14
Multivariate Analysis

Multivariate data is used when researchers deal with several variables on each "case" in their sample. In the majority of cases investigators are dealing with multivariate problems. Sometime it may be convenient to isolate a variable and study it as a single variable problem, however in most cases the problem must be addressed by using a multivariate approach. The reason is that often variables are related to each other and therefore to elucidate relationships and dependencies it is necessary to analyze them simultaneously and with respect to each other.

There are several statistical methods used to multivariate analysis, some are descriptive and other inferential. The main objective of multivariate analysis are:

1. Data exploration
2. Classification
3. Multidimensional statistical inference
4. Graphical representation

Depending on the statistical problems, the most important methods used in multivariate analysis are:

1. Principal components analysis
2. Factorial Analysis
3. Correspondences Analysis
4. Canonical correlation
5. Discriminant Analysis
6. Cluster Analisys

In this chapter, first data type, matrix representation of data and the main inferential tools are presented. Afterwhile the techniques described above will be discussed.

14.1 Data type

In general, when preparing data for an analysis, it is convenient to organize it by creating a matrix or a data frame, where the rows contains the cases (the units of the sample) while the columns are the different variables as shown in Figure 14.5. Let's assume that we are investigating the height, age and weight of an animal population sample. Each individual would be a each row, and the values of height, age and weight in three columns.

Variable

$$\begin{array}{c} \text{C} \\ \text{a} \\ \text{s} \\ \text{e} \end{array} \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1q} \\ x_{21} & x_{22} & \cdots & x_{2q} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nq} \end{pmatrix}$$

Fig. 14.1

for $i = 1...n$ and $j = 1...q$, where the vector $\mathbf{X} = (x_{ij}) = (X_1 \ldots X_q)$ is the column vector and $(X^1 \ldots X^n)$ is the row vector. $\mathbf{X}$ is a matrix while X_j (with $j = 1...q$) is a vector.

14.2 Summary statistics

The first analysis that is usually performed is a summary statistics of the individual variables and later begin by analysing variables by pairs. In the first case *means* and *variances* are used as described in previous chapters, and for the latter we usually take pairs of variables at a time and look at their *covariances* or *correlations*. An example is presented (Everitt, 2005) where the age and height of husband and wife is listed in a table as below for ten couples. The data file used for the example is called `hus_wife.dat` and the R script is called `multivariate_1.R`

```
   Hage Hheight Wage Wheight Hagefm
1    49    1809   43    1590     25
2    25    1841   28    1560     19
3    40    1659   30    1620     38
4    52    1779   57    1540     26
5    58    1616   52    1420     30
6    32    1695   27    1660     23
7    43    1730   52    1610     33
8    47    1740   43    1580     26
9    31    1685   23    1610     26
10   26    1735   25    1590     23
```

The means are:

```
Hage  Hheight Wage Wheight Hagefm
40.3  1728.9  38   1578    26.9
```

The variances are:

```
Hage  Hheight Wage   Wheight Hagefm
130.2 4706.9  164.6  4173.3  29.8
```

14.2.1 Covariances

The *population covariance* is defined as:

$$\mathrm{Cov}(x_i, x_j) = E(x_i - \mu_i)(x_j - \mu_j). \tag{14.1}$$

If $i = j$, the population covariance of the variable with itself is simply its variance, and therefore there is no need to define variances and covariances independently in the multivariate case. The covariance of x_i and x_j is usually denoted by σ_{ij} (so the variance of the variable x_i is often denoted by σ_{ii} rather than σ^2_{ii}). When we have a number q of variables, x_1, x_2, ... x_q, there are q variances and $\frac{q(q-1)}{2}$ covariances. These quantities are commonly arranged in a q x q symmetric matrix:

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1q} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2q} \\ \dots & \dots & \dots & \dots \\ \sigma_{q1} & \sigma_{q2} & \dots & \sigma_{qq} \end{pmatrix} \tag{14.2}$$

The variances for pairs are equal such that $\sigma_{ij} = \sigma_{ji}$. Numerically, the matrix Σ is computed by using:

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})' \tag{14.3}$$

where $(x_i - \bar{x})'$ is the vector of observations for each individual, therefore the data contained in each row. With the instruction `cov()` in R, the covariance matrix of the dataframe is computed as shown below. Note that the values on the main diagonal are the same computed for the variances of each individual variables, since they correspond the the variances of the vector column with itself, which correspond to the variances computed above.

```
> cov(hus_wife)
           Hage      Hheight        Wage     Wheight     Hagefm
Hage     130.23333  -192.18889  128.55556 -436.0000   28.03333

Hheight -192.18889  4706.98889   25.88889  876.4444   -229.34444

Wage     128.55556   25.88889  164.66667  -456.6667   21.66667

Wheight -436.00000  876.44444 -456.66667   4173.3333  -8.00000

Hagefm    28.03333 -229.34444   21.66667   -8.0000    29.87778
```

