INTRODUCTION TO GENOME-WIDE ASSOCIATION STUDIES (GWAS)

Filippo Biscarini, Oscar Gonzalez-Recio, Christian Werner

23 - 27 June 2025

https://www.physalia-courses.org/

This course manual will guide you through the steps involved in performing a GWAS on the rice data set we analyzed during the course. Although every data set has unique features and requires customized data handling, a GWAS pipeline routinely comprises:

1. Data preparation
2. Data pre-processing
    - Initial and exploratory data analysis (IDA & EDA)
    - Quality control
    - Imputation
3. Genome-Wide Association Study

While during the course we will primarily work on the AWS cloud computing platform, the code provided in this manual will enable you to run the rice GWAS pipeline on your local computer and make use of the code to create your own GWAS pipelines.

Software

Building a pipeline to perform data preparation, data pre-processing and GWAS will mostly require different software and/or packages. There is a variety of efficient open-source solutions available for each of those steps. Choosing software and packages is often driven by personal preferences rather than substantial differences in performance and features.

To guide you through the different steps of a GWAS workflow, we will build a pipeline using three well-established and relatively user-friendly programs, which will allow efficient data handling even with larger data sets ("massive" data sets may require more computationally efficient solutions, though). However, this approach is not written in stone and we do encourage you to try different options and keep yourself up-to-date as soon as you can handle the basics.

RStudio (R)

R is a programming language and free software environment for statistical computing and graphics. The R language is widely used for data manipulation and analysis. R comes with a large diversity of additional packages which can be downloaded from the Comprehensive R Archive Network (CRAN). You can likely find a package there which will help you solve your problem. R runs on a wide variety of UNIX platforms, Windows and macOS. While R has a command line interface, there are several third-party graphical user interfaces (IDEs), with RStudio probably being the most popular.

During this course, we're going to use various R packages for data preparation, data analysis, and graphical illustration of the data. We also will run the GWAS in R using the R package **rrBLUP**. While rrBLUP might not be the most powerful and flexible GWAS package out there, it is fairly easy to use and a suitable option to run your first GWAS and produce a Manhattan plot. However, be curious and try other GWAS software to find what works best for you and for your data set. During the course, we will introduce other packages which are also great for beginners.

If you would like to improve your R programming skills for data analysis, check "R for Data Science" by Hadley Wickham & Garrett Grolemund. The book is freely available online (https://r4ds.hadley.nz/).

PLINK

PLINK is a powerful open-source genome analysis toolset. It offers a plethora of computationally efficient functions for data pre-processing, filtering, formatting, and analysis.

PLINK 1.90 beta is available for Linux, macOS, and Windows. It comes as a binary and has to be called using the command line. We will use PLINK for data pre-processing (quality control) and conversion between different data formats.

You can download PLINK 1.9 (stable beta version) and get detailed information on all the functions and arguments here:

https://www.cog-genomics.org/plink/

PLINK 2.00 alpha is also available. We recommend using the stable PLINK 1.90 beta version, though, which provides all the features needed for our analysis.

Beagle

Beagle is a free software package for phasing genotypes and imputation of ungenotyped (missing) markers. It is widely used and provides a fast and accurate all-in-one solution for phasing and imputation under default settings. Beagle version 5.5 is the most recent version of the software. It requires Java version 8 and can be run on Linux, macOS, and Windows using the command line.

You can download the beagle.jar package and a short manual here:

https://faculty.washington.edu/browning/beagle/beagle.html

During the course, we will also cover two other options for imputation: mean imputation and K-Nearest Neighbor Imputation (KNNI). While not covered here, KNNI is a suitable imputation method when you're working with a species without a reference genome (i.e., the position and order of the SNP markers are unknown)

MobaXterm

MobaXterm is only necessary for **Windows users**. While Linux and macOS provide a UNIX-based terminal by default, the Windows terminal is not UNIX-based and makes working in the command line a bit complicated and unpleasant. MobaXterm provides all the important UNIX commands and remote network tools to the Windows desktop. If you are already using another emulator (e.g., PuTTY, Cygwin, ...) you won't need MobaXterm and can use your preferred solution instead. During the course, will use MobaXterm mainly to access the AWS cloud computing platform, but you can also start a "local terminal" and use it on your local machine to navigate through your file system and perform various tasks.

Data preparation

Data preparation comprises all the steps required to ensure your data files are in good shape and can be read in by the software and packages you use. While this may be necessary several times during your pipeline workflow (when different software and packages are called), initial data preparation is crucial to fix potential inconsistencies and errors in our data set. Frequent issues include non-matching IDs in genotype and phenotype files, missing or redundant whitespace and tab characters, incorrect genotype formats, and missing or unnecessary headers.

All the code shown in this manual to read in files, perform individual tasks, run software and call scripts, is either run in the **command line** or in **RStudio**. A clear and well-organized folder structure to store your data and the files produced during your GWAS workflow can considerably simplify the job. In this manual, however, we will skip this part and not provide full paths and locations for the files.

Rice data set

The rice data set comprises GBS marker data and phenotypic records for several continuous traits on 153 accessions from 5 subpopulations. This data set is freely available online.

We use the **command line** to move to the directory where we would like to store the data set, download it from an online server using `wget`, and then `tar` to extract the genotype files.

```
wget https://zenodo.org/record/50803/files/GBSgenotypes.tar.gz
wget https://zenodo.org/record/50803/files/plantgrainPhenotypes.txt

tar -xvzf GBSgenotypes.tar.gz
```

The extracted data set comprises two genotype files in standard PLINK format (***GBSnew.ped***, ***GBSnew.map***) and a phenotype file (***plantgrainPhenotypes.txt***).

```
ls -l
total 287084
-rw-r--r--    1 USER  UsersGrp  4096   29381995 May 14 08:57 GBSgenotypes.tar.gz
-rw-r--r--    1 USER  UsersGrp  4096    4291480 Aug  2  2013 GBSnew.map
-rw-r--r--    1 USER  UsersGrp  4096  260286484 Aug  2  2013 GBSnew.ped
-rw-r--r--    1 USER  UsersGrp  4096       7364 May 23 20:58 plantgrainPhenotypes.txt
```

Data examination

The **command line** provides various options to examine text files at different levels of detail. We are going to show some of these. This will help us get a first idea of what our data set looks like.

1. Command line text editors like **vim**, **emacs** or **nano** provide the most powerful and flexible way to open, view and manipulate entire text files. They are much more efficient than graphical text editors like Notepad (**Notepad++** is a great alternative to the Windows notepad application if you are looking for a more efficient graphical text editor).

```
vim GBSnew.ped # Open text file using vim

:q # Used in VIM to close the text file
```

GBSnew.ped contains a large amount of information, including the marker calls for the 153 rice genotypes. It turns out that using a text editor is not the most efficient way to retrieve some basic information when the text file is large.

2. Utility commands like `head` and `wc` (word count) are more suitable to extract key information quickly rather than viewing the entire document.

```
wc -l GBSnew.ped
391 GBSnew.ped # File contains 391 lines (391 genotypes).

head -5 GBSnew.map # print the first 5 lines.
1 S1_3417  0 3417
1 S1_5039  0 5039
1 S1_5170  0 5170
1 S1_11884  0 11884
1 S1_11995  0 11995

wc -l GBSnew.map # File contains 166,418 lines (markers).
166418 GBSnew.map

head -5 plantgrainPhenotypes.txt # Print first 5 lines of the phenotype file.
Accession       PH      PL      FLL     FLW     SL      SW      SR
A201    93.40   23.27   328.67  10.80   10.38   2.43    4.27
A301    81.20   18.40   217.40  10.27   10.56   2.72    3.88
ADELAIDECHIAPPELLI      100.07  18.96   208.33  9.20    10.26   4.13    2.49
AIACE   79.40   18.23   203.33  9.47    9.82    2.94    3.35

wc -l plantgrainPhenotypes.txt # File contains 154 lines (153 accessions + header).
154 plantgrainPhenotypes.txt
```

Data preparation

As we can see above, the file ***plantgrainPhenotypes.txt*** contains accession IDs and values for 7 traits. Here, we will perform a GWAS on plant height (PH) only and remove all other traits from the phenotype file. To correct for population structure during the GWAS, we will add a column with subpopulation information. Using the file ***rice_group.reference***, each rice accession will be assigned to one of five subpopulations.

Note: the file ***rice_group.reference*** is not available from the online repository but will be shared with you during the course. It is located in the "cross_reference" folder of the git repository.

We use **RStudio**

```r
library("data.table") # Package for fast and efficient data manipulation.
library("dplyr")

phenotypes <- fread("plantgrainPhenotypes.txt")
ref_group <- fread("rice_group.reference")

## Add subpopulation to phenotype file.
phenotypes$population = ref_group$population[match(phenotypes$Accession, ref_group$id)]

## Rename column and check.
names(phenotypes)[1] <- "id"
head(phenotypes)

## Write out new phenotype file.
fwrite(x = phenotypes[ , c("id", "population", "PH"), with=FALSE],
       file = "rice_phenotypes.txt", sep=" ")

## Create file "ids": used with PLINK to indicate which genotypes to keep.
phenotypes$fam <- rep("NF1", nrow(phenotypes))
fwrite(x = phenotypes[, c("fam","id")], file = "ids", sep="\t", col.names = FALSE)

## "update.ids" is used later to calculate stratified allele frequencies.
ref_group$oldfam <- rep("NF1", nrow(ref_group))
ref_group$oldid <- ref_group$id
ref_group <- ref_group %>%
  select(oldfam, oldid, population, id)

fwrite(x = ref_group, file = "update.ids", col.names = FALSE, sep = "\t")
```

rice_phenotypes.txt is a newly created phenotype file that we will use from now on. We also generated the file ***ids***, which contains all accession names and will be used with **PLINK** to show which genotypes we want to keep. The file ***update.ids*** will be used later for the calculation of stratified allele frequencies.

We can go back to the **command line** and have a look at some of the new files.

```
wc -l rice_phenotypes.txt
wc -l ids
```

To make sure the accession IDs in the two genotype files and the new phenotype file match, we need to remove substrings and the letter "a" from the ***GBSnew.ped*** file. We use `sed` for that.

```
sed -i 's/\_//g' GBSnew.ped
sed -i 's/a//g' GBSnew.ped
```

Note: non-matching IDs in different files are a very common issue that can occur before or during data processing and will adversely affect your analysis, if not taken care of. The replacements done above are specific for our data set here, and your own data set will likely require different corrections (if any). Always check IDs thoroughly and make sure they match!

