Introduction to GWAS

Bash tutorial (Review)

Write down what each of this functions do.

Basic functions

- cd:
- ls:
- wc:
- head:
- tail:

Additional functions

- awk:
- sed:
- grep:

Introduction to GWAS practice

Today we will run several GWAS of simulated phenotypes using the 1000 Genomes data generated on the Affy6 genotyping array. A lot of 1000 Genomes data exists and openly accessible to anyone—this includes a lot of whole exome and genome sequence data, as well as genotyping array data on the Illumina Omni2.5 and Affy6.0 platforms.

The 1000 Genomes phase 3 (final) dataset consists of 2,504 individuals from 26 different populations across Africa, Europe, East Asia, South Asia, and the Americas, which you can learn more about here: http://www.internationalgenome.org/category/population/. The final release of sequencing data includes all unrelated individuals.

The genotype data has more individuals (including relatives), that are almost entirely overlapping, with 3450 individuals from these same populations. The dataset we will use today has 226,227 SNPs. For future reference, you could download all of the raw data with many more SNPs than we will use today from this website (but don't worry about this right now since it's already in your docker or on your laptop): http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/hd_genotype_chip/

Details of populations and super populations ancestry groups

- *Super-population* ancestry groups: European (EUR), African (AFR), East Asian (EAS), South Asian (SAS), American (AMR).

- *Populations*: African Caribbean in Barbados (ACB), People with African Ancestry in Southwest USA (ASW), Esan in Nigeria (ESN), Gambian in Western Division, Mandinka (GWD), Luhya in Webuye, Kenya (LWK), Mende in Sierra Leone (MSL), Yoruba in Ibadan, Nigeria (YRI), Colombians in Medellin, Colombia (CLM), People with Mexican Ancestry in Los Angeles, CA, USA (MXL), Peruvians in Lima, Peru (PEL), Puerto Ricans in Puerto Rico (PUR), Chinese Dai in Xishuangbanna, China (CDX), Han Chinese in Beijing, China (CHB), Han Chinese South, China (CHS), Japanese in Tokyo, Japan (JPT), Kinh in Ho Chi Minh City, Vietnam (KHV), Utah residents (CEPH) with Northern and Western European ancestry (CEU), Finnish in Finland (FIN), British from England and Scotland, UK (GBR), Iberian Populations in Spain (IBS), Toscani in Italia (TSI), Bengali in Bangladesh (BEB), Gujarati Indians in Houston, TX, USA (GIH), Indian Telugu in the UK (ITU), Punjabi in Lahore, Pakistan (PJL), Sri Lankan Tamil in the UK (STU).

We will analyzed simulated genetic phenotypes, one continuous and one binary.

1. Go to Plink2 webpage

We are going to use the software plink. This is a very useful software to analysis genetic data. Go to: www.cog-genomics.org/plink/2.0/ Note that there are two versions of plink, we will focus on plink2 which is the more updated one. In this practice, we will tell you which commands to use and how to use them, but we will continue to reference the webpage so you are familiar with it and can use it as a future reference. You can see the formats of the output files: https://www.cog-genomics.org/plink/2.0/formats

```
### Lets first run plink help command
./bin/plink2 --help –freq

# Windows option

bin\plink2 --help –freq
```

Basic statistics

Plink is a tool that allows us to get specific statistics based on genotype and phenotype data.

Allele frequency

```
./bin/plink2 --bfile data/ALL.wgs.nhgri_coriell_affy_6.20140825.genotypes_has
_ped.225k_sites.chr1-22 --freq --out output/ALL.225k_sites_freq

### Windows version

bin\plink2 --bfile data\ALL.wgs.nhgri_coriell_affy_6.20140825.genotypes_has_p
ed.225k_sites.chr1-22 --freq --out output\ALL.225k_sites_freq
```

Describe the output

Hint: Go to the plink2 tutorial and find the description of the `*.afreq` file. Also, use the `head` command in the terminal to see the first lines of your output.

Missing rate for both SNPs and Individuals

```
./bin/plink2 --bfile data/ALL.wgs.nhgri_coriell_affy_6.20140825.genotypes_has
_ped.225k_sites.chr1-22 --missing --out  output/ALL.225k_sites_miss
```

Describe the output

Hardy-Weinberg equilibrium

```
./bin/plink2 --bfile data/ALL.wgs.nhgri_coriell_affy_6.20140825.genotypes_has
_ped.225k_sites.chr1-22 --hardy --out  output/ALL.225k_sites_miss
```

Describe the output

Visualize some of this statistics

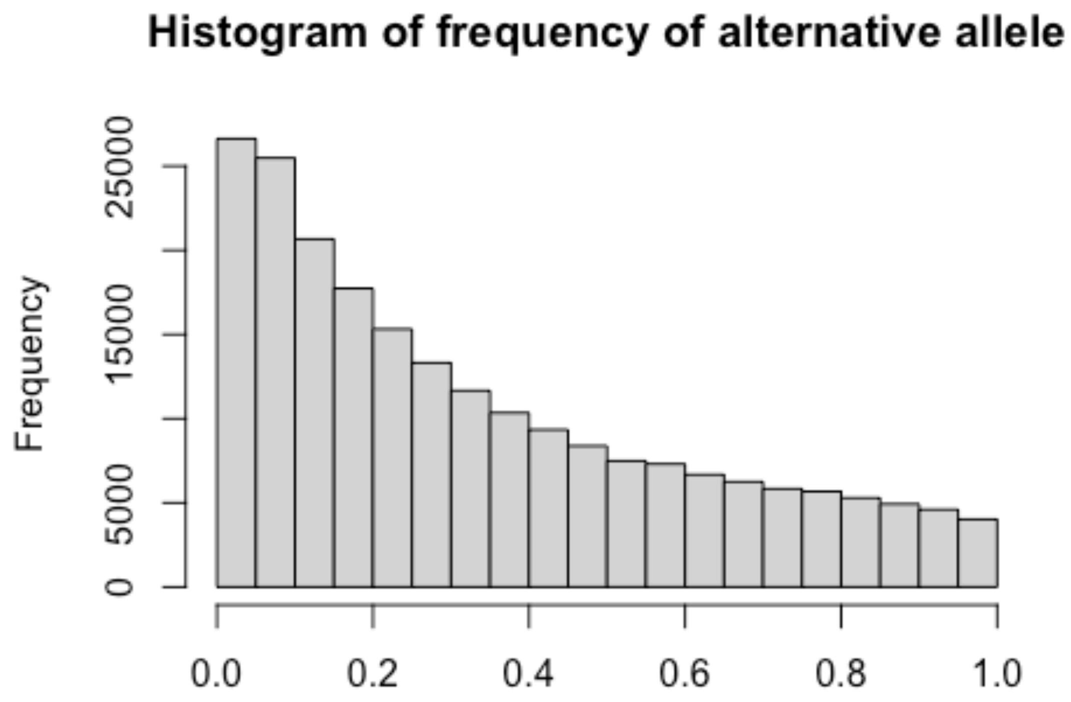
Now we go to R. Open R Studio and set your working directory using the function `setwd`. See the Supplementary material *R tutorial*.

