INTRODUCTION TO GENOME-WIDE ASSOCIATION STUDIES (GWAS)

Filippo Biscarini, Oscar Gonzalez-Recio, Christian Werner

21-25 June 2021

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

Introduction

This course manual will guide you through the steps involved in performing a GWAS on the rice data set. Although every data set has unique features and requires customised data handling, the overall procedure of performinng a GWAS is fairly standardized and can be divided into:

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 platfrom, 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 perfrom data preparation, data pre-processing and a GWAS will require different software and packages. There is a variety of efficient open-source solutions available for each of those steps. Choosing a certain combination of softare and packages is often driven by personal preferences rather than substantial differences in performance.

We will build our pipeline using three well-established and relatively user-friendly programs which will allow efficient data handling even with larger data sets. 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 are able to 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). It is very likely that you can find a package there which will help you to 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 many different purposes including 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 successfully run your first GWAS

and produce a Manhattan plot. As mentioned above, however, be curious and and try other GWAS software to find what works best for your. During the course, we will introduce a few other packages which also do a great job and are suitable for beginners.

PLINK

PLINK is a very 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.

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 5.2 requires Java version 8 and can be run on Linux, MacOS and Windows using the command line.

During the course, we will also cover a two other options for imputation: mean imputation and K-Nearest Neighbor Imputation (KNNI).

MobaXterm

MobaxTerm is only necessary for **Windows users**. While Linux and MacOS provide a UNIX-based terminal by default, the Windows terminal is different and makes working with it complicated and unpleasent. MobaXterm provides all the important Unix commands and remote network tools to the Windows desktop. If you are already using another emulator (e.g., Cygwin, ...) you won't need MobaXterm and can use your preferred solution instead.

Data preparation

Data preperation comprises the initial steps required to ensure your data files are in good shape and can be read in by the software and packages you intend to use. While this may be neccessary several times during your pipeline, during initial data preperation we need to eliminate potential inconsistencies and errors from our data set. Some common issues inlcude non-matching IDs in genotype and phenotype files, missing or redundant whitepsace and tab characters, incorrect genotype formats and missing or unnecessary headers.

All the code shown in this manual is either run in the **command line** or in **RStudio**. Both these interfaces require to set the path to the files you want to read in or the programs you need to call (PLINK, Bealge). While we suggest to definea clear, well-organized folder structure to run your pipeline and store your results, we will skip that in this manual and not provide full pathes and locations of the files.

Rice data set

The rice data set comprises GBS marker data and phenotypic records for several continous traits on 153 accessions from 5 sub-populations. 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 `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 143584
-rw-r--r--   1 USER  UsersGrp  29381995 Jun  1 05:56 GBSgenotypes.tar.gz
-rw-r--r--   1 USER  UsersGrp   4291480 Aug  2  2013 GBSnew.map
-rw-r--r--   1 USER  UsersGrp 260286484 Aug  2  2013 GBSnew.ped
-rw-r--r--   1 USER  UsersGrp      7364 Jun  8 00:55 plantgrainPhenotypes.txt
```

Data examination

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

1. Command line text editors like **vim**, **emacs** or **nano** provide the most powerful and flexible way to open and manipulate entire text files. They are much more efficient than graphical text editors like Notepad.

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

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

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

2. Utility commands like `head` and `wc` are more suitable to quickly extract key information.

```
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 just saw, the file ***plantgrainPhenotypes.txt*** contains accession IDs and values for 7 traits. We will perform a GWAS for plant height (PH) only and remove all other traits from the phenotype file. Furthermore, in order to correct for population structure during the GWAS, we will add a column with sub-population information. Based on the file ***rice_group.reference***, each rice accession will be assigned to one of five sub-populations.

We use **RStudio**

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

phenotypes = fread("plantgrainPhenotypes.txt")
ref_group = fread("rice_group.reference")

## Add sub-population 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" used later to calculate stratified allele frequencies.
ref_group$oldf am <- rep("NF1", nrow(ref_group))
ref_group$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 the new phenotype file which we will be using from now on. We also generated the file *ids* which contains all accession names and will be used with PLINK to indicate which genotypes to keep. The file *update.ids* will be used later for calculation of stratified allele frequencies.