Especially when data sets are large, some steps of a GWAS pipeline can be computationally demanding and might require quite some time to run. We bypass this problem by reducing the size of the data set with **PLINK**. Since we already know the position of potentially causal genes in the rice data set, we can pre-select 4 of the 12 rice chromosomes. We only keep those accessions with phenotypic information in the genotype files, as indicated by the ***ids*** file.

Note: `./` is used to call an executable (***plink.exe*** here) from the command line, given that you are currently in the directory where the executable is located. If you are in a different directory, specify the exact path. The same applies to the file(s) that you would like to manipulate (***GBSnew.map*** & ***GBSnew.ped*** here). In the example below, we assume that the executable and the files are in the same directory.

```
./plink --file GBSnew --keep ids --chr 1,2,6,7 --recode --out rice

PLINK v1.90b7 64-bit (16 Jan 2023)              www.cog-genomics.org/plink/1.9/
(C) 2005-2023 Shaun Purcell, Christopher Chang   GNU General Public License v3
Logging to rice.log.
Options in effect:
  --chr 1,2,6,7
  --file GBSnew
  --keep ids
  --out rice
  --recode

16162 MB RAM detected; reserving 8081 MB for main workspace.
.ped scan complete (for binary autoconversion).
Performing single-pass .bed write (62169 variants, 391 people).
--file: rice-temporary.bed + rice-temporary.bim + rice-temporary.fam written.
62169 variants loaded from .bim file.
391 people (0 males, 0 females, 391 ambiguous) loaded from .fam.
Ambiguous sex IDs written to rice.nosex .
--keep: 147 people remaining.
```

```
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 147 founders and 0 nonfounders present.
Calculating allele frequencies... done.
Total genotyping rate in remaining samples is 0.92673.
62169 variants and 147 people pass filters and QC.
Note: No phenotypes present.
--recode ped to rice.ped + rice.map ... done.
```

Plink produces a new ***rice.ped*** file and a ***rice.map*** file, which contain our reduced rice data set. It also writes the above information into a ***.log*** file and creates a ***rice.nosex***. The latter two files are of no use for us can be deleted.

```
rm rice.log rice.nosex
```

Data pre-processing

Initial data analysis (IDA) and **exploratory data analysis (EDA)** comprise the inspection, cleaning, transforming and modeling of our data before addressing the actual research question (e.g., a GWAS). The purposes of IDA and EDA are to ensure that the subsequent statistical analysis can be performed efficiently and to minimize the risk of incorrect or misleading results. IDA and EDA often occupy most of the time allocated to the analysis workflow. Although IDA and EDA comprise different tasks during the upstream data analysis, the transition between them is somewhat fluid and definitions often vary. Hence, the definitions below may not exactly match those you find elsewhere. They cover, however, the process of upstream data analysis as a whole, which we focus on here.

Initial data analysis mainly focuses on data cleaning, a first screening, and transformation (if necessary) to ensure data quality and confirm that our data set meets the relevant distributional and model assumptions. Data cleaning may include steps such as elimination of duplicate records, handling of missing values, identification of systematic errors, or correction of coding inconsistencies.

Exploratory data analysis is used to examine data sets and summarize their main characteristics. EDA helps to discover patterns in the data, spot anomalies and outliers, test a hypothesis, and check our assumptions. EDA tells us what data can reveal beyond the formal modeling or hypothesis testing and provides a better understanding of data set variables and their interactions. It can also help to determine if the statistical techniques you are considering for data analysis are appropriate.

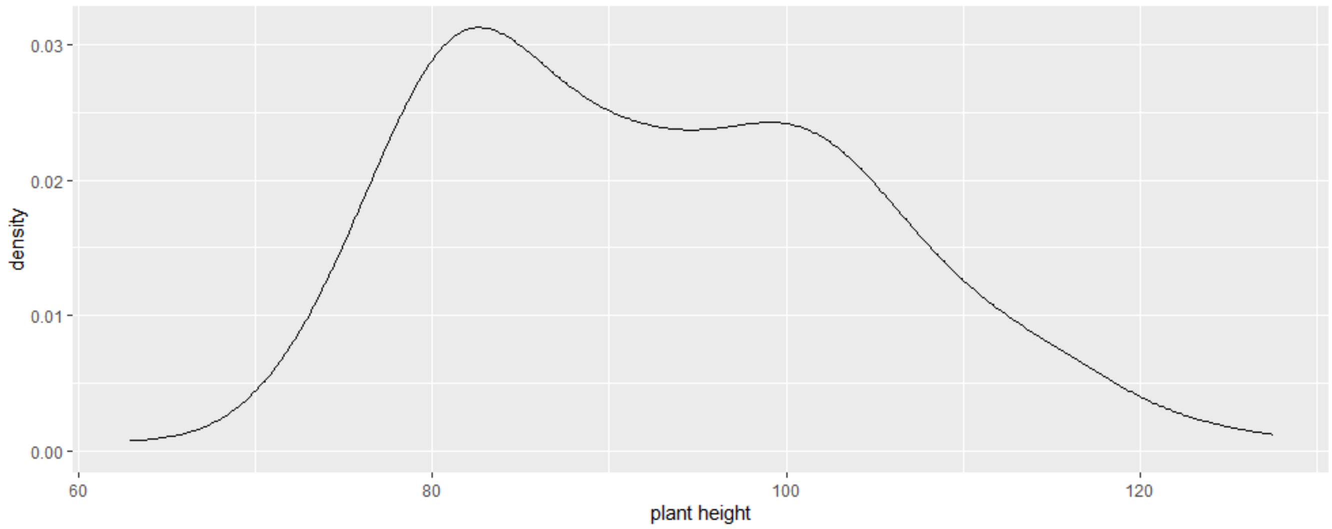
IDA and EDA often employ data visualization methods to check distributional characteristics, identify relationships between variables, identify potential cofactors, and spot inconsistencies. We will run an IDA and EDA for both our **phenotypic data** and **genotypic data**.

Phenotypic data

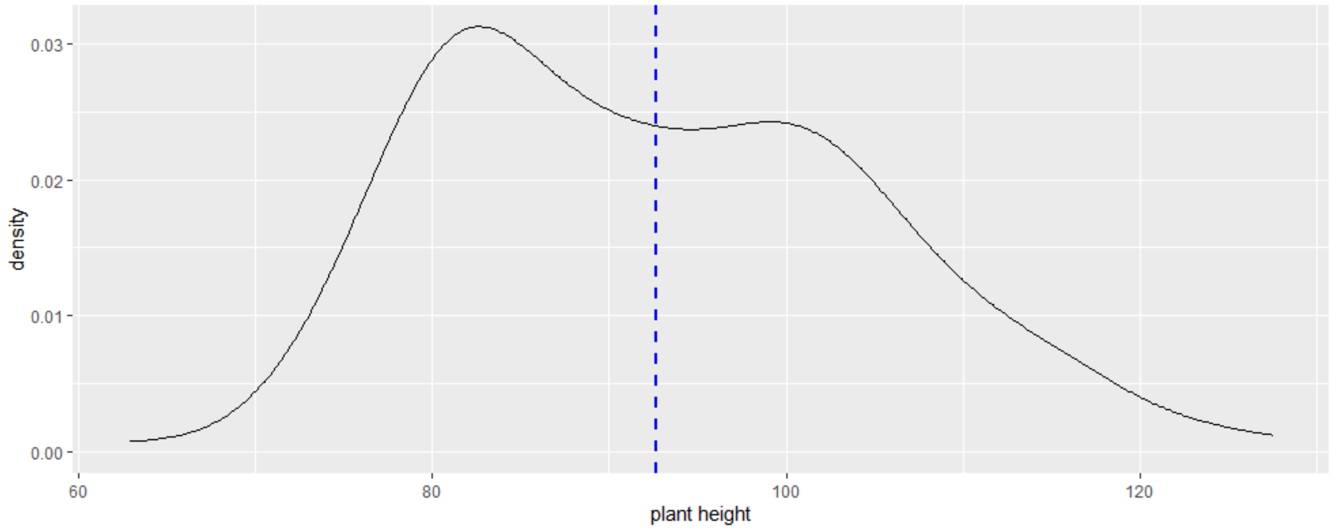
Phenotypic data analysis is a fundamental step before running a GWAS. There are many reasons our phenotypic data or the modeling approach might not be suitable to answer our research question. As a result, a GWAS might not give us meaningful results ("crap in, crap out").

Since data analysis and preparation is an entire topic on its own that cannot be covered here in detail, we made sure in advance that our data sets are suitable and have been corrected for potential cofactors. However, we give a few examples of how data visualization can be efficiently done in **RStudio** (taken from ***phenotypes.R***) using our rice data set.

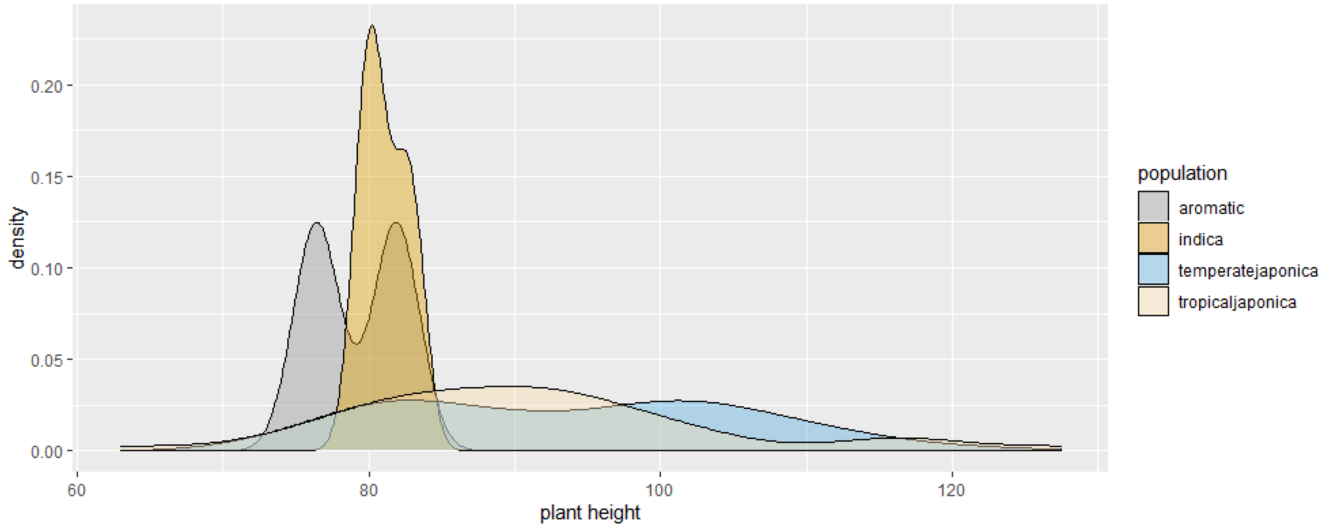
```
library("tidyr")
library("dplyr")
library("ggplot2")

pheno_continuous <- read.table("rice_phenotypes.txt", header = TRUE)

D <- pheno_continuous %>%
  group_by(population) %>%
  summarise(N=n(), avgPH=mean(PH), stdPH=sd(PH), minPH=min(PH),
            maxPH=max(PH))

D

## A tibble: 4 x 6
  population            N avgPH stdPH minPH maxPH
  <chr>             <int> <dbl> <dbl> <dbl> <dbl>
1 aromatic              2  79.1  3.87  76.4  81.9
2 indica                5  81.1  1.51  79.5  83.1
3 temperatejaponica   108  93.9 12.1   72.7 124.
4 tropicaljaponica     38  91.2 13.0   62.9 127.

## Density plot
p <- ggplot(pheno_continuous, aes(x=PH)) + geom_density()
p <- p + xlab("plant height")
p
```



