Multivariate analysis of genetic data: exploring group diversity

Thibaut Jombart[*], Caitlin Collins

Imperial College London

MRC Centre for Outbreak Analysis and Modelling

October 25, 2014

Abstract

This practical provides an introduction to the analysis of group diversity in genetic data analysis using `R` . First, simple clustering methods are used to infer the nature, and the number of genetic groups. Second, we show how group information can be used to explore the genetic diversity using the Discriminant Analysis of Principal Components (DAPC). More information on this latter topic is available in a tutorial dedicated to DAPC, accessible from the *adegenet* website or by typing `adegenetTutorial("dapc")` (whilst connected to the internet).

[*] `tjombart@imperial.ac.uk`

Contents

1 Defining genetic clusters

Group information is not always known when analysing genetic data. Even when some prior clustering can be defined, it is not always obvious that these are the best genetic clusters that can be defined. In this section, we illustrate two simple approaches for defining genetic clusters.

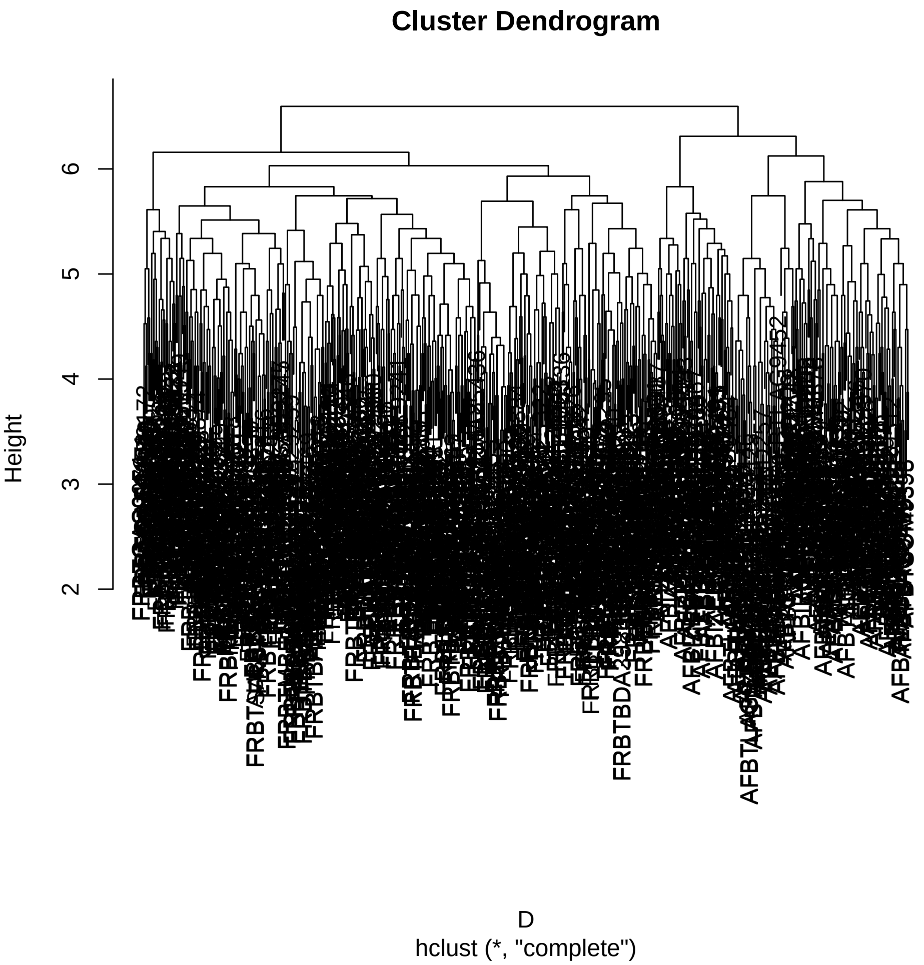
1.1 Hierarchical clustering

Hierarchical clustering can be used to represent genetic distances as trees, and indirectly to define genetic clusters. This is achieved by cutting the tree at a certain height, and pooling the tips descending from the few retained branches into the same clusters (`cutree`). Here, we load the data `microbov`, replace the missing data, and compute the Euclidean distances between individuals:

```
library(adegenet)

## Loading required package:  ade4
##    ==========================
##    adegenet 1.4-2 is loaded
##    ==========================
##
##  - to start, type '?adegenet'
##  - to browse adegenet website, type 'adegenetWeb()'
##  - to post questions/comments:  adegenet-forum@lists.r-forge.r-project.org

data(microbov)
X <- scaleGen(microbov, missing="mean",scale=FALSE)
D <- dist(X)
```

Then, we use `hclust` to obtain a hierarchical clustering of the individual, using complete linkage to obtain ”strong” groups.

```
h1 <- hclust(D, method="complete")
h1

##
## Call:
## hclust(d = D, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 704

plot(h1)
```



Groups can be defined by cutting the tree at a given height. This is performed by the function `cutree`, which can also find the right height to obtain a specific number of clusters. Here, we first look at two groups:

```
grp <- cutree(h1, k=2)
head(grp,10)

## AFBIBOR9503 AFBIBOR9504 AFBIBOR9505 AFBIBOR9506 AFBIBOR9507 AFBIBOR9508
##           1           1           1           1           1           1
## AFBIBOR9509 AFBIBOR9510 AFBIBOR9511 AFBIBOR9512
##           1           1           1           1
```

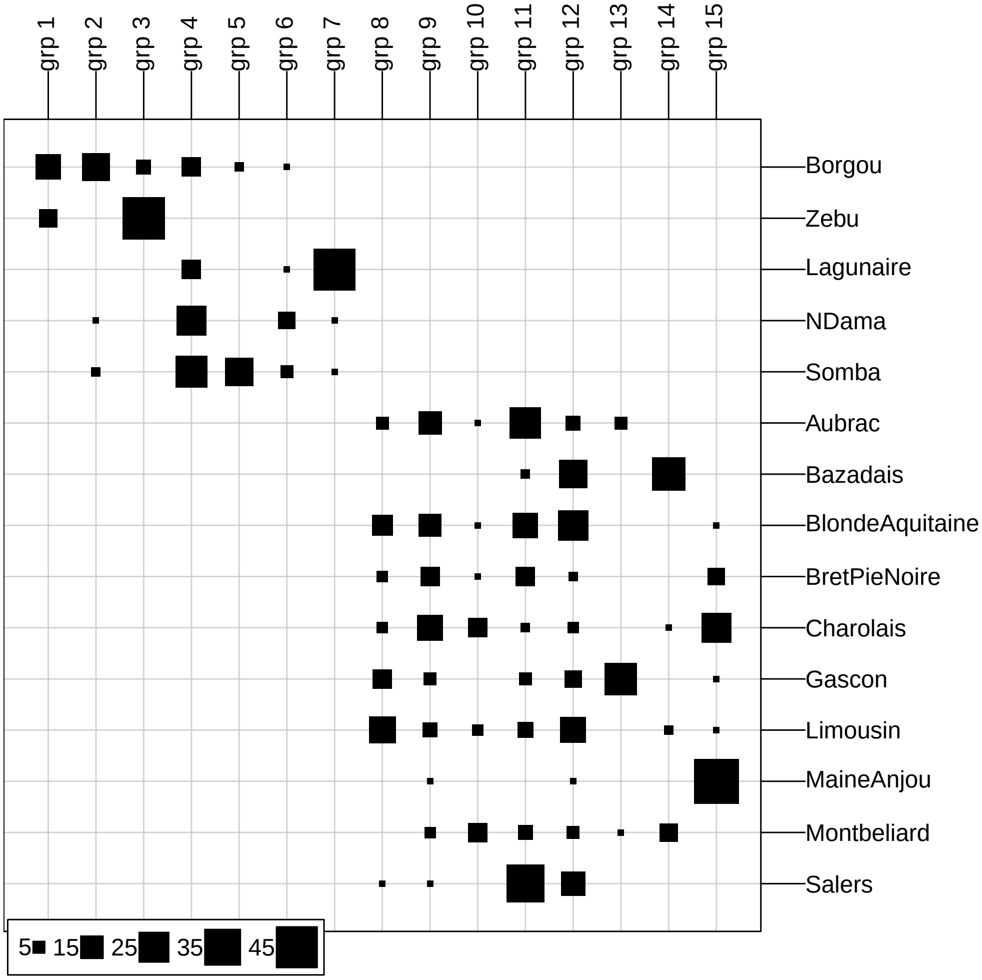
The function `table` is extremely useful, as it can be used to build contingency tables. Here, we use it to compare the inferred groups to the species and the origins of the cattles.

```
table(grp, other(microbov)$spe)

##
## grp  BI  BT
##    1 100 131
##    2   0 473

table(grp, other(microbov)$coun)

##
## grp   AF   FR
##    1 231    0
##    2   0 473
```

