Practical course using the R software

Introduction to multivariate analysis for bacterial GWAS using R

Thibaut Jombart (tjombart@imperial.ac.uk)

Imperial College London — MSc "*Modern Epidemiology*" / "*Public Health*"

Abstract

This practical illustrates how multivariate methods can be used for the analysis of bacterial genomic datasets. Principal Component Analysis (PCA, [7, 2, 3]) is introduced for assessing the diversity between sampled isolates. We also show how hierarchical clustering methods can be applied on principal components to identify groups of genetically related isolates. Discriminant Analysis of Principal Components (DAPC, [5]) is then used for identifying polymorphic sites associated with phenotypic traits such as bacterial resistance. While this tutorial uses simulated data, the procedures described are applicable to a wide range of Genome-Wide Association Studies (GWAS).

Contents

1 Introduction

1.1 Required packages

This practical requires a working version of R [8] greater than or equal to 2.15.2. To check which version of R you are using, examine the welcome message displayed when starting R, or type:

```
> R.version$version.string
```

```
[1] "R version 2.15.2 (2012-10-26)"
```

The practical relies on the package *ade4* [1] for classical multivariate analyses (PCA) and on *adegenet* [4] for the DAPC. Both packages need to be installed, which may be tricky if you do not possess administrative rights on your computer. However, most systems still "public" areas where any user has read/write access. This is exploited by a simple hack allowing one to install packages without the administrative rights. To use it, simply type (while connected to the internet):

```
> source("http://adegenet.r-forge.r-project.org/files/hackLib/hackLib.R")
> hackLib()
```

Then, the required packages can be installed using the usual procedure:

```
> install.packages("ade4", dep=TRUE)
> install.packages("adegenet", dep=TRUE)
```

and loaded using:

```
> library(ade4)
> library(adegenet)
```

1.2 Getting help

There are several ways of getting information about R functions, including some specific documentation sources for *adegenet*. The function `help.search` is used to look for help on a given topic. For instance:

```
> help.search("Hardy-Weinberg")
```

replies that there is a function `HWE.test.genind` in the *adegenet* package, and other similar functions in *genetics* and *pegas*. To get help for a given function, use `?foo` where `foo` is the function of interest. For instance:

```
> ?spca
```

will open up the manpage of the spatial principal component analysis [6]. At the end of a manpage, an "example" section often shows how to use a function. This can be copied and pasted to the R console, or directly executed from the console using `example`. For further researches on R functions, `RSiteSearch` can be used to perform online researches using keywords in R's archives (mailing lists and manpages).

adegenet has a few extra documentation sources. Information can be found from the adegenet website (`http://adegenet.r-forge.r-project.org/`), in the "documents" section, including several tutorials and a manual which compiles all manpages of the package, and a dedicated mailing list with searchable archives. To open the website from R, use:

```
> adegenetWeb()
```

Tutorials ("*vignettes*" in R's terminology) are also distributed with *adegenet*, and can be accessed using the command `vignette`. These can be listed using:

```
> vignette(package="adegenet")
```

To open a vignette, for instance the tutorial on DAPC, simply use:

```
> vignette("adegenet-dapc")
```

Lastly, several mailing lists are available to find different kinds of information on R; to name a few:

- ⋆ *R-help*: general questions about R.
  `https://stat.ethz.ch/mailman/listinfo/r-help`
- ⋆ *R-sig-genetics*: genetics in R.
  `https://stat.ethz.ch/mailman/listinfo/r-sig-genetics`
- ⋆ *R-sig-phylo*: phylogenetics in R.
  `https://stat.ethz.ch/mailman/listinfo/r-sig-phylo`
- ⋆ *adegenet forum*: adegenet and multivariate analysis of genetic markers.
  `https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/adegenet-forum`

1.3 The data

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
ind1  0 0 0 1 0 1 1 0 0  1  1  0  1  1  0  0  1  0  0  1
ind2  0 0 0 1 0 1 1 0 0  1  1  1  1  1  0  1  1  0  0  1
ind3  0 0 0 1 1 1 1 1 0  1  0  1  1  1  1  1  1  1  0  1
ind4  0 1 1 1 0 1 1 0 0  0  1  1  0  1  1  1  0  1  0  0
ind5  1 0 0 1 0 1 0 1 0  1  1  0  1  1  0  1  1  1  0  0
ind6  1 1 0 1 1 1 1 0 0  1  1  1  1  1  0  1  1  1  0  1
ind7  1 1 0 1 1 0 0 1 0  1  0  1  1  1  0  0  1  1  1  0
ind8  0 1 0 0 0 1 0 0 0  1  0  1  1  0  0  1  1  1  1  0
ind9  0 1 0 1 1 1 0 0 0  1  1  1  1  0  1  1  1  1  1  1
ind10 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" "S" "S"
[20] "S" "S" "S" "R" "S" "S" "S" "S" "S" "S" "S" "R" "R" "S" "S" "S" "S" "R" "S"
[39] "S" "R" "R" "R" "S" "S" "S" "R" "S" "S" "S" "S" "S" "S" "S" "S" "R" "R" "S"
[58] "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "R" "R" "R" "S" "S" "R" "S"
[77] "R" "R" "R" "R" "S" "S" "S" "S" "S" "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).