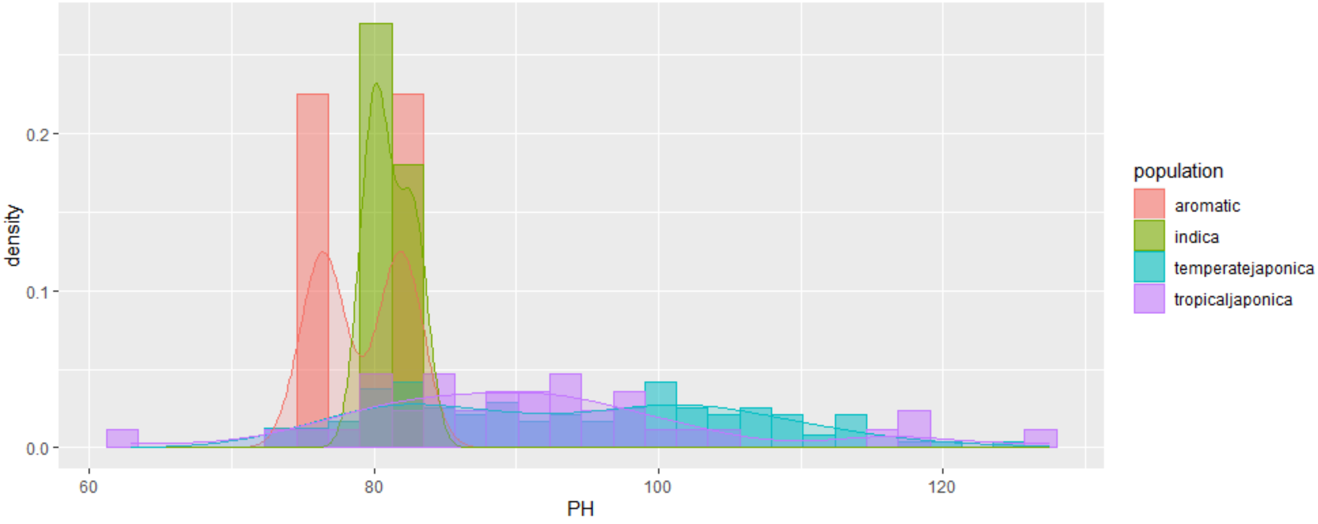
```r
## Density plot with average line
p <- ggplot(pheno_continuous, aes(x=PH)) + geom_density()
p <- p + geom_vline(aes(xintercept=mean(PH)), color="blue", linetype="dashed", linewidth=1)
p <- p + xlab("plant height")
p
```



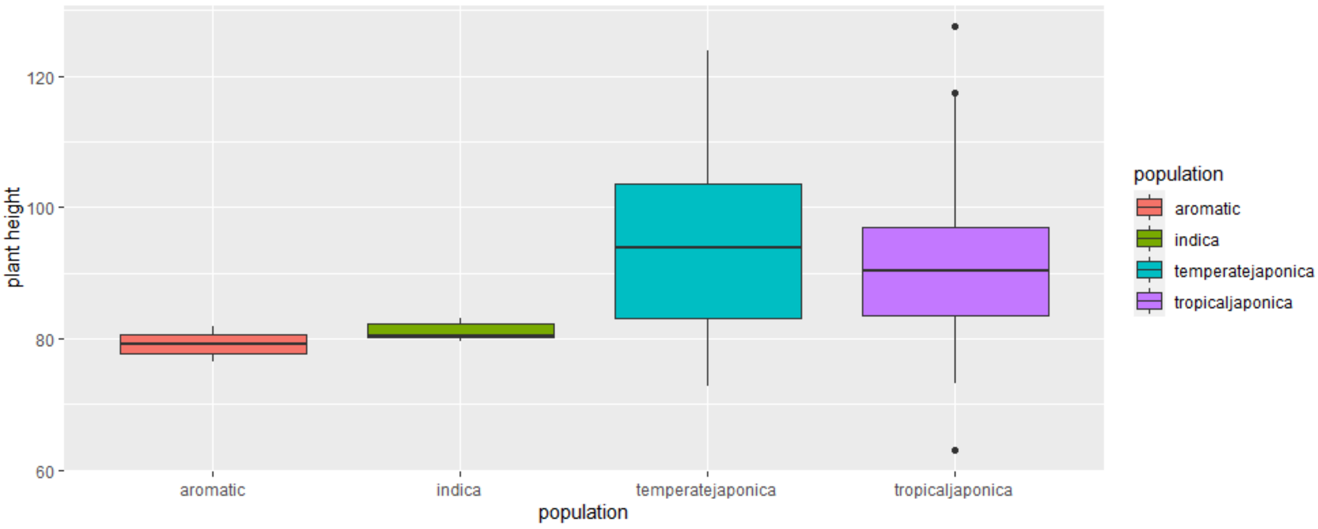
```r
## Density plot with color shades per population
p <- ggplot(pheno_continuous, aes(x=PH)) + geom_density(aes(fill=population), alpha=0.4)
p <- p + scale_fill_manual(values=c("#999999", "#E69F00", "#56B4E9","#FFE4B5"))
p <- p + xlab("plant height")
p
```



```r
## Density plot with color shades per population + histograms
p <- ggplot(pheno_continuous, aes(x=PH, color=population, fill=population))
p <- p + geom_histogram(aes(y=after_stat(density)), alpha=0.5, position="identity")
p <- p + geom_density(alpha=.2)
p
```



```r
## Boxplots
p <- ggplot(pheno_continuous, aes(x=population,y=PH)) + geom_boxplot(aes(fill=population))
p <- p + xlab("population") + ylab("plant height")
p
```



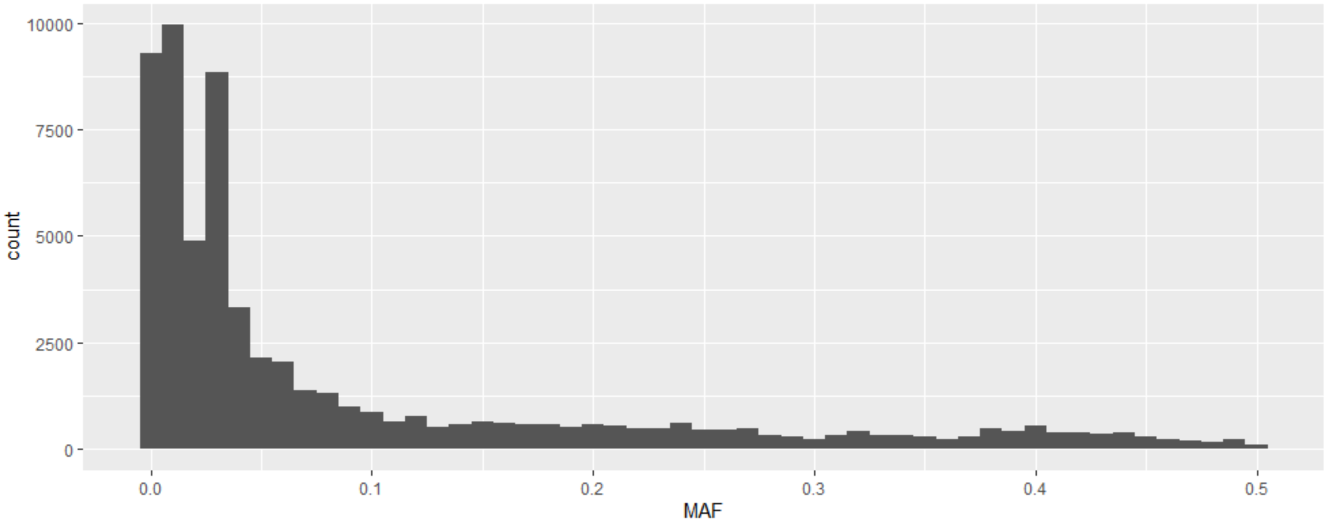
Genotypic data analysis

To explore our genotypic data, **PLINK** provides many options that are worth having a look at. While the analysis and output file generation will be done in PLINK using the command line...