What can you say about the two inferred groups? Accordingly, what is the main component of the genetic variability in these cattle breeds?

Repeat this analysis by cutting the tree into as many clusters as there are breeds in the dataset (this can be extracted by `pop`), and name the result `grp`. Using `table` as above, build a contingency table called `tab` to see the match between inferred groups and breeds. The obtained table is then visualized using `table.value`:

```
table.value(tab, col.lab=paste("grp",1:15))
```



Can some groups be identified as species or breeds? Do some species look more admixed than others?

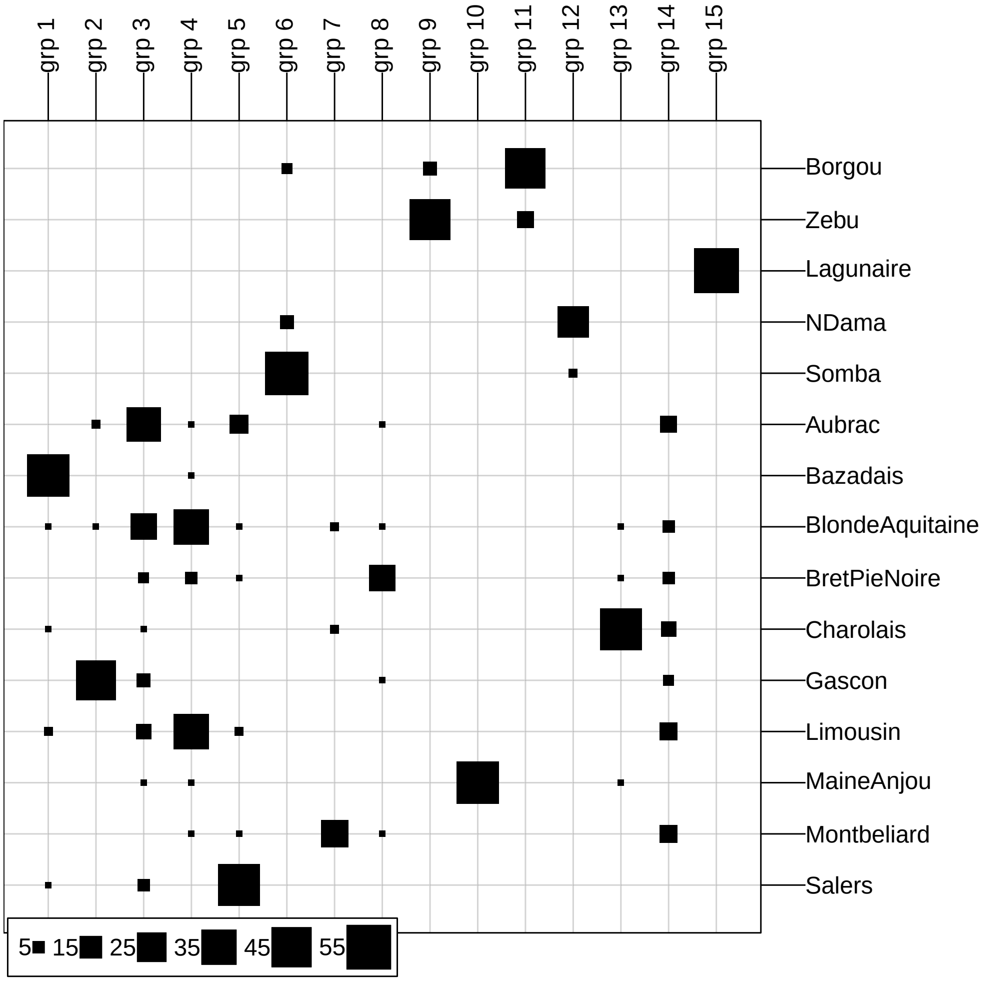
1.2 K-means

K-means is another, non-hierarchical approach for defining genetic clusters. While basic K-means is implemented in the function `kmeans`, the function `find.clusters` provides a computer-efficient implementation which first reduces the dimensionality of the data (using PCA), and optionally allows for choosing the optimal number of clusters using Bayesian Information Criteria (BIC). Use `find.clusters` to obtain 15 groups and store the result in an object called `grp`. If unsure how to use the function, remember to check the help page (`?find.clusters`).

How many clusters would you have selected relying on the BIC?

