Multivariate analysis of genetic data: exploring group diversity

Thibaut Jombart[*], Caitlin Collins

Imperial College London

MRC Centre for Outbreak Analysis and Modelling

October 28, 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). This second part will include two studies of the genetic makeup of *Elizabethis nonsensicus* populations, as well as an investigation of the origins of cattle allegedly abducted by aliens.

[*] `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:

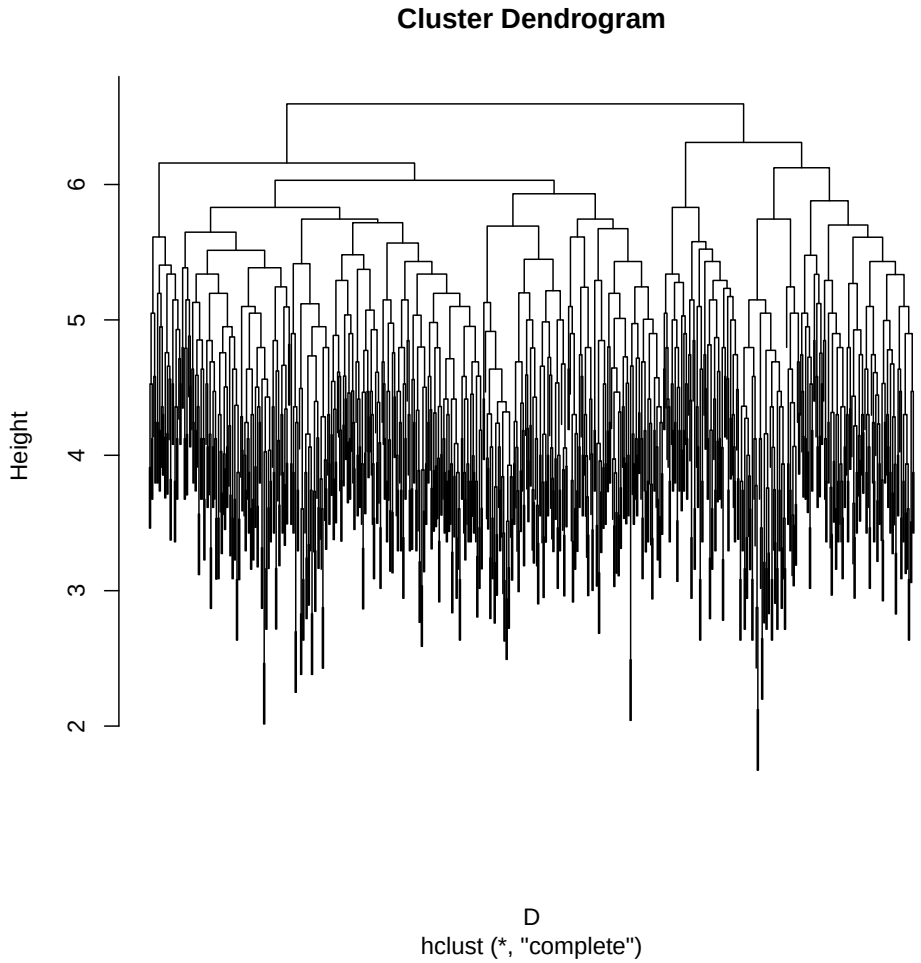
```
library(adegenet)
data(microbov)
X <- scaleGen(microbov, missing="mean",scale=FALSE)
D <- dist(X)
```

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,labels=FALSE)
```



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)
```

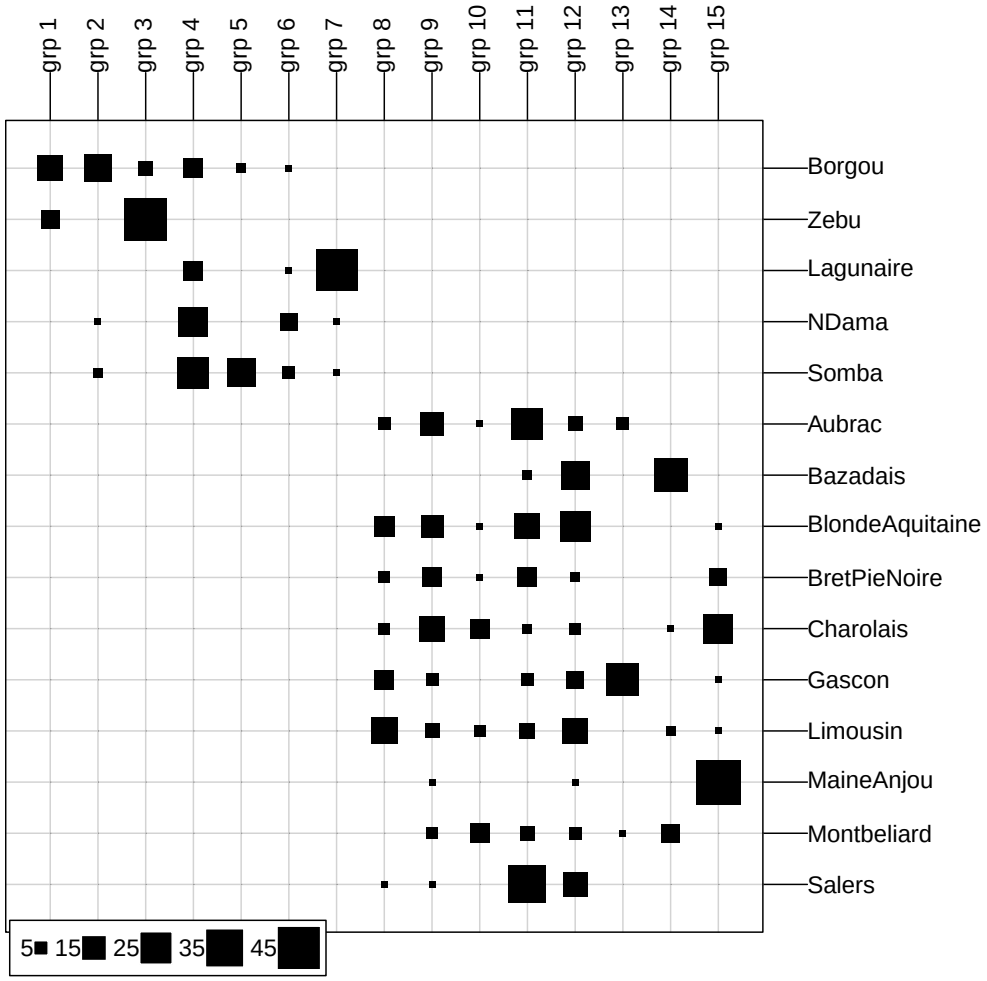
```
##
## grp  AF  FR
##   1 231   0
##   2   0 473
```