14.2.2 Correlations

Covariances matrices are sometime difficult to interpret since the number depends on the unit of the data. Therefore to obtain a better interpretation the data are normalized by

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \tag{14.4}$$

where ρ_{ij} is called *correlation coefficient* and it varies between -1 and 1. It establish linear correlations between variables. Positive numbers means that high values of x_i

are associated with high values of x_j. Negative numbers means that high values of x_i are associated with low values of x_j. With the instruction `cor()` in R, the correlation matrix of the dataframe is computed as shown below. For sample data, the correlation matrix contains the usual estimates of the ρ's, namely Pearson's correlation coefficient, and is generally denoted by R.

Note that the main diagonal values are all one and the other numbers are comprised between -1 and 1.

```
> cor(hus_wife)
 Hage    Hheight       Wage      Wheight      Hagefm
Hage     1.0000000 -0.2454684  0.8778634 -0.59140348  0.44940667

Hheight -0.2454684  1.0000000  0.0294062  0.19774762 -0.61156482

Wage     0.8778634  0.0294062  1.0000000 -0.55087737  0.30889801

Wheight -0.5914035  0.1977476 -0.5508774  1.00000000 -0.02265553

Hagefm   0.4494067 -0.6115648  0.3088980 -0.02265553  1.00000000
```

In the following code, all the instructions are combined including plotting.

```
## Code Ch_14_1.R
#
library(ggplot2)
library(GGally)
setwd("~/Didattica/R_class_4/exercises/multivariata")

hus_wife <- read.csv("hus_wife.dat", header = TRUE, sep = ",")
str(hus_wife)

mean(hus_wife$Hage)
mean(hus_wife$Hheight)
mean(hus_wife$Wage)
mean(hus_wife$Wheight)
mean(hus_wife$Hagefm)

var(hus_wife$Hage)
var(hus_wife$Hheight)
var(hus_wife$Wage)
var(hus_wife$Wheight)
var(hus_wife$Hagefm)


cov(hus_wife)

cor(hus_wife)
```

```
panel.hist <- function(x, ...)
{
       usr <- par("usr"); on.exit(par(usr))
       par(usr = c(usr[1:2], 0, 1.5) )
       h <- hist(x, plot = FALSE)
       breaks <- h$breaks; nB <- length(breaks)
       y <- h$counts; y <- y/max(y)
       rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}

pairs(hus_wife,diag.panel = panel.hist)

ggpairs(hus_wife)
ggcorr(hus_wife)
```

The covariance plots can be generated using the instruction `pairs(hus_wife,diag.panel = panel.hist)`, by previosly defining the histograms, to obtain a plot with the histograms on the main diagonal. Otherwise with the simple instruction `ggpairs(hus_wife)` and `ggcorr(hus_wife)`, the plots displayed below are obtained.

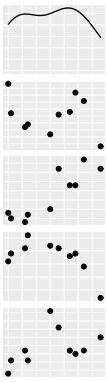

Fig. 14.2 Covariance Matrix

14.2.3 Distances

The concept of distance between observations is important for some multivariate techniques. The most common measure used in Euclidean distance. . If we have p variables X_1, X_2, ..., X_p measured on a sample of n subjects, the observed data for the subject I can be indicated as x_{i1}, $_{i2}$,..., $_{ip}$, and the observed data for the subject j can be identified as $xj1$, $xj2$,..., xjp. The Euclidean distance between these two subjects

Fig. 14.3 Correlation Matrix

(d_{ij}) is given by:

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ip} - x_{jp})^2} \tag{14.5}$$

When using a measure such as the Euclidean distance, the scale of measurement of the variables under consideration is an issue, as changing the scale will have an effect on the distance between subjects. In addition, if one variable has a much wider range than others, then this variable will tend to dominate. For example, if body measurements had been taken for a number of different people, the range (in mm) of heights would be much wider than the range in wrist circumference, say. To get around this problem each variable can be standardised (converted to z-scores). However, this standardization may presents a problem as it tends to reduce the variability (distance) between clusters, and thus we are partly changing the original structure of our initial dataset. This happens because if a particular variable separates observations well then, by definition, it will have a large variance (as the between cluster variability will be high). If this variable is standardized then the separation between clusters will become less. Despite this problem. If in doubt about the use of standardization for your dataset, one strategy would be to carry out the multivariate analysis twice — once without standardising and once with — to see how much difference, if any, this makes to, for example, the resulting clusters.