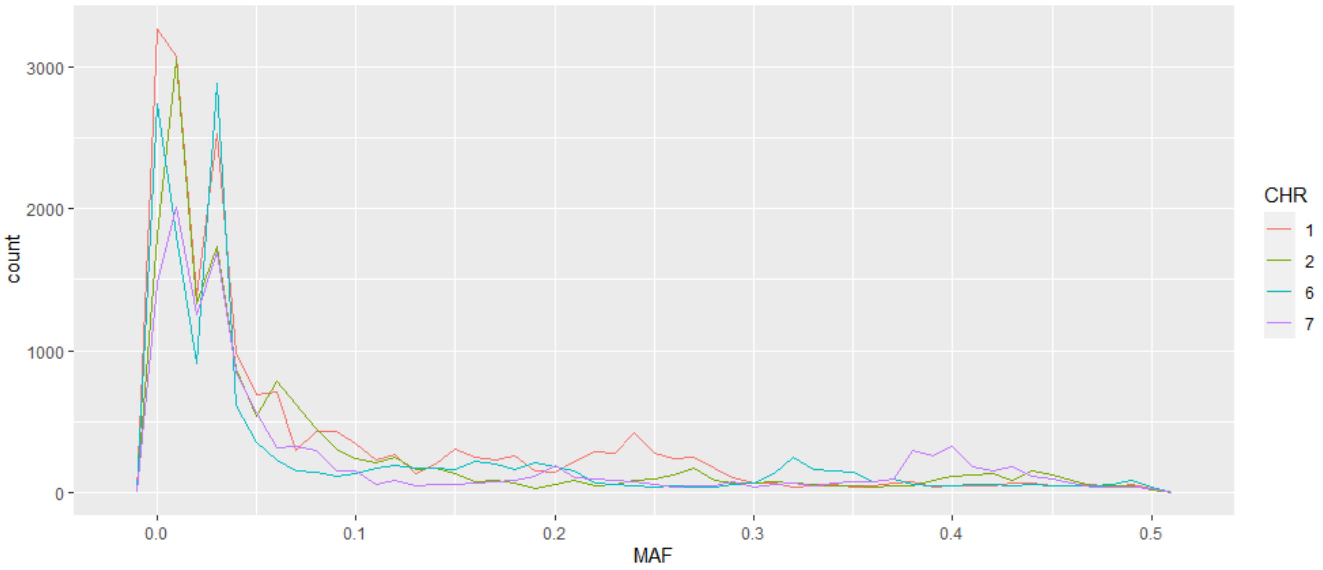
```
# 3 different levels of detail
./plink --file rice --freq --out rice         # Minor allele frequency (MAF).
./plink --file rice --freq counts --out rice  # Allele count report.
./plink --file rice --freqx --out rice        # More informative genotype count report.

## Stratified MAF and allele frequencies
./plink --file rice --update-ids update.ids --recode --out rice_pop
./plink --file rice_pop --freq --family --out rice_pop

## Missing markers per individual and missing SNP calls
./plink --file rice --missing --out rice # Sample-based variant-based missing data.

## Hardy-Weinberg equilibrium
./plink --file rice --hardy --out rice # Genotype counts and Hardy-Weinberg equilibrium exact test statistics.

## Heterozygosity
./plink --file rice --het --out rice # Observed and expected autosomal homozygous genotype counts for each sample, and reports method-of-moments F coefficient estimates.
```

... we use **RStudio** to visualize the results.

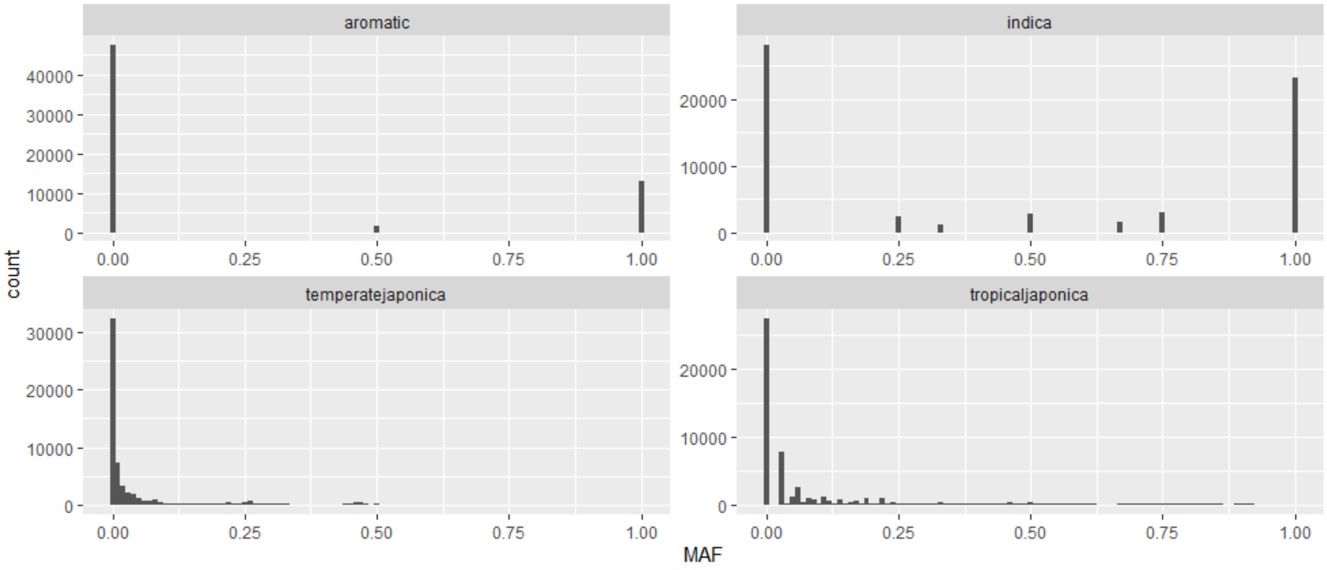
```
library("dplyr")
library("ggplot2")
library("reshape2")
library("data.table")

rice.frq <- fread("rice.frq")
rice.frq = na.omit(rice.frq)

rice.frq %>%
  summarise(avg_frq=mean(MAF), std_dev=sd(MAF), max_frq=max(MAF), min_frq=min(MAF))

rice.frq %>%
  group_by(CHR) %>%
  summarise(avg_frq=mean(MAF), std_dev=sd(MAF), max_frq=max(MAF), min_frq=min(MAF))

p <- ggplot(rice.frq, aes(MAF))
p <- p + geom_histogram(binwidth = 0.01)
p
```



```
p <- ggplot(rice.frq, aes(MAF, colour = as.factor(CHR)))
p <- p + geom_freqpoly(binwidth = 0.01)
p <- p + guides(colour=guide_legend(title="CHR"))
p
```



```
rice.frq <- fread("rice_pop.frq.strat")
rice.frq %>%
  group_by(CLST) %>%
  summarise(avg_frq=mean(MAF), std_dev=sd(MAF), max_frq=max(MAF), min_frq=min(MAF))

p <- ggplot(rice.frq, aes(MAF))
p <- p + geom_histogram(binwidth = 0.01)
p <- p + facet_wrap(~CLST, scales = "free")
p
```



Genotype filtering

Filtering markers for minor allele frequency (MAF), missing calls per marker, and missing calls per individual is a standard procedure during data pre-processing. Thresholds are not written in stone and somewhat subjective, but some common value ranges have evolved over time. Values may be different for SNP arrays, GBS, and whole-genome sequences. GWAS literature with comparable genotype data sets is always a good starting point to identify suitable thresholds.

We will filter the rice genotypes data for

- variants with a minor allele frequency below 5%
- variants with missing call rates exceeding 5%
- samples with missing call rates exceeding 20%

Using PLINK, this can be achieved with a single call to the software:

```
./plink --file rice --geno 0.05 --mind 0.2 --maf 0.05 --recode --out rice_filtered

PLINK v1.90b7 64-bit (16 Jan 2023)             www.cog-genomics.org/plink/1.9/
(C) 2005-2023 Shaun Purcell, Christopher Chang   GNU General Public License v3
Logging to rice_filtered.log.
Options in effect:
  --file rice
  --geno 0.05
  --maf 0.05
  --mind 0.2
  --out rice_filtered
  --recode

16162 MB RAM detected; reserving 8081 MB for main workspace.
.ped scan complete (for binary autoconversion).
```

```
Performing single-pass .bed write (62169 variants, 147 people).
--file: rice_filtered-temporary.bed + rice_filtered-temporary.bim +
rice_filtered-temporary.fam written.
62169 variants loaded from .bim file.
147 people (0 males, 0 females, 147 ambiguous) loaded from .fam.
Ambiguous sex IDs written to rice_filtered.nosex .
7 people removed due to missing genotype data (--mind).
IDs written to rice_filtered.irem .
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 140 founders and 0 nonfounders present.
Calculating allele frequencies... done.
Total genotyping rate in remaining samples is 0.937544.
13360 variants removed due to missing genotype data (--geno).
35101 variants removed due to minor allele threshold(s)
(--maf/--max-maf/--mac/--max-mac).
13708 variants and 140 people pass filters and QC.
Note: No phenotypes present.
--recode ped to rice_filtered.ped + rice_filtered.map ... done.
```

You will see that changing these numbers within common ranges used in the literature will most often have only a minor effect on the GWAS output. Once you have generated your pipeline, changing the filtering thresholds and comparing their effects on the number of markers, number of individuals and the final GWAS will help you gain experience in data analysis.

Imputation

Like all regression methods, GWAS cannot deal with missing values. Hence, missing marker values need to be imputed before we can regress phenotypes on genotypes to identify potentially causal variants. Imputation can be done in many different ways and qualities. There are various solutions for rather simple and sophisticated imputation methods. In this course, we are going to cover three common methods.

1. Mean imputation (e.g., with provided scripts or rrBLUP)
2. K-Nearest Neighbor Imputation (KNNI)
3. Imputation with Beagle based on a Hidden Markov Model (HMM)

Mean imputation

Mean imputation replaces missing marker calls by the mean value of the marker across the population. It is the fastest and simplest way to impute missing markers, but it can also be quite inaccurate. It may, however, be an adequate method if the fraction of missing markers is very low and the marker density is high (i.e., genomic regions are represented by many markers). Mean imputation often is the default imputation method implemented in GWAS software including rrBLUP, the package which we will use in this manual. rrBLUP also provides an EM imputation algorithm that was designed for use with GBS (genotyping-by-sequencing) markers, which tend to be high density in but with lots of missing data. We also provide scripts for mean (and median) imputation for you to run it yourself.

K-Nearest Neighbor Imputation (KNNI)

K-Nearest Neighbor Imputation is a general imputation method applicable to any type of data, including (but not limited to) genotype data. It is more accurate than mean imputation and particularly useful when no reference genome is available (i.e., marker locations on the genome are unknown). It can also be used for imputation in polyploid species, while Beagle and most other imputation programs were developed for diploid species. KNNI uses a distance matrix between samples based on data (e.g., Euclidean distances or Hamming distances between SNP genotypes). This method will be demonstrated during the course using a separate R script.

Beagle

Beagle uses a localized haplotype clustering imputation algorithm. It makes use of a Hidden Markov Model (HMM) to find the most likely haplotype pair given the genotype data for that individual and the haplotype frequency in the population. Imputation with Beagle is the most sophisticated of the three imputation algorithms and can achieve the highest imputation accuracies (in diploid species with known marker location and order).

Besides Beagle's localized haplotype clustering imputation algorithm, there are various other algorithms that use different flavors of HMMs to find an efficient balance between imputation accuracy and computational speed. Beagle is relatively user-friendly, accurate under default settings, well-supported, and widely used. We will focus on imputation with Beagle in this manual.