After data preparation, the new number of accessions in the phenotype file is 153.

```
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.

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

Note: nonmatching IDs in different files are a very common pitfall that can occur before or during data processing.

Especially when the data sets are large, some of the 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 potential causal genes in the rice data set, we preselect 4 of the 12 rice chromosomes. Furthermore, we only keep those accessions with phenotypic information in the genotype files, as indicated by the *ids* file.

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

PLINK v1.90b6.22 64-bit (16 Apr 2021)          www.cog-genomics.org/plink/1.9/
(C) 2005-2021 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 also writes that information into a *log* file and creates a *rice.nosex* file which both 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 inspecting, cleaning, transforming and modelling our data prior to 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 pipeline. 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 be universally valid. They cover, however, the process of data analysis as a whole, which is the more important point

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 conding 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 hypotheses, 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 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, to identify relationships between variables, identify potential cofactors and spot inconsistencies. We will run an IDA and EDA for both our **phenotypic data** and the **genotypic data**.

Phenotypic data

Phenotypic data analysis is a fundamental step prior to running a GWAS. There are numerous reasons why our phenotypic data or the modelling 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 appropriate 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, using the rice data set, we give a few examples on how data visualization can be efficiently done in **RStudio** (taken from **_phenotypes.R_**).

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

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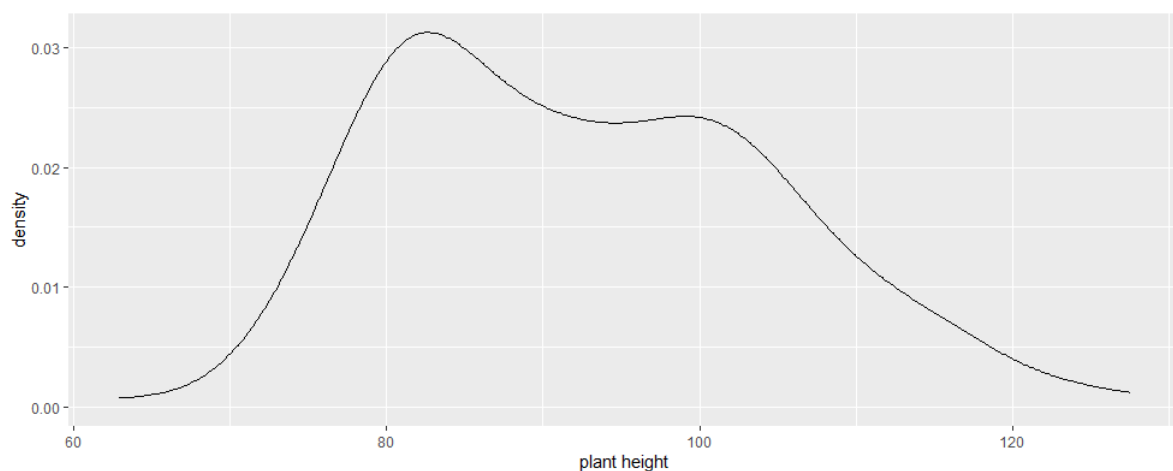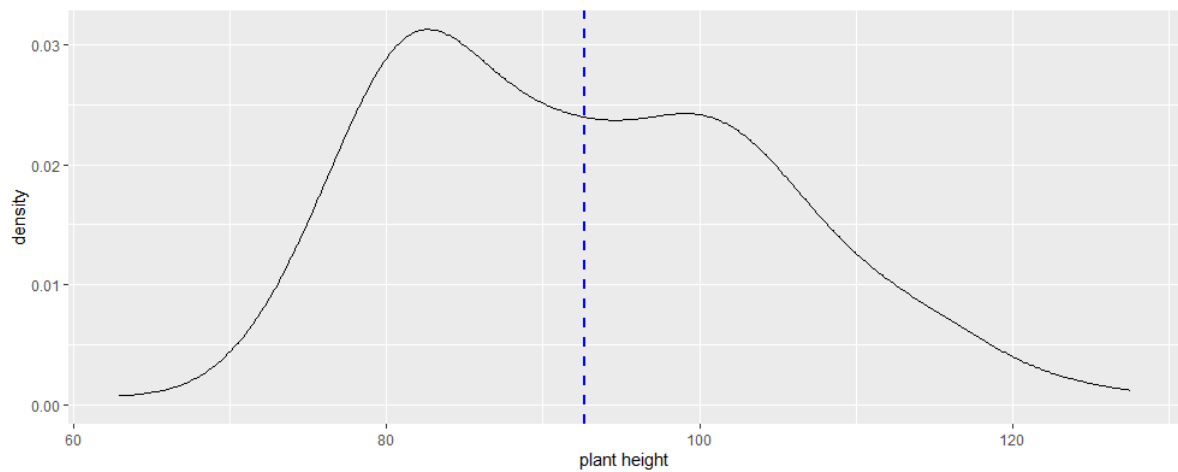
```
## Density plot with average line
p <- ggplot(pheno_continuous, aes(x=PH)) + geom_density()
p <- p + geom_vline(aes(xintercept=mean(PH)), color="blue", linetype="dashed",
size=1)
p <- p + xlab("plant height")
p
```



```
## 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
```
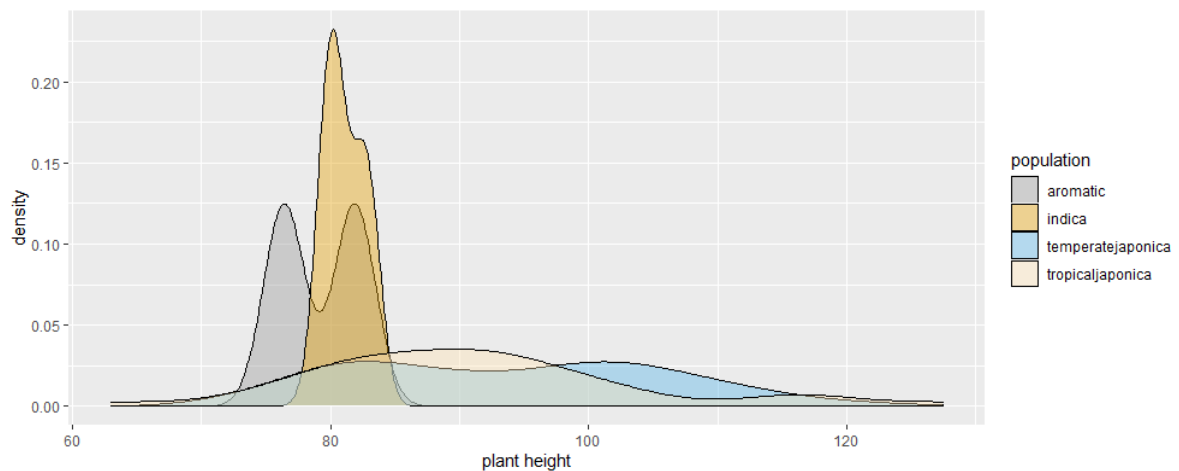