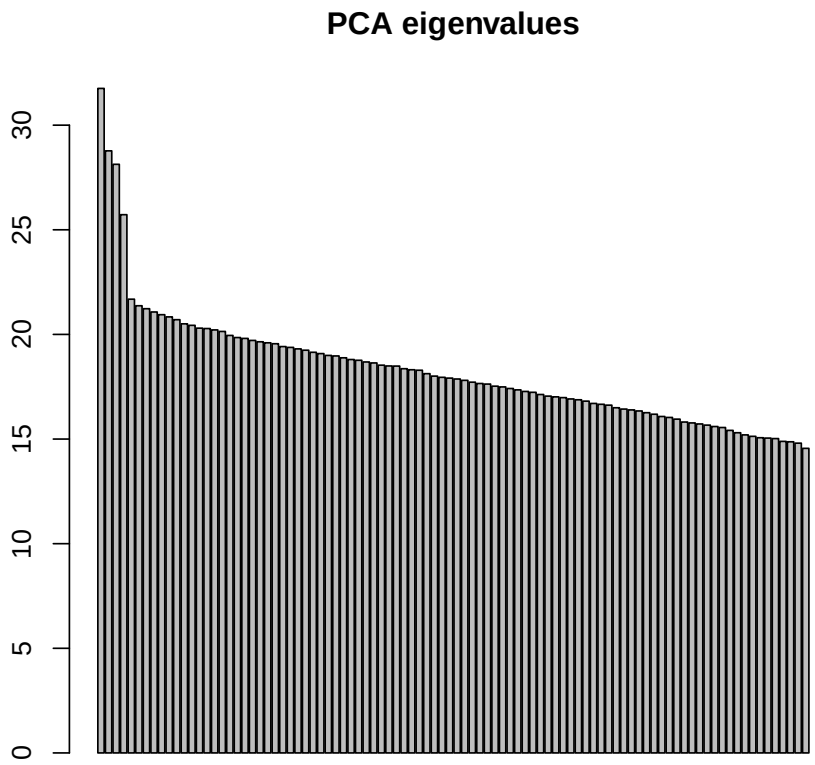
To simplify further commands, we create the new objects `snps` and `phen` from `simGWAS`:

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

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] 31.75594 28.77240 28.12875 25.72180 21.68303 21.36911
```