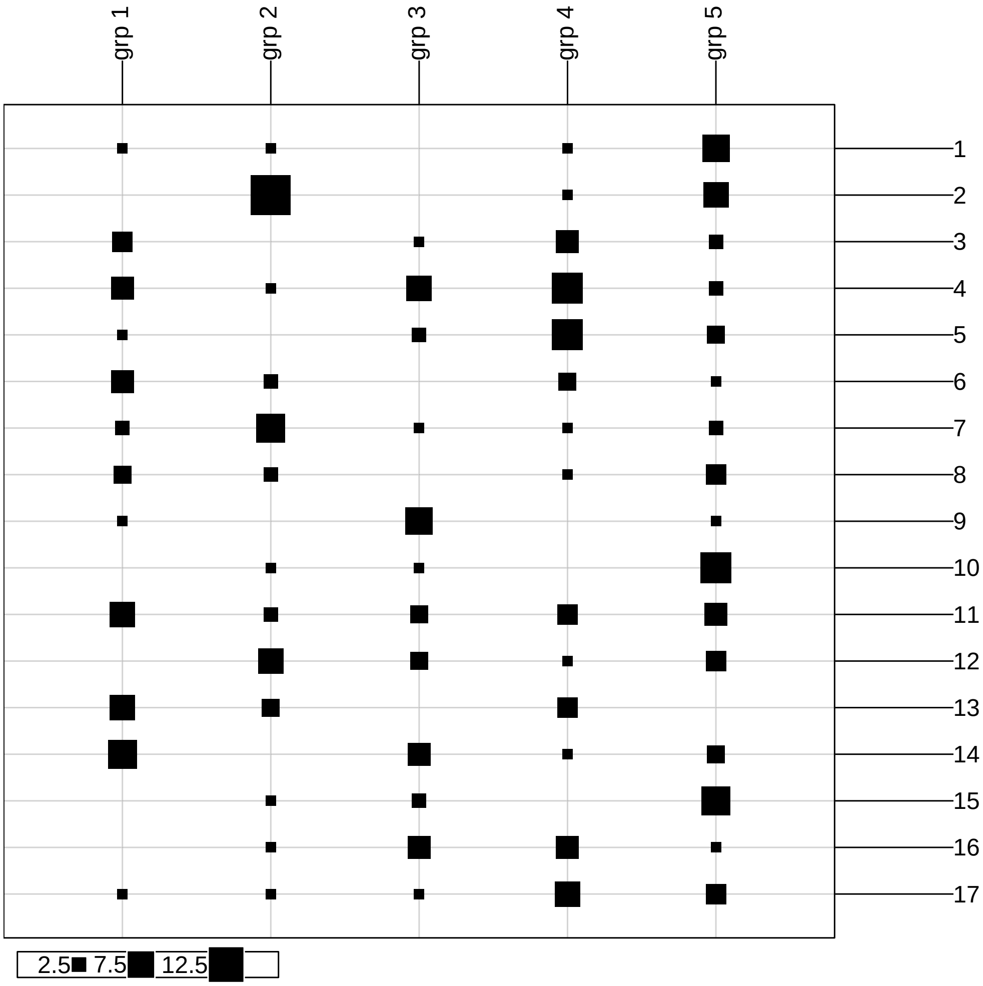
Using `table.value` as before, visualize the correspondence between inferred groups and actual breeds:

```
table.value(table(pop(microbov), grp$grp), col.lab=paste("grp", 1:15))
```



How do these results compare to the ones obtained using hierarchical clustering? What are the species which are easily genetically identified using K-means?

Repeat the same analyses for the `nancycats` data. What can you say about the likely profile of admixture between these cat colonies?



2 Describing group diversity: an application to Genome-Wide Association Study (GWAS)

2.1 The data

In this part, we use multivariate analyses to investigate genetic differences between two simulated groups of bacterial genomes, one of which is resistant to a given antibiotic.

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. 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).

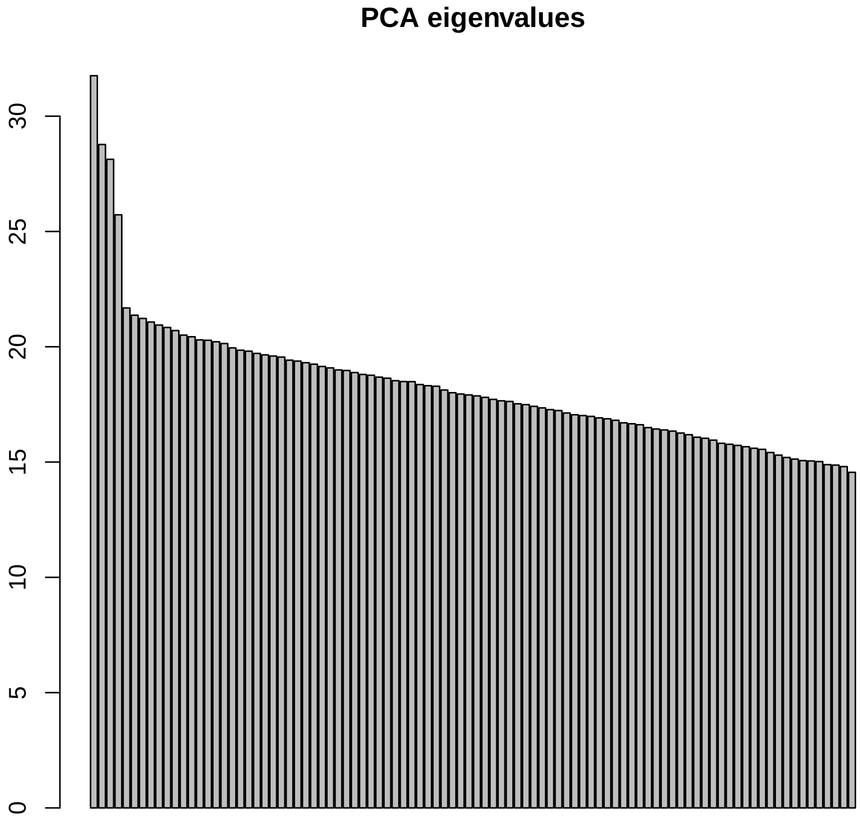
To simplify further commands, we create the new objects snps and phen from simGWAS:

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

2.2 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:

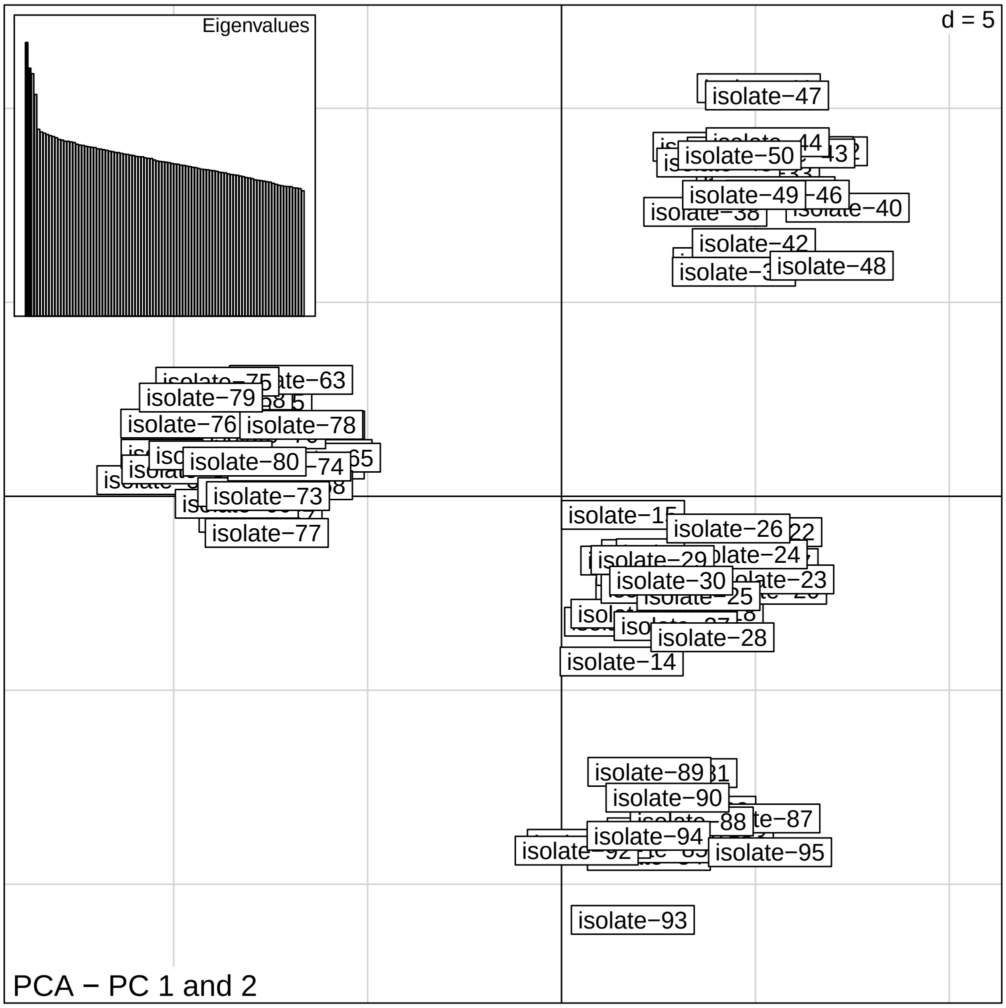
- `pca1$eig`: contains the eigenvalues of the analysis, representing the amount of information contained in each PC.
- `pca1$li`: contains the principal components.
- `pca1$c1`: contains the principal axes (loadings of the variables).

```
head(pca1$eig)

## [1] 31.76 28.77 28.13 25.72 21.68 21.37

head(pca1$li)

##           Axis1  Axis2  Axis3   Axis4
## isolate-1 3.606 -2.133  9.623  -6.302
## isolate-2 1.913 -1.657  8.734 -10.006
## isolate-3 2.317 -2.565  9.325  -7.446
## isolate-4 2.491 -2.485  8.819  -6.030
## isolate-5 2.449 -1.490  8.576  -8.776
## isolate-6 2.939 -2.693 10.877  -3.797

head(pca1$c1)
```

```
##            CS1       CS2       CS3        CS4
## X1  0.0100427  0.004292 -0.003510 -0.0092503
## X2 -0.0051457 -0.003539 -0.001471  0.0075073
## X3 -0.0000335 -0.003363  0.003798  0.0013049
## X4 -0.0010178  0.002489 -0.002323 -0.0007848
## X5  0.0070474  0.007802 -0.003475  0.0057484
## X6 -0.0101199  0.013435  0.021812 -0.0321210
```