```
## Density plot with color shades per population + histograms
p <- ggplot(pheno_continuous, aes(x=PH, color=population, fill=population))
p <- p + geom_histogram(aes(y=..density..), alpha=0.5, position="identity")
p <- p + geom_density(alpha=.2)
p
```
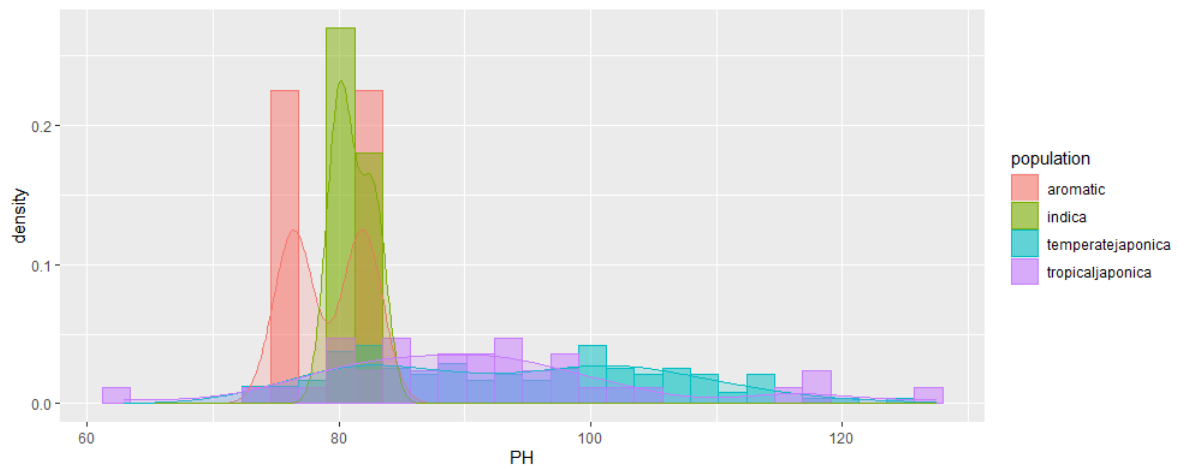
```
## Boxplots
p <- ggplot(pheno_continuous, aes(x=population,y=PH)) +
geom_boxplot(aes(fill=population))
p <- p + xlab("population") + ylab("plant height")
p
```
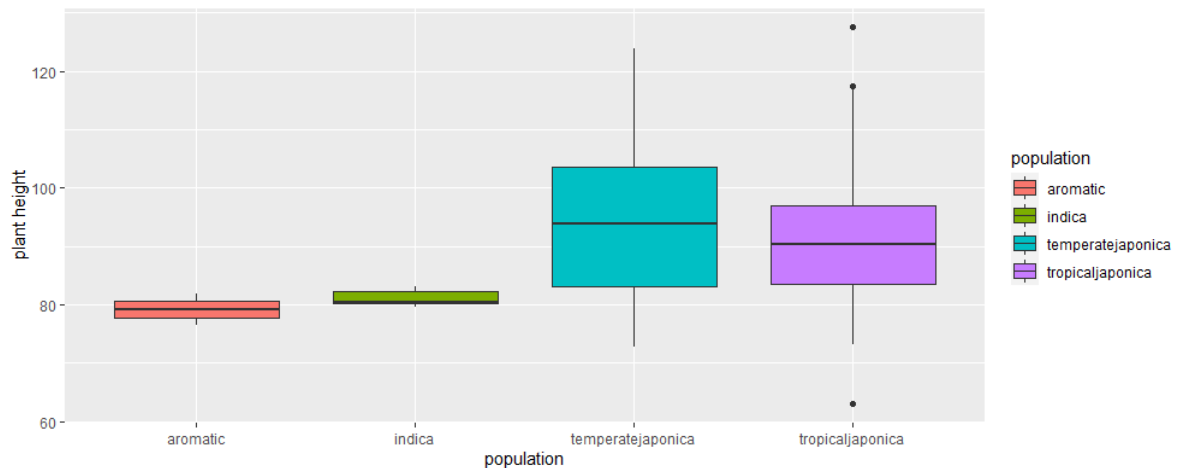

Genotypic data analysis

To explore the genotypes, **PLINK** provides many different options that are worth having a look at. While the analysis and output file generation will be done in PLINK ...

