**Lab 7: Classification and Ordination**

In this lab you are going to try more ordination techniques. The method of choice can depend on characteristics of the specific dataset you have collected, and moreover, there is usually several alternative analysis methods that would do a good job for a particular dataset. One important thing to take away from this lab is that it is not always clear cut when one should use one method over another, sometimes it is just based on preference, the particulars of the question you are asking, and traditions within certain fields. That can sound both frustrating and reassuring.

**7.1 K means clustering**

If you want to get a quick answer to the question: How many clusters (groups) does my data set likely consist of? You can use *K means* clustering. You can run K-means clustering on distance-based data or on raw continuous data, from which R can calculate distances. This clustering algorithm works by ordering data points into K number of clusters by identifying group structure so that the mean distance from the individual data points to the center (often called centroid) of their cluster is minimized, while distance between the centroids of different clusters is maximized.

We start this lab by performing non-hierarchical K-means clustering on the tadpole data set from the previous lab.

First install and load the necessary packages for many of the following exercises (you may need to install them first if you haven’t already).

pgs <- c("factoextra", "NbClust", "cluster", "fpc", "gridExtra")

install.packages("pgs")

library(factoextra)

library(NbClust)

library(cluster)

library(fpc)

library(gridExtra)

Then, we load the data set and scale it. The default to the function ***scale()*** in R is to mean-center the data (the new mean = 0) and then divide it by the standard deviation (this is what we call *mean- and variance standardization* and is what you have been doing when performing PCA on the correlation matrix of a dataset). You can find out more by typing “***?scale***”.

tadpole <- read.table("tadpole\_food.txt", header=TRUE, stringsAsFactors=T)

dat <- scale(tadpole[,3:9]) #mean and varivance std of columns 3-9

The *centers* part of the k-means function below specifies the number of clusters we want to group the data into, whereas *nstart* gives the number of rows (observations) in the data chosen to create the initial centroids upon which the iteration process proceeds (apparently choosing a higher number here can stabilize the performance; the default value is *nstart* = 1….). We will test 4 different groupings.

k2 <- kmeans(dat, centers = 2, nstart = 25)

k3 <- kmeans(dat, centers = 3, nstart = 25)

k4 <- kmeans(dat, centers = 4, nstart = 25)

k5 <- kmeans(dat, centers = 5, nstart = 25)

plot the results for the different number of clusters to compare

p2 <- fviz\_cluster(k2, geom = "point", data = dat) + ggtitle("k = 2")

p3 <- fviz\_cluster(k3, geom = "point", data = dat) + ggtitle("k = 3")

p4 <- fviz\_cluster(k4, geom = "point", data = dat) + ggtitle("k = 4")

p5 <- fviz\_cluster(k5, geom = "point", data = dat) + ggtitle("k = 5")

grid.arrange(p2, p3, p4, p5, nrow = 2)

**7.1a *Looking at the pre-assigned clusters, which one do you think fits the data the best?***

*But how do we really know?* There are several options here. One general approach that can be applied to clustering methods is to calculate *Dunn’s index*. The Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance. The Dunn Index has a value between zero and infinity, and should be maximized. Similarly, the “silhouette” method evaluates fit by quantifying how well a given data point fits to the cluster it has been assigned to using a distance metric (like the Euclidean distance). Let’s use this method to graphically evaluate which number of clusters (K) is the best fit to the data. In a normal analysis you might run this before the K-means functions to reduce your search space.

fviz\_nbclust(dat, kmeans, method = "silhouette") + labs(subtitle = "Silhouette method")

**7.1b** ***Which value of K (clusters) appears to be the optimal? Does it correspond to what you thought was the optimal grouping in the previous question and from the previous lab?*** Remember, unlike the previous PCA lab we don't color the groupings based on a pre-assigned factor (food treatment), but instead based on the groupings found by the algorithm.

How do the 7 traits (data columns 3-9) related to the clustering? You can assess how the two clusters are predicted to differ in the mean of each of the seven traits using:

k2$centers

**7.1c *Make sure you understand the result*.** There are many more summary outputs you can retrieve from the saved clustering results (if you are interested, you can type “k2” to go deeper), but let’s move on to hierarchical clustering .