```
> head(pca1$li)
```

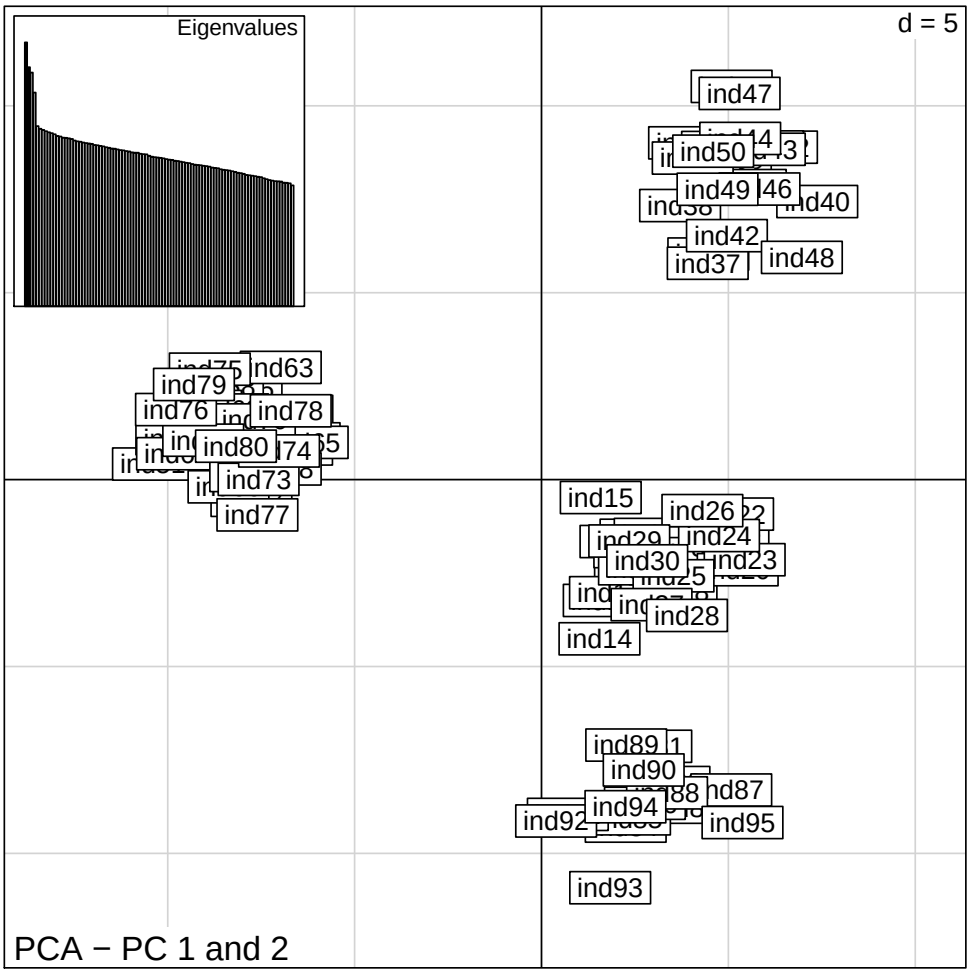
```
        Axis1     Axis2     Axis3      Axis4
ind1 3.606420 -2.132999  9.622764  -6.301912
ind2 1.912918 -1.656548  8.734490 -10.006055
ind3 2.316603 -2.564638  9.324818  -7.445660
ind4 2.490536 -2.484711  8.819193  -6.029816
ind5 2.448958 -1.489571  8.576321  -8.775661
ind6 2.938701 -2.693103 10.876804  -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
```

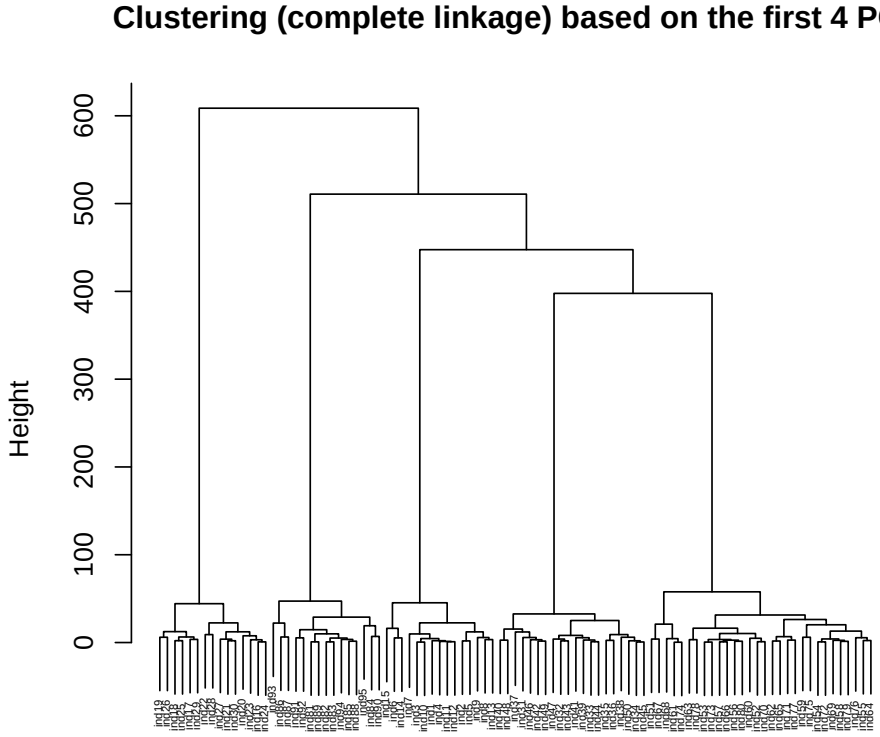
Because of the large number of variables, the usual biplot (function `scatter`) is useless to visualize the results (try `scatter(pca1) if unsure`). We represent only PCs using `s.label`:

```
> 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 the existence 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?