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 the accessor `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`:

```
grp <- cutree(h1, k=15)
tab <- table(pop(microbov), grp)
```

```
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

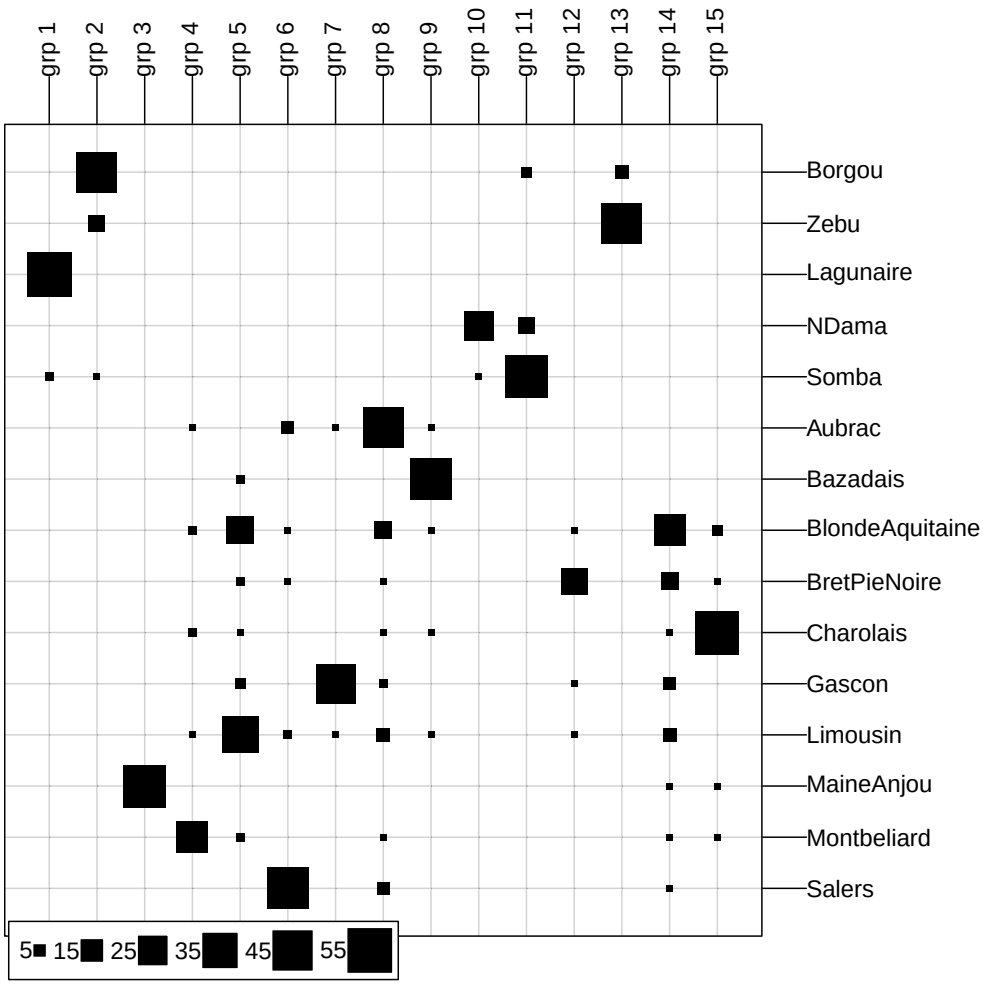
K-means is another, non-hierarchical approach for defining genetic clusters. While basic K-means is implemented in the function `kmeans`, the function `find.clusters` provides a computer-efficient implementation which first reduces the dimensionality of the data (using PCA), and optionally allows for 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`).

```
set.seed(1)
grp <- find.clusters(microbov, n.pca=100, n.clust=15)
```

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:

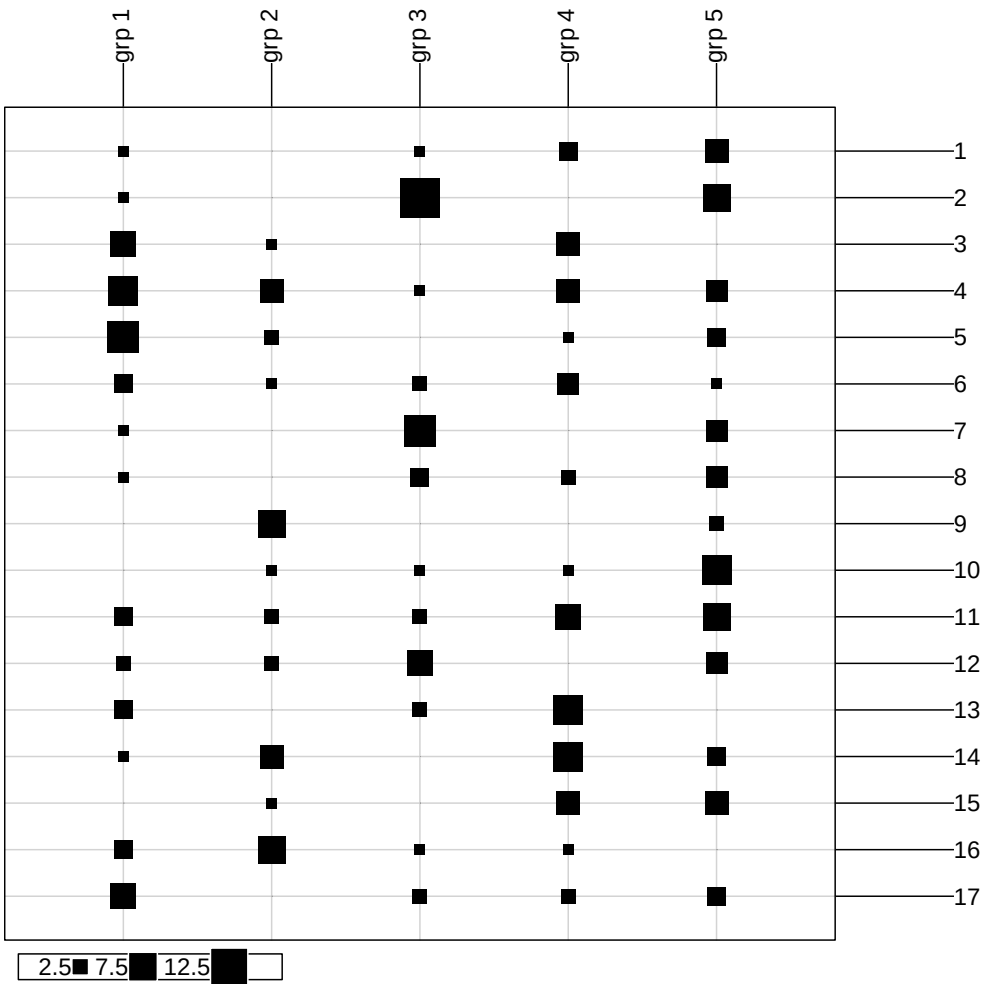
```
table.value(table(pop(microbov), grp$grp), col.lab=paste("grp", 1:15))
```



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?

```
data(nancycats)
grp <- find.clusters(nancycats, n.pca=100, n.cl=5)
table.value(table(pop(nancycats), grp$grp), col.lab=paste("grp",1:5))
```


grp 1
grp 2
grp 3
grp 4
grp 5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
2.5
7.5
12.5

2 Describing group diversity: *Elizabethis nonsensicus* populations

2.1 *Elizabethis nonsensicus*: first contact

The first study of group diversity focuses on *Elizabethis nonsensicus*, a diploid plant well-known for having a number of cryptic sub-species. A total of 600 individual plants have been sampled in the Leuven countryside and genotyped for 30 microsatellite markers. We first load the dataset, which has already been converted to a `genind` object:

```
load(url("http://adegenet.r-forge.r-project.org/files/Leuven2014/Enonsensicus1.RData"),
     verbose=TRUE)

## Loading objects:
##   Enonsensicus1

Enonsensicus1

##
##    ######################
##    ### Genind object ###
##    ######################
## - genotypes of individuals -
##
## S4 class:  genind
## @call: read.fstat(file = file, missing = missing, quiet = quiet)
##
## @tab:  600 x 140 matrix of genotypes
##
## @ind.names: vector of  600 individual names
## @loc.names: vector of  30 locus names
## @loc.nall: number of alleles per locus
## @loc.fac: locus factor for the  140 columns of @tab
## @all.names: list of  30 components yielding allele names for each locus
## @ploidy:  2
## @type:  codom
```

```
##
## Optional contents:
## @pop:  - empty -
## @pop.names:  - empty -
##
## @other: - empty -
```

The main goal of the study is to assess whether the sampled plants all belong to the same panmictic population, or whether sub-populations can be identified. First, use `find.clusters` to identify the number and nature of potential genetic clusters, and store the result in an object called `grp1`.