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`:

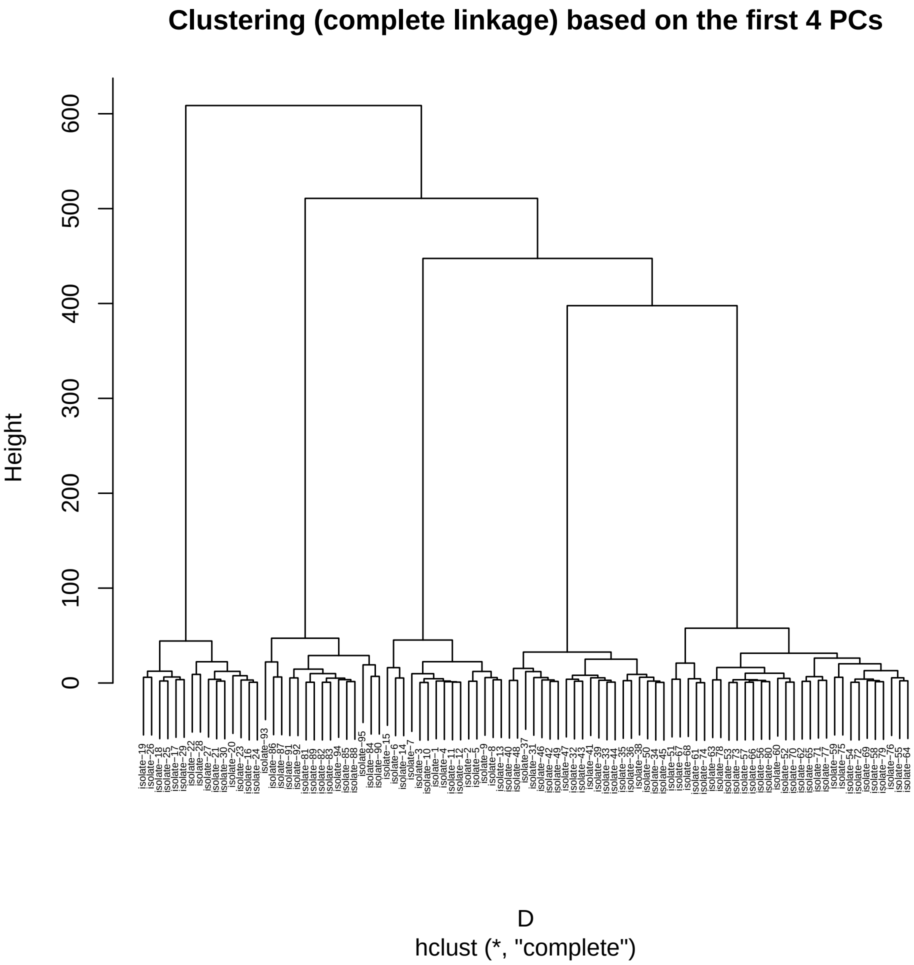
```
s.label(pca1$li, sub="PCA - PC 1 and 2")
add.scatter.eig(pca1$eig,4,1,2, ratio=.3, posi="topleft")
```



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")
```

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



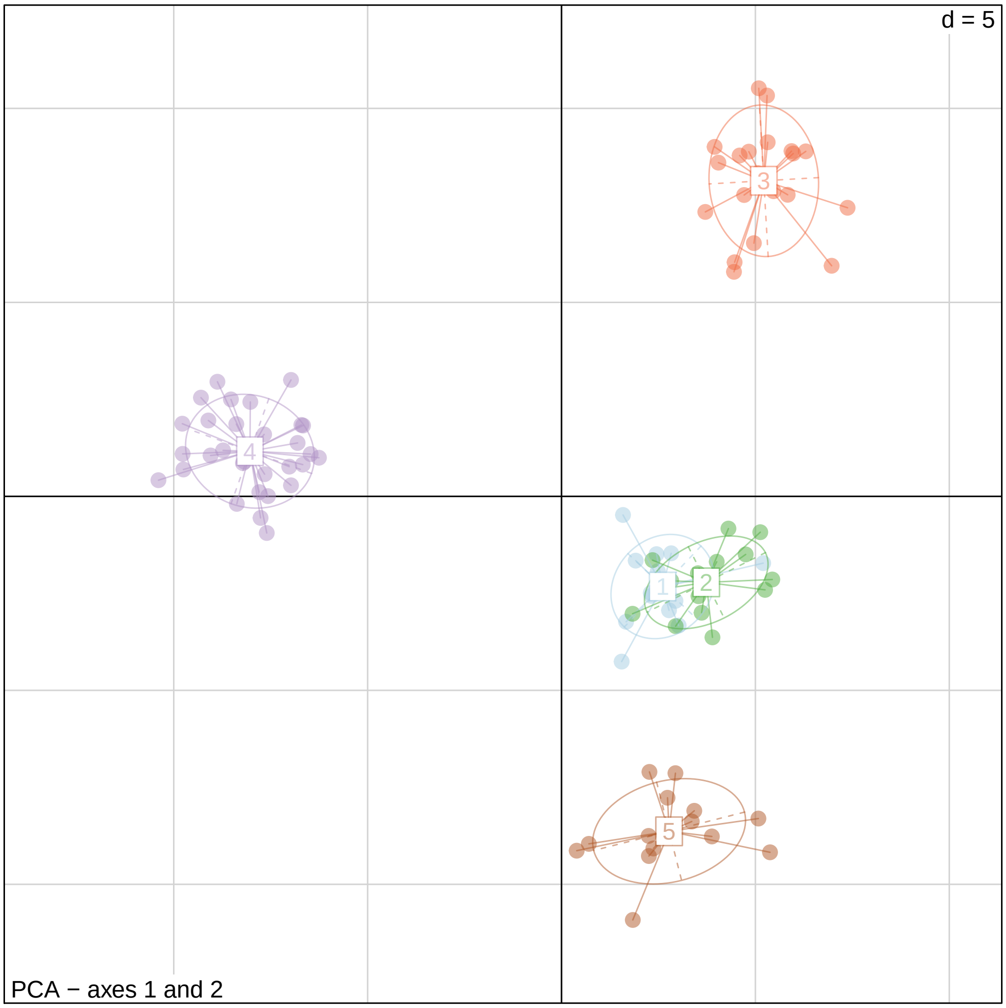
How many clusters are there in the data? How does it compare to what you would have assessed based on the first two PCs of PCA? *Bonus question*: considering that the original data are profile of binary SNPs, what does the 'height' represent in this dendrogram?

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
```