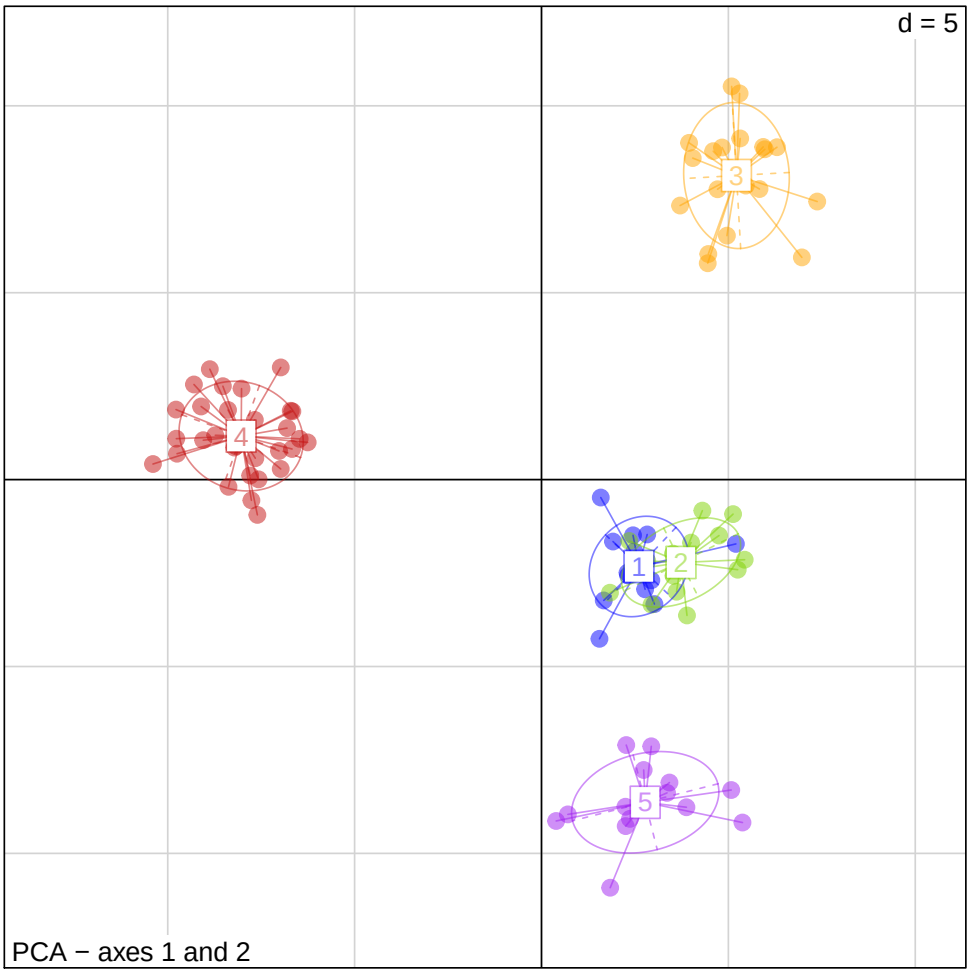
You can define clusters based on the dendrogram `clust` using `cutree`:

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

```
 ind1  ind2  ind3  ind4  ind5  ind6  ind7  ind8  ind9 ind10 ind11 ind12 ind13
    1     1     1     1     1     1     1     1     1     1     1     1     1
ind14 ind15 ind16 ind17 ind18 ind19 ind20 ind21 ind22 ind23 ind24 ind25 ind26
    1     1     2     2     2     2     2     2     2     2     2     2     2
ind27 ind28 ind29 ind30 ind31 ind32 ind33 ind34 ind35 ind36 ind37 ind38 ind39
    2     2     2     2     3     3     3     3     3     3     3     3     3
ind40 ind41 ind42 ind43 ind44 ind45 ind46 ind47 ind48 ind49 ind50 ind51 ind52
    3     3     3     3     3     3     3     3     3     3     3     4     4
ind53 ind54 ind55 ind56 ind57 ind58 ind59 ind60 ind61 ind62 ind63 ind64 ind65
    4     4     4     4     4     4     4     4     4     4     4     4     4
ind66 ind67 ind68 ind69 ind70 ind71 ind72 ind73 ind74 ind75 ind76 ind77 ind78
    4     4     4     4     4     4     4     4     4     4     4     4     4
ind79 ind80 ind81 ind82 ind83 ind84 ind85 ind86 ind87 ind88 ind89 ind90 ind91
    4     4     5     5     5     5     5     5     5     5     5     5     5
ind92 ind93 ind94 ind95
    5     5     5     5
Levels: 1 2 3 4 5
```