```
grp1 <- find.clusters(Enonsensicus1, n.pca=40, n.clust=6)
```

How many clusters do you identify? Are these dependent on how many principal components (PCs) you retain? What are the respective group sizes?

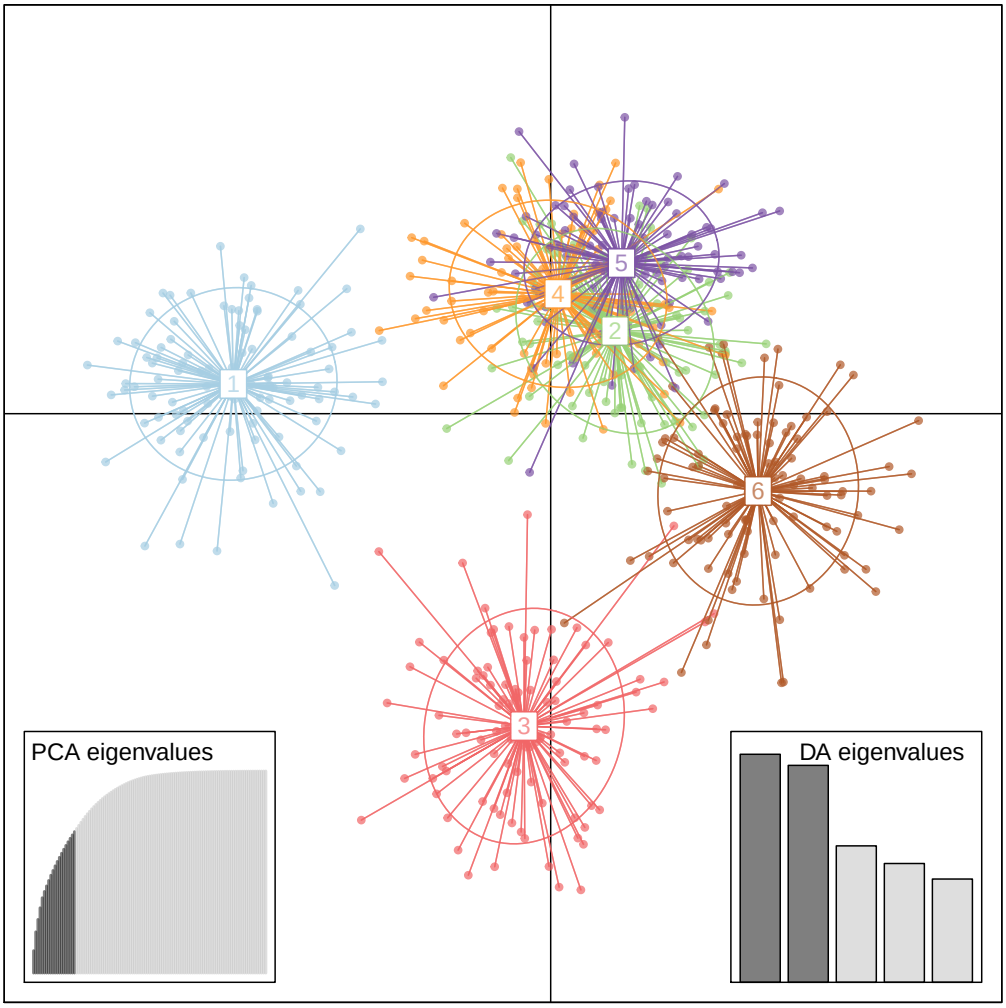
We want to assess the relationships between these groups using DAPC. Using the following command, perform the DAPC and store the results in a new object called `dapc1`:

```
dapc1 <- dapc(Enonsensicus1, pop=grp1$grp, scale=FALSE)
```

```
dapc1 <- dapc(Enonsensicus1, pop=grp1$grp, scale=FALSE, n.pca=20, n.da=5)
```

Use the function `scatter` to visualize the results. This function has many options, which are documented in `?scatter.dapc`. Your graphic should roughly ressemble:

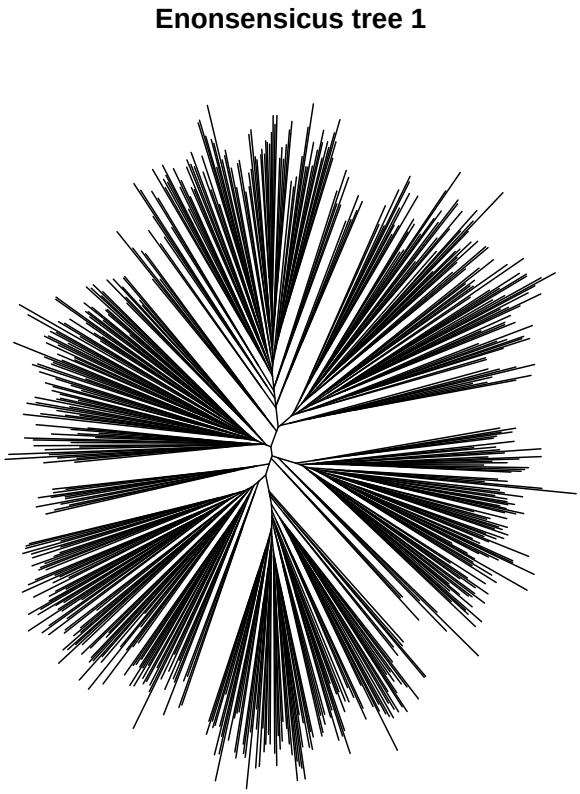
```
scatter(dapc1, col=funky(6), scree.pca=TRUE)
```



The function `scatter` plots by default the first two discriminant functions. Try visualizing other possibly relevant axes. What can you tell about the structure of this population?

It may be useful to compare these results to an alternative approach. Compute the Euclidian distances (function `dist`) between the matrix of allele frequencies `as.matrix(Enonsensicus1)`, and use them to build a Neighbour-Joining tree (implemented in the *ape* package). Examine the tree. This should look like:

```
library(ape)
tre1 <- nj(dist(as.matrix(Enonsensicus1)))
plot(tre1, type="unr", show.tip=FALSE, main="Enonsensicus tree 1")
```



What are your conclusions?

2.2 *Elizabethis nonsensicus*: the return

After the initial study of *E. nonsensicus* populations, the sampling area has been extended and new populations have been discovered. A new sample of 450 plants has been characterized for the same 30 microsatellite markers. Your task is to conduct the same kind of analysis, and assess the genetic makeup of the new population.

