Genome-Wide Association Studies

Caitlin Collins[*], Thibaut Jombart

Imperial College London

MRC Centre for Outbreak Analysis and Modelling

October 30, 2014

Abstract

This practical provides an introduction to Genome-Wide Association Studies in `R` . First, we will examine population structures within the data. Second, we will test for associations between a genome-wide SNP panel and our phenotypic trait of interest: antibiotic resistance. We will carry out this test and perform feature selection with three separate methods: The univariate Fisher's exact test, the multivariate penalized regression technique LASSO, and with an extension of the multivariate factorial method DAPC . Then, we will use PCA to correct for population stratification. Finally, we will re-run the three methods of association testing and feature selection on the "corrected" dataset and compare the results.

[*]`caitlin.collins12@imperial.ac.uk`

Contents

1 The data **3**
- 1.1 First assessment of the genetic diversity . . . 5
- 1.2 Identifying SNPs linked to antibiotic resistance . . . 12

2 Association testing and feature selection **15**
- 2.1 Univariate method . . . 15
- 2.2 Multivariate methods . . . 17
  - 2.2.1 LASSO . . . 17
  - 2.2.2 DAPC-based feature selection . . . 19
  - 2.2.3 A little more on DAPC-based feature selection . . . 23

3 Correcting for population stratification **28**

4 Association testing and feature selection after correcting **32**
- 4.1 Univariate method . . . 32
- 4.2 Multivariate methods . . . 36
  - 4.2.1 LASSO . . . 36
  - 4.2.2 DAPC-based feature selection . . . 38
- 4.3 Interpreting the significance of the SNPs selected . . . 40

5 The *adegenet* Server **44**

1 The data

```
library(adegenet)
install.packages("glmnet", dep=TRUE)
library(glmnet)
```

```
## Loading required package:  ade4
##    ==========================
##    adegenet 1.4-1 is loaded
##    ==========================
##
##  - to start, type '?adegenet'
##  - to browse adegenet website, type 'adegenetWeb()'
##  - to post questions/comments:  adegenet-forum@lists.r-forge.r-project.org
##
##
## Loading required package:  Matrix
## Loaded glmnet 1.9-5
```

The simulated data used in this practical are available online from the following address: http://adegenet.r-forge.r-project.org/files/simGWAS/simGWAS.RData. The dataset is in R's binary format (extension `RData`), which uses compression to store data efficiently (the raw csv file would be more than 4MB). R objects can be loaded into R using `load`. The instruction `url` is required to load the data directly from the internet; as data are loaded, a new object `simGWAS` appears in the R environment:

```
load(url("http://adegenet.r-forge.r-project.org/files/simGWAS/simGWAS.RData"))
ls(pattern="sim")

## [1] "simGWAS"

class(simGWAS)

## [1] "list"

names(simGWAS)

## [1] "snps" "phen"

class(simGWAS$snps)

## [1] "matrix"

class(simGWAS$phen)
```

```
## [1] "character"

dim(simGWAS$snps)

## [1]    95 10000

simGWAS$snps[1:10,1:20]

##            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## isolate-1  0 0 0 1 0 1 1 0 0  1  1  0  1  1  0  0  1  0  0  1
## isolate-2  0 0 0 1 0 1 1 0 0  1  1  1  1  1  0  1  1  0  0  1
## isolate-3  0 0 0 1 1 1 1 1 0  1  0  1  1  1  1  1  1  1  0  1
## isolate-4  0 1 1 1 0 1 1 0 0  0  1  1  0  1  1  1  0  1  0  0
## isolate-5  1 0 0 1 0 1 0 1 0  1  1  0  1  1  0  1  1  1  0  0
## isolate-6  1 1 0 1 1 1 1 0 0  1  1  1  1  1  0  1  1  1  0  1
## isolate-7  1 1 0 1 1 0 0 1 0  1  0  1  1  1  0  0  1  1  1  0
## isolate-8  0 1 0 0 0 1 0 0 0  1  0  1  1  0  0  1  1  1  1  0
## isolate-9  0 1 0 1 1 1 0 0 0  1  1  1  1  0  1  1  1  1  1  1
## isolate-10 0 0 0 1 1 1 1 0 0  1  1  0  0  0  0  0  1  1  0  1

print(object.size(simGWAS$snps), unit="Mb")

## 7.8 Mb

length(simGWAS$phen)

## [1] 95

simGWAS$phen

##  [1] "R" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "R" "S" "S" "R" "S" "S"
## [18] "S" "S" "S" "S" "S" "R" "S" "S" "S" "S" "S" "S" "S" "R" "R" "S" "S"
## [35] "S" "S" "R" "S" "S" "R" "R" "R" "S" "S" "S" "R" "S" "S" "S" "S" "S"
## [52] "S" "S" "S" "R" "R" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S"
## [69] "S" "R" "R" "R" "S" "S" "R" "S" "R" "R" "R" "R" "S" "S" "S" "S" "S"
## [86] "S" "S" "S" "S" "S" "R" "S" "S" "R" "R"

table(simGWAS$phen)

##
##  R  S
## 24 71
```

The object `simGWAS` is a list with two components: `$snps` is a matrix of Single Nucleotide Polymorphism (SNPs) data, and `$phen` is the phenotype of the different sampled isolates.In this analysis, the individuals are isolates of bacteria, and the phenotype of interest is antibiotic

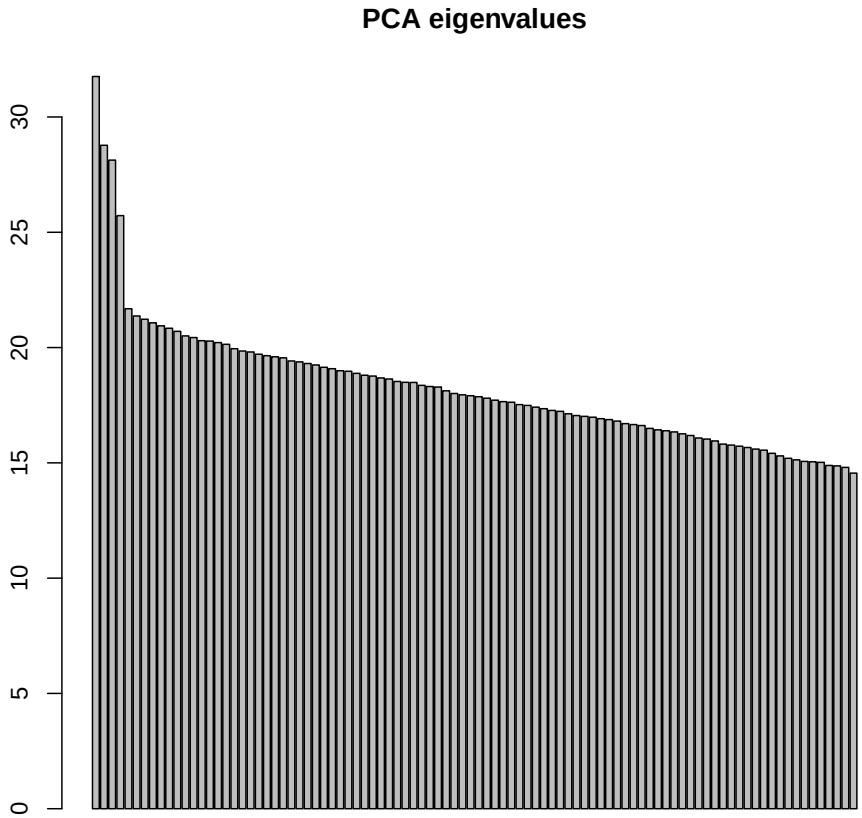
resistance. The 'R' and 'S' in the above table stand for the two levels of this phenotype: 'Resistant' and 'Susceptible'.
The SNPs data has a modest size by GWAS standards: only 95 isolates (in row) and 10000 SNPs (alleles coded as 0/1). Note that here, all SNPs are binary, so that only one allele needs to be stored. Consequently, we do not need to use the `genind` class to store the data (this would be a waste of RAM - not a problem here, but definitely a concern for larger datasets). To simplify further commands, we create the new objects `snps` and `phen` from `simGWAS`:

```
snps <- simGWAS$snps
phen <- factor(simGWAS$phen)
```

1.1 First assessment of the genetic diversity

Principal Component Analysis (PCA) is a very powerful tool for reducing the diversity contained in massively multivariate data into a few synthetic variables (the principal components — PCs). There are several versions of PCA implemented in R. Here, we use `dudi.pca` from the *ade4* package, specifying that variables should not be scaled (`scale=FALSE`) to unit variances (this is only useful when variables have inherently different scales of variation, which is not the case here):

```
pca1 <- dudi.pca(snps, scale=FALSE)
```



The method displays a screeplot (barplot of eigenvalues) to help the user decide how many PCs should be retained. The general rule is to retain only the largest eigenvalues, after which non-structured variation results in smoothly decreasing eigenvalues. How many PCs would you retain here?

```
pca1

## Duality diagramm
## class: pca dudi
## $call: dudi.pca(df = snps, scale = FALSE, scannf = FALSE, nf = 4)
##
## $nf: 4 axis-components saved
## $rank: 94
## eigen values: 31.76 28.77 28.13 25.72 21.68 ...
##   vector length mode    content
## 1 $cw    10000  numeric column weights
## 2 $lw    95     numeric row weights
## 3 $eig   94     numeric eigen values
##
```

```
##   data.frame nrow  ncol  content
## 1 $tab       95    10000 modified array
## 2 $li        95    4     row coordinates
## 3 $l1        95    4     row normed scores
## 4 $co        10000 4     column coordinates
## 5 $c1        10000 4     column normed scores
## other elements: cent norm
```

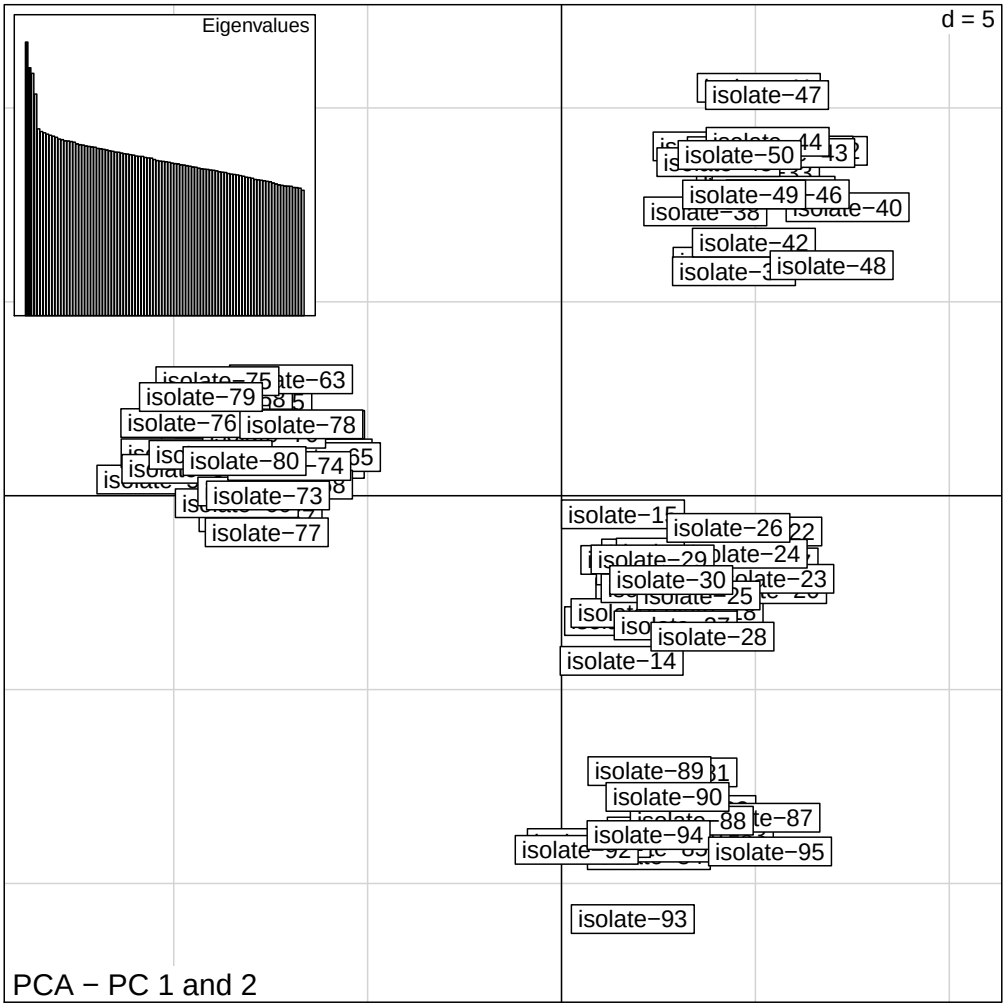
The object `pca1` contains various information. Most importantly:

- `pca1$eig`: contains the eigenvalues of the analysis, representing the amount of information contained in each PC.
- `pca1$li`: contains the principal components.
- `pca1$c1`: contains the principal axes (loadings of the variables).

```
head(pca1$eig)

## [1] 31.75594 28.77240 28.12875 25.72180 21.68303 21.36911

head(pca1$li)

##              Axis1     Axis2     Axis3      Axis4
## isolate-1 3.606420 -2.132999  9.622764  -6.301912
## isolate-2 1.912918 -1.656548  8.734490 -10.006055
## isolate-3 2.316603 -2.564638  9.324818  -7.445660
## isolate-4 2.490536 -2.484711  8.819193  -6.029816
## isolate-5 2.448958 -1.489571  8.576321  -8.775661
## isolate-6 2.938701 -2.693103 10.876804  -3.797021

head(pca1$c1)

##              CS1          CS2          CS3           CS4
## X1  1.004273e-02  0.004291539 -0.003509719 -0.0092503284
## X2 -5.145732e-03 -0.003539221 -0.001470553  0.0075073374
## X3 -3.349998e-05 -0.003362894  0.003797944  0.0013048886
## X4 -1.017829e-03  0.002489303 -0.002323418 -0.0007847613
## X5  7.047362e-03  0.007801922 -0.003475240  0.0057483659
## X6 -1.011989e-02  0.013435213  0.021812199 -0.0321210072
```

Because of the large number of variables, the usual biplot (function `scatter`) is useless to visualize the results (try `scatter(pca1)` if unsure). We represent only PCs using `s.label`:

```
s.label(pca1$li, sub="PCA - PC 1 and 2")
add.scatter.eig(pca1$eig,4,1,2, ratio=.3, posi="topleft")
```