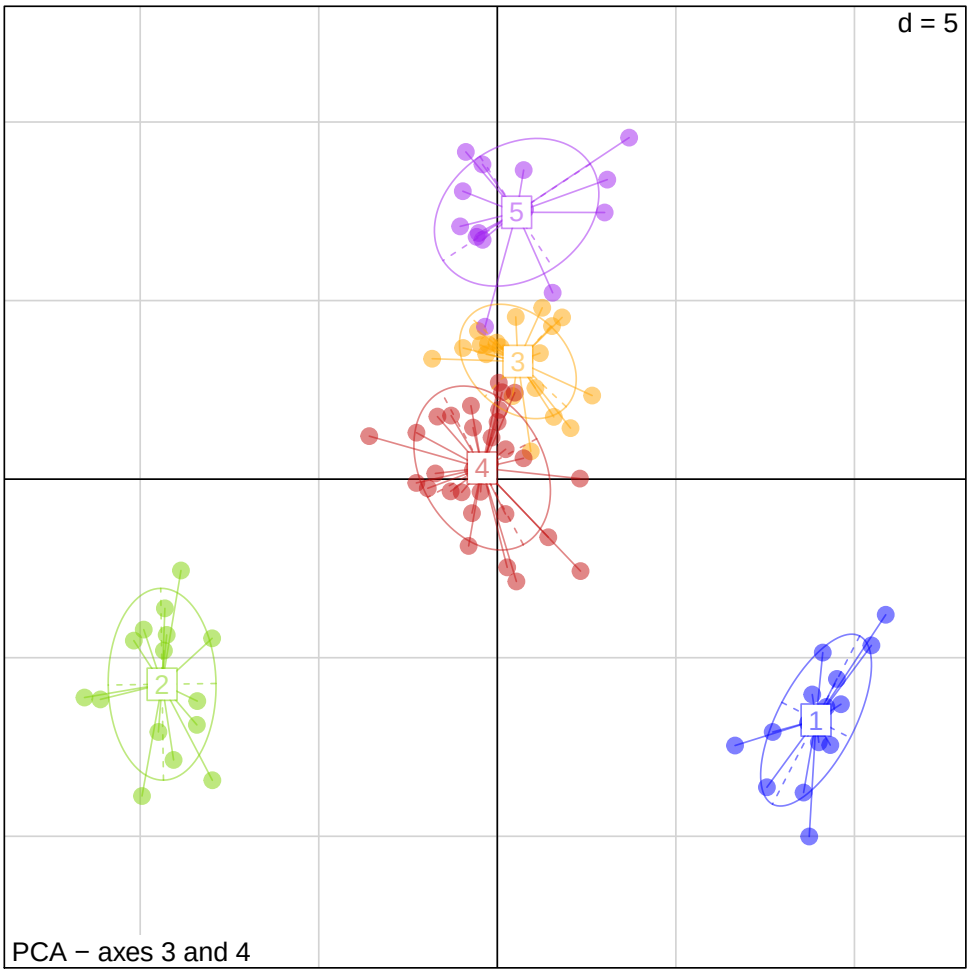
Now, we can represent these groups on top of the PCs using `s.class` (clusters are indicated by different colors and ellipses):

```
> 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?

3 Identifying SNPs linked to bacterial resistance

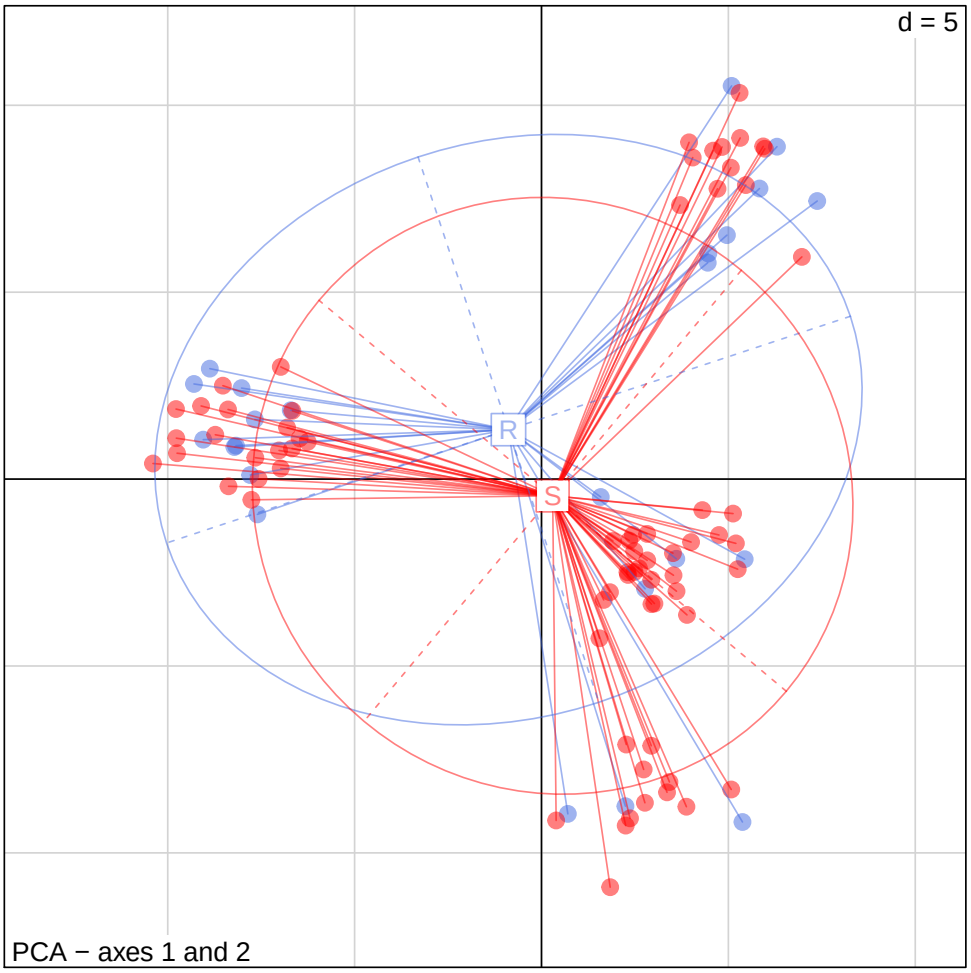
The data contained in `phen` indicate whether isolates are susceptible or resistant to a given antibiotic (`S`/`R`):