```
load(url("http://adegenet.r-forge.r-project.org/files/Leuven2014/Enonsensicus2.RData"),
     verbose=TRUE)

## Loading objects:
##   Enonsensicus2

Enonsensicus2

##
##    ####################
```

```
##    ### Genind object ###
##    ######################
## - genotypes of individuals -
##
## S4 class:  genind
## @call: .local(x = x, i = i, j = j, drop = drop)
##
## @tab:  450 x 160 matrix of genotypes
##
## @ind.names: vector of  450 individual names
## @loc.names: vector of  30 locus names
## @loc.nall: number of alleles per locus
## @loc.fac: locus factor for the  160 columns of @tab
## @all.names: list of  30 components yielding allele names for each locus
## @ploidy:  2
## @type:  codom
##
## Optional contents:
## @pop:  - empty -
## @pop.names:  - empty -
##
## @other: a list containing: elements without names
```

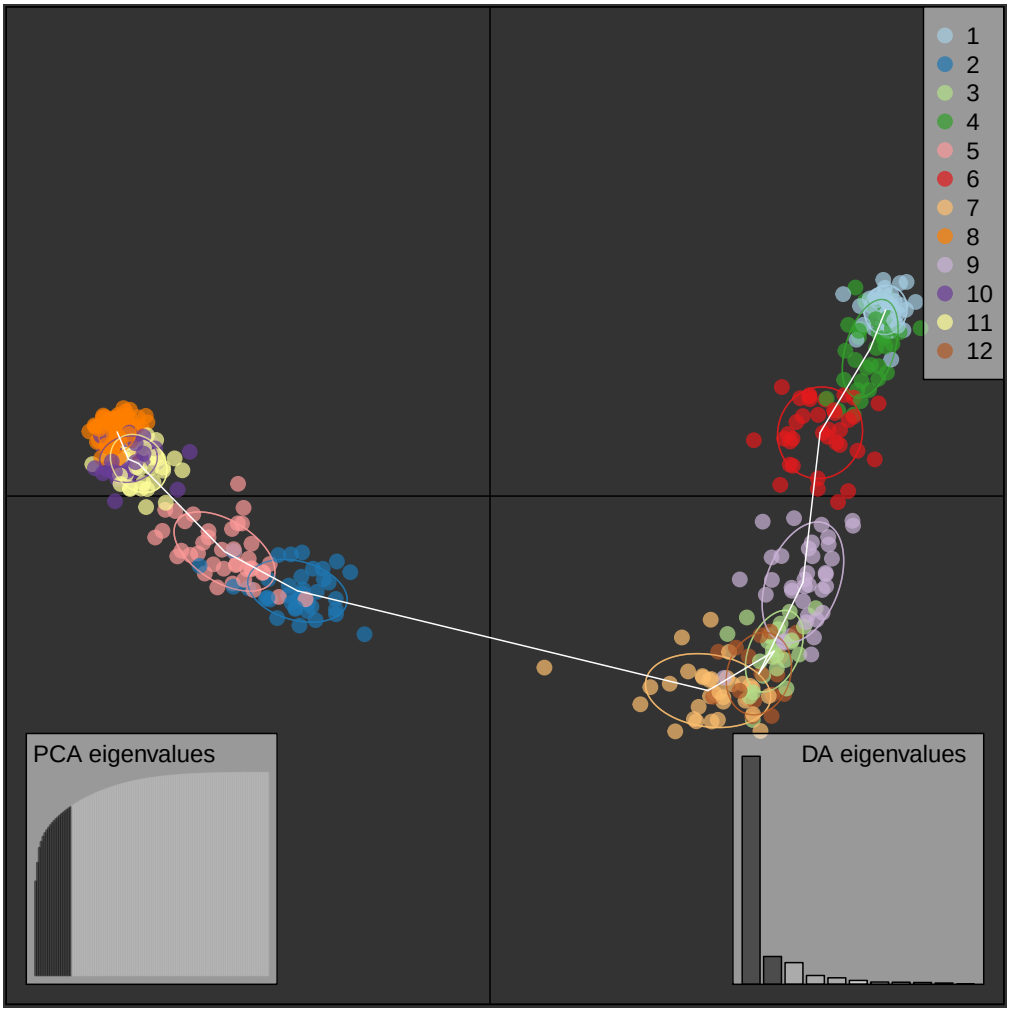
Again, use `find.clusters` to identify the number and nature of potential genetic clusters, and store the result in an object called `grp2`. How many clusters would you retain? How do the results compare to the previous study?

Try assessing the relationships between these clusters using `dapc`. If results seem unstable from one run to another, try increasing the number of starting points used in the K-means algorithm (argument `n.start`).

```
grp2 <- find.clusters(Enonsensicus2, scale=FALSE, n.start=30,
                      n.pca=80, n.clust=12)
dapc2 <- dapc(Enonsensicus2, pop=grp2$grp, scale=FALSE,
              n.pca=20, n.da=5)
```

Use the function scatter to visualize the results. Specify that you want the *minimum spanning tree* added to link together the closest populations. With a bit of customisation (see `?scatter.dapc`), your graphic should ressemble:

```
scatter(dapc2, col=funky(12), legend=TRUE, mstree=TRUE,
        cstar=0, axesell=FALSE, clab=0, cex=2, bg=grey(.2),
        scree.pca=TRUE,segcol="white")
```



What can you say about the structure of this population? Assuming this structure is essentially spatial, what kind of spatial processes could have generated the observed patterns?

3 Describing group diversity: cattle breed discrimination and alien abductions

3.1 Choosing how many components to retain

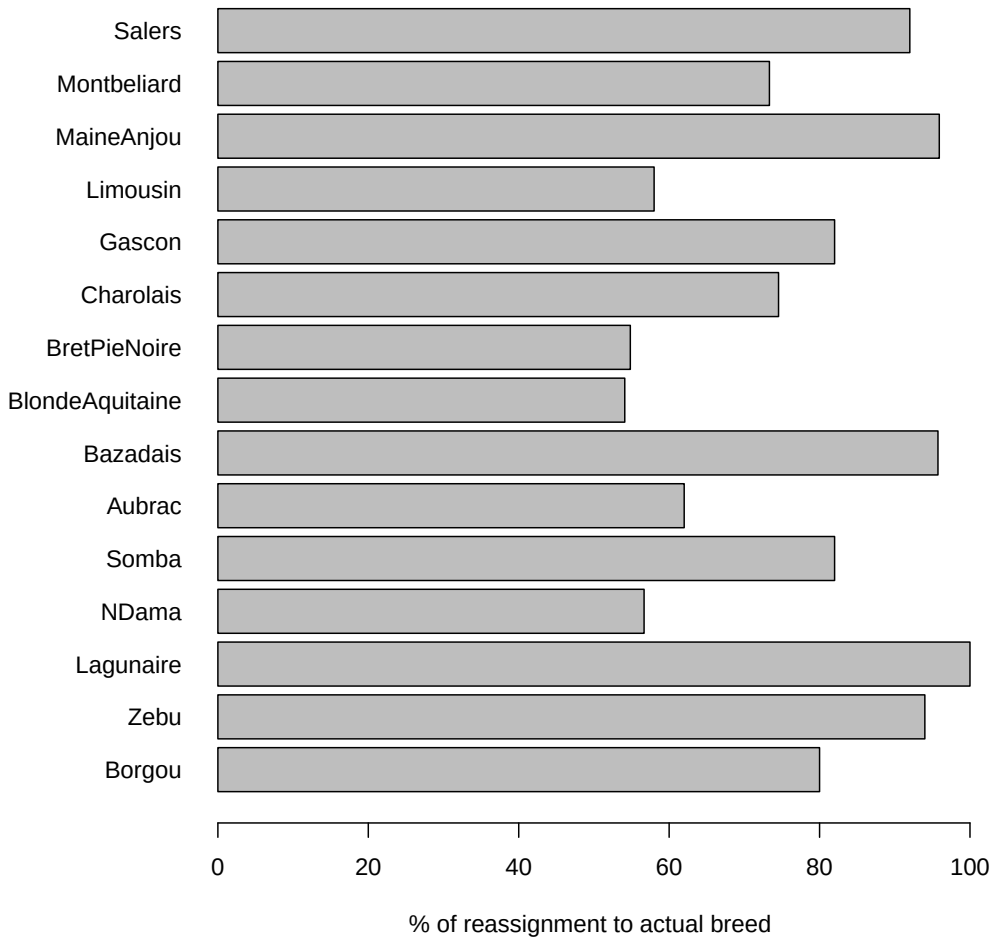
DAPC relies on a 'simplification' of the data using a PCA as a prior step to Discriminant Analysis. As always in multivariate analysis, the choice of the number of PCs to retain is not trivial. Let us illustrate the impact of this choice on the results using the `microbov` dataset (704 cattles of 15 breeds typed for 30 microsatellite markers). We first examine the % of successful reassignment (i.e., quality of discrimination) for different numbers of retained PCs. First, retaining only 10 PCs during the dimension-reduction step, and all discriminant functions:

```
data(microbov)
microbov

##
##    ####################
##    ### Genind object ###
##    ####################
## - genotypes of individuals -
##
## S4 class:  genind
## @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
## @tab:  704 x 373 matrix of genotypes
##
## @ind.names: vector of  704 individual names
## @loc.names: vector of  30 locus names
## @loc.nall: number of alleles per locus
## @loc.fac: locus factor for the  373 columns of @tab
## @all.names: list of  30 components yielding allele names for each locus
## @ploidy:  2
## @type:  codom
##
## Optional contents:
## @pop:  factor giving the population of each individual
## @pop.names:  factor giving the population of each individual
##
## @other: a list containing: coun  breed  spe

temp <- summary(dapc(microbov, n.da=100, n.pca=10))$assign.per.pop*100
```

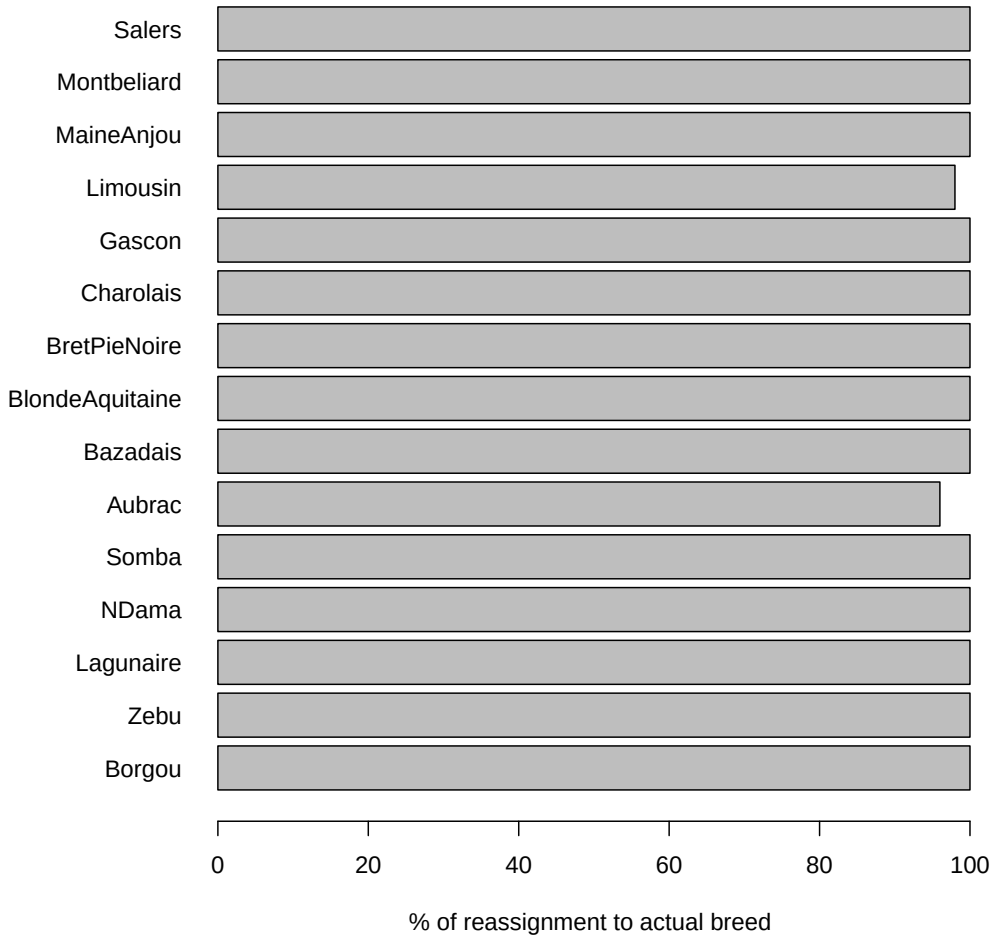
```
par(mar=c(4.5,7.5,1,1))
barplot(temp, xlab="% of reassignment to actual breed",
        horiz=TRUE, las=1)
```



We can see that some breeds are well discriminated while others are mostly overlooked by the analysis. This is because too much genetic information is lost when retaining only 10 PCs. We repeat the analysis, this time keeping 300 PCs:

```
temp <- summary(dapc(microbov, n.da=100, n.pca=300))$assign.per.pop*100
```

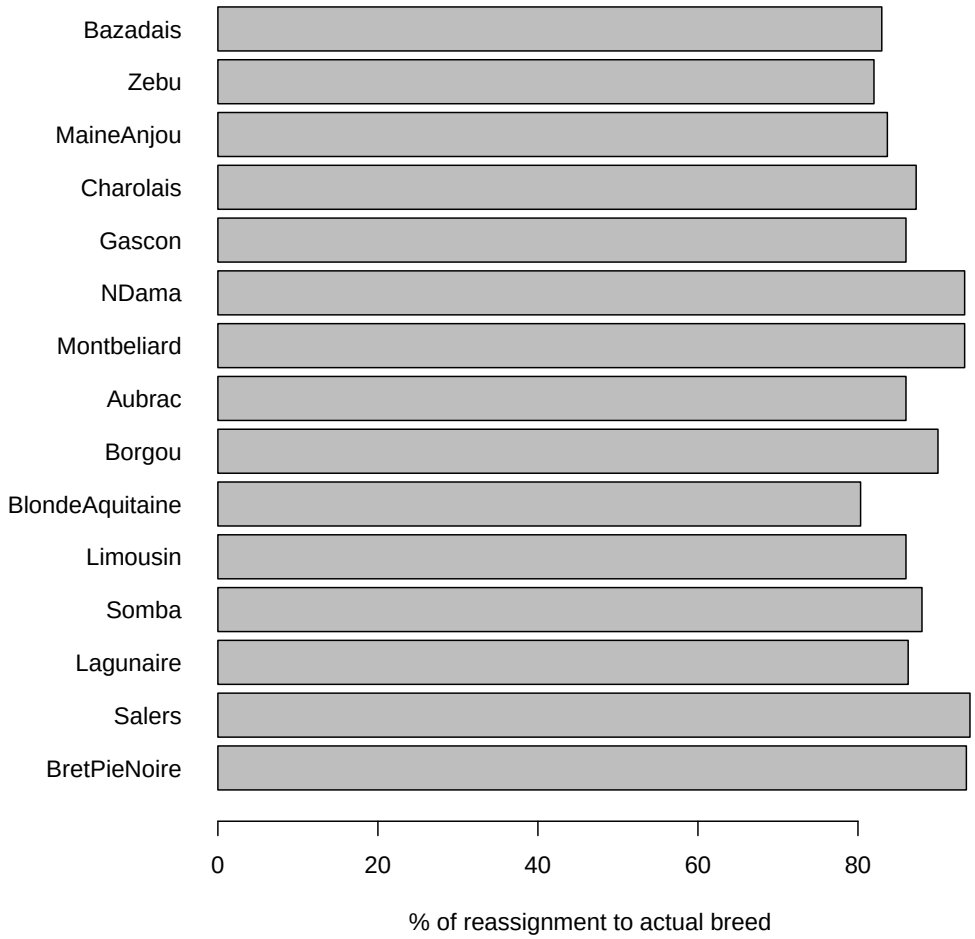
```
par(mar=c(4.5,7.5,1,1))
barplot(temp, xlab="% of reassignment to actual breed", horiz=TRUE, las=1)
```



We now obtain almost 100% discrimination for all groups. Is this result satisfying? Let us try again, this time using randomised groups in the analysis:

```
x <- microbov
pop(x) <- sample(pop(x))
temp <- summary(dapc(x, n.da=100, n.pca=300))$assign.per.pop*100
```

```
par(mar=c(4.5,7.5,1,1))
barplot(temp, xlab="% of reassignment to actual breed", horiz=TRUE, las=1)
```



Groups have been randomised, and yet we still obtain very good discrimination. Why is this?

In attempting to summarise high-dimensional data in a small number of meaningful discriminant functions, DAPC must manage a trade-off. If too *few* PCs (with respect to the number of individuals) are retained, useful information will be excluded from the analysis, and the resulting model will not be informative enough to accurately discriminate between groups. By contrast, if too *many* PCs are retained, the discriminant functions will be over-fitted and capable of discriminating any clusters. In this case, the discriminant functions will be completely tailored to the dataset, and loose any ability to generalize to new or unseen data.

3.2 Using cross-validation

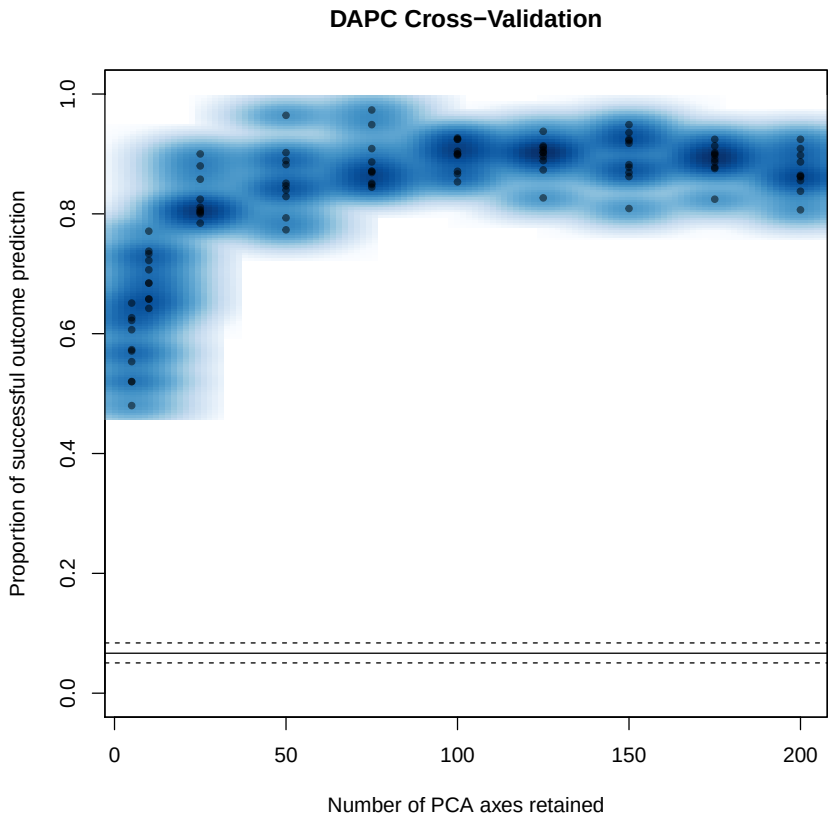
As discussed above, choosing the 'right' number of PCs in DAPC is not a trivial task. As the main goal could be formulated as finding the number of PCs which '*maximizes the probability of assigning new individuals to their actual group*', one natural approach to address this issue is cross-validation. Cross-validation (function `xvalDapc`) provides an objective optimisation procedure for identifying the 'goldilocks point' in the trade-off between retaining too few

and too many PCs in the model. In cross-validation, the data is divided into two sets: a training set (typically comprising 90% of the data) and a validation set (which contains the remainder (by default, 10%) of the data). With `xvalDapc`, the validation set is selected by stratified random sampling: this ensures that at least one member of each group or population in the original data is represented in both training and validation sets.

DAPC is carried out on the training set with variable numbers of PCs retained, and the degree to which the analysis is able to accurately predict the group membership of excluded individuals (those in the validation set) is used to identify the optimal number of PCs to retain. At each level of PC retention, the sampling and DAPC procedures are repeated `n.rep` times. Let us apply this method to the `microbov` dataset:

```
mat <- as.matrix(na.replace(microbov, method="mean"))
grp <- pop(microbov)

xval <- xvalDapc(mat, grp, n.pca.max=200, training.set=0.9,
                 result="groupMean", scale=FALSE, n.rep=10,
                 n.pca=c(5,10,seq(25,by=25,to=200)),
                 xval.plot = TRUE)
```



When `xval.plot` is `TRUE`, a scatterplot of the DAPC cross-validation is generated. The number of PCs retained in each DAPC varies along the x-axis, and the proportion of

successful outcome prediction varies along the y-axis. Individual replicates appear as points, and the density of those points in different regions of the plot is displayed in blue.

```
names(xval)

## [1] "Cross-Validation Results"
## [2] "Median and Confidence Interval for Random Chance"
## [3] "Mean Successful Assignment by Number of PCs of PCA"
## [4] "Number of PCs Achieving Highest Mean Success"
## [5] "Root Mean Squared Error by Number of PCs of PCA"
## [6] "Number of PCs Achieving Lowest MSE"
## [7] "DAPC"

xval[2:6]

## $`Median and Confidence Interval for Random Chance`
##    2.5%     50%   97.5%
## 0.05053 0.06668 0.08387
##
## $`Mean Successful Assignment by Number of PCs of PCA`
##      5     10     25     50     75    100    125    150    175    200
## 0.5724 0.6998 0.8271 0.8571 0.8873 0.8969 0.8958 0.8949 0.8893 0.8707
##
## $`Number of PCs Achieving Highest Mean Success`
## [1] "100"
##
## $`Root Mean Squared Error by Number of PCs of PCA`
##      5     10     25     50     75    100    125    150    175    200
## 0.4307 0.3028 0.1767 0.1522 0.1200 0.1060 0.1079 0.1126 0.1137 0.1335
##
## $`Number of PCs Achieving Lowest MSE`
## [1] "100"
```

The ideal result of this cross-validation procedure would be a bell-shaped relationship, indicating the optimal number of PCs to retain. Here, most solutions beyond 75 PCs seem equivalent. Make your own DAPC of `microbov` choosing your preferred number of PCs and store the result in `dapc.bov`.