- Visualize minor allele frequency distribution.

```
# First read the frequency file
dfreq<-read.table("output/ALL.225k_sites_freq.afreq", comment.char = "",heade
r=TRUE)
# See how it looks.
head(dfreq)

##   X.CHROM           ID REF ALT  ALT_FREQS OBS_CT
## 1       1   rs10458597   C   T 0.01062590   6870
## 2       1 1:564773:C:T   C   T 0.01850680   6322
## 3       1   rs11240776   A   G 0.00710763   6894
## 4       1    rs2980319   A   T 0.73194100   6894
## 5       1    rs2905036   C   T 0.97405800   6900
## 6       1    rs2341354   A   G 0.53411100   6772

# We are doing the same thing we did in our previous excercise but now in R i
nstead of the terminal.
```

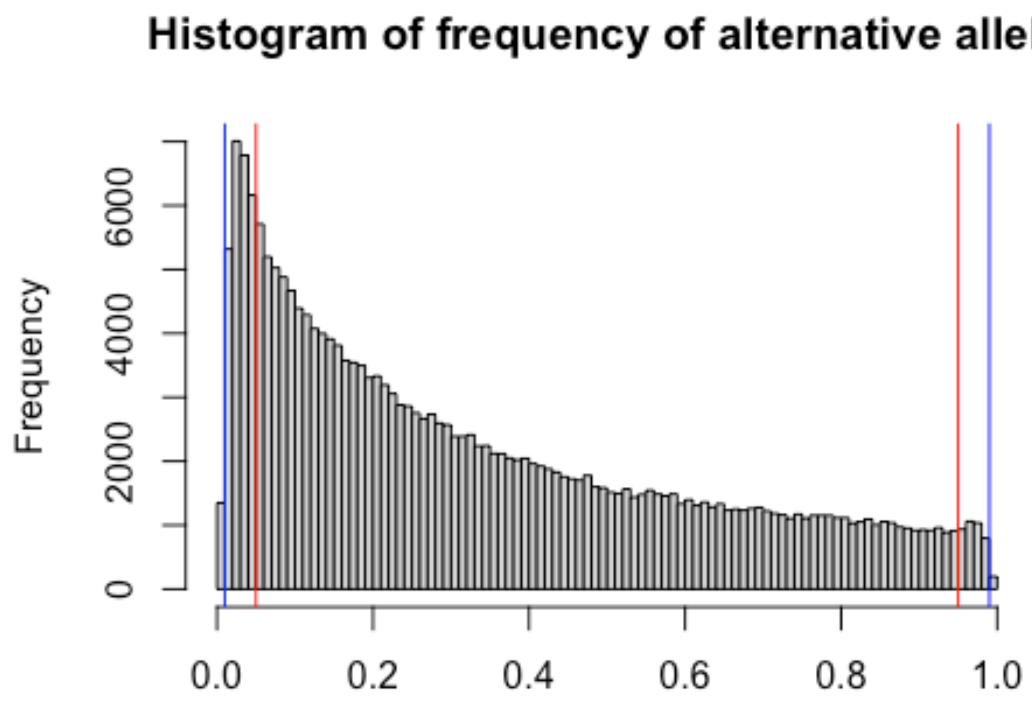
Now we will plot the distribution of the allele frequencies using the function `hist()`.

```
hist(dfreq[,5],xlab = "", main="Histogram of frequency of alternative allele"
)
```



Now try changing the breakpoints of the distribution.

```
hist(dfreq[,5],xlab = "", main="Histogram of frequency of alternative allele"
,breaks =seq(0,1,by=0.01) )

# We can use the function abline to delimit the distribution based on minor a
llele frequency thresholds.
abline(v=0.01,col="blue")
abline(v=0.99,col="blue")

abline(v=0.05,col="red")
abline(v=0.95,col="red")
```



Note: You can save the plot in your `analysis` folder. This can be don manually in Rstudio (see tutorial). Or you can do it using the command line. For example:

```
pdf("analysis/hist_maf.pdf")
hist(dfreq[,5],xlab = "", main="Histogram of frequency of alternative allele"
)
dev.off()
```

Excercise: Try saving using the commands `pdf` and `dev.off()` the histogram of that shows the alternative allele distribution with breakpoints of size 0.05.

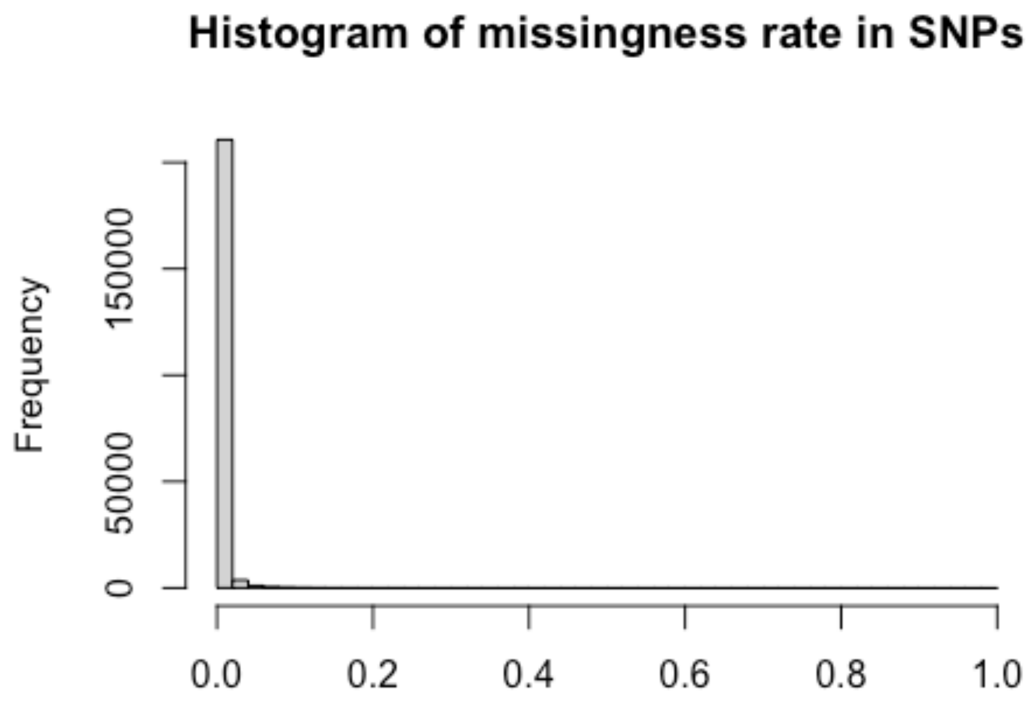
From now on you can choose your favorite way to save figures. For simplicity we will ommit the commands to save figure in the next figures.

The *Alternative Allele* used as reference to get its frequency is not necessarily the minor allele. Se we can do the following.