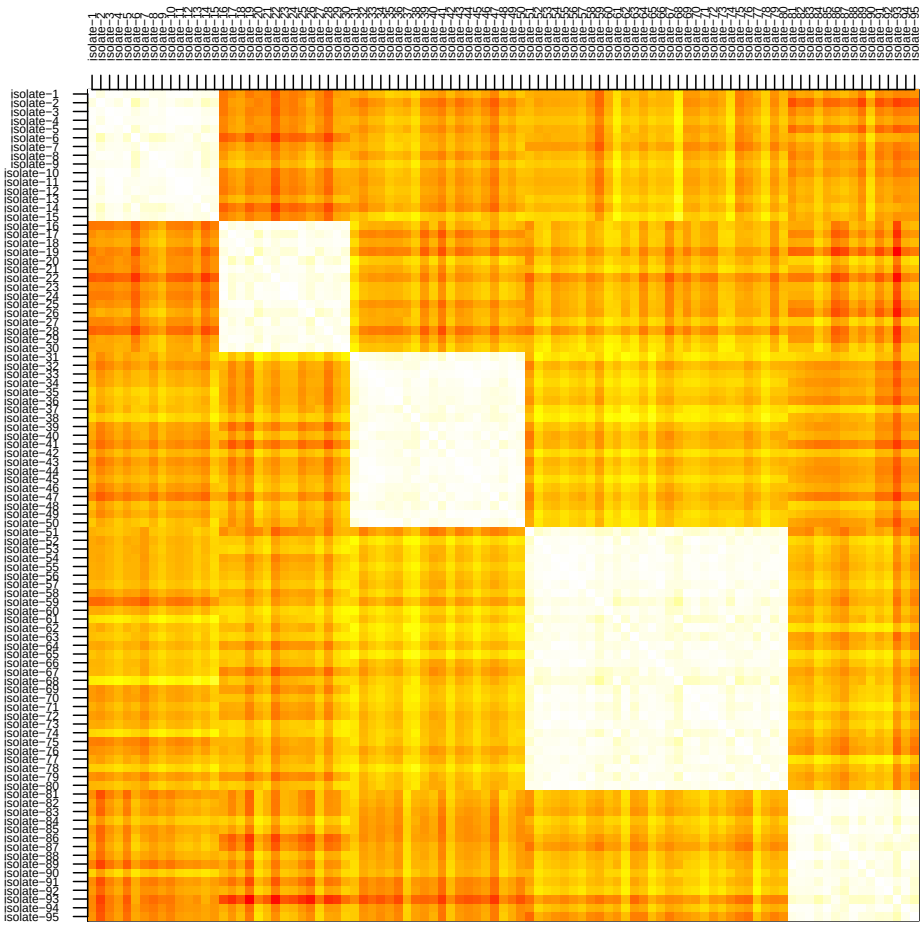
What can you say about the genetic relationships between the isolates? Are there indications of distinct lineages of bacteria? If so, how many lineages would you count?
For a more quantitative assessment of this clustering, we derive squared Euclidean distances between isolates (function `dist`) and use hierarchical clustering with complete linkage (`hclust`) to define tight clusters:

```
D <- dist(pca1$li[,1:4])^2
clust <- hclust(D, method="complete")
```

We can plot the distances stored in the `dist` object `D` in a heatmap with the follwing commands.

```
temp <- as.data.frame(as.matrix(D))
temp <- t(as.matrix(D))
temp <- temp[,ncol(temp):1]
```

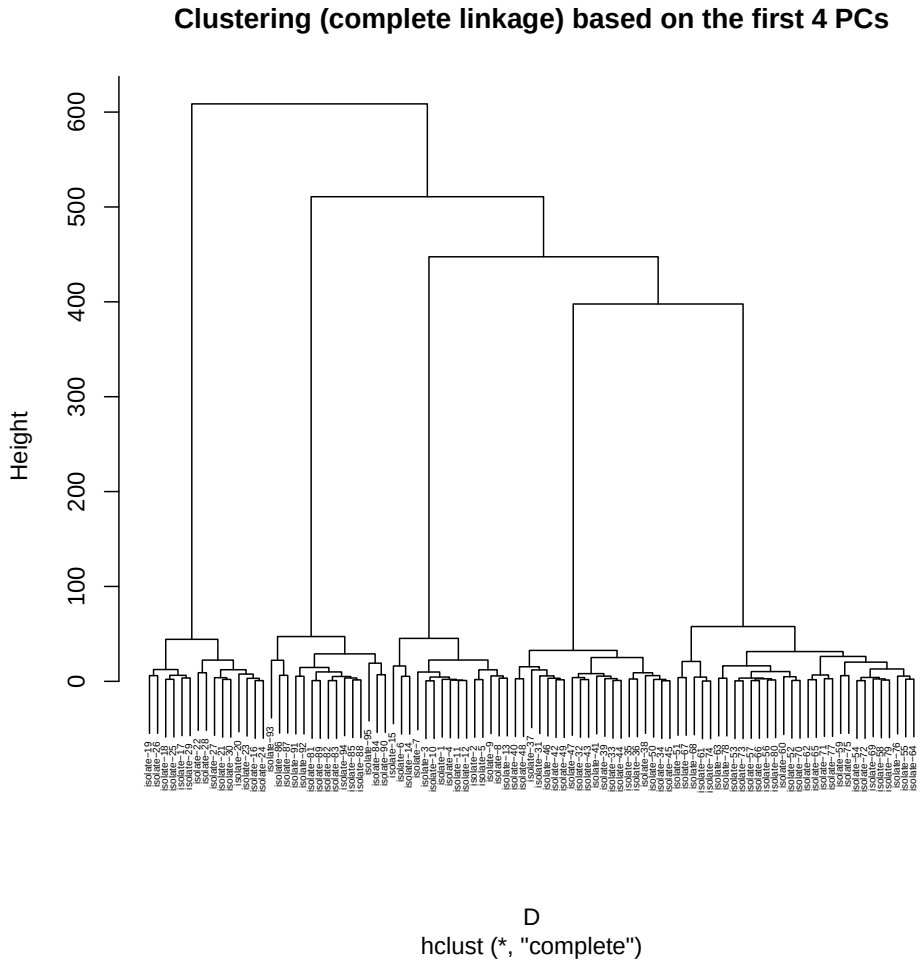
```
par(mar=c(1,5,5,1))
image(x=1:95, y=1:95, temp, col=rev(heat.colors(nlevels(as.factor(D)))),
      xaxt="n", yaxt="n",
      xlab="",ylab="")
axis(side=2, at=1:95, lab=rev(rownames(snps)), las=2, cex.axis=.46)
axis(side=3, at=1:95, lab=rownames(snps), las=2, cex.axis=.46)
```



```
par(mar=c(5.1,4.1,4.1,2.1))
```

Based on this distance matrix, what do you predict the topology of a complete-linkage hierarchical clustering tree will look like?

```
plot(clust, main="Clustering (complete linkage) based on the first 4 PCs", cex=.4)
```



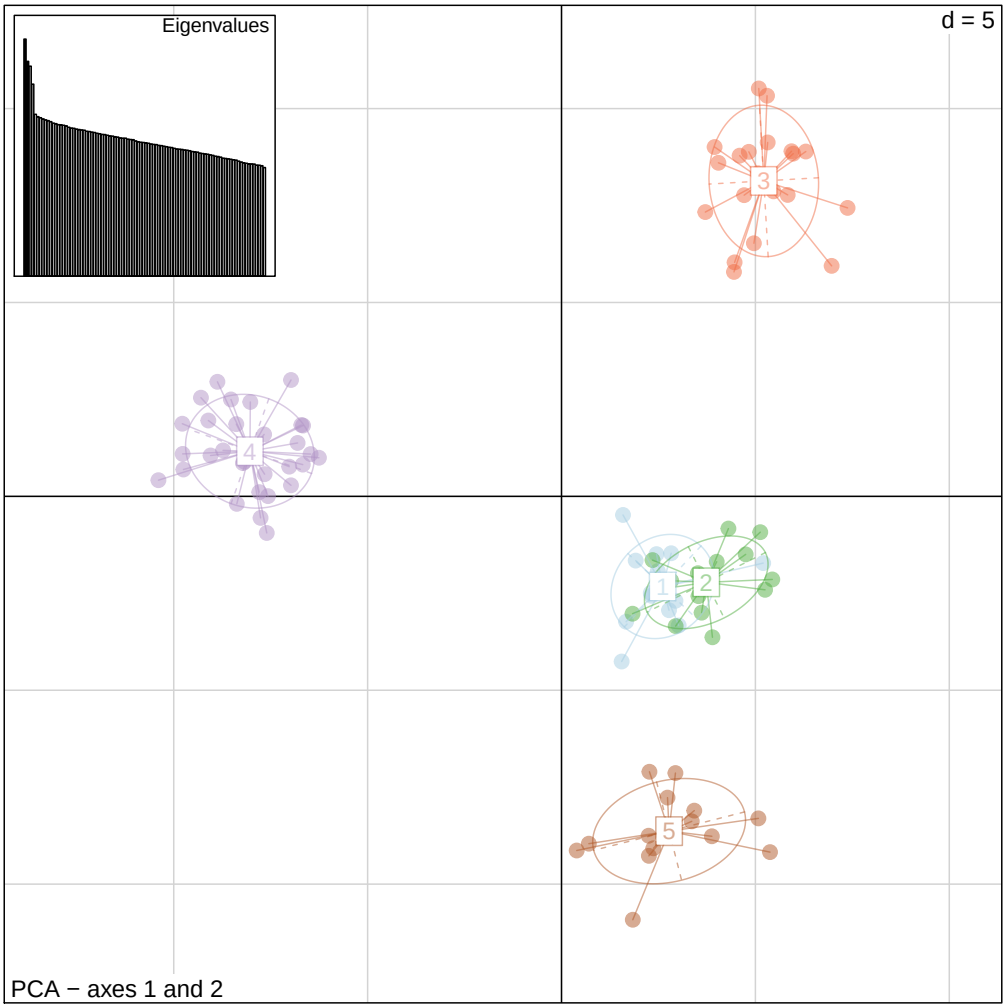
How many clusters are there in the data? How does it compare to what you would have assessed based on the first two PCs of PCA? *Bonus question*: considering that the original data are profile of binary SNPs, what does the 'height' represent in this dendrogram?

You can define clusters as before based on the dendrogram `clust`, using `cutree`:

```
pop <- factor(cutree(clust, k=5))
head(pop,20)

##  isolate-1  isolate-2  isolate-3  isolate-4  isolate-5  isolate-6
##          1          1          1          1          1          1
##  isolate-7  isolate-8  isolate-9 isolate-10 isolate-11 isolate-12
##          1          1          1          1          1          1
## isolate-13 isolate-14 isolate-15 isolate-16 isolate-17 isolate-18
##          1          1          1          2          2          2
## isolate-19 isolate-20
##          2          2
## Levels: 1 2 3 4 5
```

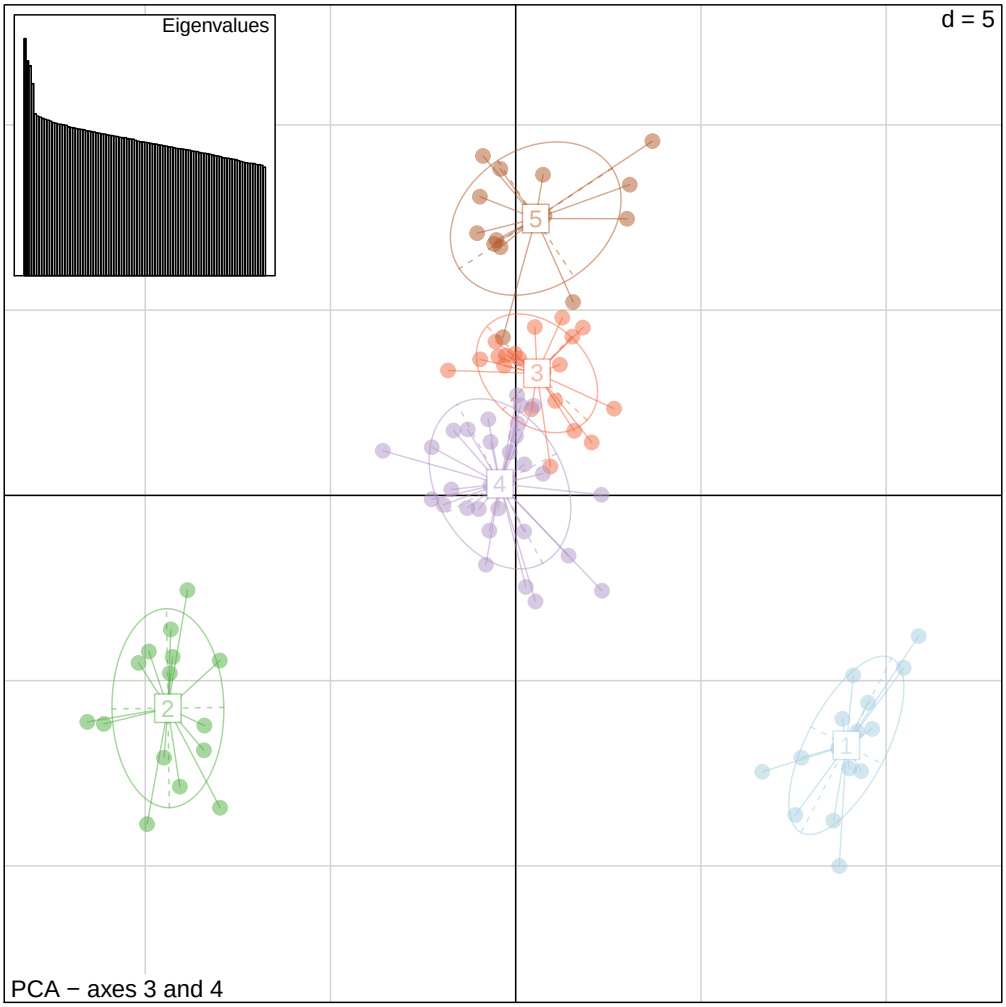
Now, we can represent these groups on top of the PCs using `s.class` (clusters are indicated by different colors and ellipses):

```
s.class(pca1$li, fac=pop, col=transp(funky(5)), cpoint=2,
        sub="PCA - axes 1 and 2")

add.scatter.eig(pca1$eig,4,1,2, ratio=.26, posi="topleft")
```



We do the same for PCs 3 and 4:

```
s.class(pca1$li, xax=3, yax=4, fac=pop, col=transp(funky(5)),
        cpoint=2, sub="PCA - axes 3 and 4")

add.scatter.eig(pca1$eig,4,1,2, ratio=.26, posi="topleft")
```



Are the clusters compatible with the results of the PCA? What is the meaning of the 3rd axis of the PCA? How many dimensions are needed to differentiate the 5 groups?