Imputation with Beagle

We will use **PLINK** to create a ***.vcf*** file that can be read by **Beagle**. Beagle requires Java to run on your local machine. The ***-Xmx[GB]g*** parameter sets an upper bound on the memory pool in gigabytes. You can try to increase the memory pool. However, depending on your computer and operating system, this may work or not. Afterwards, we use PLINK to recode the imputed ***.vcf*** file for our GWAS analysis.

```
## Rice data
./plink --file rice_filtered --recode vcf --out rice_filtered

# Imputation of genotypic data in the rice and dogs data set
java -Xmx1g -jar beagle.27Feb25.75f.jar gt=rice_filtered.vcf out=rice_imputed

# Generate .ped and .map files from Beagle's output (.vcf format)
./plink --vcf rice_imputed.vcf.gz --recode --out rice_imputed

## Prepare data for GWAS (0/1/2-coded marker calls): "rice_imputed.raw"
./plink --file rice_imputed --recode A --out rice_imputed
```

In case you get an error message when running Beagle, an outdated Java version might be the reason. Run "java -version" in the command line and check if it's up-to-date.

GWAS analysis

We will compare three GWAS models which correct for population structure in slightly different ways. These three models are commonly found in the plant genetics literature, while models 2 and 3 are less common in animal genetics. We will see that applying these three models on the rice data set produces relatively similar results. However, this might be different in other data sets and comparing different models in your own data set is never a bad idea!

Here, our main intention is to demonstrate how to correct for relatedness and population structure in a GWAS. The three GWAS models correct for population structure by including

1. A genomic relationship matrix (K model)
2. A genomic relationship matrix + known subpopulation structure (subgroup assignment)
3. A genomic relationship matrix + principal components to capture unknown subpopulation structure (P + K model)

Using a fourth GWAS model, we also demonstrate what happens if we do not correct for any population structure in the rice data set. As discussed during the course, this approach is not recommended, since it results in an inflation of marker effects if there is population structure in the data set.

We will perform the GWAS in **RStudio**. At first, we need to read in the imputed genotype file and the phenotype file, and convert them into the format required by the ***rrBLUP*** R package.

```
## Packages used
library("data.table")
library("dplyr")
library("rrBLUP")
library("qqman")

## Read in genotype file which we imputed before using Beagle
snp_matrix <- fread("rice_imputed.raw", header = TRUE)

snp_matrix[1:5,1:8]

   FID                IID PAT MAT SEX PHENOTYPE S1_59126_T S1_170244_T
1: NF1               A201   0   0   0        -9          0           0
2: NF1               A301   0   0   0        -9          0           0
3: NF1 ADELAIDECHIAPPELLI   0   0   0        -9          0           2
4: NF1              AIACE   0   0   0        -9          0           0
5: NF1        AKITAKOMACHI   0   0   0        -9          0           0


## Extract marker calls from snp_matrix
X <- as.matrix(snp_matrix[,-c(1:6)])

X <- X-1 # Change coding to -1, 0 ,1 (required to calculate GRM in rrBLUP)
colnames(X) <- gsub("\\_[A-Z]{1}$","",colnames(X))
rownames(X) <- snp_matrix$IID # Set row names using original snp_matrix file

X[1:5, 1:5]
```

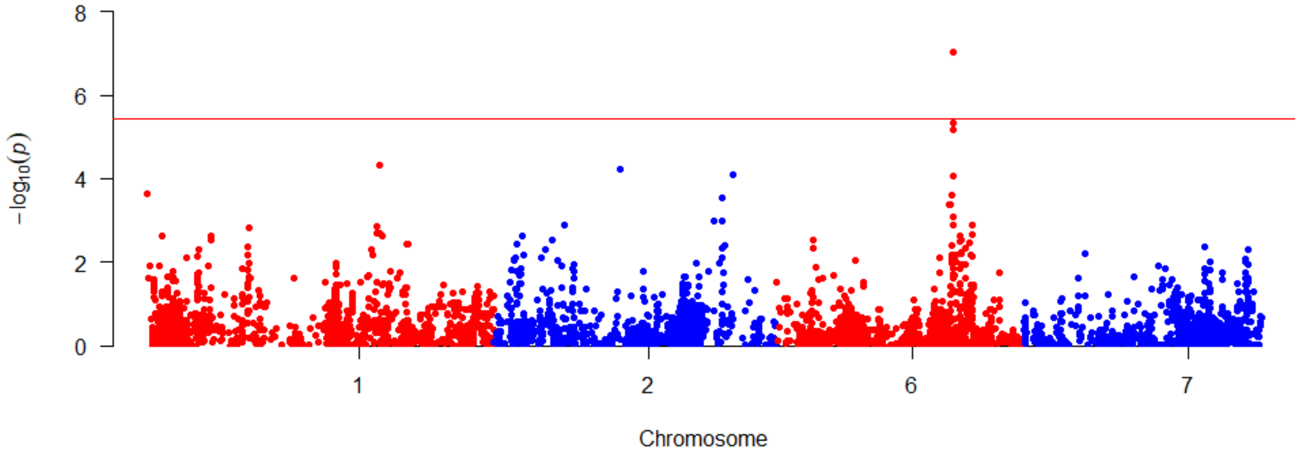
```
                   S1_59126 S1_170244 S1_330484 S1_594195 S1_628091
A201                     -1        -1        -1         1         1
A301                     -1        -1        -1         1        -1
ADELAIDECHIAPPELLI       -1         1         1        -1        -1
AIACE                    -1        -1        -1        -1        -1
AKITAKOMACHI             -1        -1        -1        -1        -1

## Calculate the genomic relationship matrix
GRM <- A.mat(X)

# Read in .map file with marker location information
SNP_INFO <- fread("rice_imputed.map")
names(SNP_INFO) <- c("Chr","SNP","cM","Pos")
SNP_INFO$cM <- NULL # Remove column with genetic distance

## Create matrix for rrBLUP with marker location and marker dosage
SNP_INFO <- bind_cols(SNP_INFO,as.data.frame(t(X)))

## Read in phenotype file with subpopulation assignment
## and extract only phenotypes with genotypic information
phenotypes_pop <- fread("rice_phenotypes.txt")
phenotypes_pop <- phenotypes_pop[phenotypes_pop$id %in% snp_matrix$IID,]

head(phenotypes_pop)

                   id        population     PH
1:               A201  tropicaljaponica  93.40
2:               A301  tropicaljaponica  81.20
3: ADELAIDECHIAPPELLI temperatejaponica 100.07
4:              AIACE  tropicaljaponica  79.40
5:       AKITAKOMACHI temperatejaponica  89.73
6:              ALICE temperatejaponica  81.73


## Check if IDs in phenotype file and SNP_INFO file are in same order
phenotypes_pop$id == rownames(X) # ... they are.

vec <- colnames(GRM) %in% phenotypes_pop$id
GRM <- GRM[vec,vec]

SNP_INFO <- as.data.frame(SNP_INFO)
SNP_INFO <- SNP_INFO[,c(TRUE,TRUE,TRUE,vec)]

SNP_INFO[1:4, 1:8]

  Chr       SNP    Pos A201 A301 ADELAIDECHIAPPELLI AIACE AKITAKOMACHI
1   1  S1_59126  59126   -1   -1                 -1    -1           -1
2   1 S1_170244 170244   -1   -1                  1    -1           -1
3   1 S1_330484 330484   -1   -1                  1    -1           -1
4   1 S1_594195 594195    1    1                 -1    -1           -1
```

GWAS Model 1 (genomic relationship matrix) & Model 2 (genomic relationship matrix + known subpopulation structure)

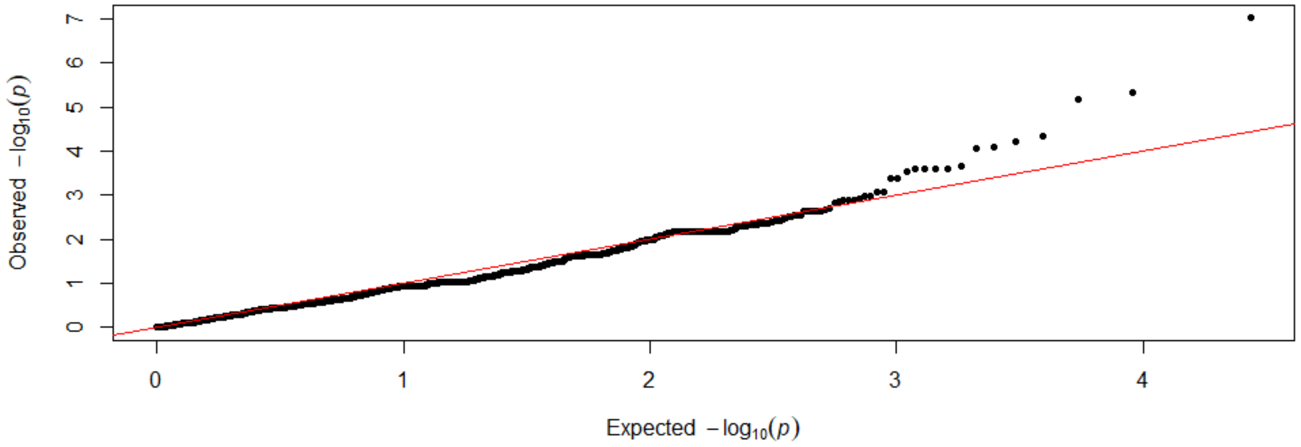
We will now compare **model 1 (genomic relationship matrix)** and **model 2 (genomic relationship matrix + known subpopulation structure)**. For illustration purposes, we will only calculate a Bonferroni correction threshold to account for multiple testing and to identify significant marker-trait associations. A p-value correction based on False Discovery Rate (FDR) is another very popular and less stringent method to account for the multiple testing problem when running a GWAS.