- Visualize missingnes rate in SNPs

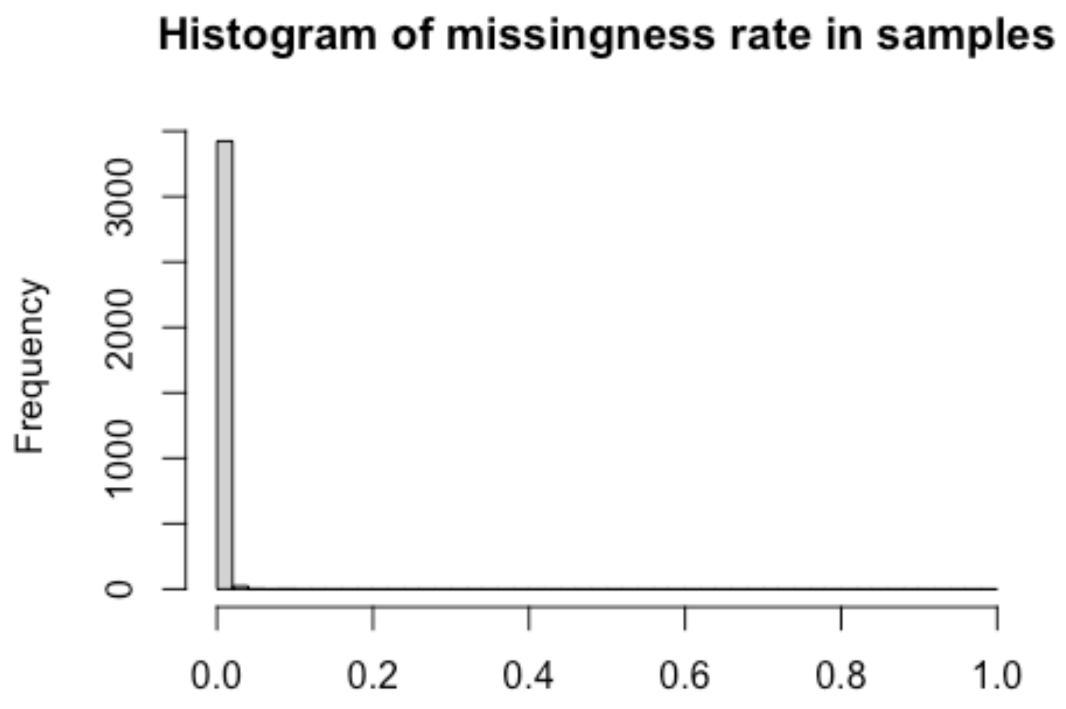
```
### read file and use head command to see file.
miss.snp<-read.table("output/ALL.225k_sites_miss.vmiss",comment.char = "",hea
der=TRUE)

head(miss.snp)
```

```
hist(miss.snp[,5], xlab = "", main="Histogram of missingness rate in SNPs", b
reaks =seq(0,1,by=0.02))
```



- Visualize missingnes rate in samples

```
miss.ind<-read.table("output/ALL.225k_sites_miss.smiss",comment.char = "",hea
der=TRUE)
hist(miss.ind[,5],xlab = "",main="Histogram of missingness rate in samples",
breaks =seq(0,1,by=0.02))
```



Quality control

Now let's apply the filters:

- `--geno 0.1` removes SNPs with genotyping rate less than 90%
- `--mind 0.1` removes individuals genotyping rate less than 90%
- `--maf 0.01` removes SNPs with MAF<1%.
- `--hwe 1e-50` removes SNPs that failed the hwe test.

```
./bin/plink2 --bfile data/ALL.wgs.nhgri_coriell_affy_6.20140825.genotypes_has
_ped.225k_sites.chr1-22 --geno 0.1 --mind 0.1 --maf 0.01 --hwe 1e-50 --make-b
ed --out  output/ALL.QC
```

Write down how many SNPs and individuals were removed.

Filter SNPs in LD

Some QC metrics that we will need to compute, such as relatedness or ancestry (via principal components), assume independent SNPs are measured. This means that they are not genetically correlated or in linkage disequilibrium (LD). We measure LD for pairs of nearby SNPs using the `--indep-pairwise` flag to compute pairwise genotypic correlation.

This function computes pairwise statistics on many SNPs, this takes a few seconds. This flag requires 3 numeric arguments, as follows:

1. window size in SNPs (e.g. 50)
2. the number of SNPs to shift the window at each step (e.g. 5)
3. the r2 threshold

Filtering out SNPs in LD is a two-step process: 1. Determining a list of SNPs in LD 2. Extracting independent SNPs

Let's try filtering out SNPs in LD from the dataset we have QC'd up until now with all populations:

```
./bin/plink2 --bfile output/ALL.QC --indep-pairwise 50 5 0.2 --out  output/AL
L.QC.ld
```

Question: How many SNPs will be removed due to LD filtering?

Now, we need to remove those SNPs from our QC file, as follows:

```
./bin/plink2 --bfile output/ALL.QC --extract output/ALL.QC.ld.prune.in --make
-bed --out  output/ALL.QC.ld02
```

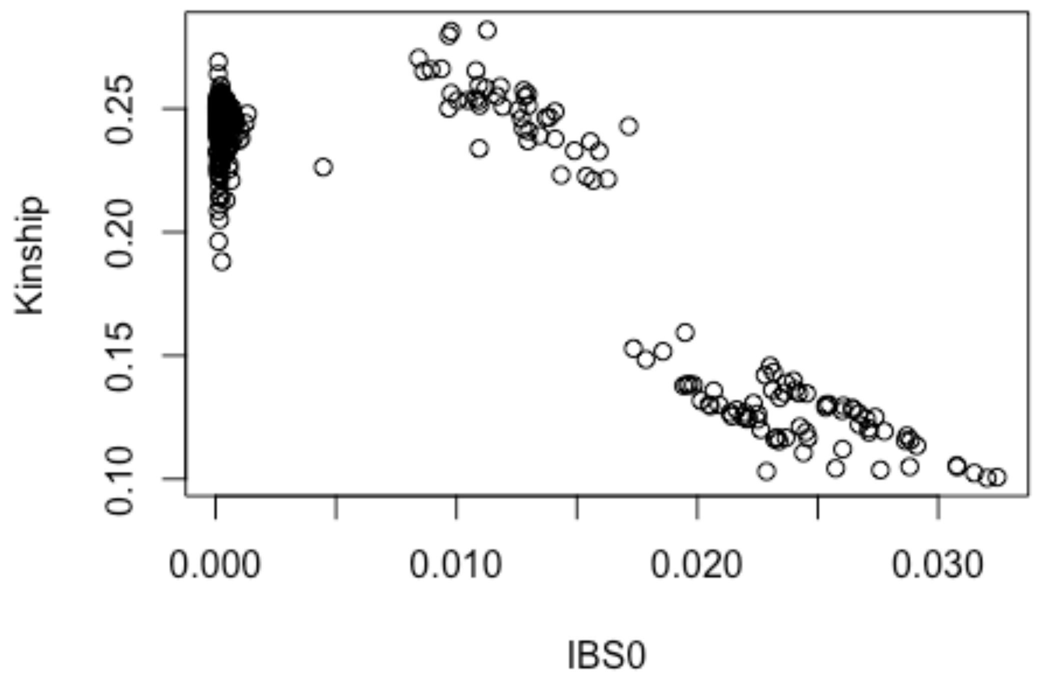
Infer relatedness and filter to unrelated individuals

Now that we have independent SNPs, we would like to compute pairwise relatedness among all pairs of individuals.

```
./bin/plink2 --bfile output/ALL.QC.ld02 --make-king-table --king-table-filter
0.1 --out output/ALL.QC.ld02_rel
```

Lets plot the relationship between the 1KG samples

```
dkin<-read.table("output/ALL.QC.ld02_rel.kin0",comment.char = "",header=TRUE)
plot(dkin$IBS0,dkin$KINSHIP,xlab="IBS0",ylab = "Kinship")
```



To filter individuals with up to a second degree relatedness can do this using the `--make-king` flag. We use the `--king-cutoff 0.125` to remove individuals with up to a second degree relatedness.

```
./bin/plink2 --bfile output/ALL.QC.ld02 --make-king --king-cutoff 0.125 --out
output/ALL.QC.ld02_rel
```

Remove one individual per pair from close relatives, using the `--remove` flag. From now on we will use the plink file `ALL.QC`.

```
./bin/plink2 --bfile output/ALL.QC --remove output/ALL.QC.ld02_rel.king.cutof
f.out.id --make-bed --out output/ALL.QC.unrel
```

Note: you could use `--keep` flag and the file `ALL.QC.ld02_rel.king.cutoff.in.id`.