1.2 Identifying SNPs linked to antibiotic resistance

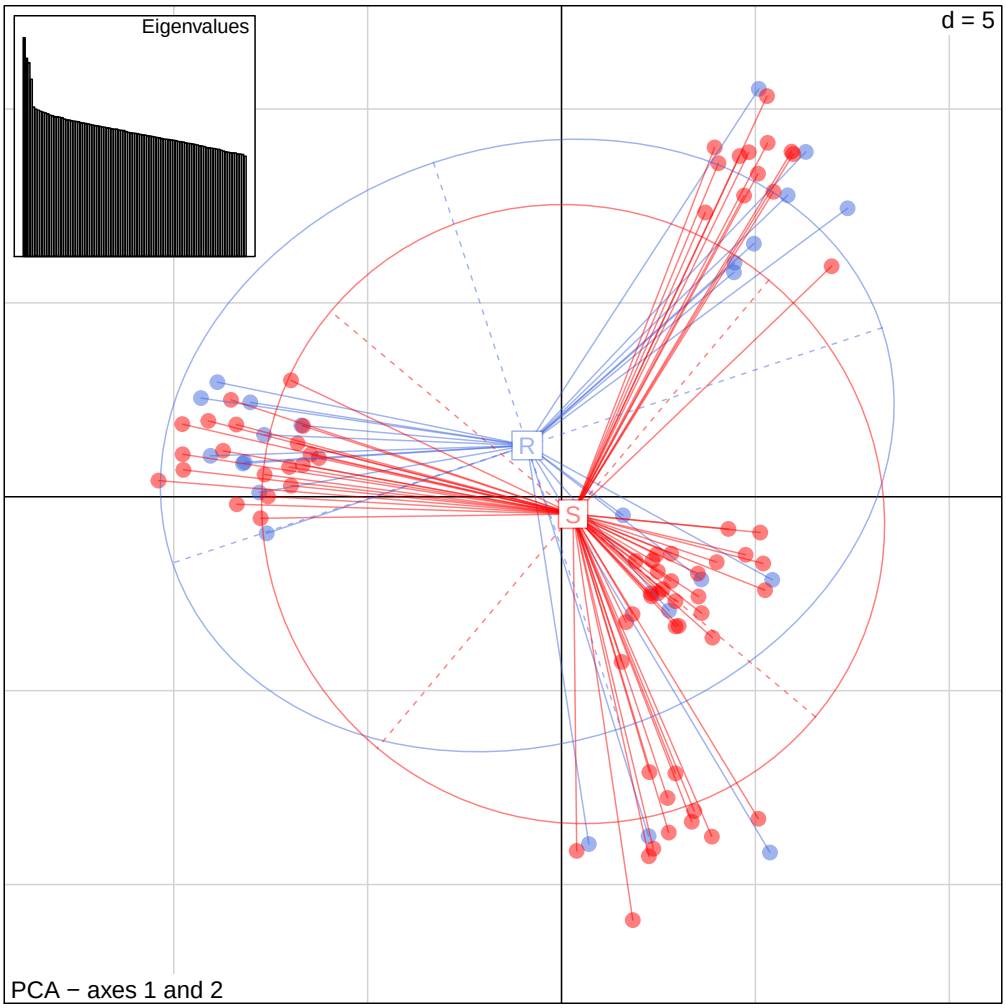
The data contained in `phen` indicate whether isolates are susceptible or resistant to a given antibiotic (`S`/`R`):

```
head(phen,10)

##  [1] R S S S S S S S S S
## Levels: R S
```

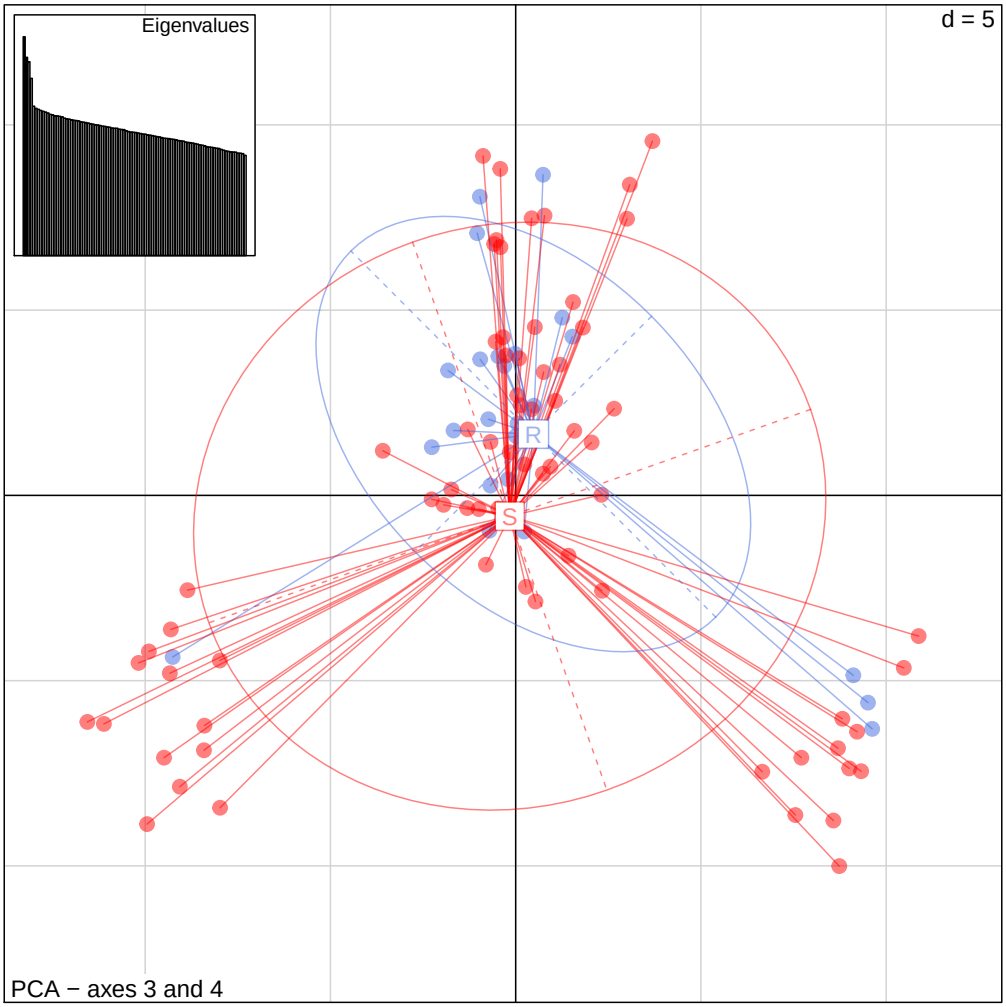
Our purpose in GWASis to attempt to identify the variables associated with the variation between specific phenotypic groups of interest (in our case, the SNPs that most contribute to the difference between the resistant and the susceptible bacteria). Let's first see if the axes of variation generated by PCA are able to discriminate between our two phenotypic groups.

As we have done with genetic clusters previously, we can represent these two groups on the PCs to assess whether antibiotic resistance correlates to some components of the genetic diversity.

```
s.class(pca1$li, fac=phen, col=transp(c("royalblue","red")), cpoint=2,
        sub="PCA - axes 1 and 2")

add.scatter.eig(pca1$eig,4,1,2, ratio=.24, posi="topleft")
```




```
s.class(pca1$li, xax=3, yax=4, fac=phen, col=transp(c("royalblue","red")),
        cpoint=2, sub="PCA - axes 3 and 4")

add.scatter.eig(pca1$eig,4,1,2, ratio=.24, posi="topleft")
```



This visual assessment can be completed by a standard Chi-square test to check if there is an association between genetic clusters and resistance:

```
table(phen, pop)

##     pop
## phen  1  2  3  4  5
##    R  3  1  7 10  3
##    S 12 14 13 20 12

chisq.test(table(phen, pop), simulate=TRUE)

##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  table(phen, pop)
## X-squared = 5.2267, df = NA, p-value = 0.2694
```

What do you conclude? Is antibiotic resistance correlated to the main genetic features of these isolates?

2 Association testing and feature selection

2.1 Univariate method

```
pval <- apply(snps, 2, function(e)
  fisher.test(table(factor(e, levels=c(0,1)), phen))$p.value)

min(pval)

## [1] 5.108221e-23

length(which(pval < 0.05))

## [1] 331
```

As we are carrying out one univariate test for every SNP in our dataset, we must now correct for multiple testing.

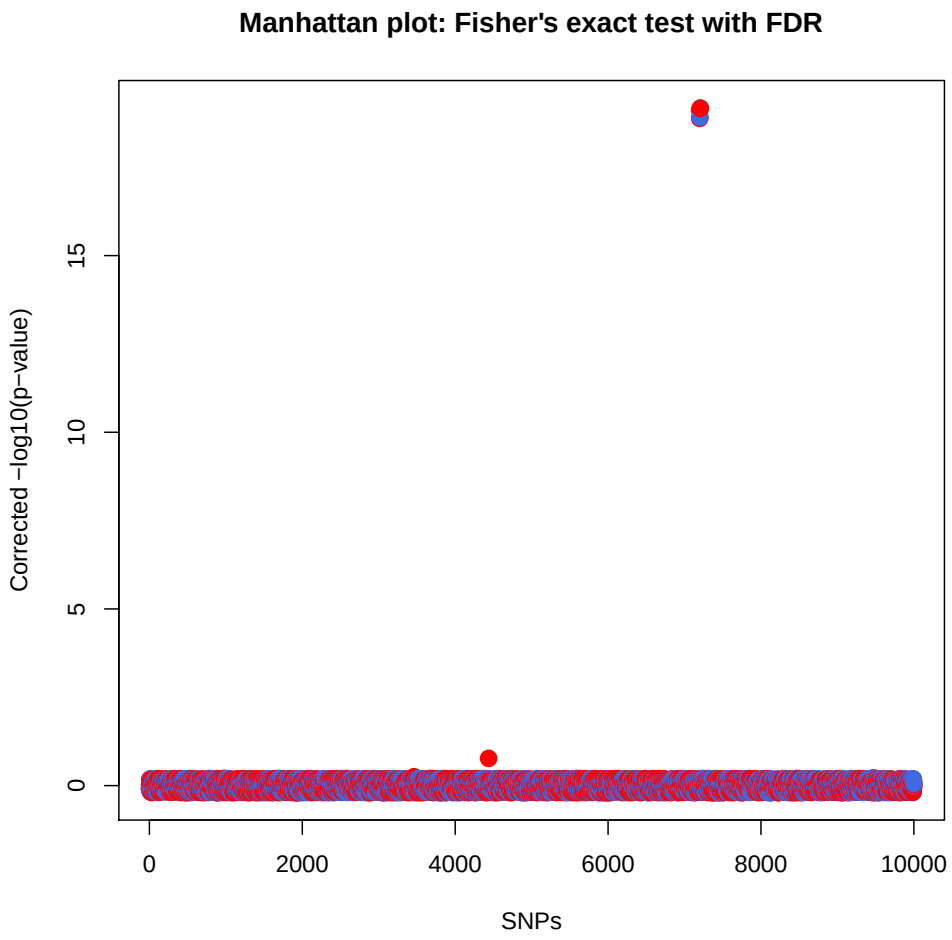
```
pval.corrected <- p.adjust(pval, method="fdr")
min(pval.corrected)

## [1] 1.021644e-19
```

We can now use these corrected p-values as our univariate selection criteria for feature selection.

```
snps.selected.univariate <- which(pval.corrected < 0.05)
n.snps.selected.univariate <- length(snps.selected.univariate)

n.snps.selected.univariate

## [1] 5

snps.selected.univariate

## 7197 7199 7202 7206 7207
## 7197 7199 7202 7206 7207
```

We can now visualise the results of this analysis with a Manhattan Plot, which is the type of plot most commonly used to represent the results of GWAS analyses.

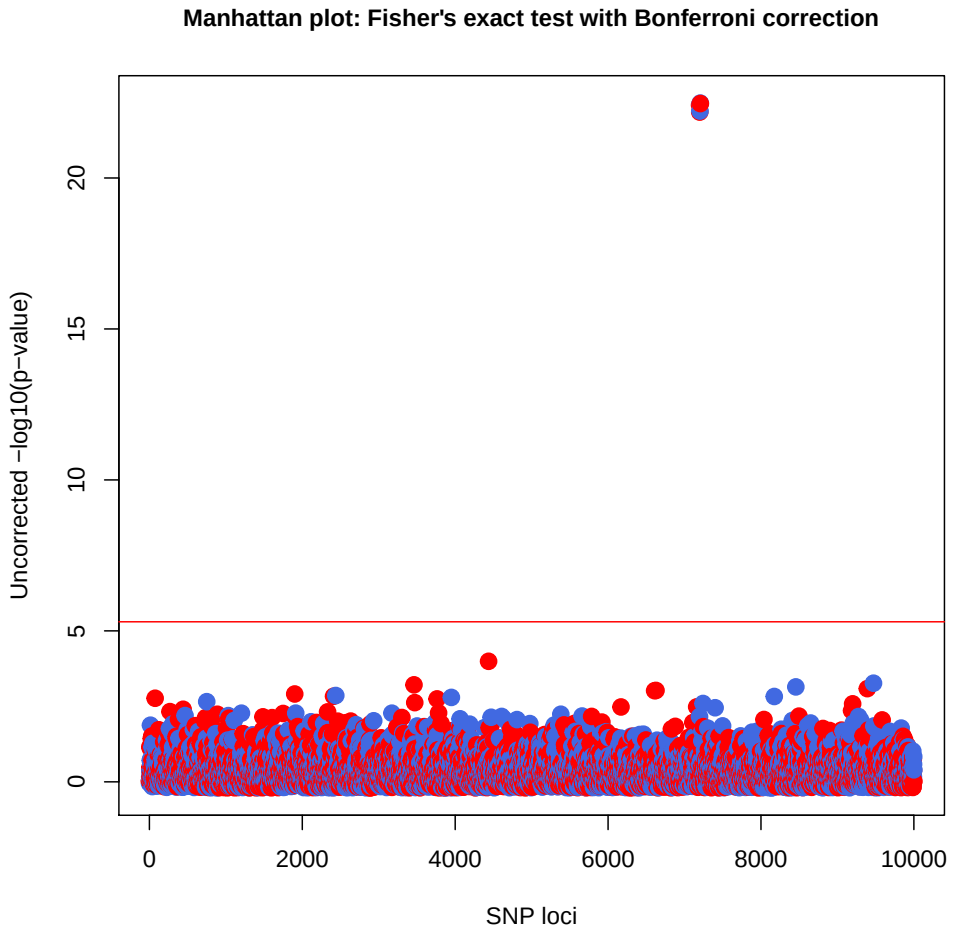
```
log.pval <- -log10(pval.corrected)
set.seed(1)
log.pval <- jitter(log.pval, amount=0.2)

plot(log.pval,
     col = c("red", "royalblue"),
     pch = 19,
     cex = 1.5,
     main="Manhattan plot: Fisher's exact test with FDR",
     xlab="SNPs", ylab="Corrected -log10(p-value)")
```