* 1. **Hierarchical clustering**

This alternative approach to K-means clustering does not force you to choose the number of clusters that you believe your data set consist of before running the analysis. It presents the clustering in a tree like structure (dendrogram), allowing you to detect substructure in the data set. It is a more quantitative approach where branch lengths in the tree are informative concerning relative similarity (or relatedness, if you are constructing a phylogeny). If you want to read more about the method check out the following blog post:

<https://www.r-bloggers.com/hierarchical-clustering-in-r-2/>

This clustering method is often used in gene expression analysis.

We will apply the approach to the same tadpole data that we have been working with. First we need to calculate a dissimilarity matrix using Euclidean distance between our observations/samples.

d <- dist(dat, method = "euclidean")

Let’s perform hierarchical clustering using the ***hclust*** function. Use ***?hclust*** to learn more about the function. To note here are two things. First, we based our clustering on Euclidean distance by calculating our distance matrix above. We could have used another distance measure if we had wished to. Second, below we choose the “*complete*” method (sometimes called “*farthest neighbour clustering*”), which is the same method as we tried in the example from lecture number 8. The choice is a bit arbitrary (to me at least).

hc1 <- hclust(d, method = "complete" )

Plot the resulting dendrogram for different values of K

par(mfrow=c(3,1))

plot(hc1, cex = 0.6, hang = -1)

rect.hclust(hc1, k = 2, border = 2:5)

plot(hc1, cex = 0.6, hang = -1)

rect.hclust(hc1, k = 3, border = 2:5)

plot(hc1, cex = 0.6, hang = -1)

rect.hclust(hc1, k = 4, border = 2:5)

**7.2a** ***Based on the dendrogram, how many clusters are there? Does it correspond to the same number as the PCA result and K-means? How do you use the branch lengths to inform you answer/what do they mean?***

* 1. **Classical multidimensional scaling (mds) also known as principal coordinate analysis (PCoA)**

The method works on distance matrices and after dimensionality reduction, it plots data points depending on dissimilarity and their respective association to the coordinate axes. In comparison with a PCA where you want to reduce the number of dimensions of a multivariate data set in order to maximize the variance explained in as few variables as possible (PCs), a classical mds/PCoA you start with a high dimensional distance matrix (which can be calculated from the same data as you performed the PCA on) and want to figure out how the distance between points and the location of those points are situated in reduced dimensional space. The key point for this method, and similar methods, is the choice of distance measure. This measure is highly data-set specific and can impact the analysis in a negative way if the wrong one is used. On the other hand, because you can choose distance measure, MDS is more flexible than PCA.

Here you will work with a data set consisting of SNP (Single Nucleotide Polymorphism) genotypes from a metapopulation of Moor frogs. 160 frogs from 9 sites were sampled and sequenced. Due to computer limitations you will only work with a small subset of the original data set.

library(ggfortify)

We load the data

dat <- read.table("SNPdata2.txt", header=TRUE)

Let’s make sure to turn “population” into a factor so that R knows what we are doing.

dat$Pop <- factor(dat$Pop)

Turn the data into a matrix

dat2 <- as.matrix(dat[, 3:1001])

Calculate Euclidian distances between individuals based on their genotypes.

d <- dist(dat2, method=”euclidean”)

Run the multidimensional scaling, reducing it to two dimensions

fit <- cmdscale(d,k=2)

Look at the results

fit

Now we plot the results

cols = c('red', 'blue', 'black', 'steelblue', "yellow", "black", "magenta", "green", "purple")

dev.off() #restore par() settings

plot(fit, xlab="Coordinate 1", ylab="Coordinate 2", main="Classical\_MDS", type="n")

points(fit, col = dat$Pop, cex=.7, pch = 19)

legend('bottomleft', col=cols, legend=levels(dat$Pop), pch = 16, cex = 0.7)

Now let’s run a PCA on the SNP data.

SNP\_PCA <- prcomp(dat2, scale=F)

Now plot it:

autoplot(SNP\_PCA, data = dat, col = cols[dat$Pop])

**7.3a** ***Does any of the two methods perform better than the other in capturing population differences? How many clusters would you say there is? How do you interpret the distance between two points in the mds? Are you surprised about the amount of variation explained by PC1 and PC2?***

You may also run the mds on another distance measure. For example, you can run the code above, but replace the Euclidean distance matrix with one based on Manhattan distance.

***Does the ordination change much?***

d2 <- dist(dat2, method=”manhattan”)