```
# 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
```
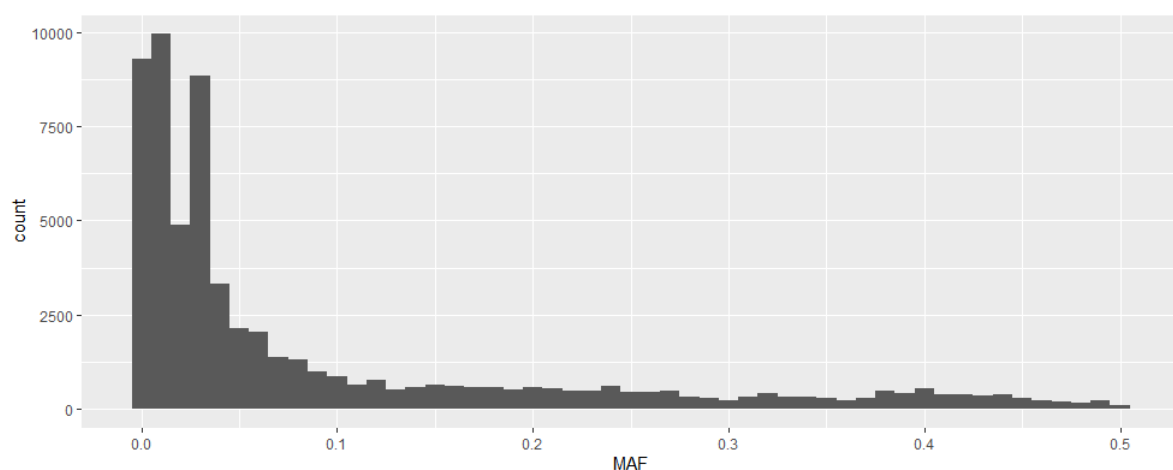


```
p <- ggplot(rice.frq, aes(MAF, colour = as.factor(CHR)))
p <- p + geom_freqpoly(binwidth = 0.01)
p <- p + guides(colour=guide_legend(title="CHR"))
p
```
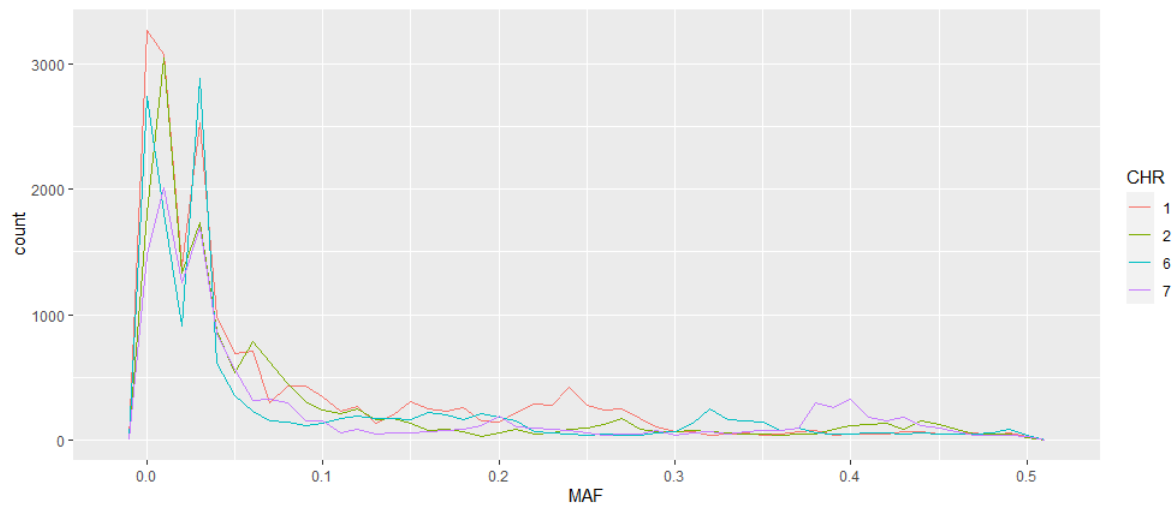
```
rice.frq <- fread("rice_pop.frq.strat")
rice.frq %>%
  group_by(CLST) %>%
  summarise(avg_frq=mean(MAF), std_dev=sd(MAF), max_frq=max(MAF),
min_frq=min(MAF))

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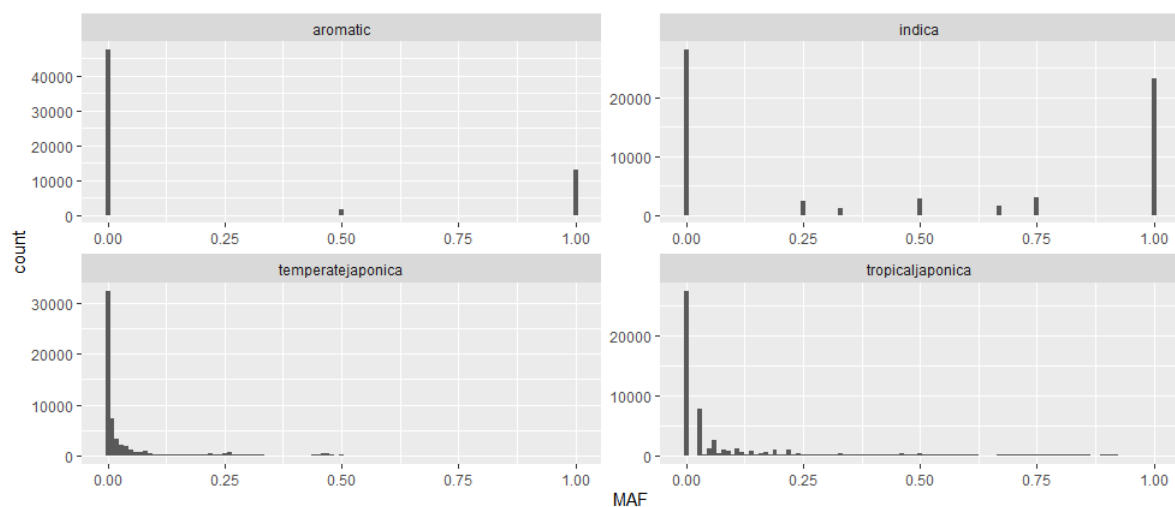

Genotype filtering

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. These value ranges may be different for SNP arrays, GBS and whole-genome sequences. Searching from thresholds in the GWAS literature with comparabel genotyping sets is always a good starting point.