For the purposes of comparison, let's also generate a Manhattan Plot with our original uncorrected p-values, this time representing our results with the Bonferroni correction for multiple testing.

```
log.pval <- -log10(pval)
set.seed(1)
log.pval <- jitter(log.pval, amount=0.2)
```

```r
plot(log.pval,
     col = c("red", "royalblue"),
     pch = 19,
     cex = 1.5,
     main="Manhattan plot: Fisher's exact test with Bonferroni correction",
     xlab="SNP loci", ylab="Uncorrected -log10(p-value)",
     cex.main=1)

bonferroni <- -log10(0.05 / ncol(snps))
abline(h=bonferroni, col = "red")
```



2.2 Multivariate methods

2.2.1 LASSO

To test for association with the LASSO penalized regression method, we use the function `cv.glmnet` from package `glmnet`. This will allow us to generate a vector of coefficients for each variable, the majority of which will be shrunk to zero in LASSO's penalization step.

```
LASSO <- cv.glmnet(snps, phen, family="binomial", lambda.min.ratio=0.01, alpha=1)
beta <- as.vector(t(coef(LASSO, s="lambda.min")))
```

What does the 'cv' in the `cv.glmnet` function stand for? And why have we set `lambda.min.ratio` to 0.01?
We can now carry out feature selection. Recall that in LASSO penalized regression, the tuning parameter `lambda` specifies the extent of the penalty on the L1 norm, and hence it also determines the shrinkage of the coefficients for each variable towards zero. In feature selection by the LASSO method, the variables selected are those with non-zero coefficients when `lambda` is at its optimal minimum.

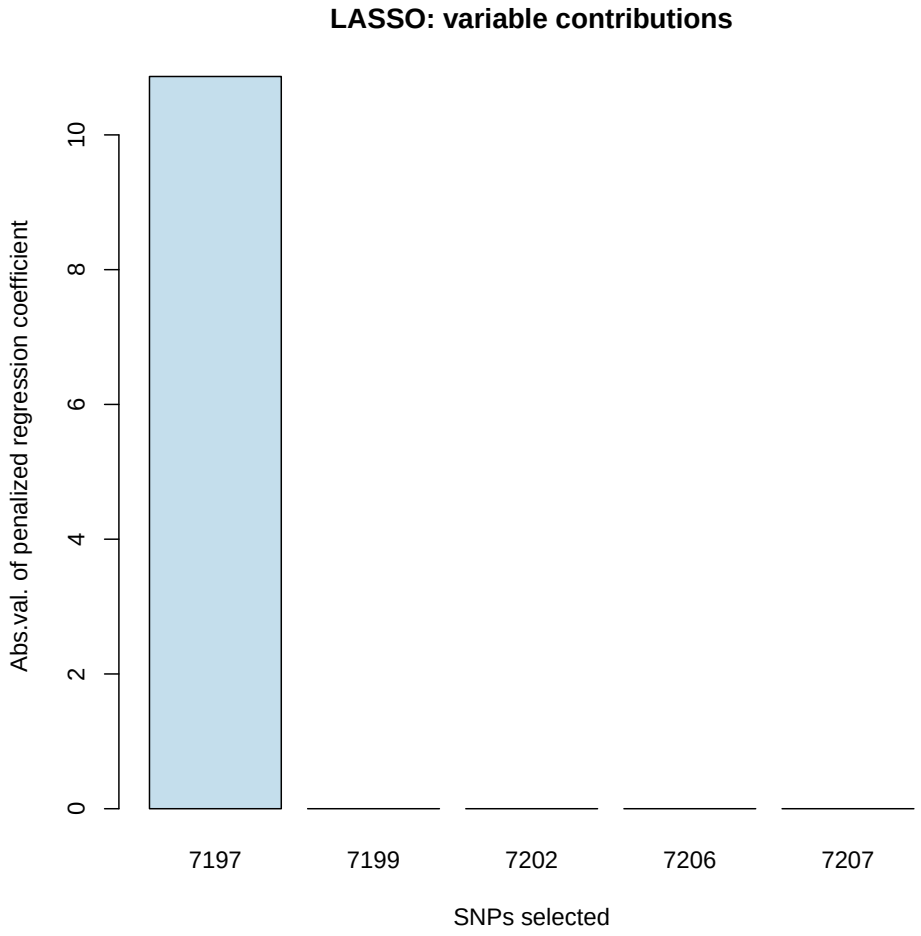
```
selected <- which(beta[-1] !=0)
n.snps.selected.LASSO <- as.integer(length(selected))
snps.selected.LASSO <- as.vector(selected)

n.snps.selected.LASSO

## [1] 5

snps.selected.LASSO

## [1] 7197 7199 7202 7206 7207
```

LASSO has selected the same 5 SNPs as the univariate Fisher's exact test.
Let's take a closer look at the coefficients assigned to each of the SNPs selected by LASSO.

```
coefs.LASSO <- beta[-1][snps.selected.LASSO]
names(coefs.LASSO) <- as.character(snps.selected.LASSO)
coefs.LASSO

##          7197          7199          7202          7206          7207
## -1.086688e+01 -6.194127e-14 -3.355152e-14 -1.290443e-15 -1.032355e-14
```

Note that the coefficient for the first SNP is substantially larger than the coefficients for the other 4 SNPs selected by LASSO, which are very near zero.
We may be able to better examine the differences between these coefficients by plotting them with a simple barplot.

```
myCol <- funky(1)
barplot(abs(coefs.LASSO),
        col=c(transp(myCol[1], 0.66)),
        xlab="SNPs selected",
        ylab="Abs.val. of penalized regression coefficient",
        main="LASSO: variable contributions")
```

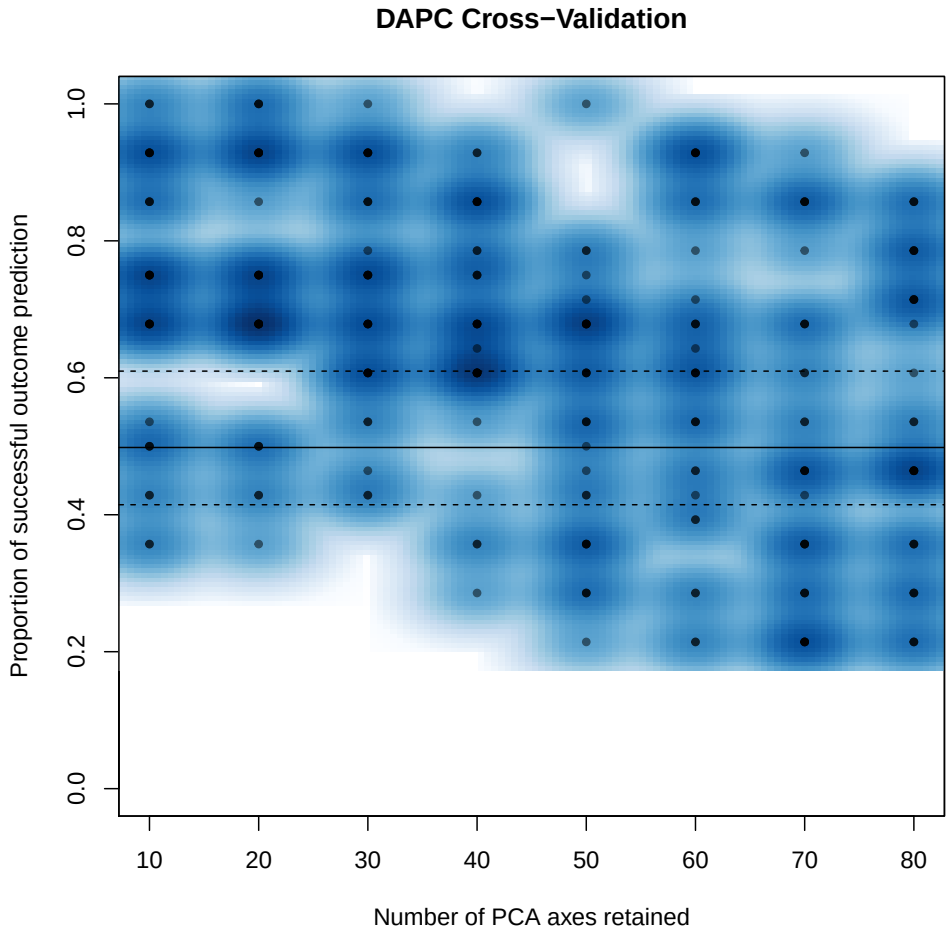


What is your interpretation of these coefficients? What do they tell you about the relative importance of each of the SNPs selected?

2.2.2 DAPC-based feature selection

We begin the DAPC approach to feature selection by running cross-validation to help us select the number of PCs of PCA to retain that will maximize our ability to discriminate between our two phenotypic groups.

```
set.seed(1)
xval <- xvalDapc(snps, phen) ## this may take a moment...

## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```




```
## NULL
```

Based on the plot generated by xvalDapc, considering that the horizontal lines are the mean (solid line) and 95% Confidence interval (dashed lines) for random chance, do you trust that cross-validation has been successful in selecting a model that is useful in assigning individuals to the correct phenotypic group?
Let's take a look at the object `xval` containing the results of cross-validation:

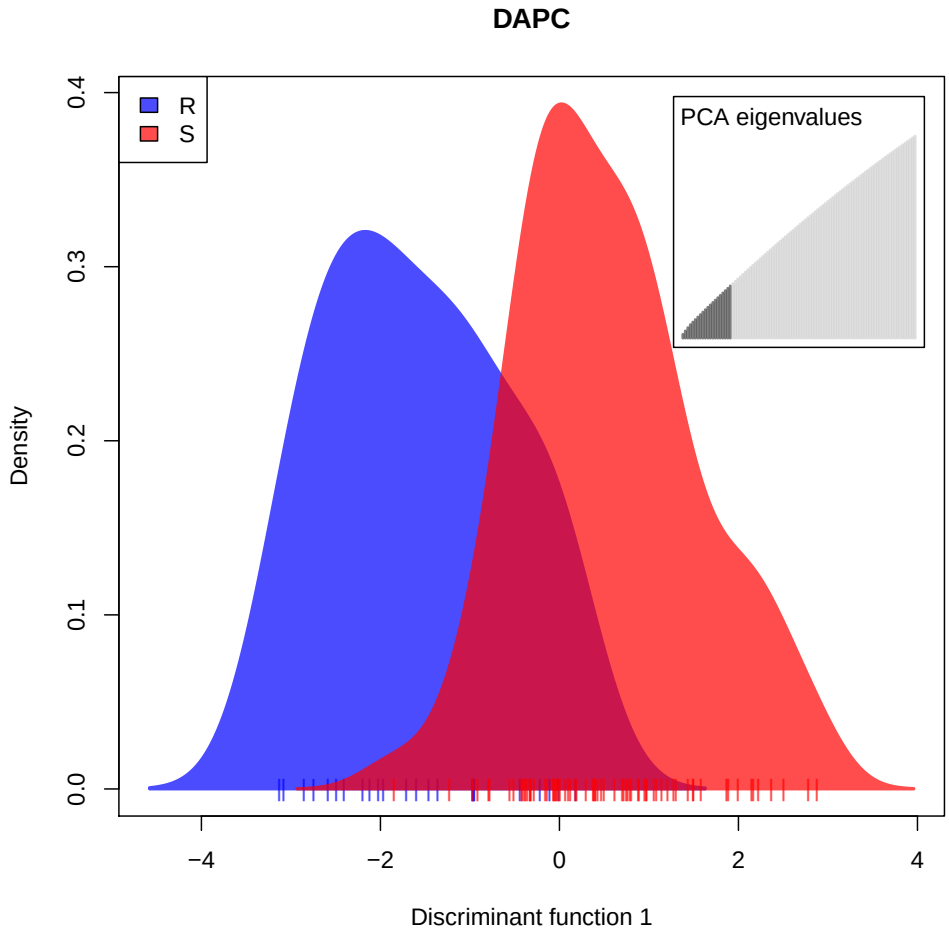
```
xval[2:6]