Note: as mentioned at the beginning, rrBLUP is not the most flexible GWAS package. If you do not provide a genomic relationship matrix (K = GRM), rrBLUP will by default calculate the GRM from the markers in the genotype data frame (geno = SNP_INFO). Hence, rrBLUP does not allow you to run a GWAS with no correction for population structure at all, nor to correct for population structure by using subpopulation information only (both not the best idea, as we will see during the course, anyway). We will also provide other scripts and examples which use alternative packages that allow for more modeling flexibility. For illustration, however, rrBLUP does a good job because of its simplicity.

```
## Copy phenotype set and remove population assignment for model 1
phenotypes_noPop <- phenotypes_pop[, -2]

head(phenotypes_pop)
head(phenotypes_noPop)

## Threshold for Bonferroni correction
bf <- -log10(0.05 / nrow(SNP_INFO))



## GRM (K model)
model_1 <- GWAS(
  pheno = phenotypes_noPop,
  geno = SNP_INFO,
  K = GRM,
  fixed = NULL,
  plot = FALSE)

gwasResults_1 <- model_1[,c("SNP","Chr","Pos","PH")]
names(gwasResults_1) <- c("SNP","CHR","BP","P")

## rrBLUP calculates -log10(p) values. These are converted into p-values
gwasResults_1$P <- 10^((-gwasResults_1$P))

## Manhattan plot of -log10(p)-values
manhattan(gwasResults_1, suggestiveline = FALSE, col = c("red","blue"),
          genomewideline = bf, logp = TRUE)
```

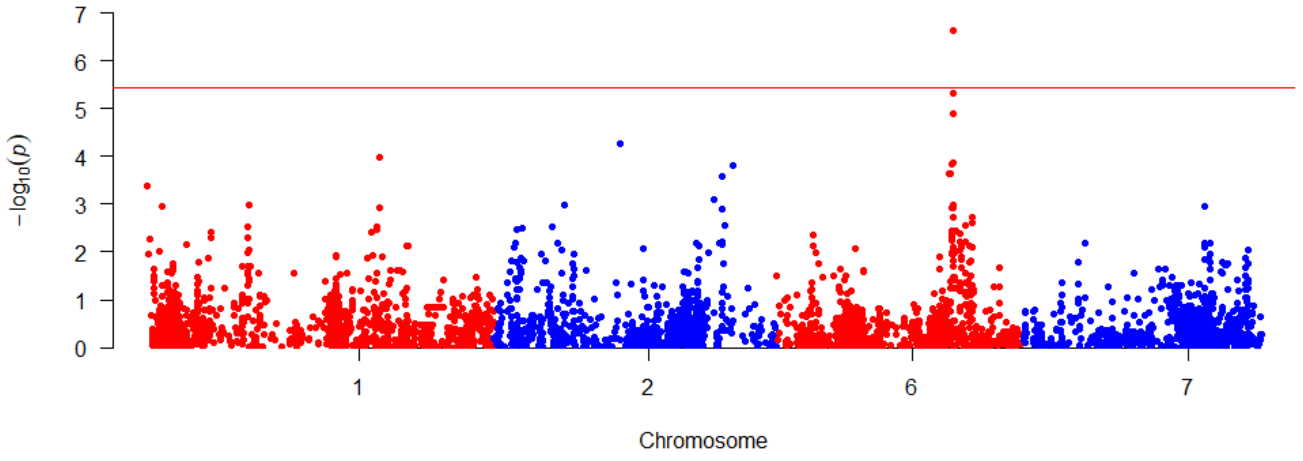


```
## qq-plot
qqman::qq(gwasResults_1$P)
```

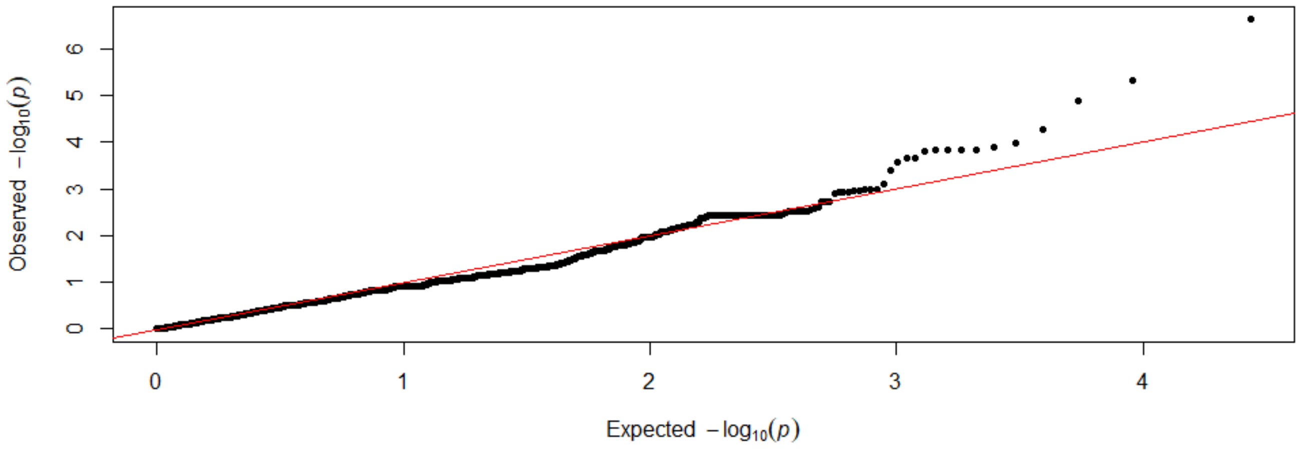


```
## GRM + sup-population information

## We have individuals from two subpopulations left in the data set
unique(phenotypes_pop$population)
[1] "tropicaljaponica"  "temperatejaponica"

model_2 <- GWAS(
  pheno = phenotypes_pop,
  geno = SNP_INFO,
  fixed = "population",  # correction for subpopulation structure.
  K = GRM,
  plot = FALSE)

gwasResults_2 <- model_2[,c("SNP","Chr","Pos","PH")]
names(gwasResults_2) <- c("SNP","CHR","BP","P")
gwasResults_2$P <- 10^((-gwasResults_2$P))
```

```
## Manhattan plot of -log10(p)-values
manhattan(gwasResults_2, suggestiveline = FALSE, col = c("red","blue"),
          genomewideline = bf, logp = TRUE)
```



```
## qq-plot
qqman::qq(gwasResults_2$P)
```



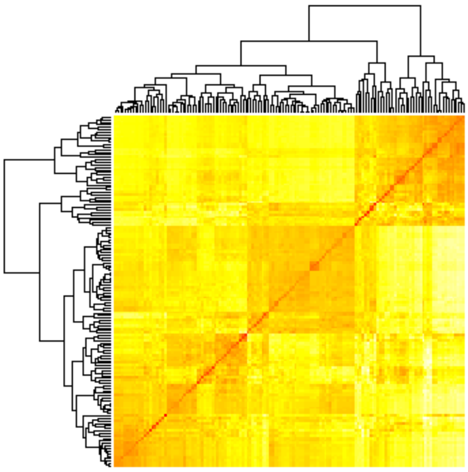
As we can see in the two Manhattan plots and the qq-plots, correcting for subpopulation structure (via known subgroup assignment) has resulted in a slight deflation of the p-values additional to including the genomic relationship matrix only. The differences are, however, marginal and we find the same peak on chromosome 6 at approximately the same -log10(p)-value in both analyses. It's not too surprising that we observe hardly any differences. First, we know that the genomic relationship matrix on its own already does a good job correcting for relatedness and population structure that exists between the individuals. Second, after filtering for individuals with both genotypic and phenotypic information, we end up with accessions from only two subpopulations: *temperate japonica* and *tropical japonica*. The population means of the two subpopulations are relatively similar (as shown in the box plots above). Hence, correction for differences in the mean using subpopulation as a fixed factor does not have a strong effect in the rice data set.

However, the effect of subpopulation structure might be more pronounced in other data sets. If the differences in the mean performance of multiple subpopulations are large, including subpopulation as a fixed factor can be advantageous. The combination of subpopulation information to capture high-level population structure and the genomic relationship matrix to represent detailed relationship information between individuals might result in a better model fit. Again, comparing different GWAS models and matching them with your assumptions is always recommended.

GWAS Model 3 (genomic relationship matrix + principal components to capture unknown subpopulation structure)

If we have no information on population structure, but we assume that (or want to analyze if) our data set contains individuals from different genetic backgrounds, we can also use principal components to capture high-level population structure.

```
## First step to get an idea of subpopulation structure based on the relationship captured
in the genomic relationship matrix.
heatmap(GRM,labRow = FALSE, labCol = FALSE, col=rev(heat.colors(75)))
```

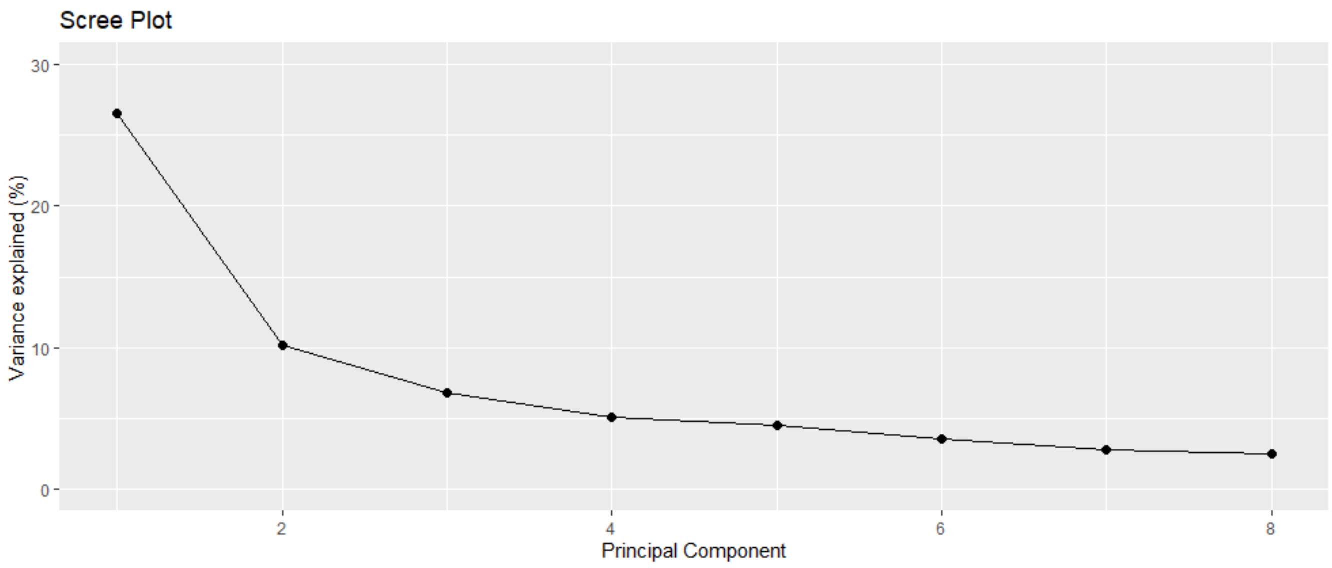


The heatmap roughly indicates four subgroups. Without further information, we cannot draw any conclusions from the heatmap on the genetic background of the four subgroups and the accessions. Nevertheless, we can use this information to correct for (hidden) population structure, as we will see now.

Note: although we know the accessions belong to only two subpopulations, it's not unusual to observe more subgroups when comparing or grouping individuals based on marker information. In plants, crossing individuals from two different subpopulations (genetic pools) is often used to introgress traits from one subpopulation into the other. A heatmap (or PCA) based on the genomic relationship matrix, therefore, might give us a clearer and more detailed idea about high-level population structure compared to known subgroup assignments only.