Write down how many individuals you have:

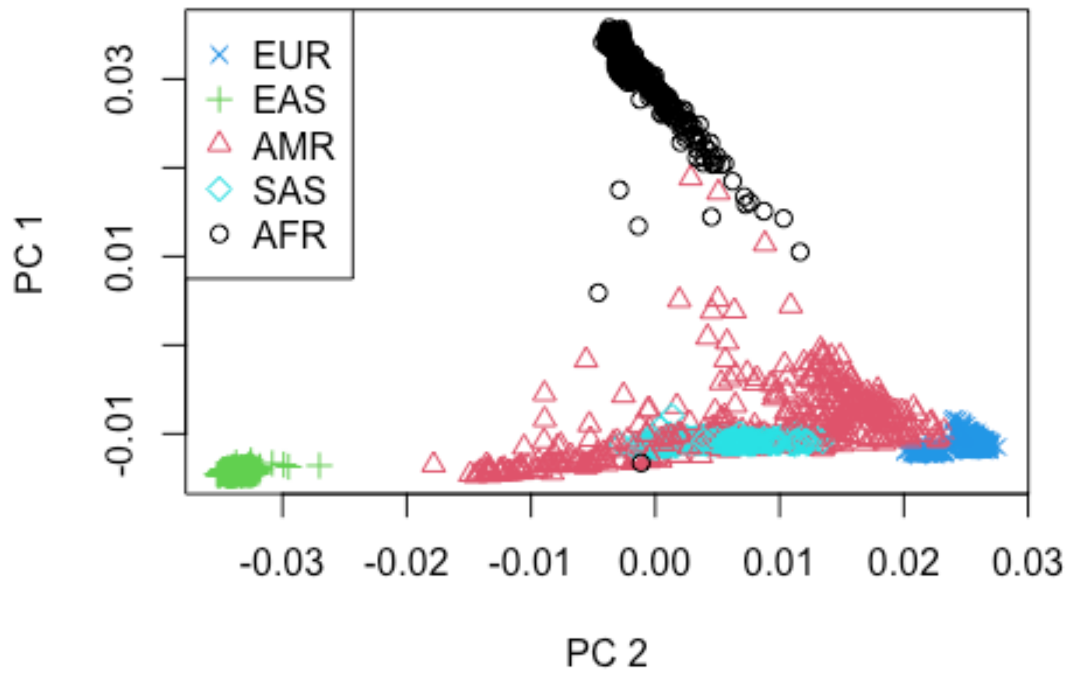
Principal Components Analysis

```
./bin/plink2 --bfile output/ALL.QC.ld02 --remove output/ALL.QC.ld02_rel.king.
cutoff.out.id --pca --out output/ALL.QC.unrel.ld02_pca
```

Lets look at the output.

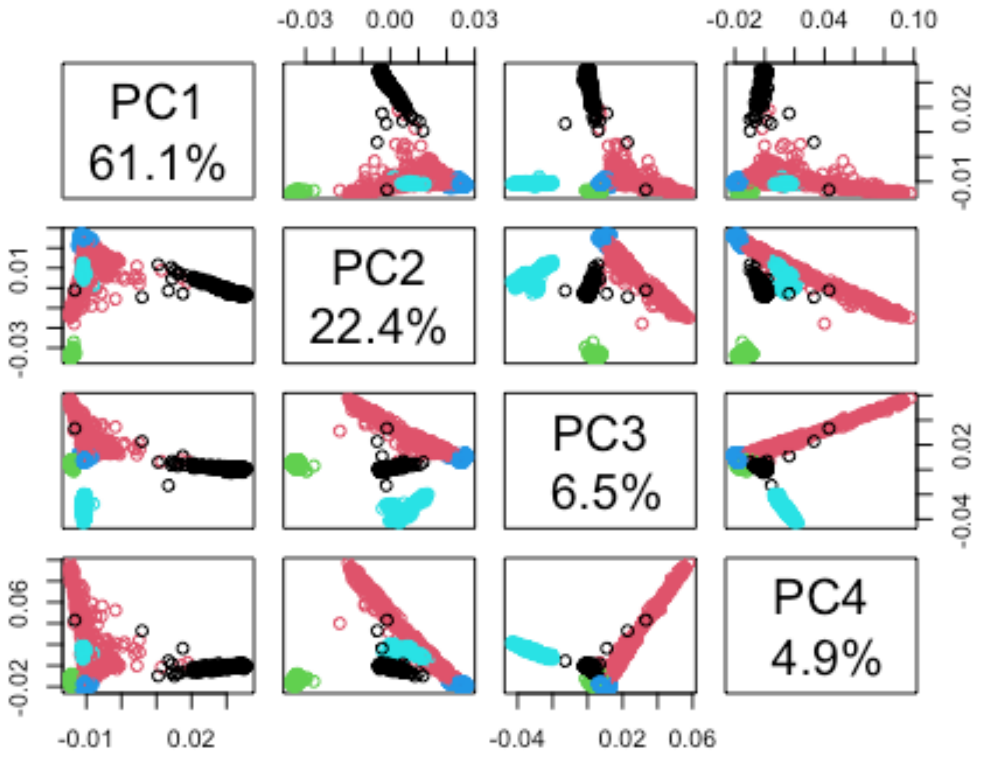
We have two different outputs: `*.eigenvec` and `*.eigenval`.

```
d.info<-read.table("data/sample_info.txt",header=TRUE)
d.pca <- read.table("output/ALL.QC.unrel.ld02_pca.eigenvec",comment.char = ""
,header=TRUE)
colnames(d.pca)<-c("FID", "IID",paste0("PC",1:10))
d.pca2<-merge(d.pca,d.info[,2:4],by="IID")

print(head(d.pca2))

plot(d.pca2$PC2, d.pca2$PC1, col=as.integer(factor(d.pca2$Super_pop)),
     pch=as.integer(factor(d.pca2$Super_pop)), xlab="PC 2", ylab="PC 1")
legend("topleft",legend =unique(d.pca2$Super_pop),
       col=unique(as.integer(factor(d.pca2$Super_pop))),
       pch=unique(as.integer(factor(d.pca2$Super_pop)))   )
```



We can also plot the principal component pairs for the first four PCS:

```
d.eval <- read.table("output/ALL.QC.unrel.ld02_pca.eigenval")
d.eval[,1]<-(d.eval[,1]/sum(d.eval))*100
lbls <- paste("PC", 1:4, "\n", format(d.eval[1:4,1], digits=2), "%", sep="")
pairs(d.pca2[,3:6], col=as.integer(factor(d.pca2$Super_pop)), labels=lbls)
```