## $`Median and Confidence Interval for Random Chance`
##      2.5%       50%     97.5%
## 0.4146127 0.4982394 0.6097418
##
## $`Mean Successful Assignment by Number of PCs of PCA`
##        10        20        30        40        50        60        70
## 0.7130952 0.7357143 0.7190476 0.6738095 0.5488095 0.6190476 0.5035714
##        80
```

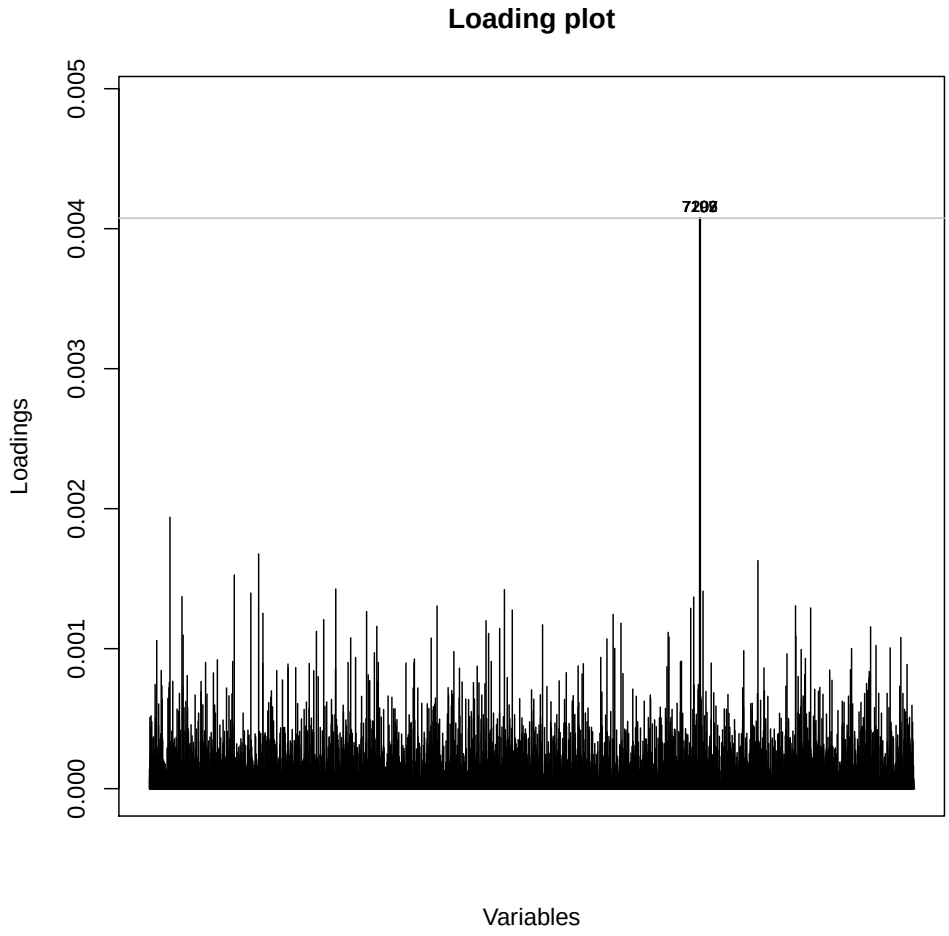
```
## 0.5428571
##
## $`Number of PCs Achieving Highest Mean Success`
## [1] "20"
##
## $`Root Mean Squared Error by Number of PCs of PCA`
##        10        20        30        40        50        60        70
## 0.3428695 0.3198373 0.3216930 0.3649154 0.4868596 0.4400255 0.5455058
##        80
## 0.5034743
##
## $`Number of PCs Achieving Lowest MSE`
## [1] "20"
```

Does this help you answer the previous question?
Based on the results of cross-validation, how many PCs of PCA should you retain in your DAPC? For the purpose of this analysis (hint: and for all Case-Control GWAS analyses), how many DA axes should you retain? Why?
Create a DAPC object called `dapc1` by running a DAPC with the `n.pca` and `n.da` you wish to retain:

We can now use the function `snpzip` to perform feature selection and visualise our results.

```
set.seed(1)
result <- snpzip(snps, dapc1,
                 method="single", xval.plot = FALSE,
                 plot = TRUE, loading.plot = TRUE)
```


DAPC
R
S
PCA eigenvalues
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
Discriminant function 1




```
par(ask=FALSE)

snps.selected.dapc <- result$FS[[2]]
n.snps.selected.dapc <- length(snps.selected.dapc)

n.snps.selected.dapc

## [1] 5

snps.selected.dapc

## [1] 7197 7199 7202 7206 7207
```

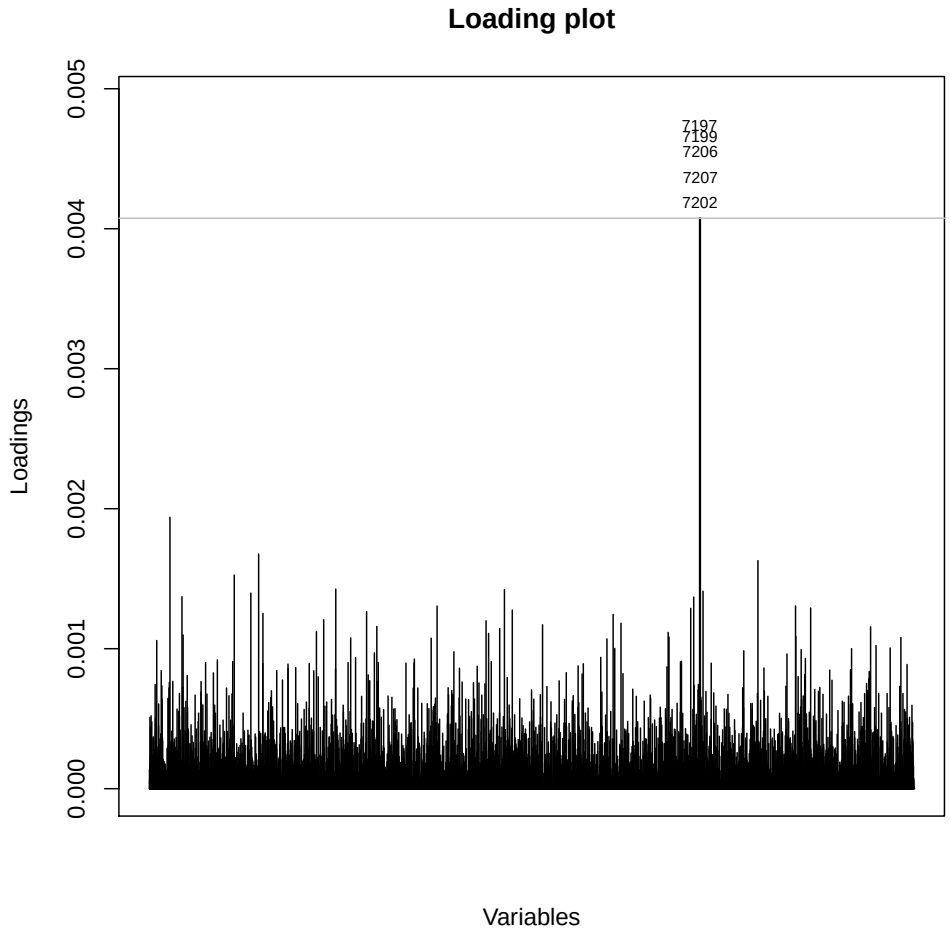
Again, we have managed to select the same 5 SNPs!

2.2.3 A little more on DAPC-based feature selection

As an aside, you may have noticed that the snps selected are so closely clustered together both in the alignment and on the y-axis (according to their similar loadings) that we may

want to re-generate the loadingplot separately to modify the placement of the labels. We can do this as so:

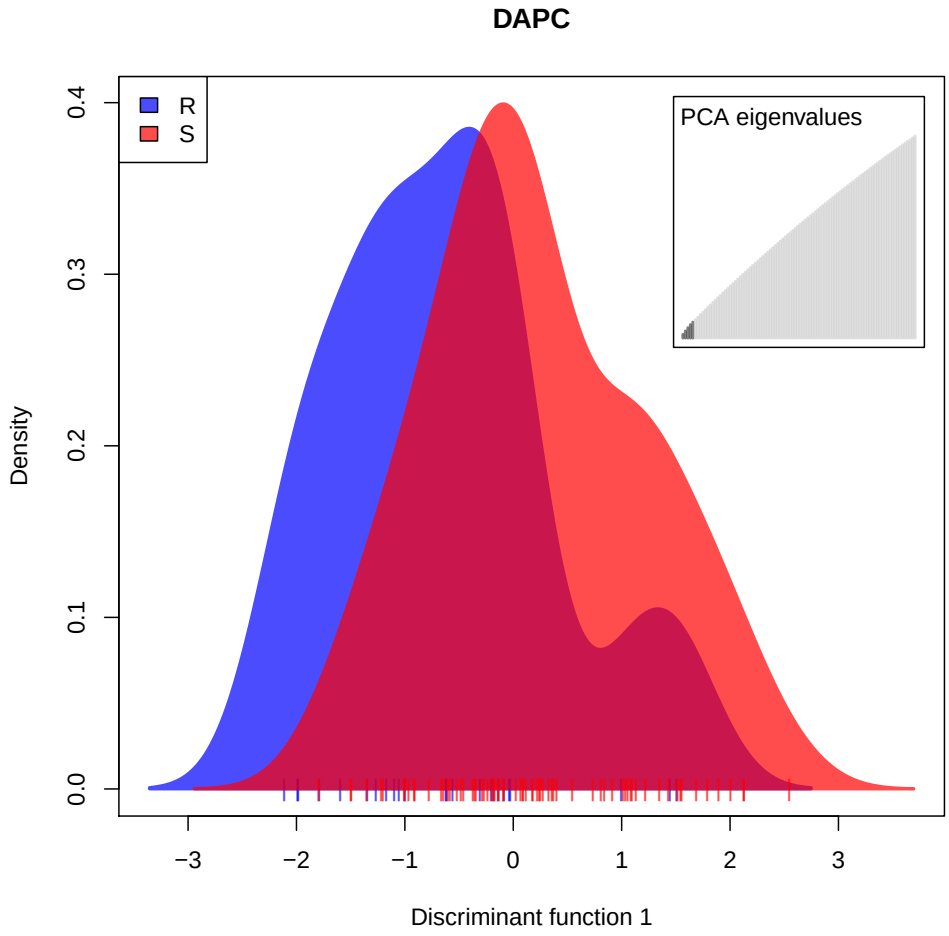
```
min.var.selected <- abs(dapc1$var.contr[snps.selected.dapc]
                        [(which.min(dapc1$var.contr[snps.selected.dapc]))])-0.000001
set.seed(41)
l.plot <- loadingplot(dapc1$var.contr[,1],
                      threshold=c(min.var.selected), lab.jitter=7.5)
```

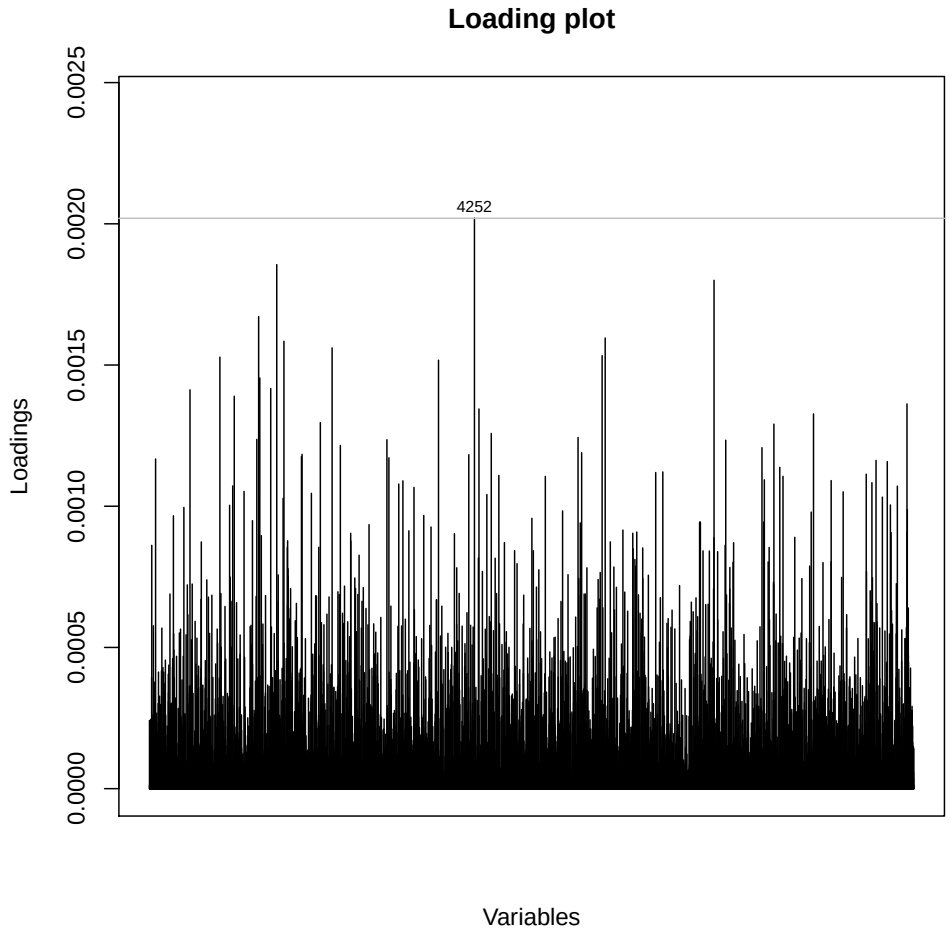


Not perfect, but much better!

Let's also examine what would have resulted from DAPC-based feature selection if: 1) We had only retained the first 5 PCs of PCA:

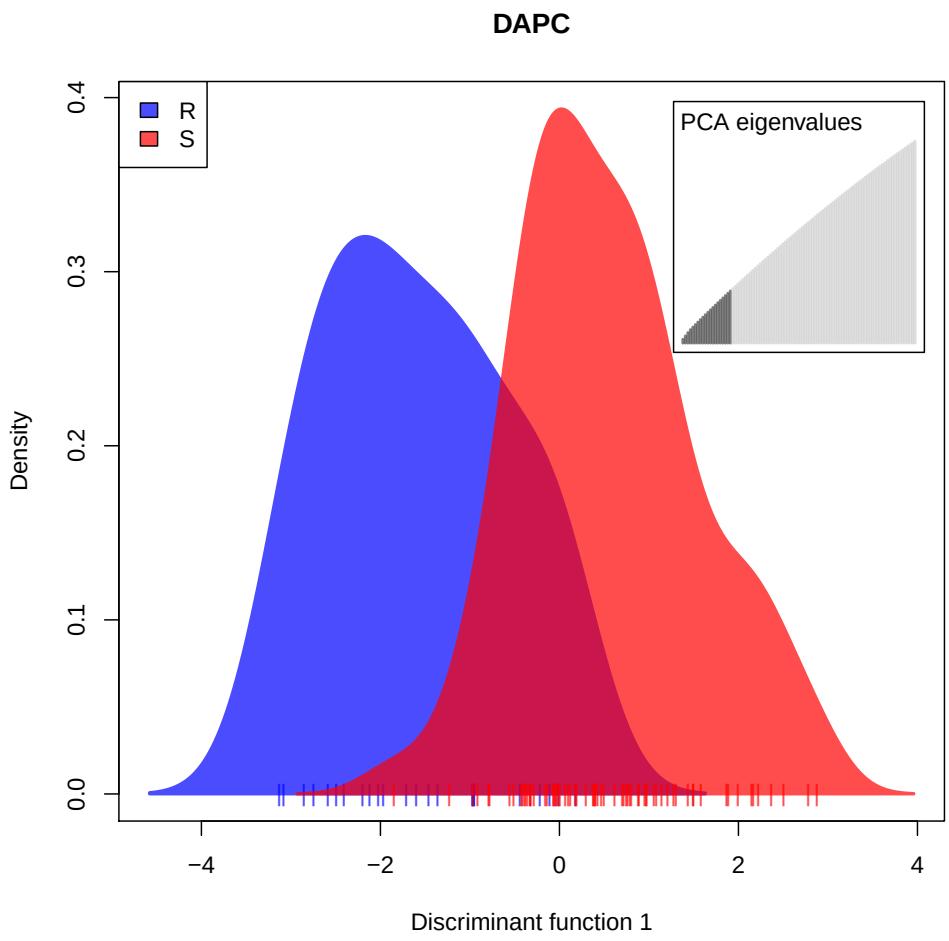
```
result5 <- snpzip(snps, dapc(snps, phen, n.da=1, n.pca=5),
                  method="single", xval.plot = FALSE,
                  plot = TRUE, loading.plot = TRUE)
```

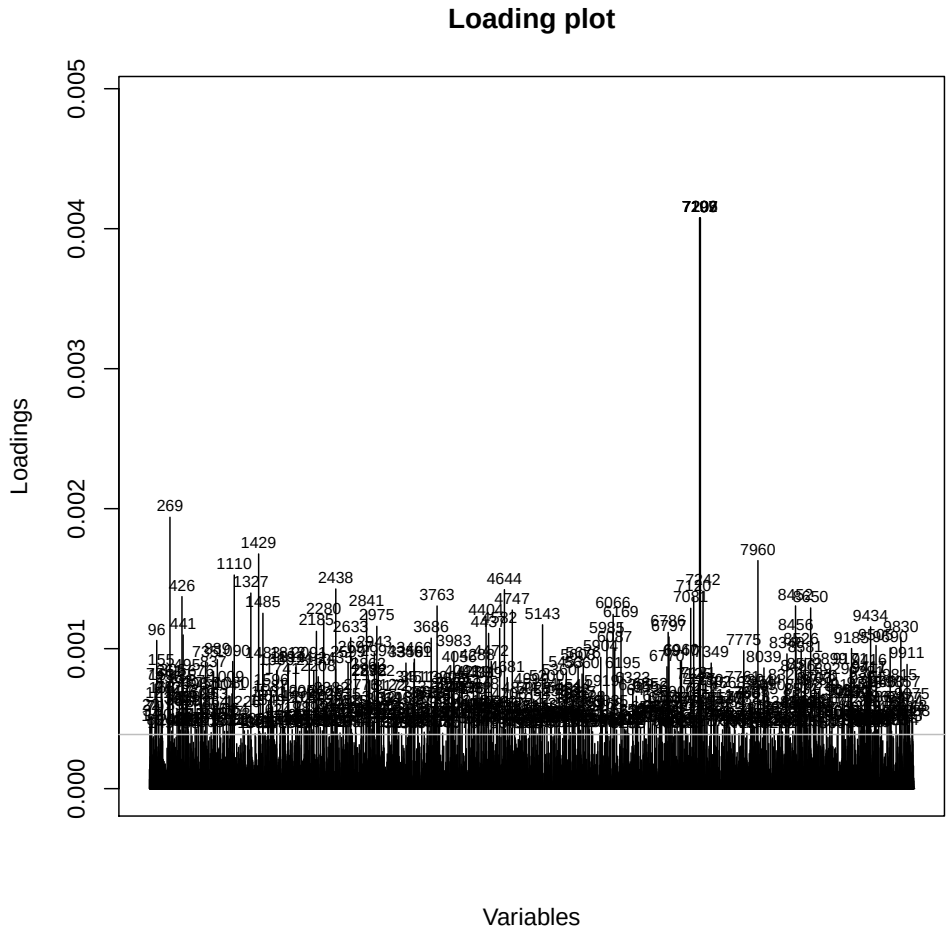

DAPC
R
S
PCA eigenvalues
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
Discriminant function 1



2) We had selected a different hierarchical clustering method for performing feature selection:

```
result.ward <- suppressWarnings(snpzip(snps, dapc1,
                    method="ward", xval.plot = FALSE,
                    plot = TRUE, loading.plot = TRUE))

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```


DAPC
R
S
PCA eigenvalues
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
Discriminant function 1



3 Correcting for population stratification

As we have already run a PCA to visualise the population structure in our first assessment of genetic diversity, we have already generated the object `pca1` that we will use to correct our SNPs matrix for population stratification. We do this by regressing along the axes of PCA required to visually diagnose the presence of our population clusters. In our case, this means we will regress along the first 4 PCs of PCA, which were needed to separate the 5 groups we identified.

```
snps.corrected <- apply(snps, 2, function(e)
  residuals(lm(e~pca1$li[,1]+pca1$li[,2]+pca1$li[,3]+pca1$li[,4]))) # may take a minute
```

Let's take a look at our corrected SNPs matrix.

```
dim(snps.corrected)

## [1]    95 10000
```

```
snps.corrected[1:10,1:10]

##                     1          2           3          4          5
## isolate-1  -0.2094808 -0.3485827 -0.06695461  0.1316558 -0.3917387
## isolate-2  -0.2319002 -0.3291088 -0.05720198  0.1237753 -0.3653154
## isolate-3  -0.2063008 -0.3485991 -0.06582533  0.1298276  0.6262579
## isolate-4  -0.1970682  0.6412060  0.93452210  0.1297420 -0.3854874
## isolate-5   0.7728263 -0.3352291 -0.05762730  0.1245033 -0.3780183
## isolate-6   0.8272011  0.6290381 -0.07689193  0.1372498  0.6072959
## isolate-7   0.7646633  0.6636476 -0.06132759  0.1296980  0.5936000
## isolate-8  -0.2236320  0.6578480 -0.05729049 -0.8754255 -0.3735933
## isolate-9  -0.2256344  0.6540961 -0.05196910  0.1214080  0.6049000
## isolate-10 -0.2111378 -0.3465306 -0.06255007  0.1277205  0.6185463
##                      6          7          8           9          10
## isolate-1   0.084415426  0.4540041 -0.4679964 -0.03934675  0.09755296
## isolate-2  -0.038729468  0.4386883 -0.4309456 -0.01333862  0.09546631
## isolate-3   0.046922304  0.4385147  0.5420833 -0.03620780  0.10738401
## isolate-4   0.104115777  0.4475439 -0.4692778 -0.04520639 -0.88562099
## isolate-5   0.007423331 -0.5511860  0.5586868 -0.01996659  0.09629192
## isolate-6   0.138289522  0.4436872 -0.4987127 -0.06728938  0.10651813
## isolate-7  -0.901639560 -0.5256087  0.5435540 -0.02454683  0.08954542
## isolate-8   0.046535326 -0.5538419 -0.4391931 -0.02178992  0.11585796
## isolate-9   0.095098715 -0.5328484 -0.4456315 -0.02360528  0.11303332
## isolate-10  0.049522790  0.4470624 -0.4568193 -0.03345280  0.10245325

range(snps.corrected)

## [1] -1.076927  1.082378
```

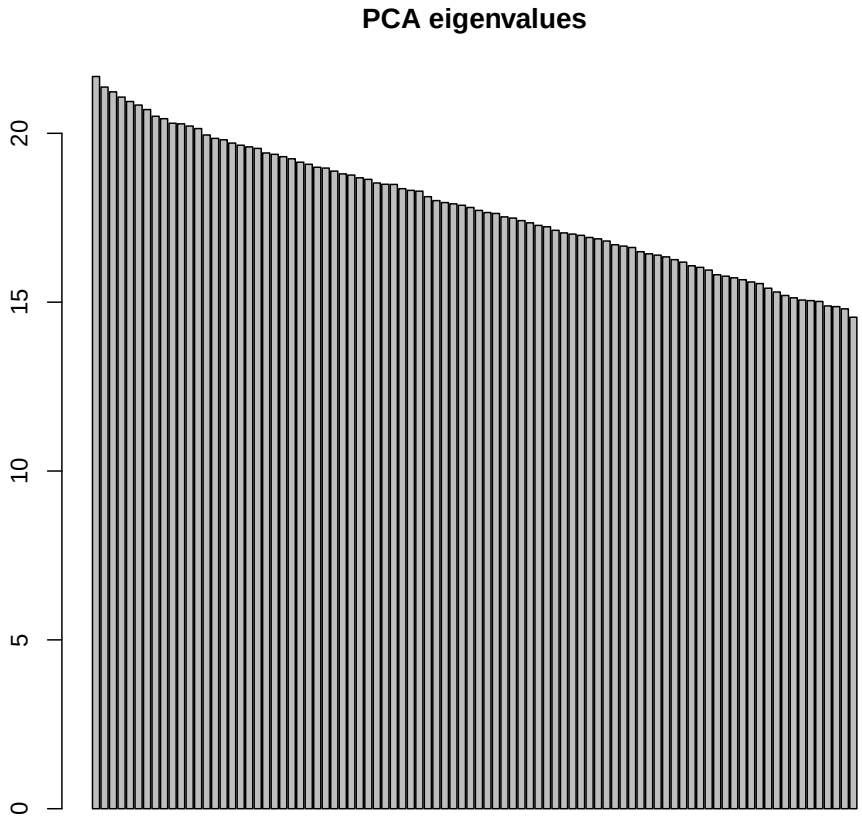
What kind of variable does our corrected SNPs matrix contain?

We can now run a second PCA analysis with the corrected SNPs matrix to visually assess whether our correction for population stratification has been successful:

```
pca2 <- dudi.pca(snps.corrected, scale=FALSE, scannf=FALSE, nf=4)
```

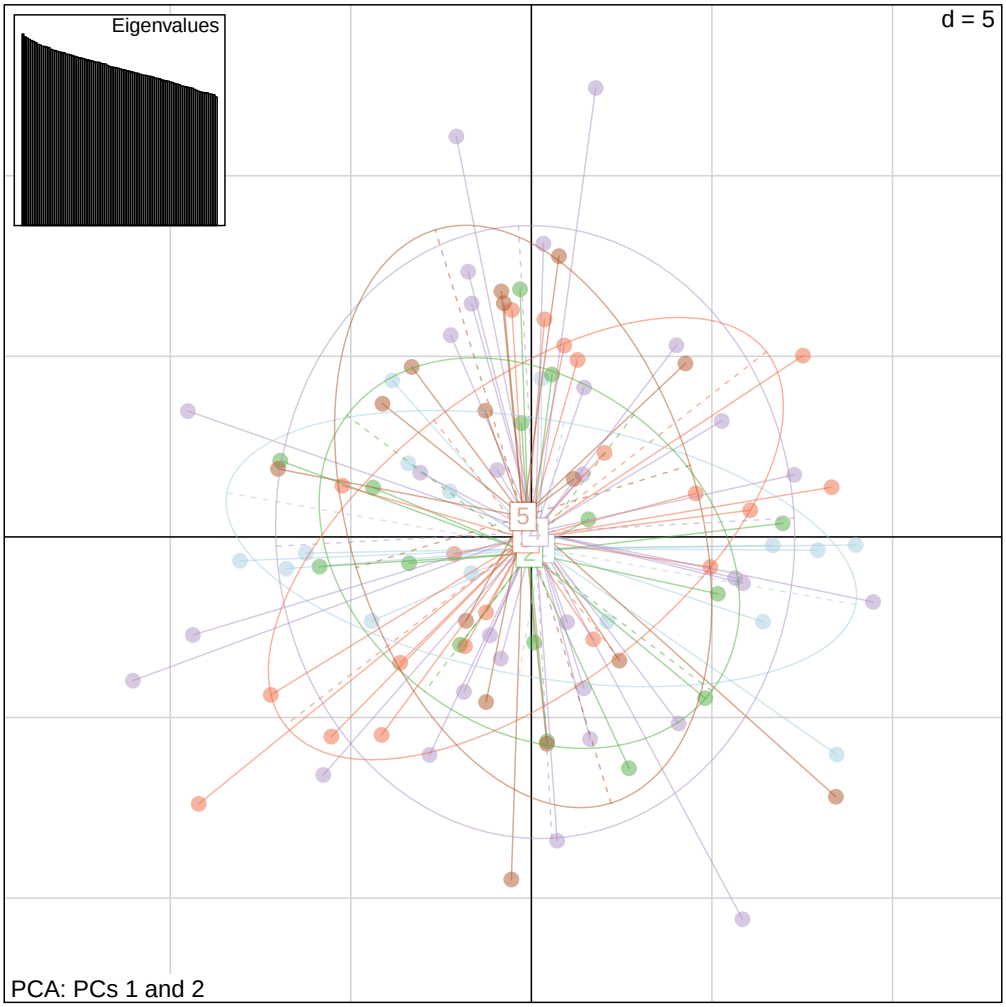
First we can take a look at the eigenvalues for `pca2`.

```
barplot(pca2$eig, main="PCA eigenvalues")
```