```
> head(phen,10)
```

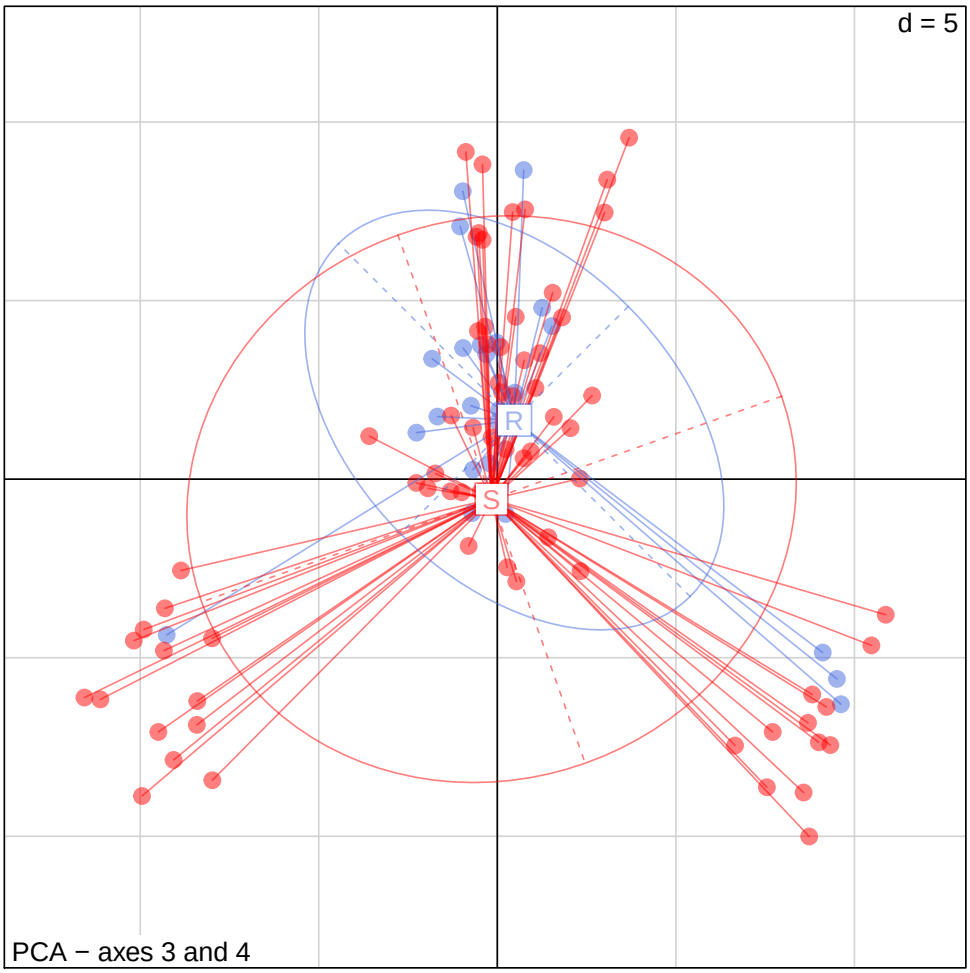
```
 [1] R S S S S S S S S S
Levels: R S
```

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 = 5.2267, df = NA, p-value = 0.2419
```

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 predefined 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`).