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 2%

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

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 potential 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., rrBLUP)
2. K-Nearest Neighbor Imputation (KNNI)
3. Imputation with Beagle based on Hidden Markov Model (HMM)

Mean imputation

Mean imputation replaces missing markers calls by the mean value of the marker across the population. It is the fastest and simplest way to impute missing markers, but it is also a quite inaccurate. It can, 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 will be used 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 but with lots of missing data.

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). KNNI uses distance matrix between samples based on available 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 localised haplotype clustering imputation algorithm. It makes use of a Hidden Markov Model 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.

Besides Beagle's localised haplotype clustering imputation algorithm, there are various other algorithms which 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 which 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 operation system, this may work or not. Afterwards, we use PLINK to recode the imputed *.vcf* file.

```
## 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.29May21.d6d.jar gt=rice_filtered.vcf out=rice_imputed

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

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 and human genetics. We will see that applying these three models on the rice data set does produce relatively similar results. 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 model. The three GWAS models correct for population structure by including

1. A genomic relationship matrix (K model)
2. A genomic relationship matrix + known sub-population structure
3. A genomic relationship matrix + principle components to capture unknown sub-population structure (P + K model)

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 rrBLUP.

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

## Read in genotype file which we imputed before using Beagle
```

```r
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 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 sub-population structure)

We will now compare **model 1 (genomic relationship matrix)** and **model 2 (genomic relationship matrix + known sub-population structure)**. For illustration purposes, we will only calculate a threshold for Bonferroni correction to identify significant marker-trait associations.

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 contained in the genotype data frame (geno = SNP_INFO). Hence, rrBLUP does not allow you to run a GWAS without any correction for population structure at all, nor to correct for population structure by using sub-population information only (both in general 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 model flexibility. For illustration purposes, however, rrBLUP does a good job due to 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)
```
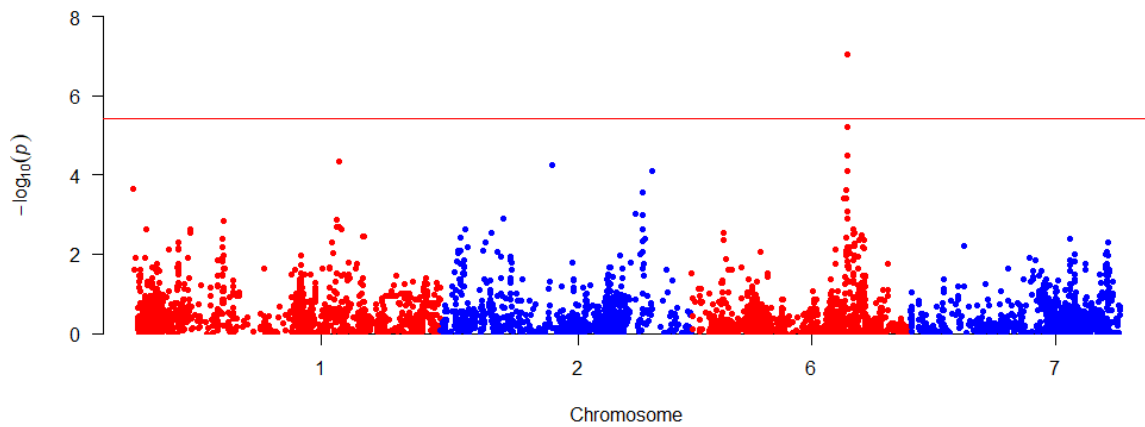


```
## qq-plot
qqman::qq(gwasResults_1$P)
```
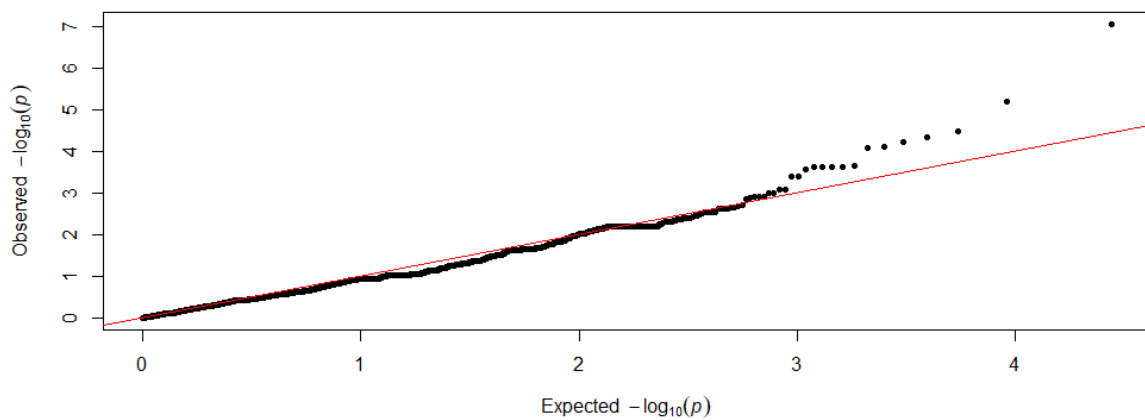


```
## GRM + sup-population information

## We have individuals from two sub-populations left in the pre-processed data
set
unique(phenotypes_pop$population)
[1] "tropicaljaponica"  "temperatejaponica"

model_2 <- GWAS(
  pheno = phenotypes_pop,
  geno = SNP_INFO,
  fixed = "population",  # correction for sub-population 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)
```
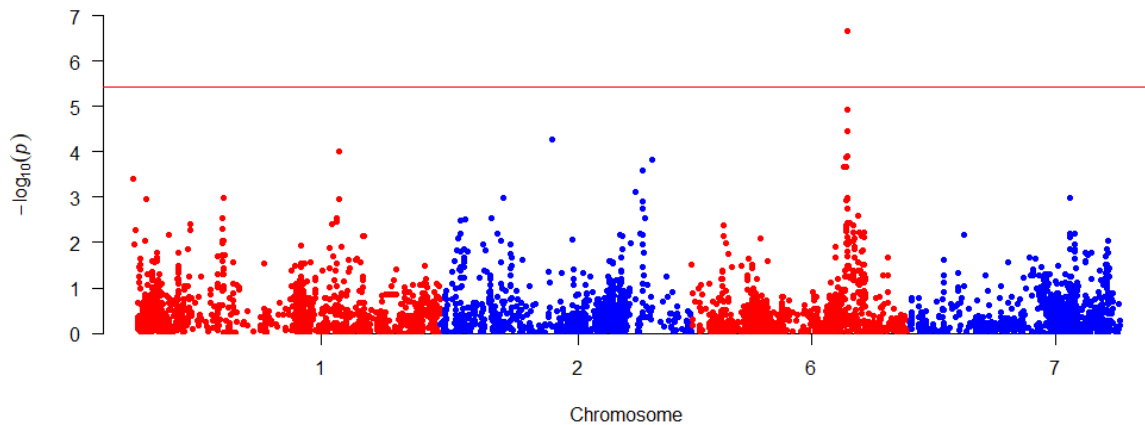


```
## qq-plot
qqman::qq(gwasResults_2$P)
```
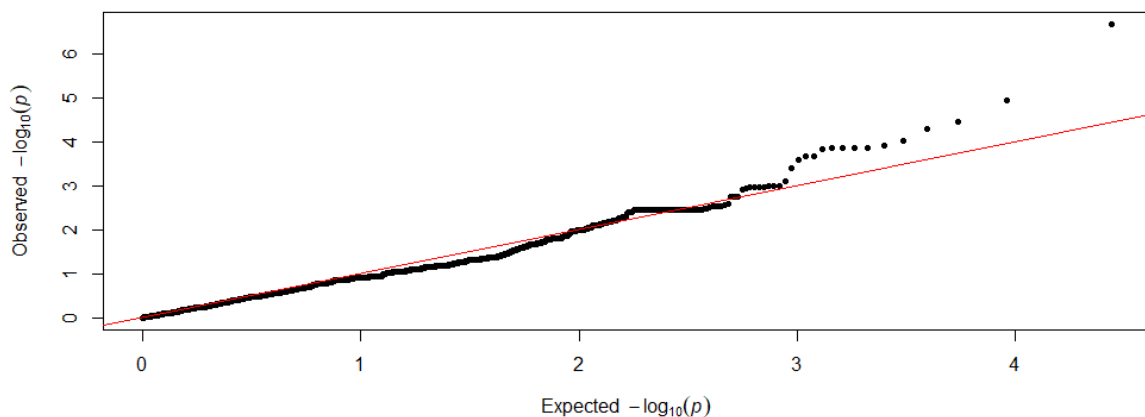