Association analyses

Once we have computed our PCs, we need to make a covariate file that has PCS and other covariates of interest (in our case we will add Gender).

```
d.info<-read.table("data/sample_info.txt",header=TRUE)
d.pca <- read.table("output/ALL.QC.unrel.ld02_pca.eigenvec",
                    comment.char = "",header=TRUE)
colnames(d.pca)[1]<-"FID"
d.covs<-merge(d.pca,d.info[,-1],by="IID")
print(head(d.covs))

##       IID FID        PC1       PC2        PC3        PC4          PC5
## 1 HG00096 GBR -0.0116789 0.0257270 0.01052830 -0.0179968 -0.001479420
## 2 HG00097 GBR -0.0116669 0.0261898 0.00979013 -0.0180333 -0.000718431
## 3 HG00098 GBR -0.0116162 0.0256472 0.00948426 -0.0165324  0.000115123
## 4 HG00099 GBR -0.0115379 0.0257724 0.01034540 -0.0176784 -0.001000880
## 5 HG00100 GBR -0.0111550 0.0257968 0.01050450 -0.0166039  0.001209080
## 6 HG00101 GBR -0.0113188 0.0258685 0.01061030 -0.0170896 -0.001700360
##          PC6          PC7        PC8          PC9         PC10 Gender Supe
r_pop
## 1 0.00377325 -0.004569100 0.01049980 -0.000358435 -0.002823170   male
```

```
EUR
## 2 0.00350280 -0.000480694 0.00999964  0.001835820 -0.000233069 female
EUR
## 3 0.00388170 -0.002516310 0.00833334 -0.001513800 -0.012795100   male
EUR
## 4 0.00781816 -0.003088670 0.01231630  0.004743440 -0.004304530 female
EUR
## 5 0.00346880 -0.003136840 0.00726526  0.004271430 -0.015643300 female
EUR
## 6 0.00159130 -0.004799010 0.01269510  0.001143040 -0.007603360   male
EUR

d.covs<-d.covs[,c(2,1,3:14)]
write.table(d.covs,"output/sample_covariates.txt",row.names = F,
            col.names = T, quote = F)
```

First we are going to run an unadjusted analysis. In this case a simple linear regresssion with no covariates.

```
./bin/plink2 --bfile output/ALL.QC.unrel --glm allow-no-covars --pheno data/p
heno.txt --out output/ALL.QC.unrel_pheno_unadj

# Take a look at the output
head output/ALL.QC.unrel_pheno_unadj.phi.glm.linear
```

Now, we are going to see what happens if we adjust by the first 10 PCs.

```
./bin/plink2 --bfile output/ALL.QC.unrel --glm hide-covar --covar output/samp
le_covariates.txt --covar-name PC1,PC2,PC3,PC4,PC5,PC6,PC7,PC8,PC9,PC10,Gende
r  --pheno data/pheno.txt --out output/ALL.QC.unrel_pheno_covs

# Take a look at the output
head  output/ALL.QC.unrel_pheno_covs.phi.glm.linear
```

Visualization of GWAS results

We will use the package qqman to make the QQ-plots and Manhattan plots

```
#install.packages('qqman')
library(qqman)

##

## For example usage please run: vignette('qqman')

##

## Citation appreciated but not required:

## Turner, (2018). qqman: an R package for visualizing GWAS results using Q-Q
and manhattan plots. Journal of Open Source Software, 3(25), 731, https://doi
.org/10.21105/joss.00731.
```

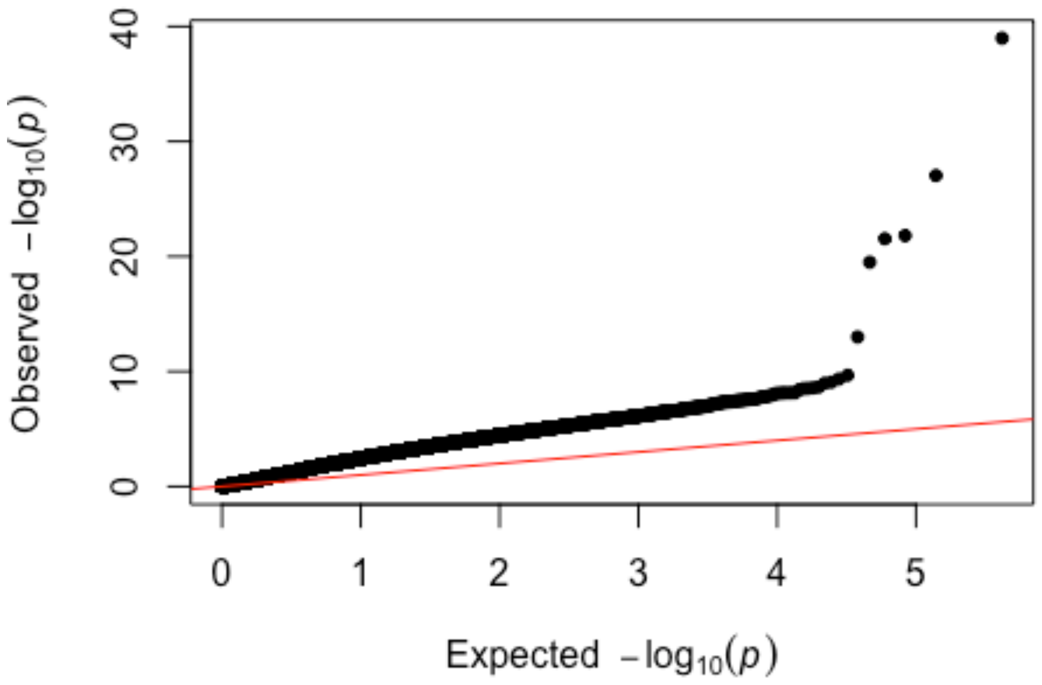
```
##
```

Read association analysis

```
gwas_unadj <- read.table("output/ALL.QC.unrel_pheno_unadj.phi.glm.linear",
                         comment.char = "",header=TRUE)
```

QQ plot

```
qq(gwas_unadj$P)
```



```
# Genomic control
median(gwas_unadj$T_STAT^2)/0.456

## [1] 3.821111
```