```
dapc.bov <- dapc(microbov,n.pca=100,n.da=14)
```

3.3 Alien abductions

```
set.seed(1)

## generate new Salers individuals
salers <- seppop(microbov)$"Salers"
unknown1 <- hybridize(salers, salers, hyb.label="unknown",n=10)
pop(unknown1) <- unknown1$call <- NULL


## generate new Zebu individuals
Zebu <- seppop(microbov)$"Zebu"
unknown2 <- hybridize(Zebu, Zebu, hyb.label="unknown",n=5)

## generate new Aubrac individuals
Aubrac <- seppop(microbov)$"Aubrac"
temp <- hybridize(Aubrac, Aubrac, hyb.label="unknown",n=5)
temp$call <- NULL
unknown2 <- repool(unknown2,temp)

## generate new Somba individuals
Somba <- seppop(microbov)$"Somba"
temp <- hybridize(Somba, Somba, hyb.label="unknown",n=10)
unknown2 <- repool(unknown2,temp)

## repool all
unknown2 <- repool(unknown2, microbov)[1:nInd(unknown2),]
pop(unknown2) <- unknown2$call <- NULL
```

After your analysis of the optimal discrimination of cattle breeds, you are contacted by some governmental officers to investigate the possible origin of blood samples coming from cattles allegedly abducted by aliens. Blood samples have been found in two different saucepans. The resulting datasets are respectively named `unknown1` and `unknown2` (governmental officers notoriously lack originality). The files are available as `RData` from the following URLs:

```
load(url("http://adegenet.r-forge.r-project.org/files/Leuven2014/unknown1.RData"),
     verbose=TRUE)

## Loading objects:
##   unknown1

unknown1

##
##    ####################
##    ### Genind object ###
##    ####################
## - genotypes of individuals -
##
## S4 class:  genind
## @call: NULL
##
## @tab:  10 x 373 matrix of genotypes
##
## @ind.names: vector of  10 individual names
## @loc.names: vector of  30 locus names
## @loc.nall: number of alleles per locus
## @loc.fac: locus factor for the  373 columns of @tab
## @all.names: list of  30 components yielding allele names for each locus
## @ploidy:  2
## @type:  codom
##
## Optional contents:
## @pop:  - empty -
## @pop.names:  - empty -
##
## @other: - empty -

load(url("http://adegenet.r-forge.r-project.org/files/Leuven2014/unknown2.RData"),
     verbose=TRUE)

## Loading objects:
##   unknown2

unknown2

##
##    ####################
##    ### Genind object ###
##    ####################
```

```
## - genotypes of individuals -
##
## S4 class:  genind
## @call: NULL
##
## @tab:  20 x 373 matrix of genotypes
##
## @ind.names: vector of  20 individual names
## @loc.names: vector of  30 locus names
## @loc.nall: number of alleles per locus
## @loc.fac: locus factor for the  373 columns of @tab
## @all.names: list of  30 components yielding allele names for each locus
## @ploidy:  2
## @type:  codom
##
## Optional contents:
## @pop:  - empty -
## @pop.names:  - empty -
##
## @other: a list containing: elements without names
```

As seen before, DAPC can be used to predict group memberships of individuals based on their scores on the discriminant functions. One advantage of this approach is that the same can be done with new individuals, provided the new data have exactly the same variables as the ones used in the analysis. First, let us check that the loci and alleles in the two new datasets (`unknown1` and `unknown2`) are indeed identical:

```
## look at the first ones
head(locNames(microbov, withAlleles=TRUE),10)

##  [1] "INRA63.167" "INRA63.171" "INRA63.173" "INRA63.175" "INRA63.177"
##  [6] "INRA63.179" "INRA63.181" "INRA63.183" "INRA63.185" "INRA5.137"

head(locNames(unknown1, withAlleles=TRUE),10)

##  [1] "INRA63.167" "INRA63.171" "INRA63.173" "INRA63.175" "INRA63.177"
##  [6] "INRA63.179" "INRA63.181" "INRA63.183" "INRA63.185" "INRA5.137"

head(locNames(unknown2, withAlleles=TRUE),10)

##  [1] "INRA63.167" "INRA63.171" "INRA63.173" "INRA63.175" "INRA63.177"
##  [6] "INRA63.179" "INRA63.181" "INRA63.183" "INRA63.185" "INRA5.137"

## look at the last ones
tail(locNames(microbov, withAlleles=TRUE),10)
```

```
##  [1] "TGLA53.191" "SPS115.242" "SPS115.244" "SPS115.246" "SPS115.248"
##  [6] "SPS115.250" "SPS115.252" "SPS115.254" "SPS115.256" "SPS115.258"

tail(locNames(unknown1, withAlleles=TRUE),10)

##  [1] "TGLA53.191" "SPS115.242" "SPS115.244" "SPS115.246" "SPS115.248"
##  [6] "SPS115.250" "SPS115.252" "SPS115.254" "SPS115.256" "SPS115.258"

tail(locNames(unknown2, withAlleles=TRUE),10)

##  [1] "TGLA53.191" "SPS115.242" "SPS115.244" "SPS115.246" "SPS115.248"
##  [6] "SPS115.250" "SPS115.252" "SPS115.254" "SPS115.256" "SPS115.258"

## formally compare them
identical(locNames(microbov, withAlleles=TRUE),locNames(unknown1, withAlleles=TRUE))

## [1] TRUE

identical(locNames(microbov, withAlleles=TRUE),locNames(unknown2, withAlleles=TRUE))

## [1] TRUE
```

Note that if these were not identical, the function `repool` could be used to pool data with different alleles.

Look at the documentation of `predict.dapc`, and use the function to predict where the abducted cattles came from.

```
pred1 <- predict(dapc.bov, newdata=unknown1)
100*round(pred1$posterior,2)

##    Borgou Zebu Lagunaire NDama Somba Aubrac Bazadais BlondeAquitaine
## 01      0    0         0     0     0      0        0               0
## 02      0    0         0     0     0      0        0               0
## 03      0    0         0     0     0      0        0               0
## 04      0    0         0     0     0      0        0               0
## 05      0    0         0     0     0      0        0               0
## 06      0    0         0     0     0      0        0               0
## 07      0    0         0     0     0      0        0               0
## 08      0    0         0     0     0      0        0               0
## 09      0    0         0     0     0      0        0               0
## 10      0    0         0     0     0      0        0               0
##    BretPieNoire Charolais Gascon Limousin MaineAnjou Montbeliard Salers
## 01            0         0      0        0          0           0    100
## 02            0         0      0        0          0           0    100
## 03            0         0      0        0          0           0    100
```

```
## 04            0         0      0        0          0            0    100
## 05            0         0      0        0          0            0    100
## 06            0         0      0        0          0            0    100
## 07            0         0      0        0          0            0    100
## 08            2         0      0        0          0            0     98
## 09            0         0      0        0          0            0    100
## 10            0         0      0        0          0            0    100
```

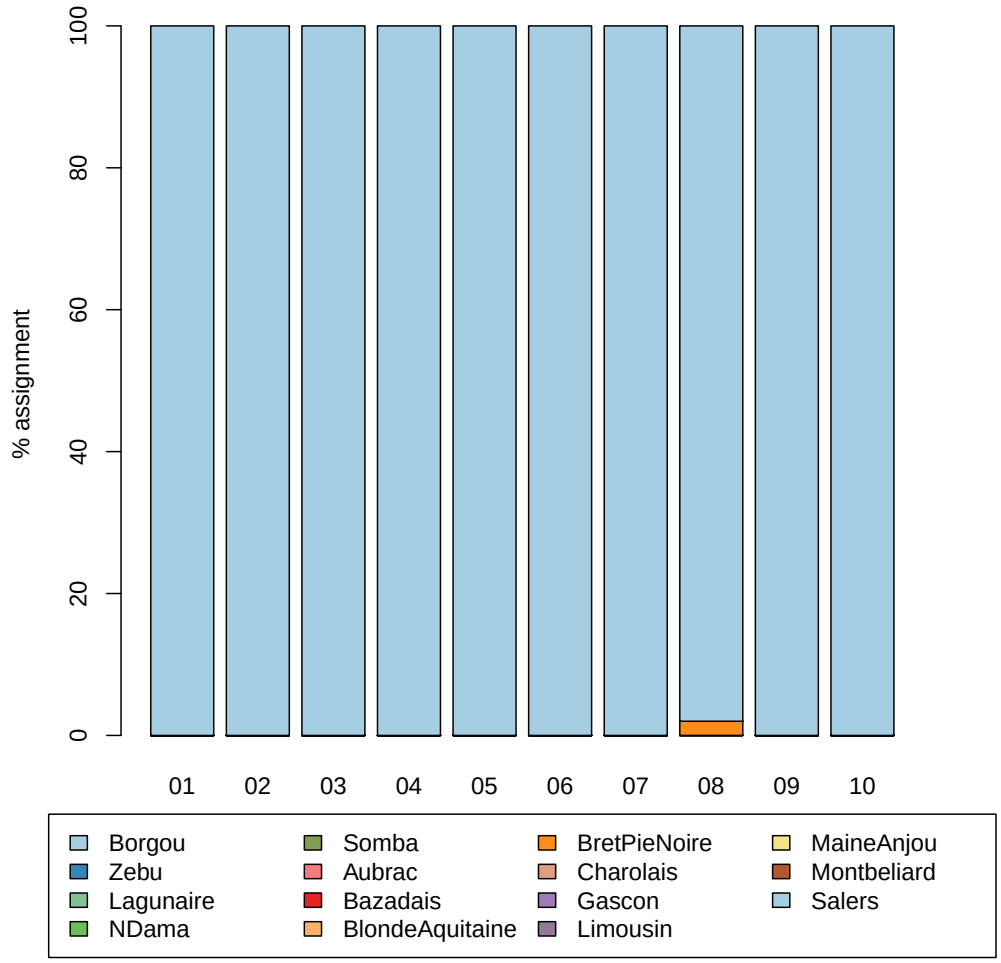
```
pred2 <- predict(dapc.bov, newdata=unknown2)
100*round(pred2$posterior,2)
```