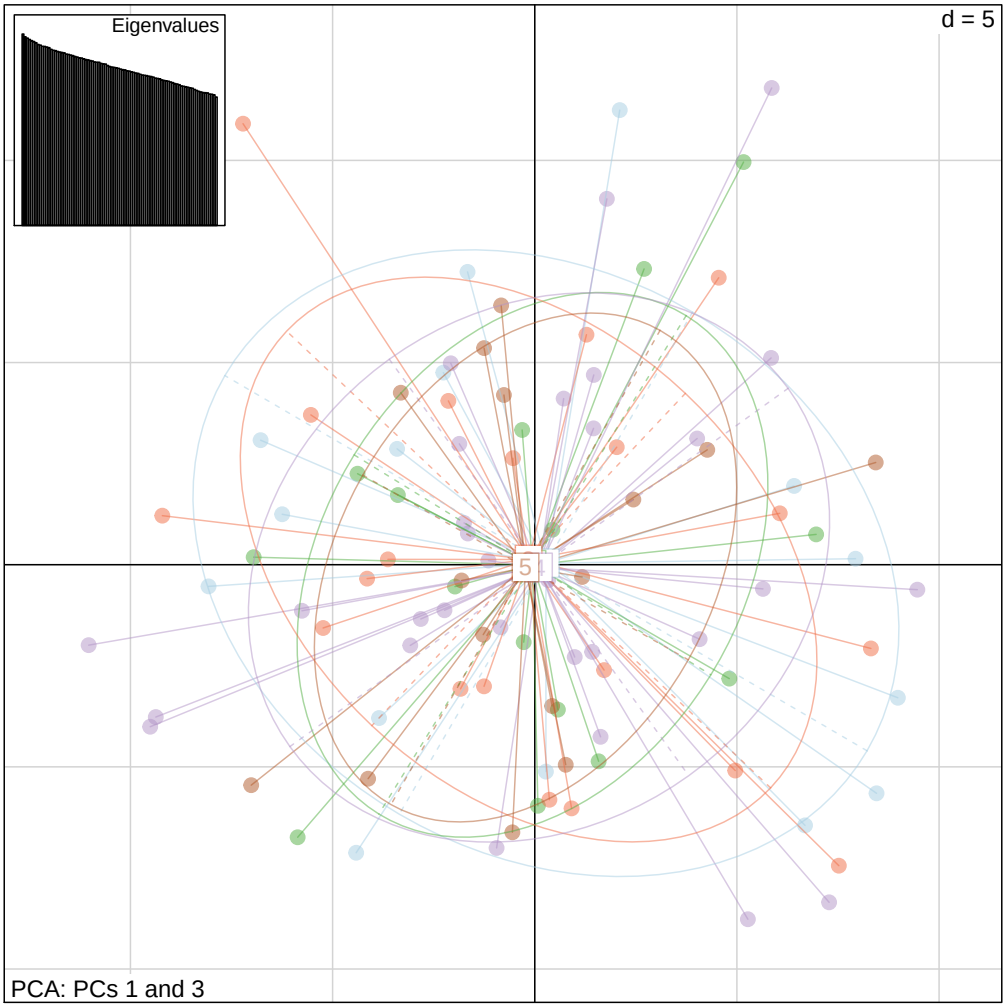
What do you notice about these eigenvalues? What can you infer from this?
Next, we can visualise our original population substructure in this new PCA space.

```
s.class(pca2$li, fac=pop, col=transp(funky(5)), cpoint=2,
        sub="PCA: PCs 1 and 2")
add.scatter.eig(pca2$eig,4,1,2, ratio=.21, posi="topleft")
```



Our population clusters are no longer separated along axes 1 and 2!
What about axes 3 and 4?

```
s.class(pca2$li, xax=1, yax=3, fac=pop, col=transp(funky(5)), cpoint=2,
        sub="PCA: PCs 1 and 3")
add.scatter.eig(pca2$eig,4,1,2, ratio=.21, posi="topleft")
```



It seems our correction for population stratification has successfully eliminated the potential confounding population structure!

4 Association testing and feature selection after correcting

We can now re-run the three methods to test for associations between our corrected SNPs and the antibiotic resistance phenotype.

4.1 Univariate method

While both multivariate methods for association testing and feature selection will be directly repeatable in application to our newly corrected SNPs matrix, our univariate approach is no longer valid!
Why do you think Fisher's exact test is no longer appropriate here?
Instead of Fisher's exact test we will use an alternative univariate approach that consists of two stages. First, we generate a simple linear model between each column of our corrected SNPs matrix and our phenotypic trait. Second, we run an analysis of variance (ANOVA) on

each model generated, specifying a Chi-squared test of association. From this,we can retrieve a p-value for the significance of association between each corrected SNP and the resistance phenotype.

```
pval2 <- numeric(0)
for(i in 1:ncol(snps.corrected)){
foo <- suppressWarnings(glm(phen ~ snps.corrected[,i], family="binomial"))
ANOVA <- anova(foo, test="Chisq")
pval2[i] <- ANOVA$"Pr(>Chi)"[2]
}

min(pval2, na.rm=TRUE)

## [1] 3.657423e-25
```

We can then correct for multiple testing as we did in our initial univariate analysis, using the False Discovery Rate.

```
pval.corrected2 <- p.adjust(pval2, method="fdr")
snps.selected.univariate2 <- which(pval.corrected2 < 0.05)
n.snps.selected.univariate2 <- length(snps.selected.univariate2)

n.snps.selected.univariate2

## [1] 5

snps.selected.univariate2

## [1] 7197 7199 7202 7206 7207
```

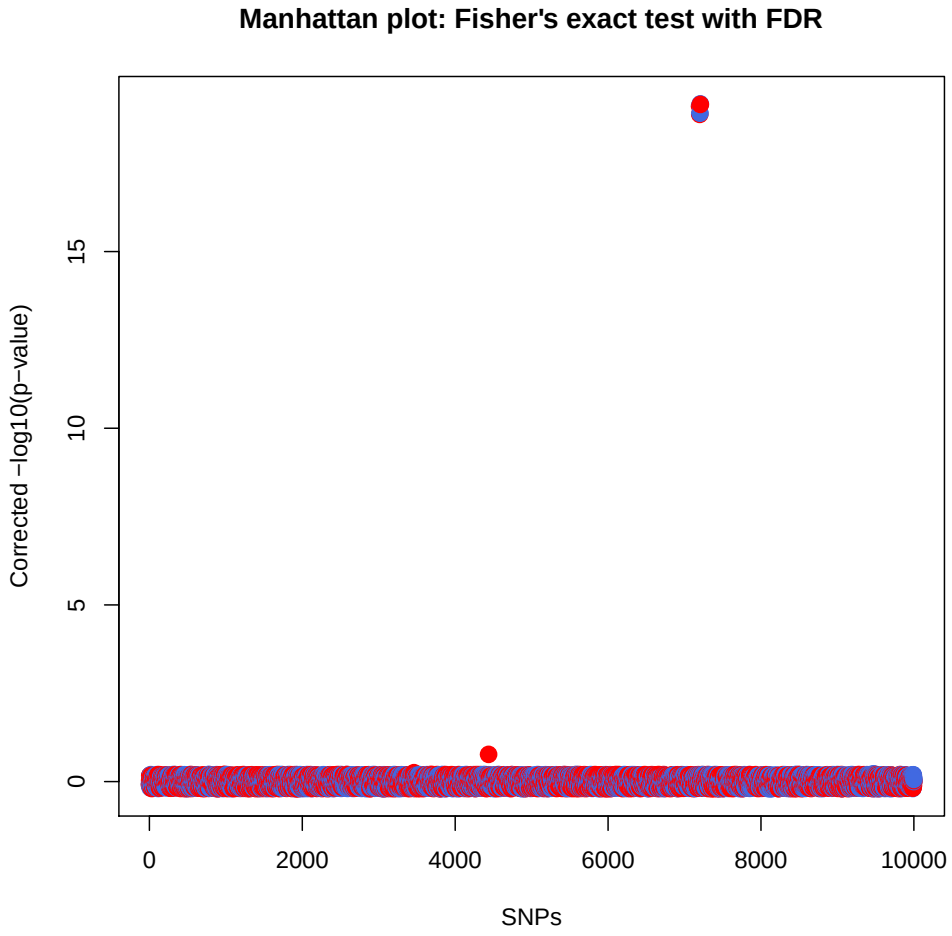
Once again, we have selected the same 5 SNPs as the set of genetic variables associated with the resistance phenotype!
However, a comparison of the FDR-corrected p-values from before and after the correction for population stratification will reveal that the p-values *after* correction are more significant than those from before.

```
pval.corrected[snps.selected.univariate]

##         7197         7199         7202         7206         7207
## 1.021644e-19 1.021644e-19 1.021644e-19 1.021644e-19 1.021644e-19

pval.corrected2[snps.selected.univariate2]

## [1] 7.240967e-22 7.240967e-22 7.240967e-22 7.240967e-22 7.240967e-22
```

```
pval.combined <- rbind(pval.corrected[snps.selected.univariate],
                       pval.corrected2[snps.selected.univariate2] )
rownames(pval.combined) <- c("Before", "After")
pval.combined

##                7197         7199         7202         7206         7207
## Before 1.021644e-19 1.021644e-19 1.021644e-19 1.021644e-19 1.021644e-19
## After  7.240967e-22 7.240967e-22 7.240967e-22 7.240967e-22 7.240967e-22
```

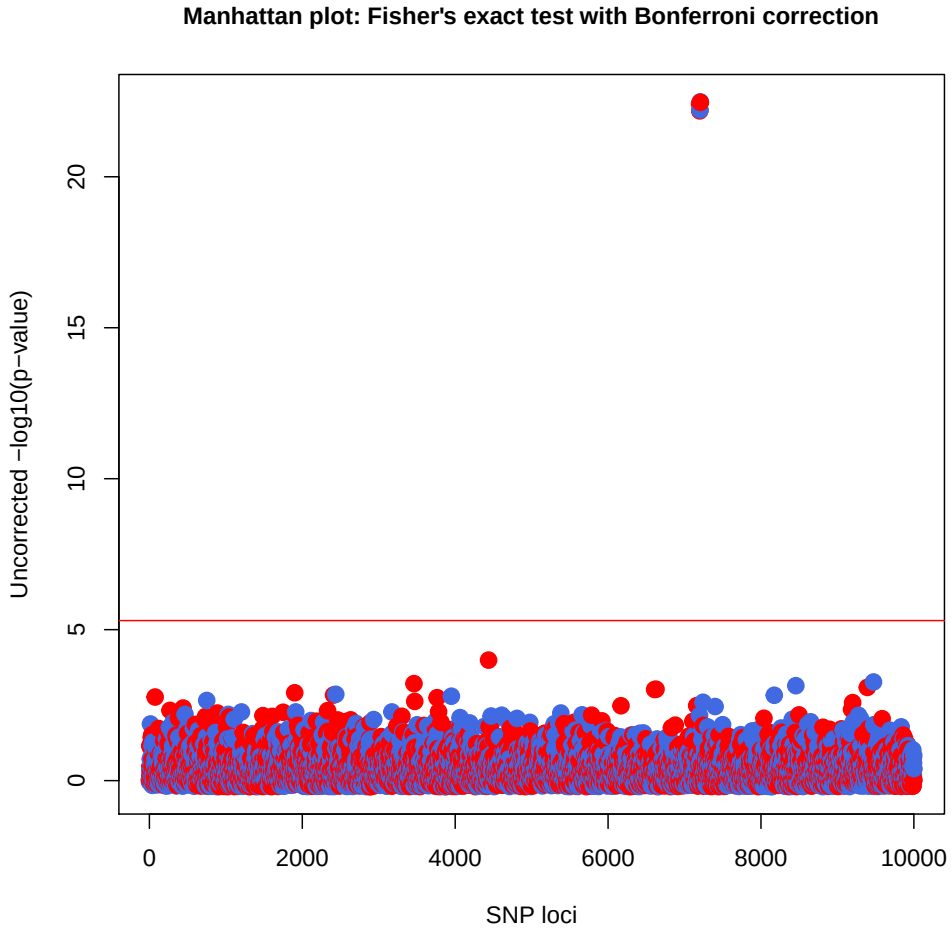
What can you infer from this difference between the p-values?
Finally, we can generate the Manhattan Plots, as before:

```
log.pval <- -log10(pval.corrected)
set.seed(1)
log.pval <- jitter(log.pval, amount=0.2)

plot(log.pval,
     col = c("red", "royalblue"),
     pch = 19,
     cex = 1.5,
     main="Manhattan plot: Fisher's exact test with FDR",
     xlab="SNPs", ylab="Corrected -log10(p-value)")
```



Once again, we can do the same with the uncorrected p-values and plot the Bonferroni correction threshold.

```
log.pval <- -log10(pval)
set.seed(1)
log.pval <- jitter(log.pval, amount=0.2)

plot(log.pval,
     col = c("red", "royalblue"),
     pch = 19,
     cex = 1.5,
     main="Manhattan plot: Fisher's exact test with Bonferroni correction",
     xlab="SNP loci", ylab="Uncorrected -log10(p-value)",
     cex.main=1)

bonferroni <- -log10(0.05 / ncol(snps))
abline(h=bonferroni, col = "red")
```



4.2 Multivariate methods

4.2.1 LASSO

```
LASSO2 <- cv.glmnet(snps.corrected, phen, family="binomial",
                    lambda.min.ratio=0.01, alpha=1)
beta2 <- as.vector(t(coef(LASSO2, s="lambda.min")))

selected2 <- which(beta2[-1] !=0)
n.snps.selected.LASSO2 <- as.integer(length(selected2))
snps.selected.LASSO2 <- as.vector(selected2)

n.snps.selected.LASSO2
```

```
## [1] 5
```

```
snps.selected.LASSO2
```

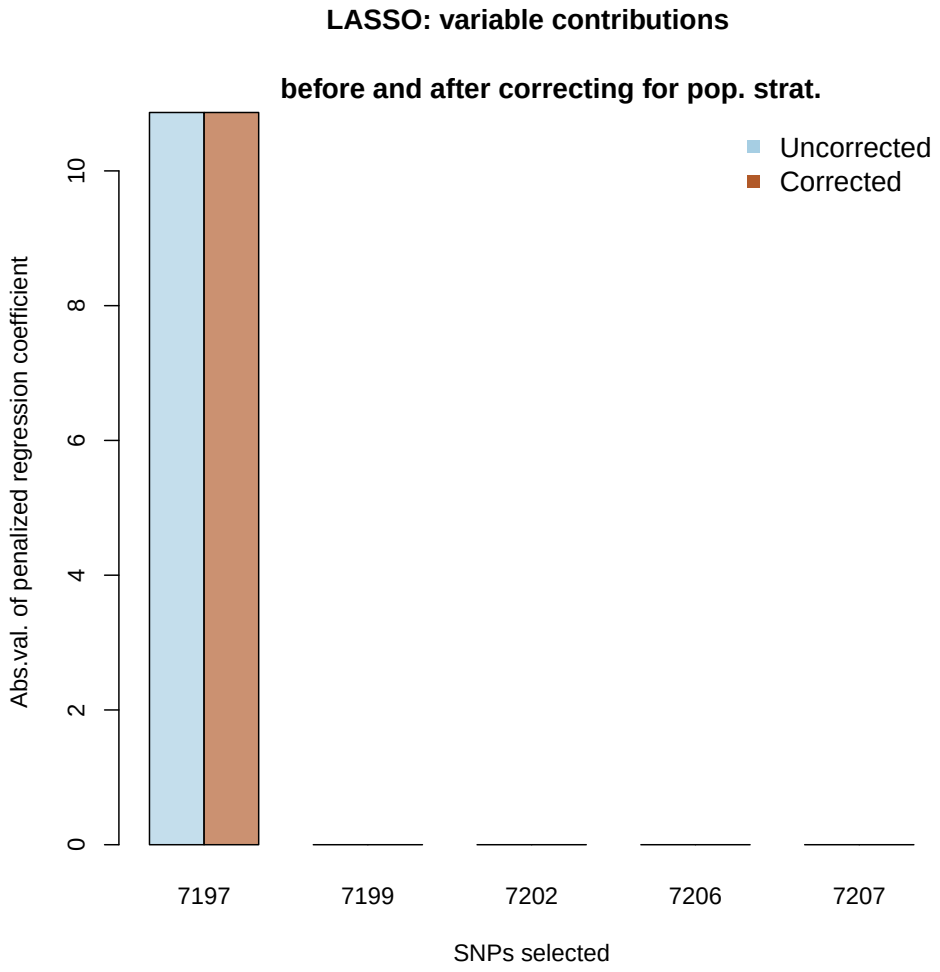
```
## [1] 7197 7199 7202 7206 7207
```

LASSO has selected the same 5 SNPs as the univariate Fisher's exact test.
Once again, we can examine the non-zero coefficients.

```
coefs.LASSO2 <- beta[-1][snps.selected.LASSO2]
names(coefs.LASSO2) <- as.character(snps.selected.LASSO2)
coefs.LASSO2

##          7197          7199          7202          7206          7207
## -1.086688e+01 -6.194127e-14 -3.355152e-14 -1.290443e-15 -1.032355e-14
```