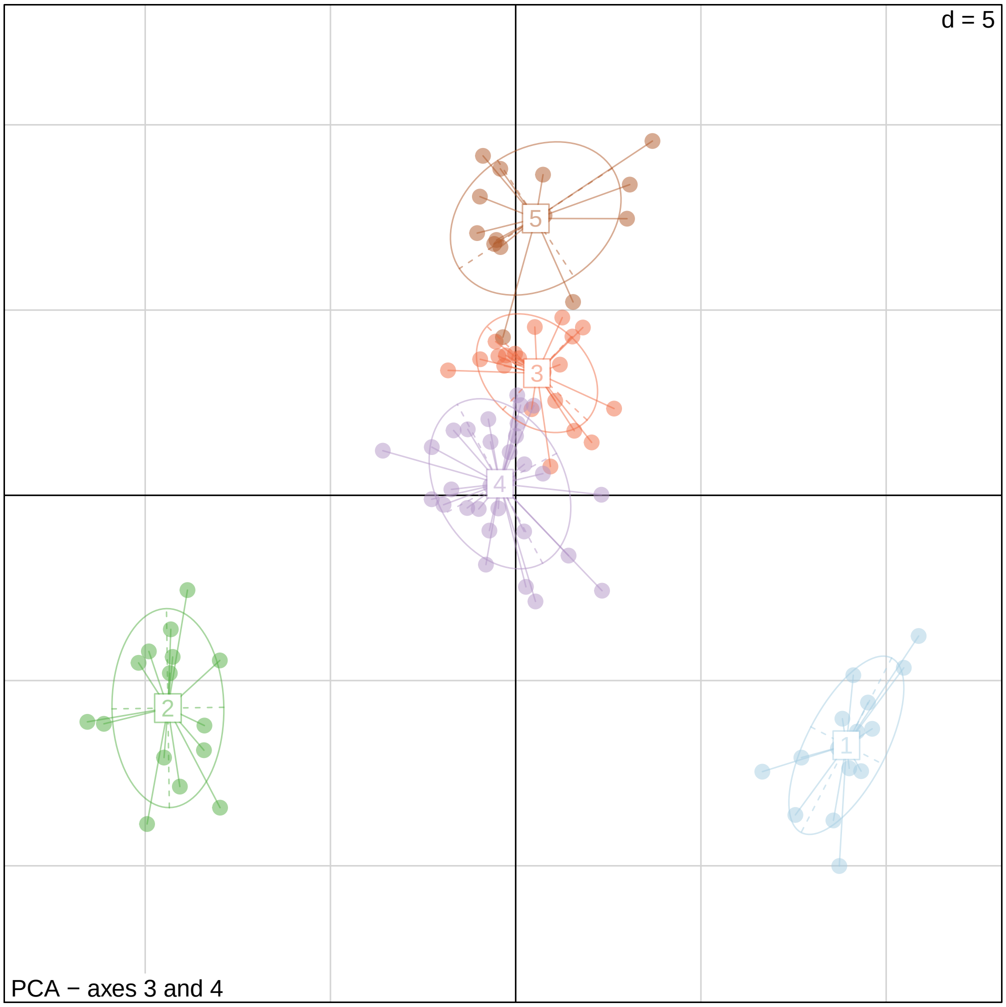
Now, we can represent these groups on top of the PCs using `s.class` (clusters are indicated by different colors and ellipses):