```
##    Borgou Zebu Lagunaire NDama Somba Aubrac Bazadais BlondeAquitaine
## 01      0  100         0     0     0      0        0               0
## 02      0  100         0     0     0      0        0               0
## 03      0  100         0     0     0      0        0               0
## 04      0  100         0     0     0      0        0               0
## 05      2   98         0     0     0      0        0               0
## 06      0    0         0     0     0    100        0               0
## 07      0    0         0     0     0    100        0               0
## 08      0    0         0     0     0    100        0               0
## 09      0    0         0     0     0    100        0               0
## 10      0    0         0     0     0    100        0               0
## 11      0    0         0     0   100      0        0               0
## 12      0    0         0     0   100      0        0               0
## 13      0    0         0     0   100      0        0               0
## 14      0    0         0    37    63      0        0               0
## 15      0    0         0     0   100      0        0               0
## 16      0    0         0     0   100      0        0               0
## 17      0    0         0     0   100      0        0               0
## 18      0    0         0     0   100      0        0               0
## 19      0    0         0     0   100      0        0               0
## 20      0    0         0     0   100      0        0               0
##    BretPieNoire Charolais Gascon Limousin MaineAnjou Montbeliard Salers
## 01            0         0      0        0          0            0      0
## 02            0         0      0        0          0            0      0
## 03            0         0      0        0          0            0      0
## 04            0         0      0        0          0            0      0
## 05            0         0      0        0          0            0      0
## 06            0         0      0        0          0            0      0
## 07            0         0      0        0          0            0      0
## 08            0         0      0        0          0            0      0
## 09            0         0      0        0          0            0      0
## 10            0         0      0        0          0            0      0
## 11            0         0      0        0          0            0      0
## 12            0         0      0        0          0            0      0
```

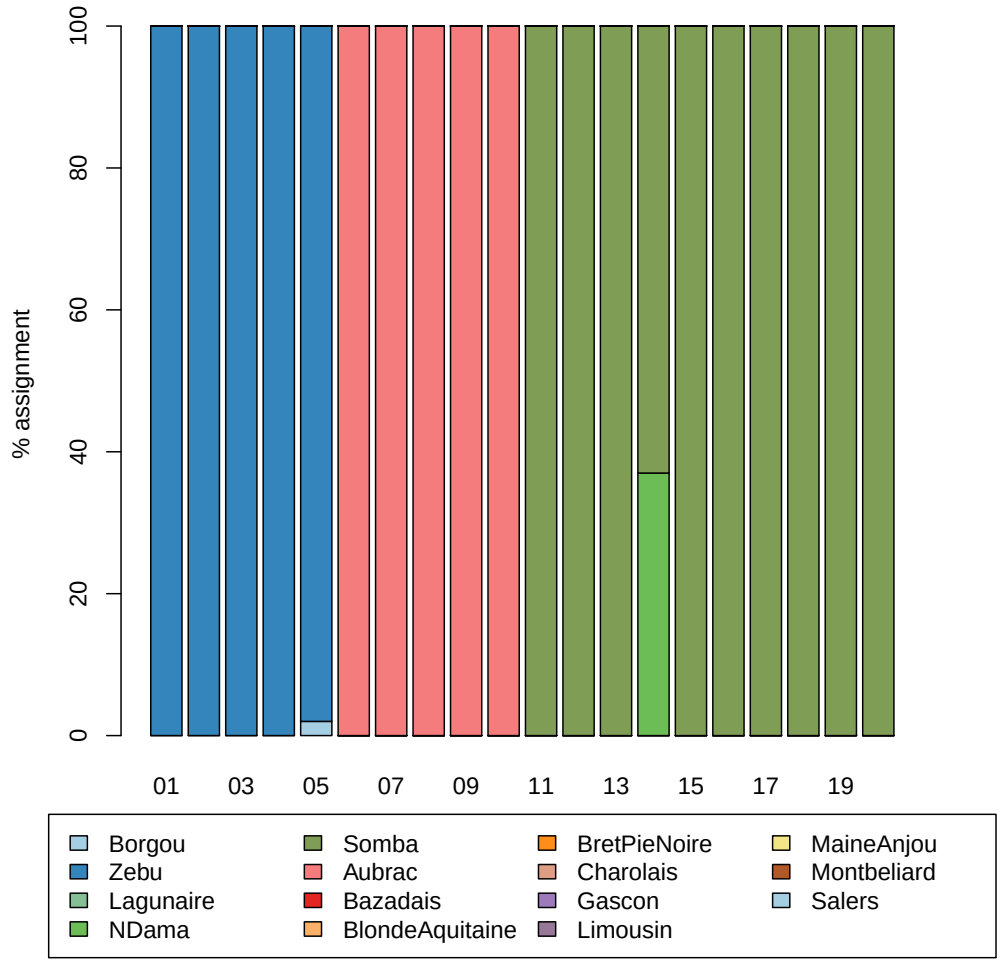
```
## 13  0  0  0  0  0  0  0
## 14  0  0  0  0  0  0  0
## 15  0  0  0  0  0  0  0
## 16  0  0  0  0  0  0  0
## 17  0  0  0  0  0  0  0
## 18  0  0  0  0  0  0  0
## 19  0  0  0  0  0  0  0
## 20  0  0  0  0  0  0  0
```

If the output of `predict` are called `pred1` and `pred2`, you can visualise the predicted group memberships using:

```
par(xpd=TRUE, mar=c(8,4,2,3))
barplot(t(100*round(pred1$posterior,2)), col=funky(14),ylab="% assignment")
legend("bottom", fill=funky(14),
       legend=levels(pop(microbov)),
       ncol=4,inset=c(0,-.3))
```



```
par(xpd=TRUE, mar=c(8,4,2,3))
barplot(t(100*round(pred2$posterior,2)), col=funky(14),ylab="% assignment")
legend("bottom", fill=funky(14),
       legend=levels(pop(microbov)),
       ncol=4,inset=c(0,-.3))
```



What are your conlcusions?

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")
```