And we can visually compare the coefficients generated by LASSO both before and after the correction for population stratification with a side-by-side barplot.

```
coefs.combined <- rbind(abs(coefs.LASSO), abs(coefs.LASSO2))
coefs.combined

##          7197         7199         7202         7206         7207
## [1,] 10.86688 6.194127e-14 3.355152e-14 1.290443e-15 1.032355e-14
## [2,] 10.86688 6.194127e-14 3.355152e-14 1.290443e-15 1.032355e-14

myCol <- funky(2)
barplot(coefs.combined, beside=TRUE,
     col=c(transp(myCol[1], 0.66), rep(transp(myCol[2], 0.66), 4)),
     xlab="SNPs selected",
     ylab="Abs.val. of penalized regression coefficient",
     main="LASSO: variable contributions \n
     before and after correcting for pop. strat."
)

legend("topright", c("Uncorrected","Corrected"), pch=15,cex=1.2,
       col=myCol,
       bty="n")
```

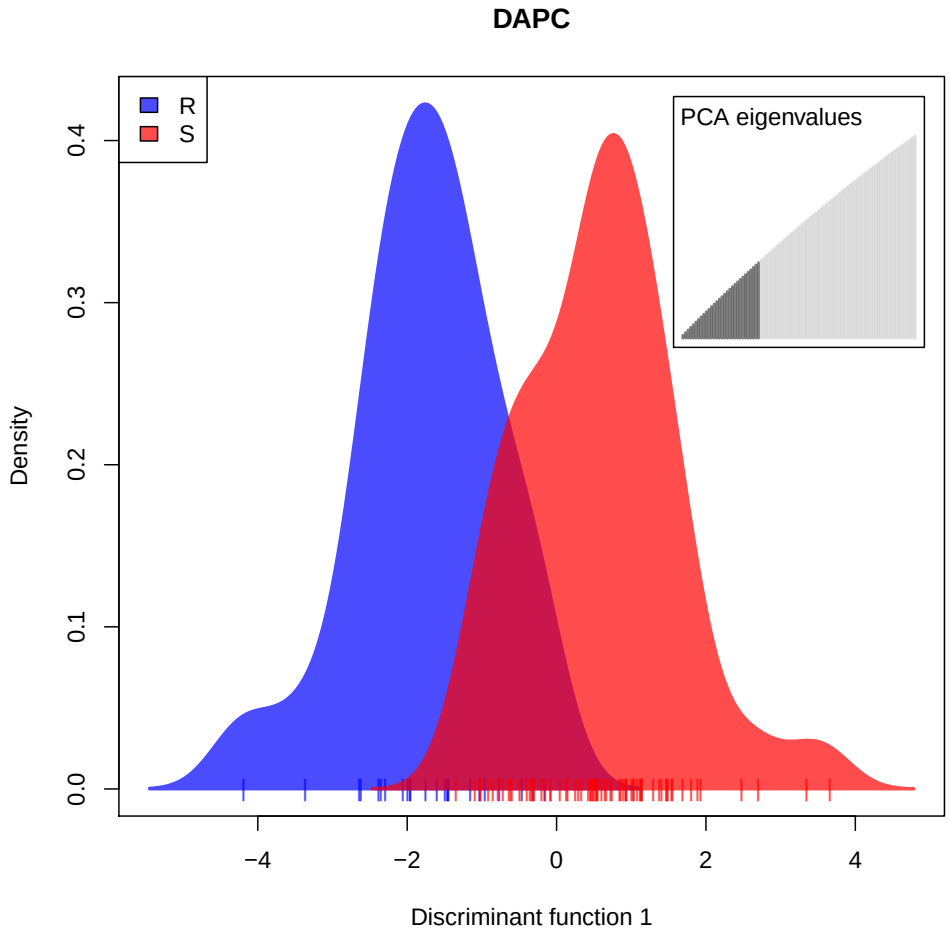


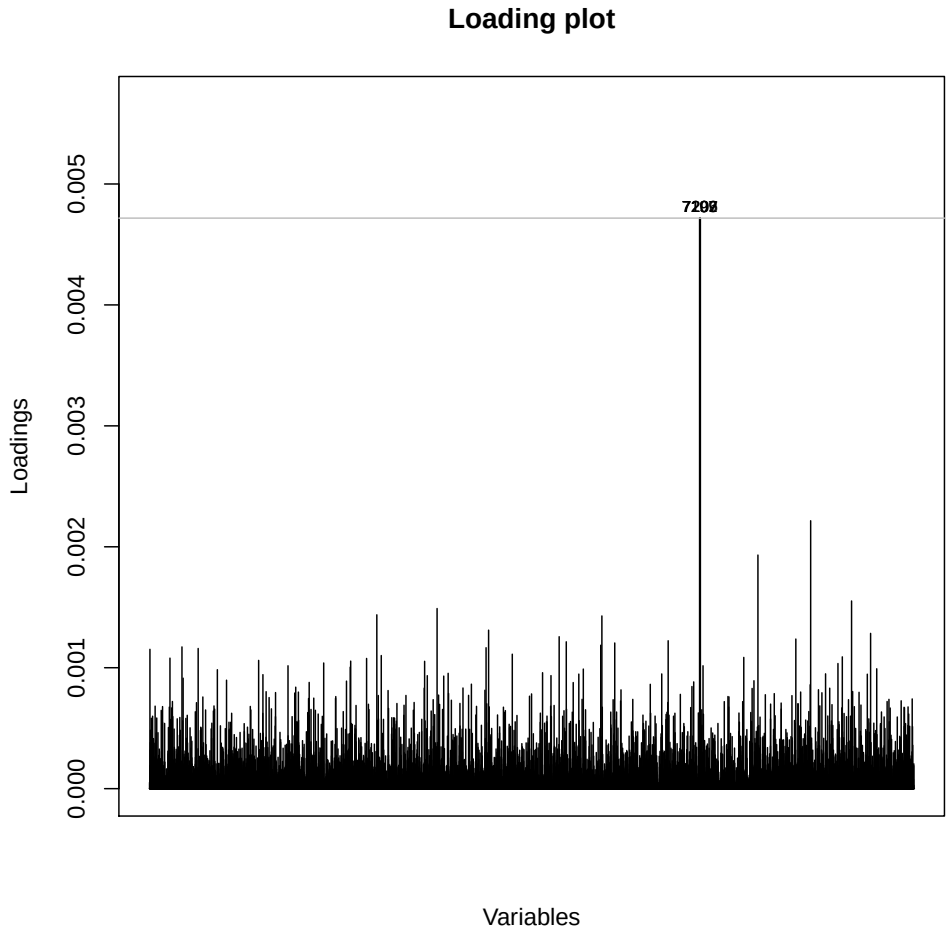
Have the coefficients been affected by the correction for population stratification? Why might that be?

4.2.2 DAPC-based feature selection

We will repeat the DAPC-based feature selection procedure on the corrected SNPs matrix, but this time, we can let the function `snpzip` do all the work by specifying its second argument to be `phen` rather than a `dapc` object. (Type `?snpzip` for more information).

```
set.seed(1)
result <- snpzip(snps.corrected, phen, method="single", xval.plot = FALSE, plot = TRUE,
```


DAPC
R
S
PCA eigenvalues
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
Discriminant function 1

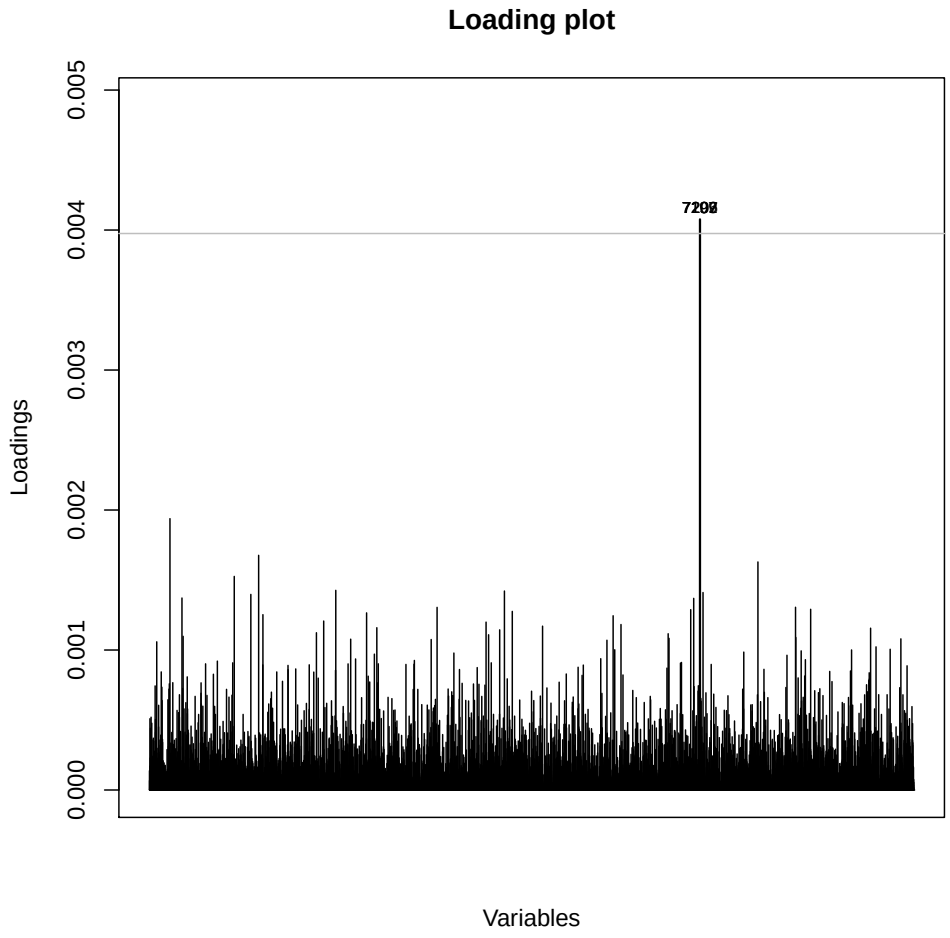



```
par(ask=FALSE)

snps.selected.dapc <- result$FS[[2]]
n.snps.selected.dapc <- length(snps.selected.dapc)

n.snps.selected.dapc

## [1] 5

snps.selected.dapc

## [1] 7197 7199 7202 7206 7207
```

4.3 Interpreting the significance of the SNPs selected

The loadingplot function also invisibly returns information on the annotated variables. We can store this in an object called `sel.snps`.

```
sel.snps <- loadingplot(dapc1$var.contr, thres=min.var.selected-.0001)
```



The object should look like this:

```
sel.snps

## $threshold
## [1] 0.003975383
##
## $var.names
## [1] "7197" "7199" "7202" "7206" "7207"
##
## $var.idx
## 7197 7199 7202 7206 7207
## 7197 7199 7202 7206 7207
##
## $var.values
##        7197        7199        7202        7206        7207
## 0.004076383 0.004076383 0.004076383 0.004076383 0.004076383
```

Which SNPs are the most strongly correlated to antibiotic resistance?
The following command derives allelic profiles of these SNPs for each isolate:

```
sel.profiles <- apply(snps[,sel.snps$var.idx],1,paste,collapse="-")
head(sel.profiles)

##   isolate-1   isolate-2   isolate-3   isolate-4   isolate-5   isolate-6
## "1-1-1-1-1" "0-0-0-0-0" "0-0-0-0-0" "0-0-0-0-0" "0-0-0-0-0" "0-0-0-0-0"

table(sel.profiles)

## sel.profiles
## 0-0-0-0-0 1-1-1-1-1
##        71        24

head(cbind.data.frame(phen,sel.profiles),10)

##            phen sel.profiles
## isolate-1     R    1-1-1-1-1
## isolate-2     S    0-0-0-0-0
## isolate-3     S    0-0-0-0-0
## isolate-4     S    0-0-0-0-0
## isolate-5     S    0-0-0-0-0
## isolate-6     S    0-0-0-0-0
## isolate-7     S    0-0-0-0-0
## isolate-8     S    0-0-0-0-0
## isolate-9     S    0-0-0-0-0
## isolate-10    S    0-0-0-0-0

tail(cbind.data.frame(phen,sel.profiles),10)

##            phen sel.profiles
## isolate-86    S    0-0-0-0-0
## isolate-87    S    0-0-0-0-0
## isolate-88    S    0-0-0-0-0
## isolate-89    S    0-0-0-0-0
## isolate-90    S    0-0-0-0-0
## isolate-91    R    1-1-1-1-1
## isolate-92    S    0-0-0-0-0
## isolate-93    S    0-0-0-0-0
## isolate-94    R    1-1-1-1-1
## isolate-95    R    1-1-1-1-1
```

A contingency table between phenotype and SNPs profile can be created using `table`:

```
table(phen,sel.profiles)

##     sel.profiles
## phen 0-0-0-0-0 1-1-1-1-1
##    R         0        24
##    S        71         0
```

What can you conclude on these SNPs? Assuming that their position in the dataset reflects their original position in the genome, would you think that each of these SNPs actually determines the antibiotic resistance? How would you address this question?

5 The *adegenet* Server

As of version 1.4-0 of adegenet, a web interface for DAPC can be started from R using:

```
adegenetServer("DAPC")
```