At first, we will perform a principal component analysis (PCA) to check how many principal components (PCs) we should use to capture high-level population structure.

```r
### Principal Component Analysis on GRM using single value decomposition
svd <- svd(GRM)

### Extract eigenvalues and eigenvectors
eigenvalues <- svd$d
eigenvectors <- svd$u

### Variances explained by first 8 principal components, respectively
var_explained <- round((eigenvalues/sum(eigenvalues)*100), 2)[1:8]
var_explained

### Total variance explained by the first 1 - 8 principal components
for(i in 1:8){
  varExp <- (var_explained)[1:i]
  print(paste0("Var explained by first ", i, " PCs: ", round(sum(varExp), 2), "%"))
}


### Scree plot
var_df <- data.frame(PC = c(1:8),
                     Var = var_explained)

var_df %>%
  ggplot(aes(x=PC,y=var_explained, group=1))+
  geom_point(size=2)+
  geom_line()+
  xlab("Principal Component") +
  ylab("Variance explained (%)") +
  ggtitle("Scree Plot") +
  ylim(0, round_any(var_explained[1], 10, f = ceiling))
```



We learn from the PCA and the scree plot that the first PC explains 26.5% of the high-level variation in the data set and that the first 3 PCs together explain 43.5% of the high-level variation.

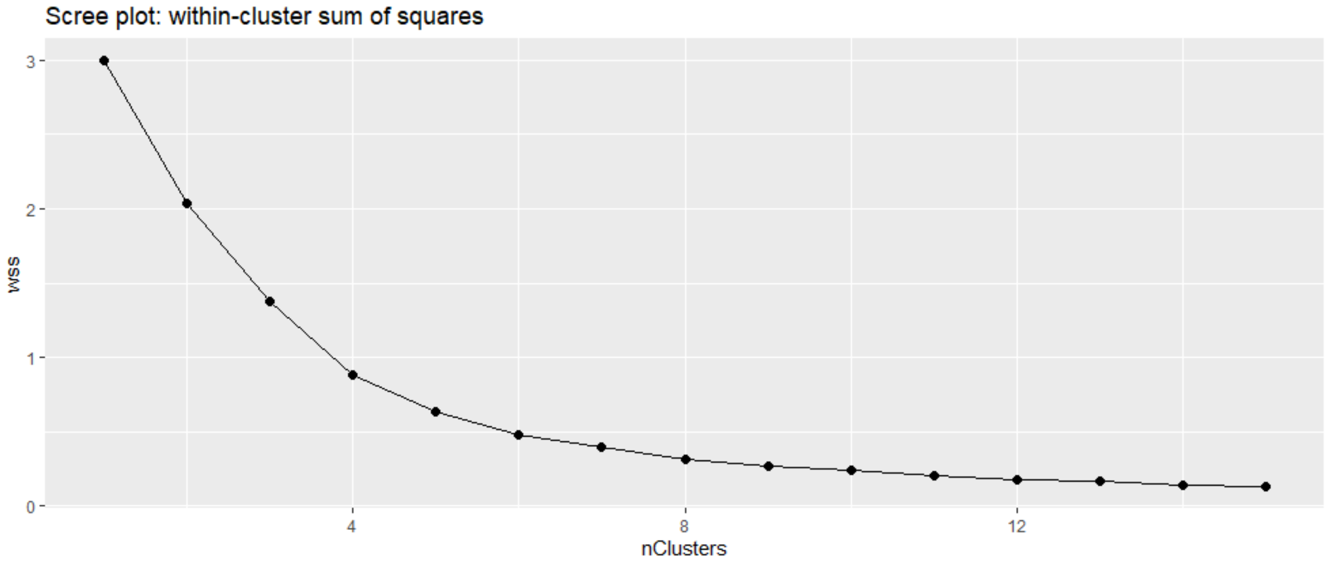
When combining a GRM and principle components in the GWAS model, we intend to only capture high-level population structure in the PCs (fixed effects), while detailed relationship information is captured in the GRM (random effect). If we include too many PCs, the information captured in the PCs approximates the information captured in the GRM, as the variance explained by the PCs increases. This will add little benefit to the analysis.

One rule of thumb is to include as many PCs as there are subgroups in the data set. We might not know the real number of subgroups, but we can use the number of the clusters observed in the heatmap as a first approximation. Another approach is to choose the number of PCs where the line in the scree plot shows the strongest bend. Since after this strong bend the variation explained by the PCs becomes rather low and starts to flatten out, adding more PCs to explain high-level variation in the population doesn't provide much benefit. While the first approach suggests 4 subgroups, the second approach suggests 2 subgroups. We arbitrarily choose 3 subgroups as a compromise between the two results. However, this is to some extent subjective and a few more or less PCs also could be used to represent the high-level population structure.

If the first few PCs do not explain a substantial part of the total variation in your data set, there probably is no detectable subpopulation structure. In this case, including PCs in the GWAS model additional to the GRM will be of little or no benefit.

We will now use the first 3 PCs in an iterative K-means clustering to investigate how many subgroups those PCs may indicate in our data set. In K-means clustering, we first set the number of clusters and then the K-means algorithm assigns individuals to those clusters they most likely belong to. Running a series of K-means clusterings with different numbers of K clusters allows identifying what a reasonable number of subgroups might be. The goodness of clustering can be assessed by looking at the within-cluster sums of squares (WSS). We will compare the WSS of 1 to 15 clusters in another scree plot.

```r
### Extract first 3 principal components which explain ~44% of the total variance
pc <- eigenvectors[,1:3]
colnames(pc) <- paste0("PC", 1:ncol(pc))

### K-means clustering based on first 4 PC's and comparison of within-cluster sum of squares
### to identify sub-populations (eyeball method)

clusters <- 15 # max. number of clusters to compare
withinSS <- data.frame(nClusters = seq(1:clusters),
                       wss = NA)
for(i in 1:clusters){
  withinSS[i,2] <- sum(kmeans(pc, centers=i, nstart=25)$withinss)
}

### The strongest bend in the graph indicates a suitable number of clusters
### (eyeball method)
withinSS %>%
  ggplot(aes(x=nClusters,y=wss, group=1))+
  geom_point(size=2)+
  geom_line()+
  labs(title="Scree plot: within-cluster sum of squares")
```

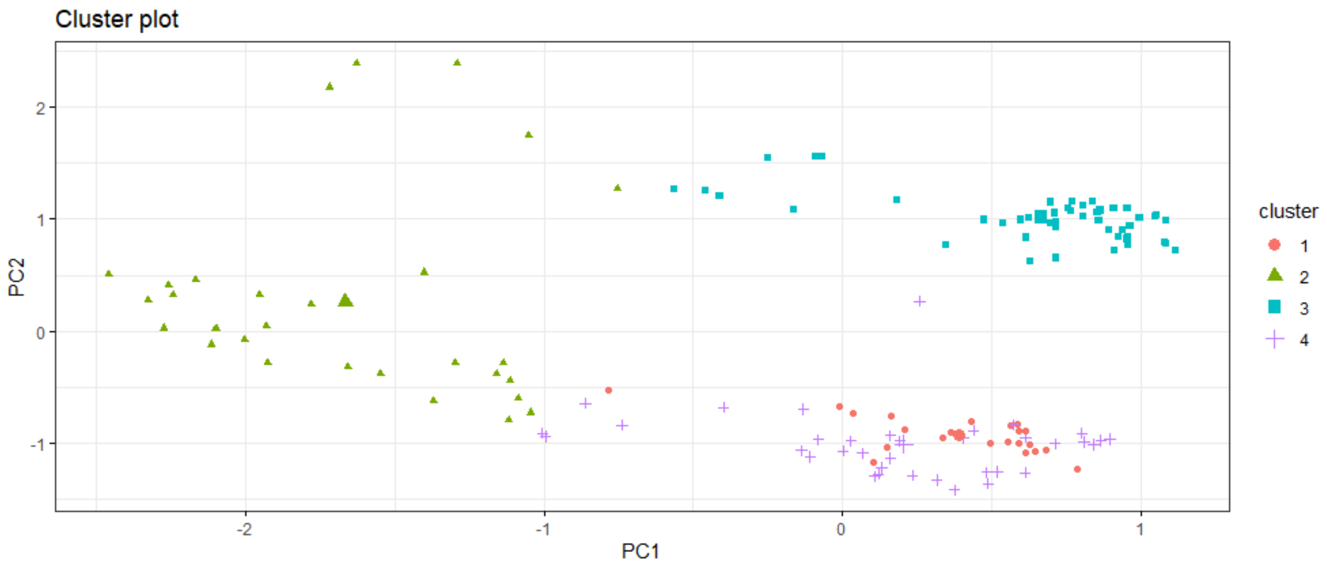


Here, it's also a common rule of thumb that a reasonable number of populations based on the chosen PCs is where the line in the scree plot bends. The bend indicates a good trade-off between too few subpopulations (high WSS in the clusters) and too many subpopulations (low WSS in the clusters) is found.

We see the bend indicates 4 clusters, which also was the number of clusters observable in the heatmap. This may indicate that the first 3 PCs were a suitable choice for capturing the high-level population structure in the rice data set. However, repeating the clustering analysis with more or fewer PCs and checking if and how the scree plot changes can help to get a better understanding of the data set.