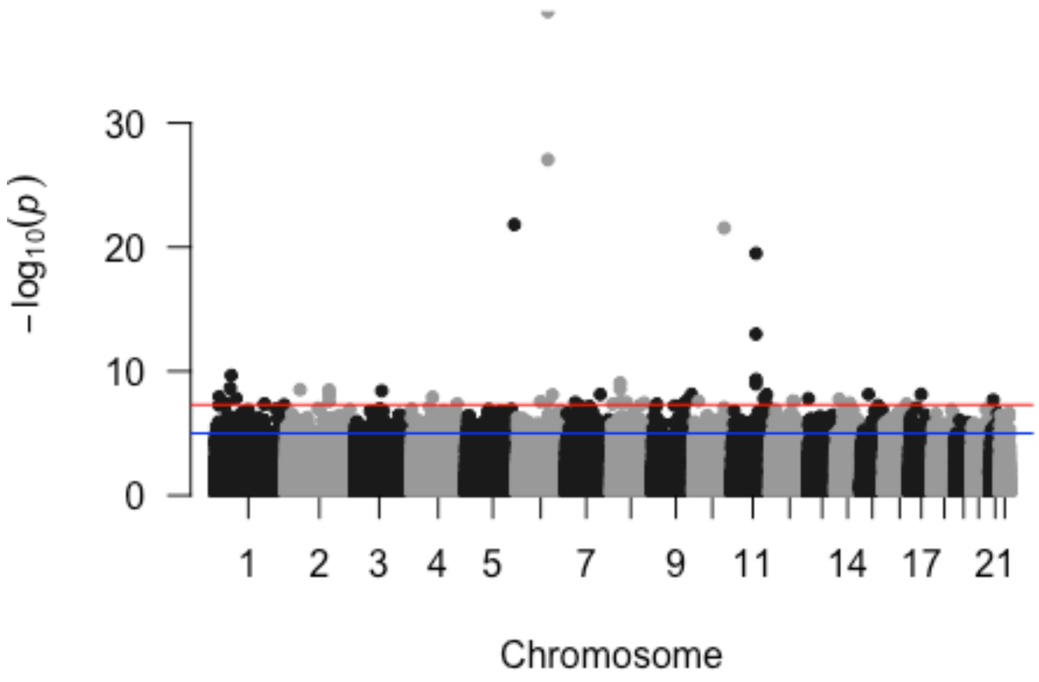
Manhattan plot.

First use the command `help(manhattan)` to see the requirements of the function to make manhattan plots.

Use the function `head` or `colnames` to identify the relevant columns that we need to use to make the manhattan plot.

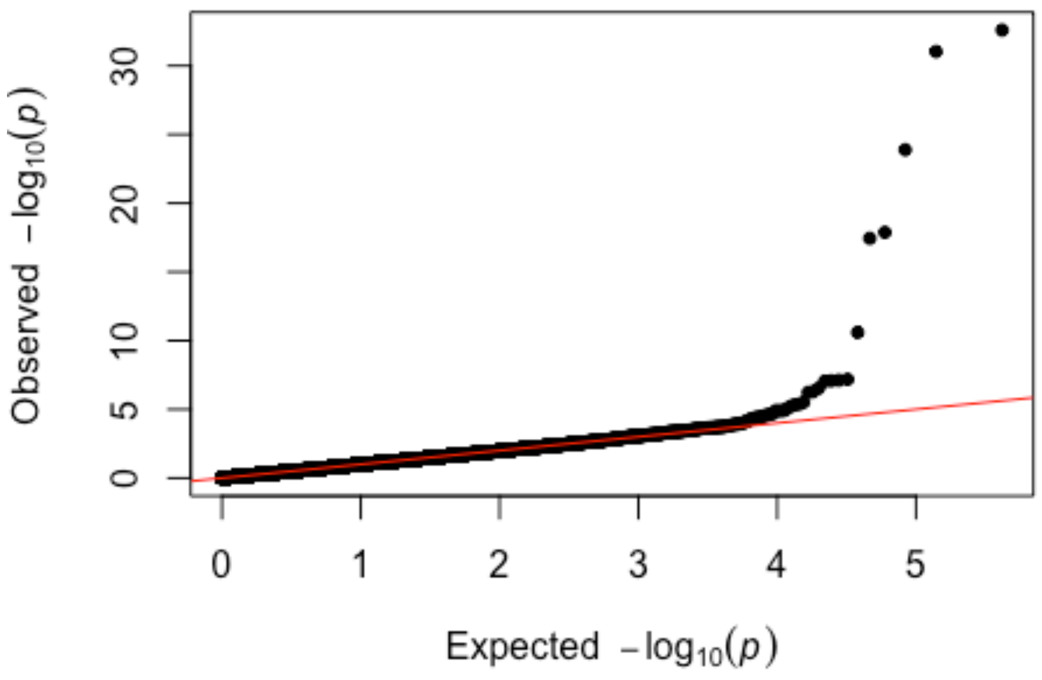
```
print(colnames(gwas_unadj))
```

```
# If you have X, Y, or MT chromosomes, be sure to renumber these 23, 24, 25,
etc.
manhattan(gwas_unadj,chr="X.CHROM", bp="POS", p="P", snp="ID")
```



Now we will read the association file when adjusting by 10 PCs. And we will plot its QQ-plot

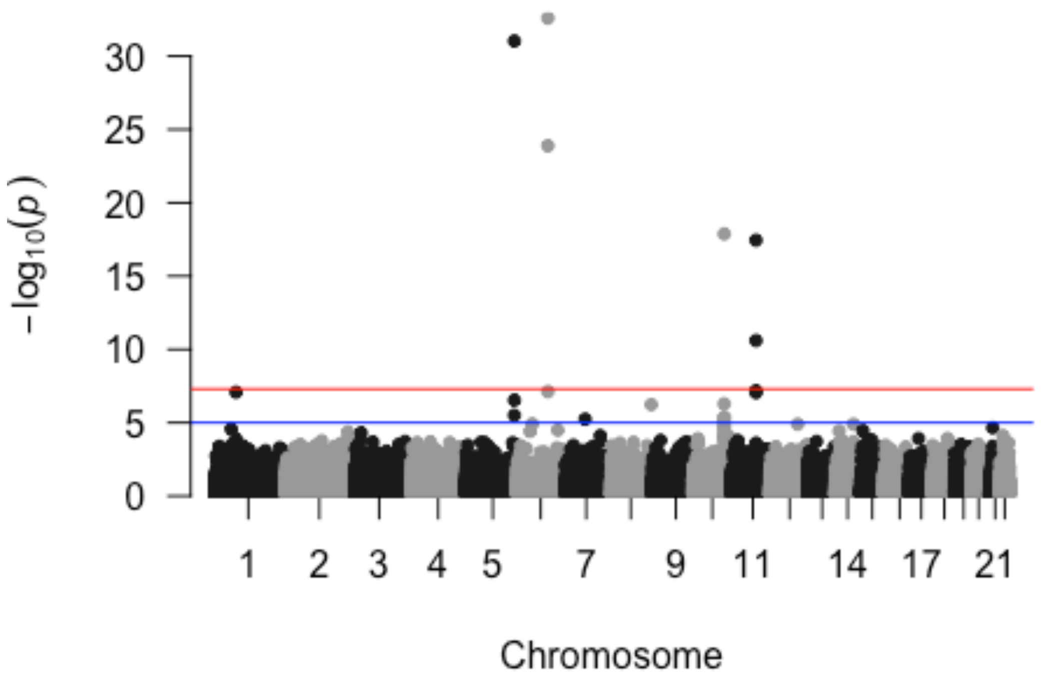
```
gwas <- read.table("output/ALL.QC.unrel_pheno_covs.phi.glm.linear",
                   comment.char = "",header=TRUE)
qq(gwas$P)
```



```
# Genomic control
median(gwas$T_STAT^2)/0.456

## [1] 0.9870646
```

Let's see the results in a Manhattan plot.

```
manhattan(gwas,chr="X.CHROM", bp="POS", p="P", snp="ID")
```