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



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")
```



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?

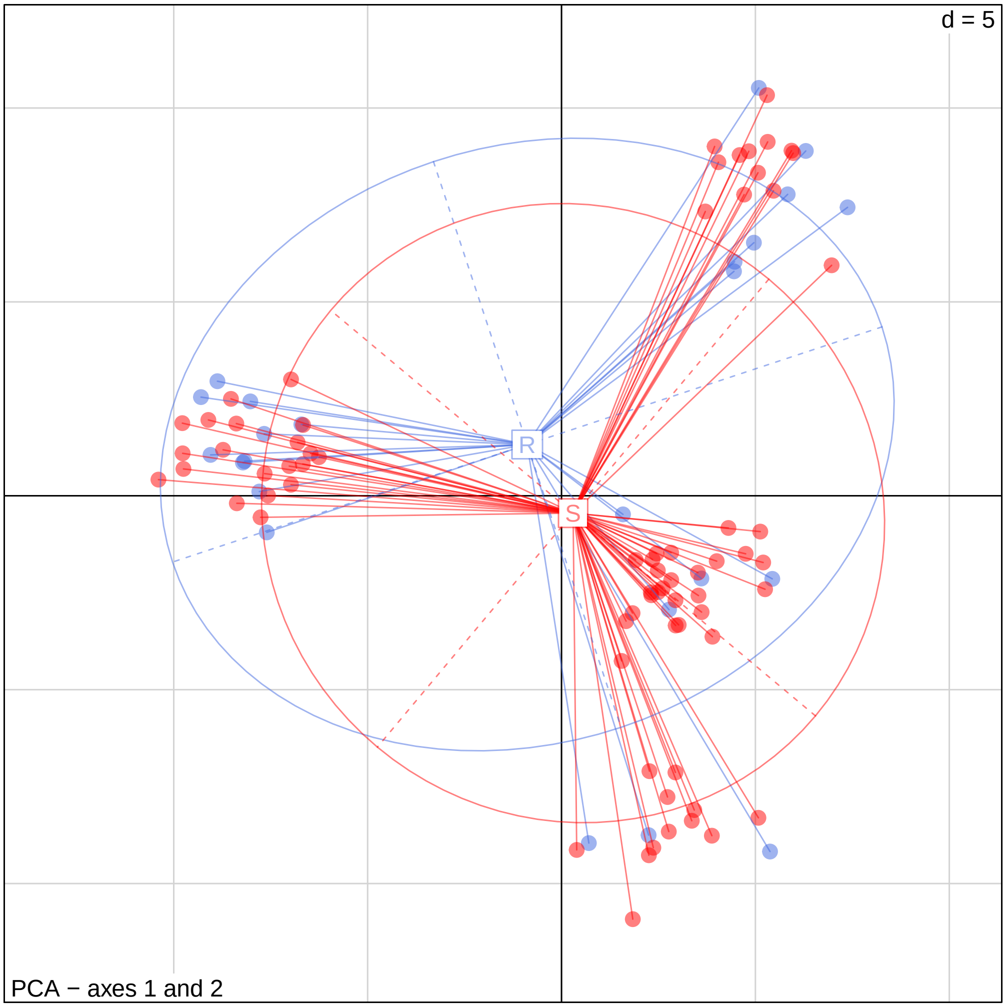
2.3 Identifying SNPs linked to antibiotic resistance

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

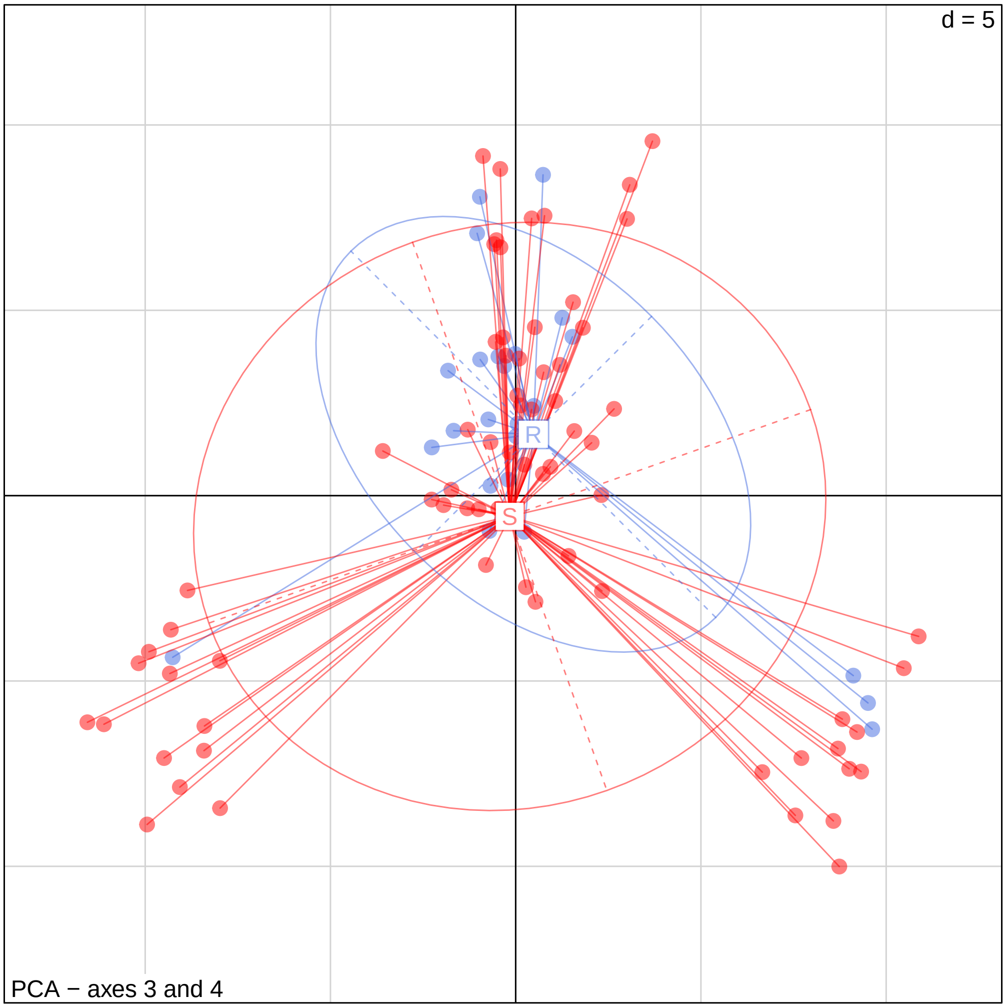
```
head(phen,10)

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

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")
```



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



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 = 5.227, df = NA, p-value = 0.2779
```

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

It is important to keep in mind that PCA optimizes the representation of the overall genetic diversity, and does not explicitly look for distinctions between pre-defined groups of isolates. If only a few loci are correlated to bacterial resistance, PCA may well overlook these, especially if stronger structures such as separate lineages or populations are present. To look for combinations of SNPs correlated to a given partition of individuals, DAPC is much more appropriate. We apply the method using the function `dapc`, specifying the input data `snps` and the groups of individuals to distinguish (susceptible/resistant, `phen`).

