariant_burden_results
)
print_result_summary(
  df = burden_results,
  analysis_name = "Burden analysis",
  id_column = "Gene"
```

```
)
head(burden_results)
```

SKAT-O Analysis

Same idea as the burden analysis above, but filtered to the `ADD-SKATO` test statistic for the `mask1` mask.

In [ ]:
```
# Merge and create finalized result
prepare_skato_results <- function(input_file) {

  read_regenie_results(input_file) %>%
    filter(
      ALLELE1 == "mask1.0.01",
      TEST == "ADD-SKATO"
    ) %>%
    mutate(
      Gene = sub("\\..*$", "", ID),
      Mask = sub("^[^.]+\\.", "", ID)
    ) %>%
    transmute(
      Gene,
      Mask,
      ID,
      Chromosome = CHROM,
      Position = GENPOS,
      Test = TEST,
      Chi_square = CHISQ,
      LOG10P = LOG10P,
      P = P
    ) %>%
    arrange(P)
}
skato_results <- prepare_skato_results(
  "./output_rarevariant/1000G.EUR.mwe.chr21.step2_rarevariant_skato_results_
)
print_result_summary(
  df = skato_results,
  analysis_name = "SKAT-O analysis",
  id_column = "Gene"
)
head(skato_results)
```

Questions

Use the outputs generated above — in particular the tables printed by `print_result_summary()` — to answer the following.

- How many genes did you analyze in each test?

Hint: look at the `Unique_results` count that `print_result_summary()` printed for the burden and SKAT-O analyses above (`id_column = "Gene"`) — that's the number of distinct genes actually tested in each of those two aggregate tests.

- What's the gene with the lowest p-value in SKAT-O?

Hint: `skato_results` is already sorted by ascending P (see `arrange(P)` inside `prepare_skato_results()`), and `print_result_summary()` prints the first 10 rows in that order — the top row's `Gene` is your answer.

Additional exercise

Perform the analysis again but this time using chromosome 22.

Hint: copy commands from above and change `chr` to '22'