As we can see in the two Manhattan plots and the qq-plots, correcting for sub-population structure has resulted in a slight deflation of the p-values additional to including the genomic relationship matrix only. The differences are, however, marginal and we find the same peak on chromosome 6 at approximately the same -log10(p)-value in both analyses.

It's not too surprising that we observe hardly any differences. Firstly, 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. Secondly, after filtering for individuals with both genotypic and phenotypic information, we ended up with accessions from only two sub-populations: *temperate japonica* and *tropical japonica*. The population means of the two sub-

populations are relatively similar (as shown in the box plots above). Hence, correction for differences in the mean using sub-population as a fixed factor does not have a strong effect in this data set.

However, the effect of sub-population structure might be more pronounced in other data sets. If the differences in the mean performance of multiple sub-populations are large, including sub-population as a fixed factor can be advantageous. The combination of sub-population 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 can always be recommended.

GWAS Model 3 (genomic relationship matrix + principle components to capture unknown sub-population structure)

If we have no information on population structure but assume that our data set may contain individuals from different genetic backgrounds, we can also use principle components to capture high-level population structure.

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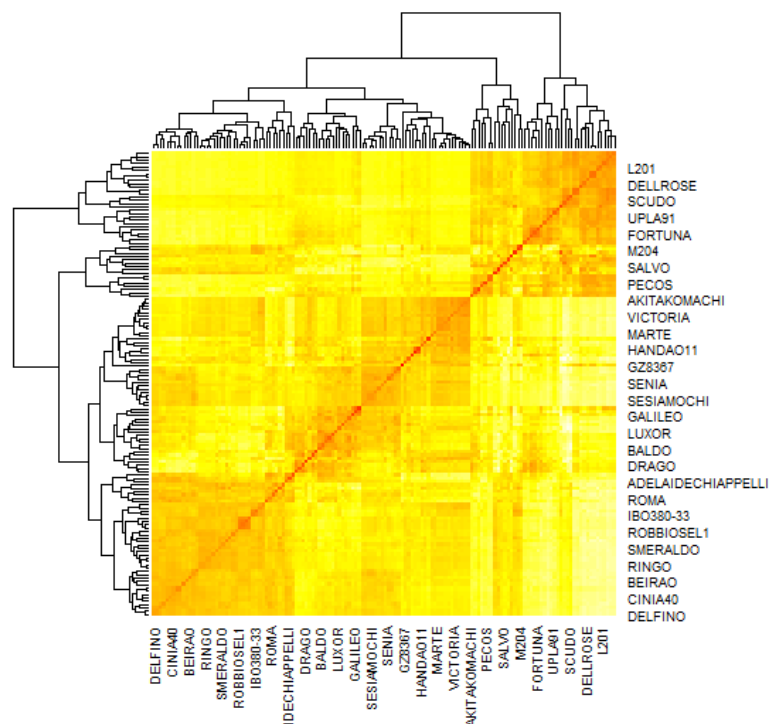

The heatmap indicates four sub-groups. Without further information, we cannot draw any conclusions from the heatmap about the original genetic background of the for sub-groups and the accessions. Nevertheless, we can use this information to correct for (hidden) population structure, as we will see now.

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

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

```
## Run pca on GRM to detect high-level population structure
pca <- prcomp(GRM, scale = TRUE, center = TRUE)

## Extract eigenvalues from PCA
eigenvalues <- pca$sdev^2

## Variances explained by first 8 principal components, respectively
round((eigenvalues/sum(eigenvalues)*100), 2)[1:8]
[1] 54.90 18.60  9.60  4.61  2.44  1.71  1.35  0.99

## Total variance explained by the first 1 - 8 principal components
for(i in 1:8){
  varExp <- (eigenvalues/sum(eigenvalues)*100)[1:i]
  print(paste0("Var explained by first ", i, " PCs: ", round(sum(varExp), 2),
"%"))
}

[1] "Var explained by first 1 PCs: 54.9%"
[1] "Var explained by first 2 PCs: 73.5%"
[1] "Var explained by first 3 PCs: 83.1%"
[1] "Var explained by first 4 PCs: 87.71%"
[1] "Var explained by first 5 PCs: 90.16%"
[1] "Var explained by first 6 PCs: 91.86%"
[1] "Var explained by first 7 PCs: 93.21%"
[1] "Var explained by first 8 PCs: 94.2%"


## Scree plot
fviz_eig(pca, ncp = 8)
```

Scree plot

We learn from the PCA and the scree plot that the first PC explains most of the high-level variation in the data set (54.9%), and that the first 3 PCs together explain 83.1% of the high-level variation.

When combining a GRM and principle components in the GWAS model, we intend to only capture high-level population strucutre in the PCs (fixed effects) and detailed relationship information 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. A common rule of thumb is to include those PCs which explain around 80% of the total variation. However, this is highly subjective and a few more or less PCs also could be used.

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

We will apply the 80%-rule and use the first 3 PCs in our GWAS. Additional to that, we will use the first 3 PCs in an iterative K-means clustering to investigate how many sub-populations there might be in our data set. To determine. In K-means clustering, we set the number of clusters and the algorithm assigns individuals to those clusters which they most likely belong to. Running a series of K-means clusterings with different number K of clusters allows to identify what a reasonable number of sub-populations might be. The goodness of a 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.

```
### Extract first X principal components which explain ~80% of the total
variance
pc <- pca$rotation[,1:3]

### K-means clustering 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")
```


Scree plot: within-cluster sum of squares

Another common rule of thumb is that a reasonable number of populations based on the chosen PCs is where the line in the scree plot bends. The bend indicates the point where a good trade-off between too few sub-populations (high WSS in the clusters) and too many sub-populations (very low WSS in the clusters) is found.

We see that the bend indicates a number of 4 clusters, which also was the number of clusters observed in the heatmap. This implies that the first 3 PCs were a reasonable choice to capture the high-level population structure. However, redoing the clustering analysis with more or less PCs and checking if and how the scree plot changes can help to get a better understanding of your data.