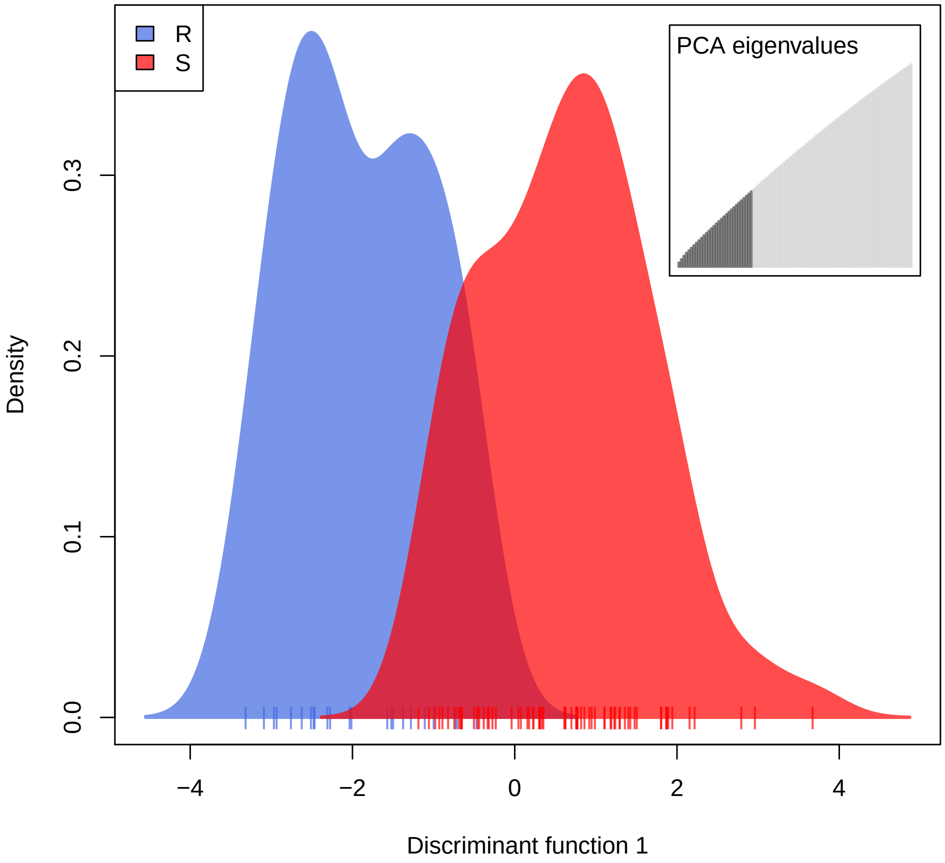
```
dapc1 <- dapc(snps, phen)
```

The function asks for a number of principal components to retain for the dimension-reduction step (PCA, retain 30 PCs) and for the subsequent discriminant analysis (DA). For the latter, only one axis can be retained (the maximum number of axes in DA is always the number of groups minus 1).

```
dapc1

##  #################################################
##  # Discriminant Analysis of Principal Components #
##  #################################################
## class: dapc
## $call: dapc.data.frame(x = as.data.frame(x), grp = ..1, n.pca = 30,
##     n.da = 1)
##
## $n.pca: 30 first PCs of PCA used
## $n.da: 1 discriminant functions saved
## $var (proportion of conserved variance): 0.371
##
## $eig (eigenvalues): 116.4  vector    length content
## 1 $eig      1      eigenvalues
## 2 $grp      95     prior group assignment
## 3 $prior    2      prior group probabilities
## 4 $assign   95     posterior group assignment
## 5 $pca.cent 10000  centring vector of PCA
## 6 $pca.norm 10000  scaling vector of PCA
## 7 $pca.eig  94     eigenvalues of PCA
##
##   data.frame    nrow  ncol
## 1 $tab          95    30
## 2 $means        2     30
## 3 $loadings     30    1
## 4 $ind.coord    95    1
## 5 $grp.coord    2     1
## 6 $posterior    95    2
## 7 $pca.loadings 10000 30
## 8 $var.contr    10000 1
##   content
## 1 retained PCs of PCA
## 2 group means
## 3 loadings of variables
## 4 coordinates of individuals (principal components)
## 5 coordinates of groups
## 6 posterior membership probabilities
## 7 PCA loadings of original variables
## 8 contribution of original variables
```

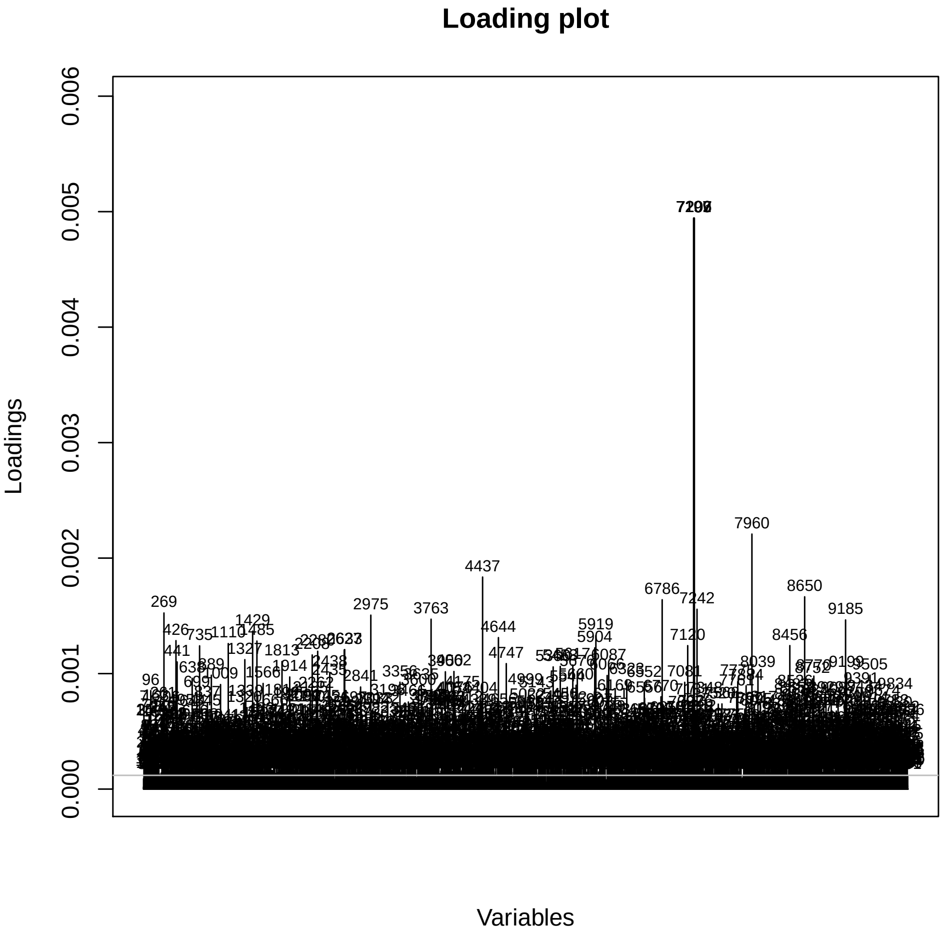
The function `scatter` can be used to visualize the results of DAPC. It produces usual plots of the principal components, using colors and ellipses to indicate groups. However, whenever only one axis has been retained, `scatter` plots the density of the individuals on the first principal component:

```
scatter(dapc1, bg="white", scree.da=FALSE, scree.pca=TRUE,
        posi.pca="topright", col=c("royalblue","red"),
        legend=TRUE, posi.leg="topleft")
```

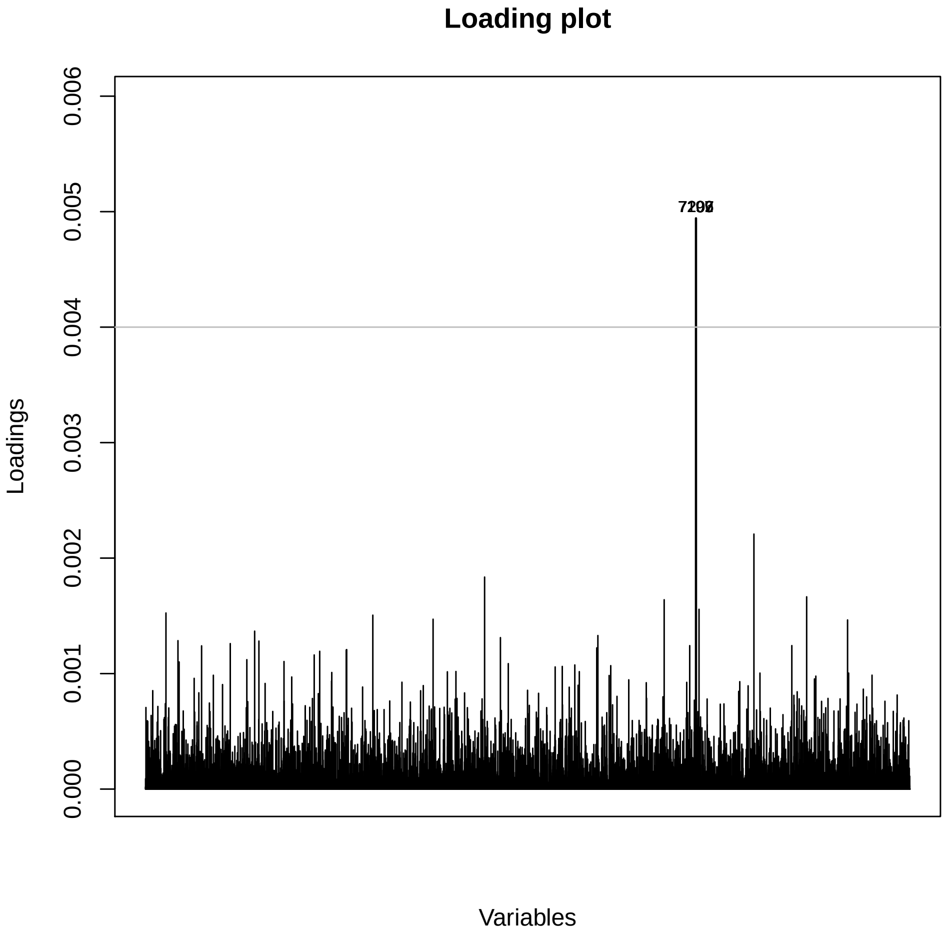


The contribution of each variable to the separation of the two groups (susceptible/resistant) is stored in `dapc1$var.contr`; it can be visualized using `loadingplot`, which displays all contributions as bars and annotates variables with the largest contributions (see argument `threshold` in `?loadingplot`):

```
loadingplot(dapc1$var.contr)
```



The function also invisibly returns information on the annotated variables. Recall `loadingplot`, specifying a higher threshold so that only the few outlying variables are retained, and store this result in an object called `sel.snps`.