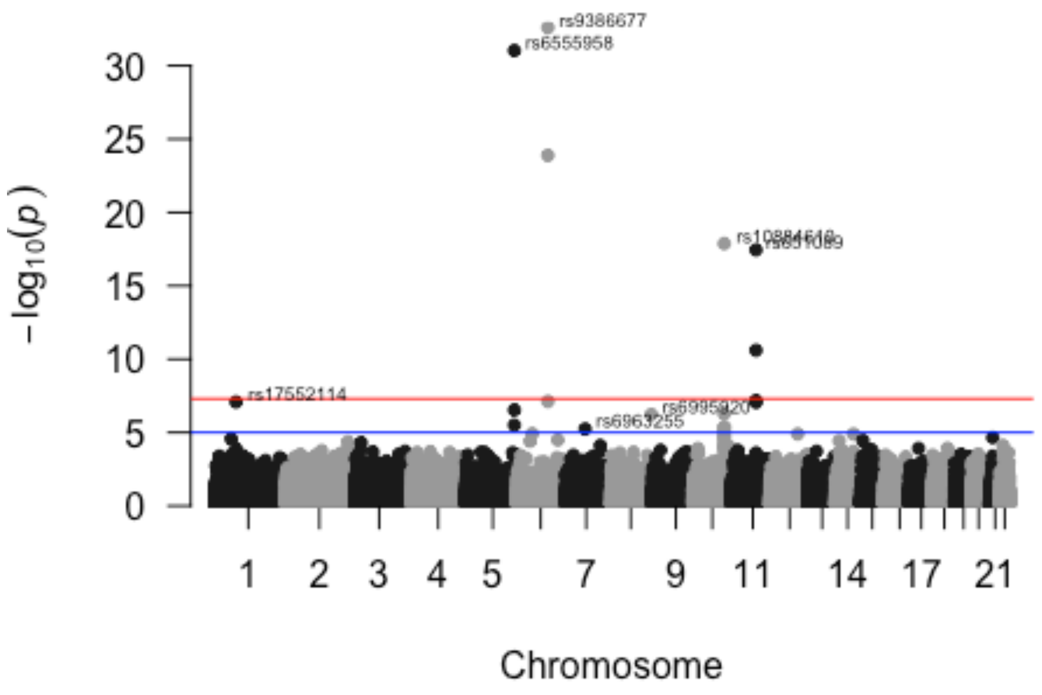
Questions:

- How many GWAS significant SNPs have you identified in the unadjusted model? Do you believe this signals are true? What factors make this signals to achieve a singificant p-value threshold.
- What happens if you only adjust by discrete populations compared to adjusting with PCs?
- How many GWAS significant SNPs have you identified in the adjusted by 10 PCs model? Do you believe this signals are true? What are the next steps to interpret this results?

Additional features for Manhattan plot

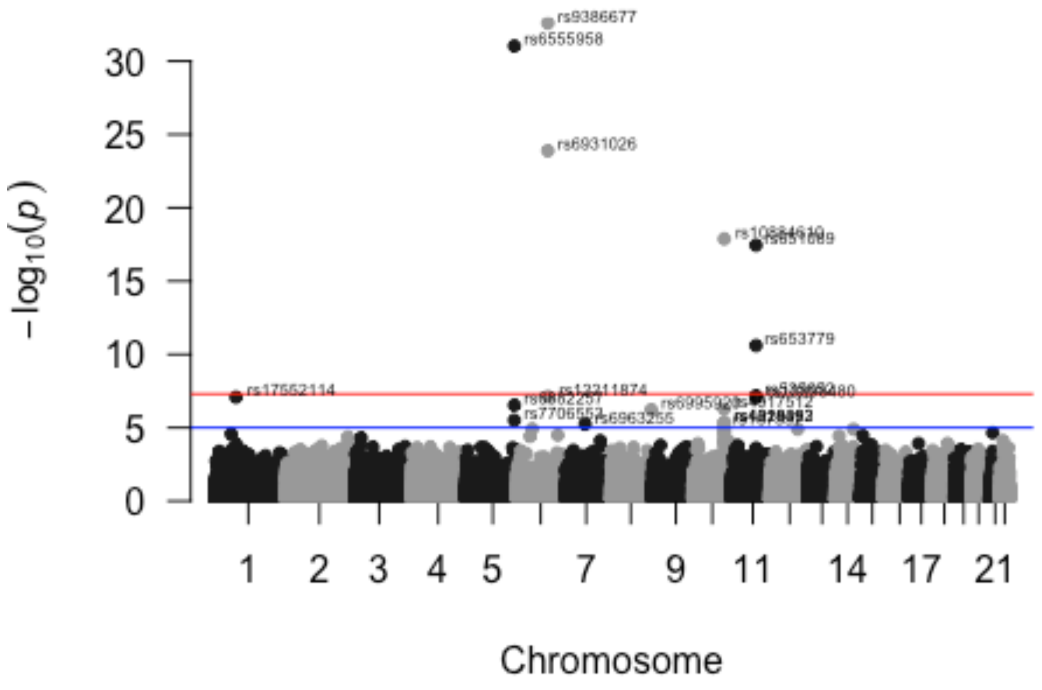
We can also annotate SNPs based on their p-value. By default, this only annotates the top SNP per chromosome that exceeds the annotatePval threshold.

```
manhattan(gwas,chr="X.CHROM", bp="POS", p="P", snp="ID", annotatePval = 1e-5)
```



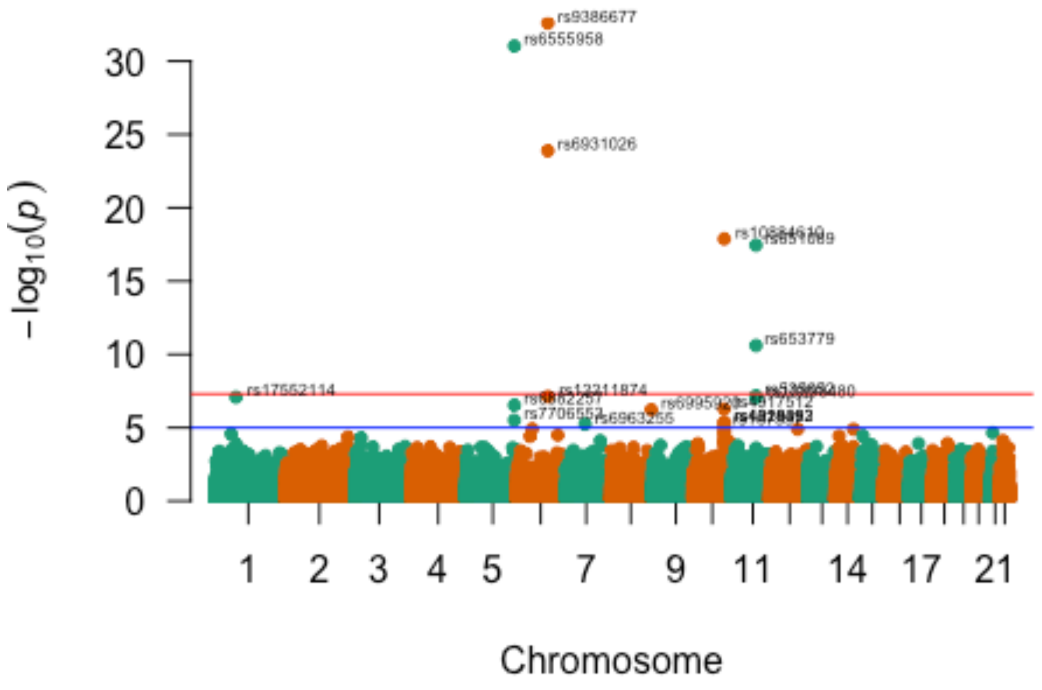
If we want all snps that exceeds that p value threshold we add annotateTop = FALSE)

```
manhattan(gwas,chr="X.CHROM", bp="POS", p="P", snp="ID", annotatePval = 1e-5,
annotateTop = FALSE)
```