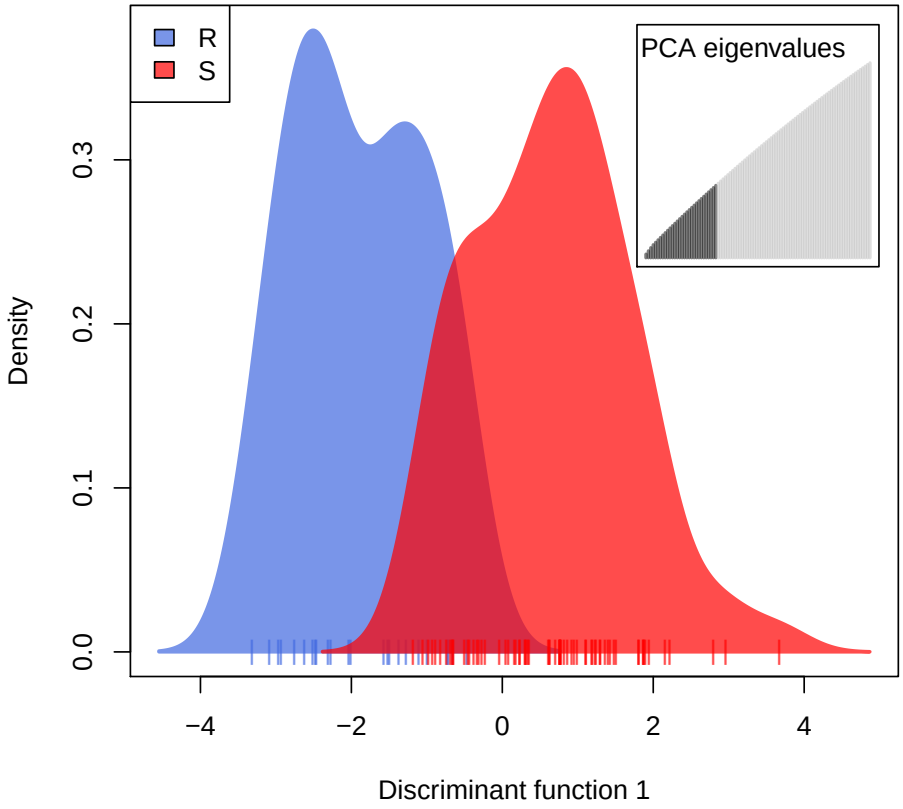
```
> dapc1 <- dapc(snps, phen)
```

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 content
1 $tab          95    30   retained PCs of PCA
2 $means        2     30   group means
3 $loadings     30    1    loadings of variables
4 $ind.coord    95    1    coordinates of individuals (principal components)
5 $grp.coord    2     1    coordinates of groups
6 $posterior    95    2    posterior membership probabilities
7 $pca.loadings 10000 30   PCA loadings of original variables
8 $var.contr    10000 1    contribution of original variables
```

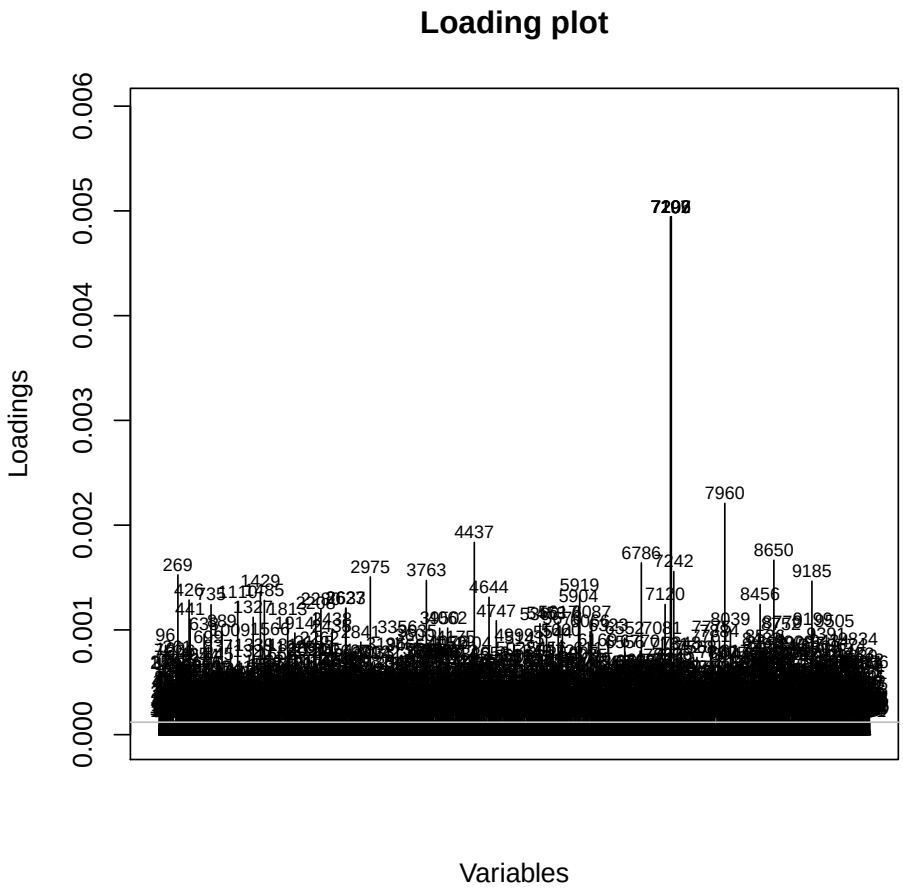
The function `scatter` can be used to visualize the results of DAPC. It produces usual plots of the principal components, using colors and ellipses to indicate groups. However, whenever only one axis has been retained, `scatter` plots the density of the individuals on the first principal component:

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

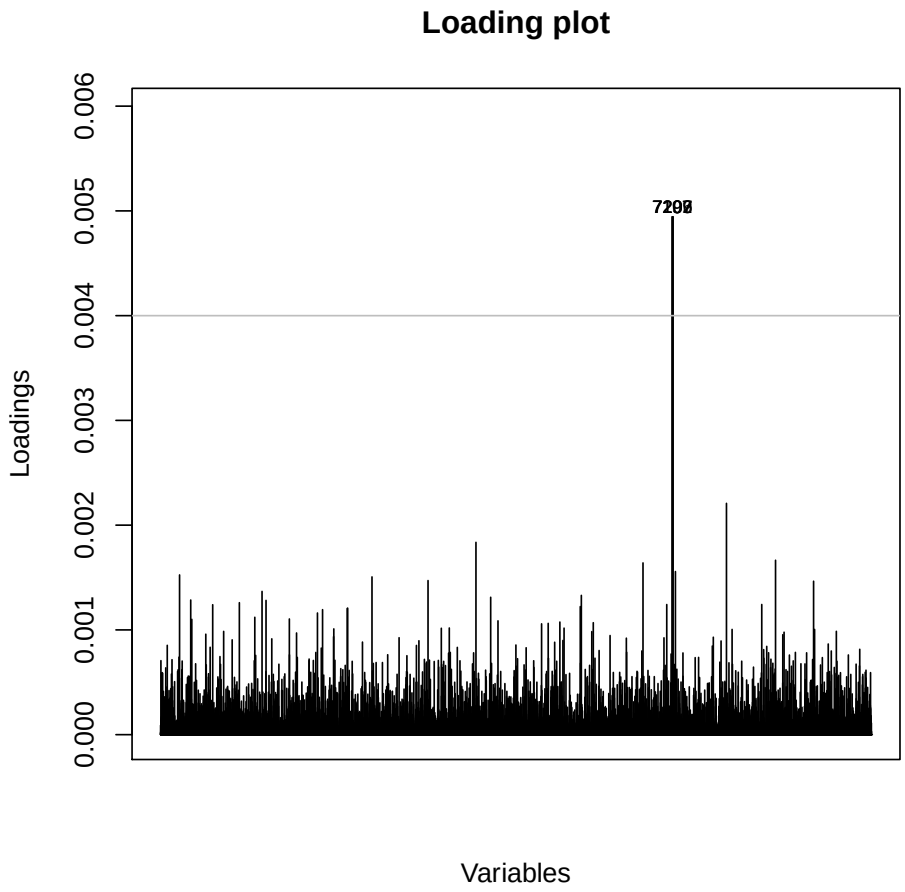


The contribution of each variable to the separation of the two groups (susceptible/resistant) is stored in `dapc1$var.contr`; it can be visualized using `loadingplot`, which displays all contributions as bars and annotates variables with the largest contributions (see argument `threshold` in `?loadingplot`):