The object should look like this:

```
sel.snps

## $threshold
## [1] 0.004
##
## $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
## 0.004944 0.004944 0.004944 0.004944 0.004944
```

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?

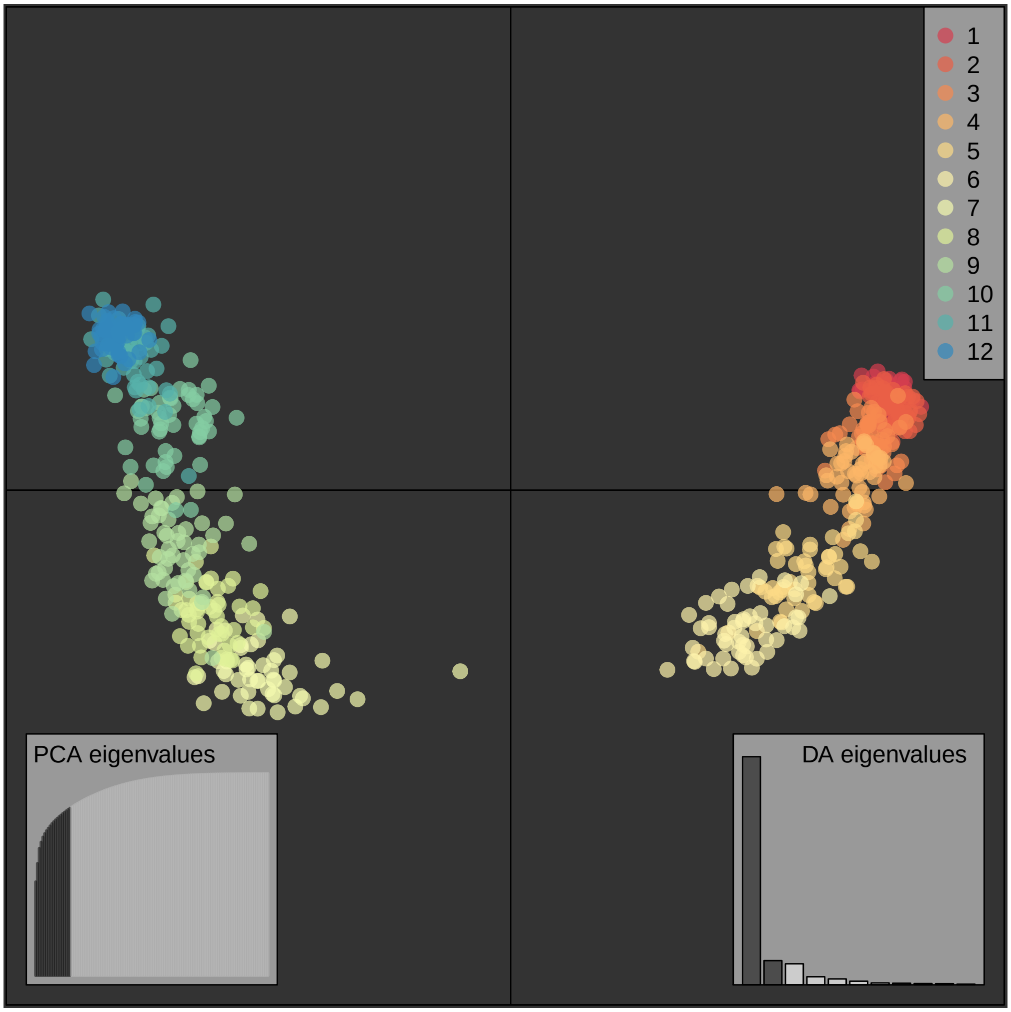
3 Supplementary exercise

Load the dataset `dapcIllus` and look at the description of the data in the help page. Try to answer to following questions:

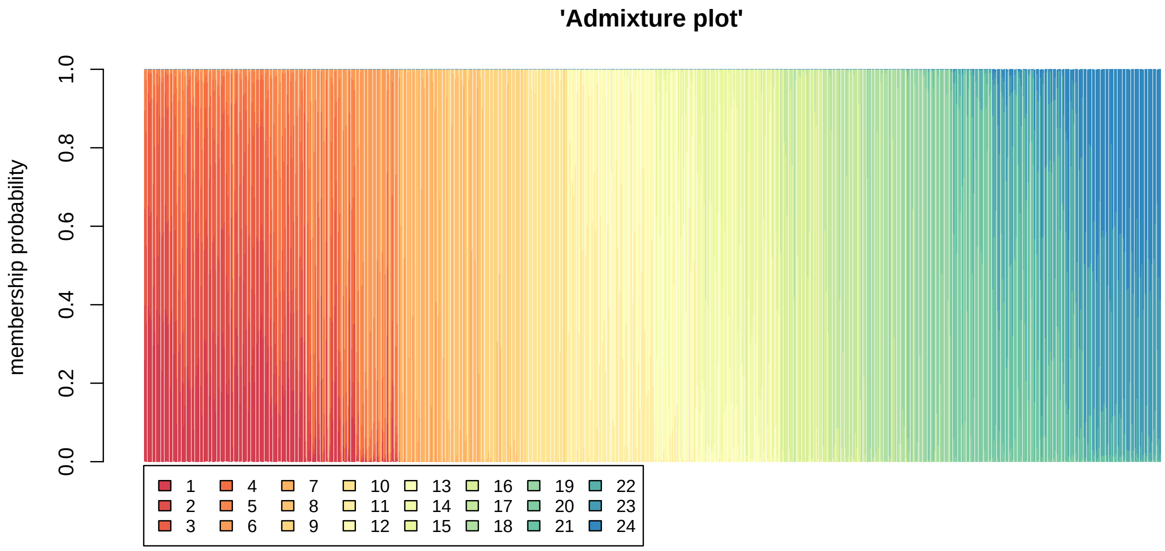
- When looking for clusters using K-means (preceded by dimension reduction using PCA), what are the differences in terms of BIC profiles for different population genetics models?

- What are the differences in the typologies identified by DAPC for different population genetics models?

- Customizing the scatterplot of DAPC, reproduce and interpret the figure below (based on `dapcIllus$c`):



- After looking at what the `compoplot` function does, reproduce and interpret the following figure (based on `dapcIllus$d`):



What was the model used to simulate these data? What can you say about the number of clusters in these data, and its characterization by `find.clusters` and the DAPC scatterplot?

4 To go further

DAPC is more extensively covered in a dedicated tutorial which you can access from the *adegenet* website:
http://adegenet.r-forge.r-project.org/

or by typing:

```
adegenetTutorial("dapc")
```

The paper presenting the method is in open access online:
http://www.biomedcentral.com/1471-2156/11/94

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

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