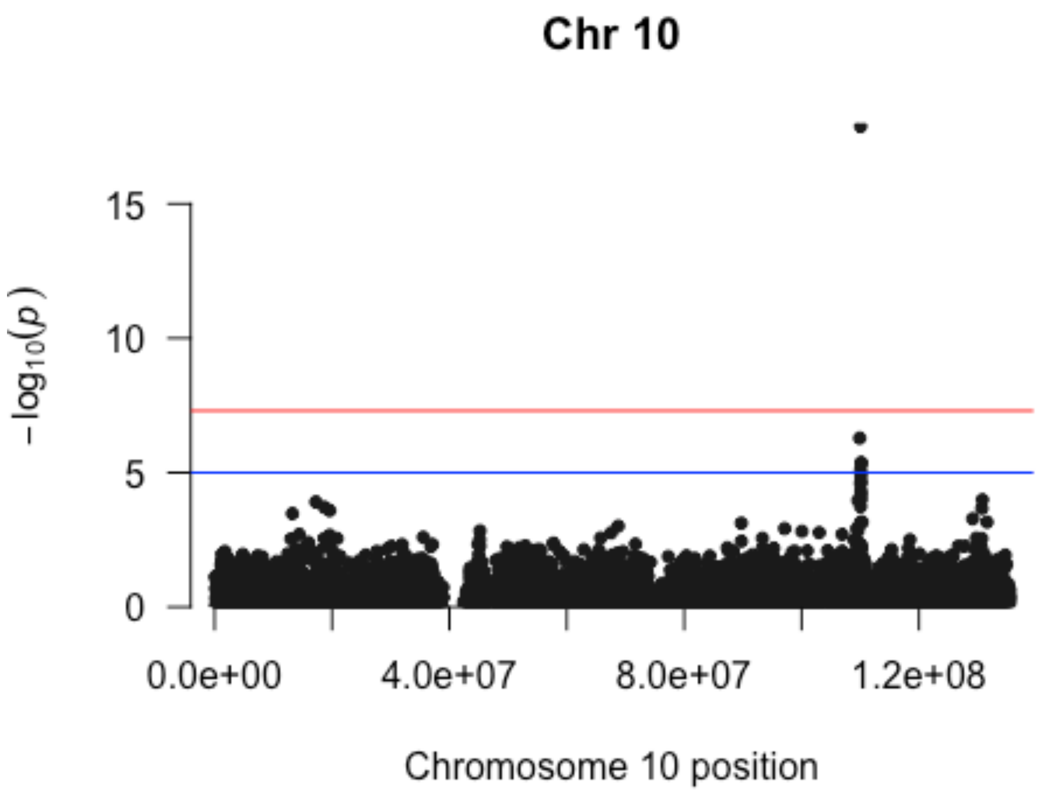
We can change the color of the bars. Go into ColorBrewer2 and choose a combination of two colors.

```
manhattan(gwas,chr="X.CHROM", bp="POS", p="P", snp="ID", annotatePval = 1e-5,
annotateTop = FALSE, col=c("#1b9e77","#d95f02"))
```



We can limit the plot to a single chromosome

```
manhattan(subset(gwas, X.CHROM == 10), chr="X.CHROM", bp="POS", p="P", snp="I
D", main = "Chr 10")
```



Lets do a Locus Zoom plot

First we are going to create a file that only has a region of SNPs. Lets focus on Chromosome 10. Locus Zoom accepts tab-delimited files in either `.txt` or `.gz` formats. So we need to add the specification `sep="\t"` when we write the file.

```
write.table(subset(gwas, X.CHROM == 10),"output/assoc_chr10.txt",row.names =
F, col.names = T, quote = F,sep="\t")

# Take a look at how the order of the columns. You need to specify them in th
e webpage.
head(subset(gwas, X.CHROM == 10))
```

Then we will upload the file we created in the Locus Zoom page.

Choose another chromosome and repeat the previous excercise

Additional excercise for PCA

If we want to focus in only one group and capture population specific structure we can run PCA only in this population. For example, lets look only into the AMR population.

First we extract the subgroup of AMR population, and create and additional file.

```
d.info<-read.table("data/sample_info.txt",header=TRUE)
d.amr<-subset(d.info,Super_pop=="AMR")
write.table(d.amr,"output/sample_amr.txt",row.names = F,
            col.names = T, quote = F)

./bin/plink2 --bfile output/ALL.QC.unrel --keep output/sample_amr.txt --make-
bed --out output/ALL.QC.unrel.amr
```

Now we keep only a subset of independent SNPs

```
./bin/plink2 --bfile output/ALL.QC.unrel.amr --indep-pairwise 50 5 0.2 --out
output/ALL.QC.unrel.amr.ld

./bin/plink2 --bfile output//ALL.QC.unrel.amr --extract output/ALL.QC.unrel.a
mr.ld.prune.in --make-bed --out output/ALL.QC.unrel.amr.ld02
```

Let's run the principal component analysis

```
./bin/plink2 --bfile output/ALL.QC.unrel.amr.ld02 --pca --out output/ALL.QC.u
nrel.amr.ld02_pca
```

Now lets read the our data

```
d.pca <- read.table("output/ALL.QC.unrel.amr.ld02_pca.eigenvec",comment.char
= "",header=TRUE)

colnames(d.pca)[1]<-c("pop")

head(d.pca)

plot(d.pca$PC2, d.pca$PC1, col=as.integer(factor(d.pca$pop)),
     pch=as.integer(factor(d.pca$pop)), xlab="PC 2", ylab="PC 1")

legend("bottomright",legend =unique(d.pca$pop),
       col=unique(as.integer(factor(d.pca$pop))),
       pch=unique(as.integer(factor(d.pca$pop)))  )
```

We can also plot the principal component pairs for the first four PCS:

```
d.eval <- read.table("output/ALL.QC.unrel.amr.ld02_pca.eigenval")
d.eval[,1]<-(d.eval[,1]/sum(d.eval))*100
lbls <- paste("PC", 1:4, "\n", format(d.eval[1:4,1], digits=2), "%", sep="")
pairs(d.pca[,3:6], col=as.integer(factor(d.pca$pop)),
      pch=unique(as.integer(factor(d.pca$pop))),labels=lbls)
par(xpd = TRUE)
legend("bottomright",legend =unique(d.pca$pop),
       col=unique(as.integer(factor(d.pca$pop))),
       pch=unique(as.integer(factor(d.pca$pop)))  )
```

Additional excercise for GWAS

Run another GWAS analysis but instead of adjusting for 10 PCs, use the “Super population” labels as covariates.

```
./bin/plink2 --bfile output/ALL.QC.unrel \
--glm 'hide-covar' \
--covar output/sample_covariates.txt \
--covar-name Super_pop  \
--pheno data/pheno.txt \
--out output/ALL.QC.unrel_pheno_adjPop

gwasPop <- read.table("output/ALL.QC.unrel_pheno_adjPop.phi.glm.linear",
                   comment.char = "",header=TRUE)
qq(gwasPop$P)
```

Compare the results with the previous analyses.