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

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

### Clusters based on first 2 principal components
fviz_cluster(kCluster, data = pc[, 1:2],
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse = FALSE,
             #ellipse.type = "convex",
             ggtheme = theme_bw()
)
```


Cluster plot
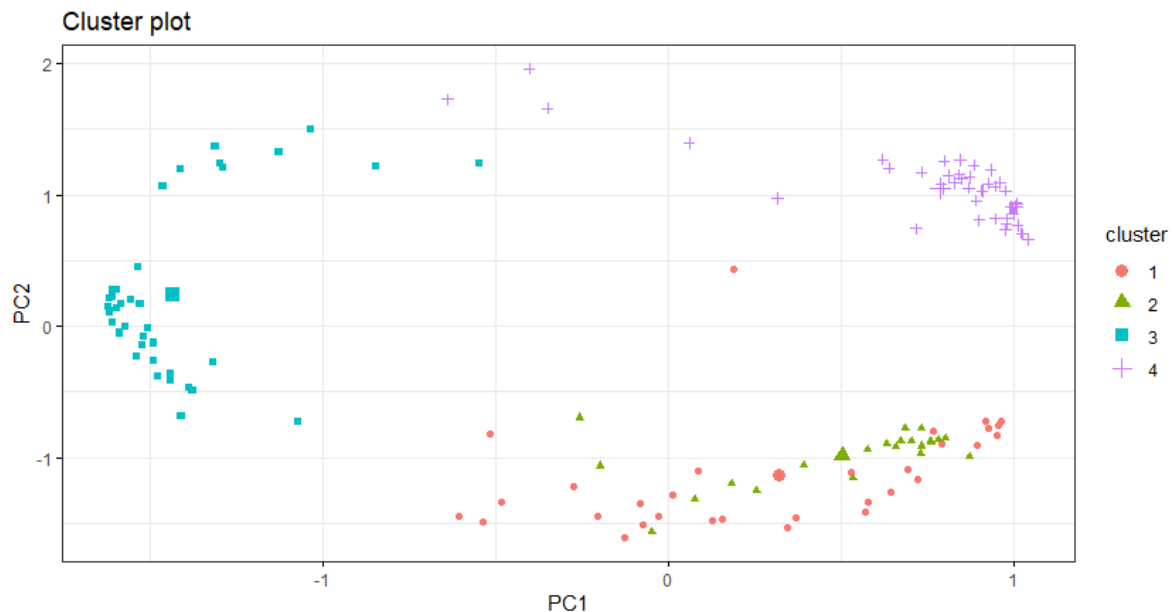
Again, without any additional information, we cannot draw any useful conclusions from the number of clusters about the true genetic backgrounds. However, this type of analysis can be very helpful to get a better understanding of our data set and test or confirm our assumptions on the origin of the material. In order to correct for population structure in a GWAS using PCs, knowing the original genetic background is not necessary though.

We will now run our third GWAS model and include the first 3 PCs as fixed effects to correct for sub-population 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 demonstrate it here in the "traditional" way.

```
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 sub-population 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)
```
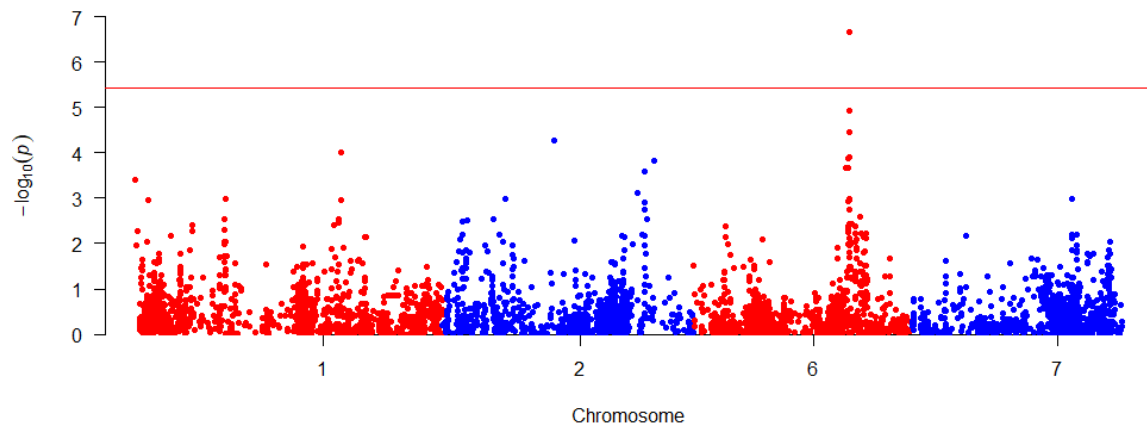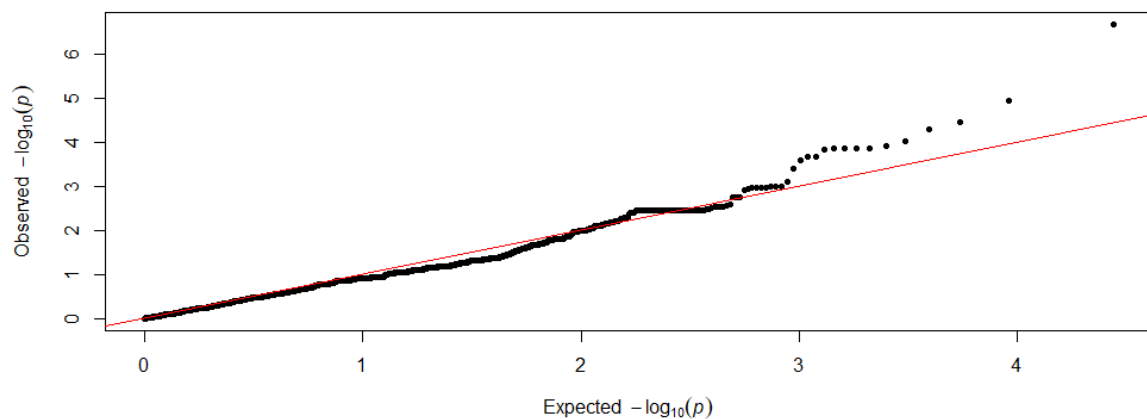


```
qqman::qq(gwasResults_3$P)
```

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