We will now plot the four clusters (subpopulations) in a two-dimensional plot based on the first two PCs.

```
# K-means clusters showing the group assignment of each individual
kCluster <- kmeans(pc, 4, nstart = 25)

### Clusters based on first 2 principal components
pc_df <- as.data.frame(pc)[1:3]
colnames(pc_df) <- paste0("PC", 1:3)

fviz_cluster(kCluster, data = pc_df[, 1:2],
             geom = "point",
             ellipse = FALSE,
             ggtheme = theme_bw()
)
```

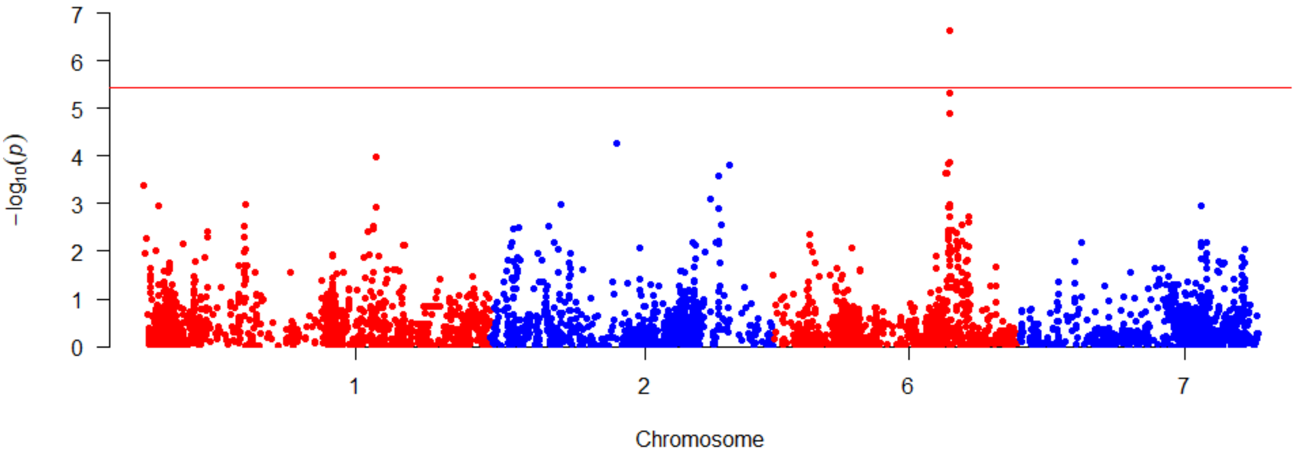


Again, it is very important to remember that with no additional information, we cannot draw strong conclusions from the number of clusters about the true genetic backgrounds. However, this type of analysis can be helpful to get a better understanding of our data set and test or confirm our assumptions about the origin of the individuals in the data set. To correct for population structure in a GWAS using PCs, knowing the original genetic background is not necessary, though. We can include the PCs in the GWAS model to correct for population structure even if we do not have a detailed understanding of the genetic background.

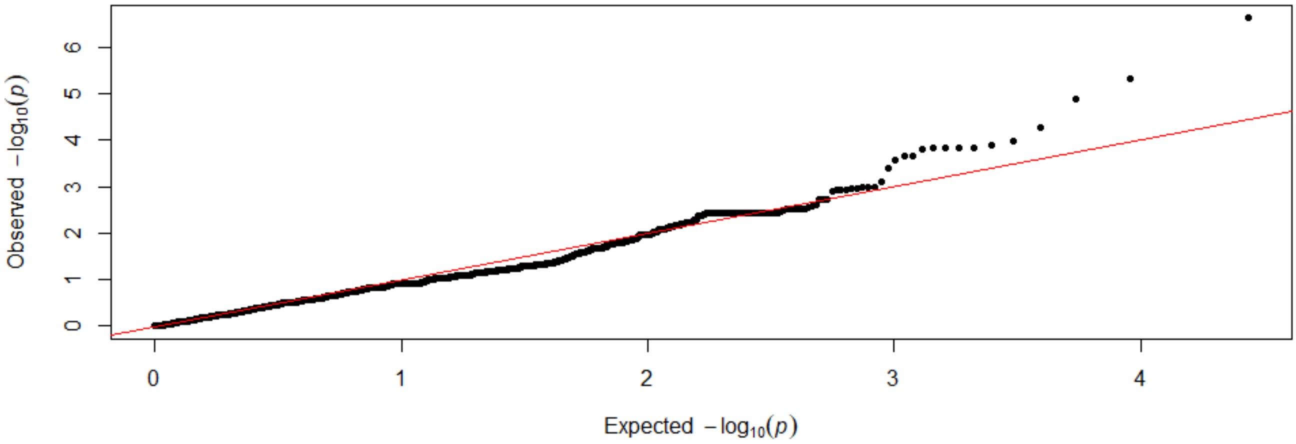
We will now run our third GWAS model and include the first 3 PCs as fixed effects to correct for subpopulation structure. rrBLUP also allows you to set the number of PCs using the "n.PC" argument of the "GWAS" function. The defined number of PCs will be automatically calculated from the GRM and does not need to be provided as a fixed factor. Most other packages, however, do not give this option. Hence, we show here how to do it.

```
phenotypes_pca <- cbind(phenotypes_noPop, round(pc,3))
head(phenotypes_pca)
                   id     PH    PC1    PC2    PC3
1:               A201  93.40 -0.112  0.007  0.001
2:               A301  81.20 -0.110  0.001  0.008
3: ADELAIDECHIAPPELLI 100.07  0.080  0.052 -0.162
4:              AIACE  79.40 -0.104  0.004  0.001
5:        AKITAKOMACHI  89.73  0.077 -0.084  0.158
6:              ALICE  81.73  0.103  0.066  0.036

# Array with strings containing the PCs to be included as fixed factors to correct for
subpopulation structure
pcomps <- colnames(pc)

model_3 <- GWAS(
  pheno = phenotypes_pca,
  geno = SNP_INFO,
  fixed = pcomps,
  K = GRM,
  plot = FALSE)

gwasResults_3 <- model_2[,c("SNP","Chr","Pos","PH")]
names(gwasResults_3) <- c("SNP","CHR","BP","P")
```

```
gwasResults_3$P <- 10^((-gwasResults_3$P))

manhattan(gwasResults_3, suggestiveline = FALSE, col = c("red","blue"),
          genomewideline = bf, logp = TRUE)
```



```
qqman::qq(gwasResults_3$P)
```

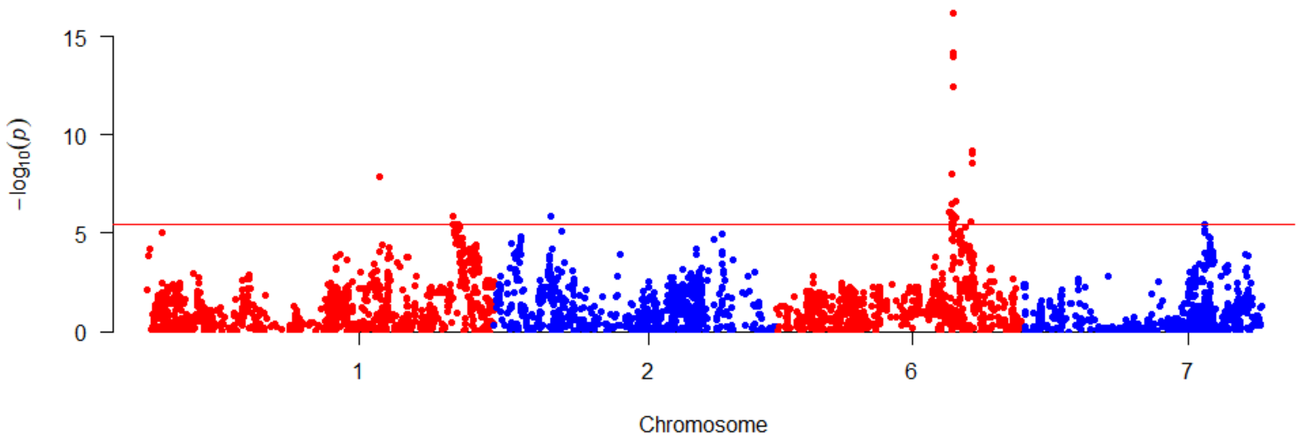


Again, we hardly see any differences compared to the Manhattan plots and qq-plots generated by Model 1 and Model 2. Correcting for subpopulation structure using the first 3 PCs and the GRM also has resulted in a slight deflation of the p-values compared to using the GRM only. The results are comparable to Model 2.

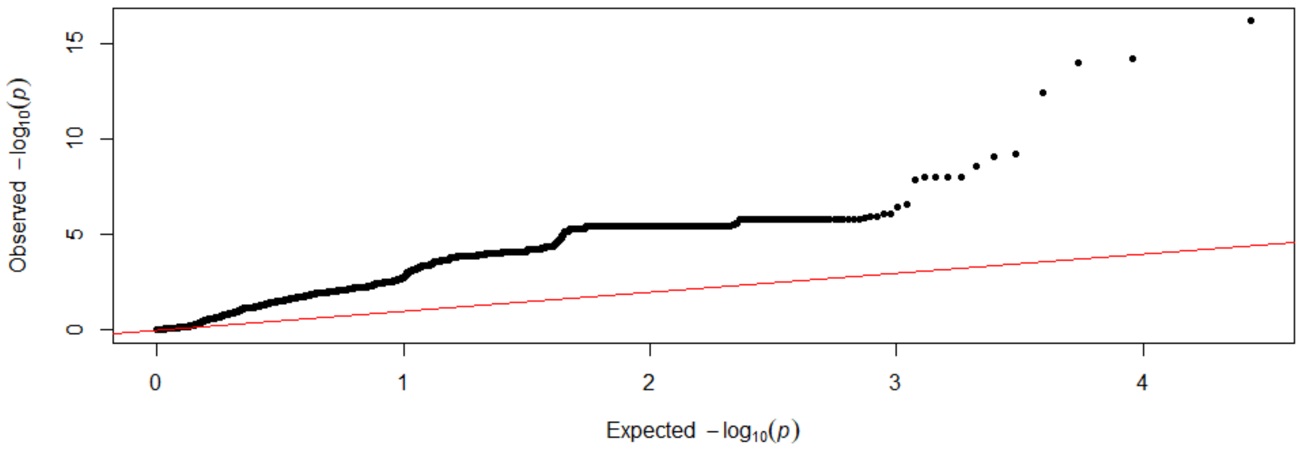
GWAS Model 4 (no correction for population structure)

If we do not specifically define a relationship matrix, rrBLUP will automatically calculate and use the genomic relationship matrix based on the marker information provided, and by default correct for population structure. However, to illustrate what happens if we do not correct for population structure, we can enforce this scenario by providing a diagonal matrix (no relationship) instead of using the genomic relationship matrix. While this is not recommended, it is useful to demonstrate how important it is to correct for population structure when running a GWAS.

```
diag_mat <- diag(nrow(GRM))
colnames(diag_mat) <- rownames(diag_mat) <- colnames(GRM)

model_4 <- GWAS(
  pheno = phenotypes_pop,
  geno = SNP_INFO,
  fixed = "population",  # correction for subpopulation structure.
  K = diag_mat,
  plot = FALSE)

gwasResults_4 <- model_4[,c("SNP","Chr","Pos","PH")]
names(gwasResults_4) <- c("SNP","CHR","BP","P")
gwasResults_4$P <- 10^((-gwasResults_4$P))

## Manhattan plot of -log10(p)-values
manhattan(gwasResults_4, suggestiveline = FALSE, col = c("red","blue"),
          genomewideline = bf, logp = TRUE)
```



```
qqman::qq(gwasResults_3$P)
```



The manhattan plot and the qq-plot both clearly suggest a high rate of false positive associations because of population structure that has not been corrected for.