```
> 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
0.004943821 0.004943821 0.004943821 0.004943821 0.004943821
```

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)

       ind1        ind2        ind3        ind4        ind5        ind6
"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
ind1     R    1-1-1-1-1
ind2     S    0-0-0-0-0
ind3     S    0-0-0-0-0
ind4     S    0-0-0-0-0
ind5     S    0-0-0-0-0
ind6     S    0-0-0-0-0
ind7     S    0-0-0-0-0
ind8     S    0-0-0-0-0
ind9     S    0-0-0-0-0
ind10    S    0-0-0-0-0
```

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

```
      phen sel.profiles
ind86    S    0-0-0-0-0
ind87    S    0-0-0-0-0
ind88    S    0-0-0-0-0
ind89    S    0-0-0-0-0
ind90    S    0-0-0-0-0
ind91    R    1-1-1-1-1
ind92    S    0-0-0-0-0
ind93    S    0-0-0-0-0
ind94    R    1-1-1-1-1
ind95    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?

References

[1] S. Dray and A.-B. Dufour. The ade4 package: implementing the duality diagram for ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007.

[2] H. Hotelling. Analysis of a complex of statistical variables into principal components. *The Journal of Educational Psychology*, 24:417–441, 1933.

[3] H. Hotelling. Analysis of a complex of statistical variables into principal components (continued from september issue). *The Journal of Educational Psychology*, 24:498–520, 1933.

[4] T. Jombart. adegenet: a R package for the multivariate analysis of genetic markers. *Bioinformatics*, 24:1403–1405, 2008.

[5] T. Jombart, S. Devillard, and F. Balloux. Discriminant analysis of principal components: a new method for the analysis of genetically structured populations. *BMC Genetics*, 11(1):94, 2010.

[6] T. Jombart, S. Devillard, A.-B. Dufour, and D. Pontier. Revealing cryptic spatial patterns in genetic variability by a new multivariate method. *Heredity*, 101:92–103, 2008.

[7] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[8] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.