14.3 Principal Components Analysis

One the main problem that we encounter in multivariate analysis is the high dimensionality of the data set. An unit can be described by a large number of analysed properties. An animal species can be described based on its genetic makeup, its age, weight, breed, color and so forth. The aim of PCA is to derive a set of new, fewer variables will account for a substantial proportion of the variation in the original variables. These new variables are called *principal components*.

The basic aim of principal components analysis (PCA) is to describe the variation in a set of correlated variables, $x1, x2, ..., xq$, in terms of a new set of uncorrelated variables, $y1, y2, ..., yq$, each of which is a linear combination of the x variables. The new variables are derived in decreasing order of "importance" in the sense that $y1$ accounts for as much of the variation in the original data amongst all linear combinations of $x1, x2, ..., xq$. Then $y2$ is chosen to account for as much as possible of the remaining variation, subject to being uncorrelated with $y1$, and so on. The new variables defined by this process, $y1, y2, ..., yq$, are the *principal components*. The general hope of principal components analysis is that the first few components will account for a substantial proportion of the variation in the original variables, $x1, x2, ..., xq$, and can, consequently, be used to provide a convenient lower−dimensional summary of these variables that might prove useful for a variety of reasons.

If we are dealing with a multivariate problem, we are trying to find a way to understand the complex structure of a problem measuring the multiple variables that are shaping the dataset structure. For example, let's assume we are dealing with a dataset that may be contained into a football-shaped ellipse. In that case, it would be useful to describe the orientation of the football shape with its long axis, that in statistical terms would be the axis with the largest variability. There could also be other directions for describing that football shape, and would be the directions at right angles to the longest axis.

Principal Components Analysis (PCA) allows for the identification of linear combinations of variables that provides the maximum variability. Those combinations are new variables made by the linear combination of the initial variables in the dataset, and are called Principal Component (PC). The first PC (PC1) allows to explain the greatest variability; the second PC (PC2) has the maximum variability among all linear combinations that are orthogonal to the first; the third PC is orthogonal to both PC1 and PC2, and so on. So, simply put, PCA reduces a large number of multivariate variables into a relatively small number of linear combinations of the initial variables, and those linear combinations (PCs) are independent of each other and are able to account and represent much of the total variability in the dataset. Generally, variables that show the greatest variance typically dominate the analysis. To avoid this bias, we often need to scale the variables in the dataset, so all variables can be compared on an equal footing. This step is particularly important when we are dealing with datasets that have different types of units (for example, cm, g, %, kg). Standardization os therefore the first step that often needs to be applied. Standardization can be applied in this way, by subtracting the mean and dividing by the standard deviation for each value of each variable:

$$Z = \frac{x_i - \mu}{\sigma} \tag{14.6}$$

where: z is the standardized value, X_i is the initial value, μ is the mean, σ is the standard deviation of the variable. In that way, after standardization, all the variables will be transformed to the same scale. After standardization, PCA performs covariance matrix computation. The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other. In other words, in that way we are testing if there is any relationship between them. As we are dealing with big datasets, often made of numerous variables, some of them may be highly correlated, and contain redundant information. So, in order to identify these correlations, we compute the covariance matrix. Just a quick remind! The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. So, if we are dealing with a two-dimensional dataset ($p = 2$; the two variables are x and y), covariance matrix is the following:

$$\begin{pmatrix} \mathrm{Cov}_{xx} & \mathrm{Cov}_{xy} \\ \mathrm{Cov}_{yx} & \mathrm{Cov}_{yy} \end{pmatrix} \tag{14.7}$$

The covariance between a variable and itself is the variance, so it can also be represented as:

$$\begin{pmatrix} \mathrm{Var}_{x} & \mathrm{Cov}_{xy} \\ \mathrm{Cov}_{yx} & \mathrm{Var}_{y} \end{pmatrix} \tag{14.8}$$

Since $\mathrm{Cov}(x, y)$ is equal to $\mathrm{Cov}(y, x)$, the matrix is symmetric, and the variances of the features lie on the principal diagonal. The covariance matrix in this two−dimensional example can assume different values depending on the shape of our data.
As it is possible to notice, the covariance matrix defines both the spread (variance) and the orientation (covariance) of our data. To this matrix we can assign two further elements: a representative vector and a number which indicates its magnitude. The vector will point into the direction of the larger spread of data, the number will be equal to the spread (variance) of that direction. These two elements are, respectively, an Eigenvector and Eigenvalue. For example, for a 3−dimensional data set with 3 variables x, y, and z, the covariance matrix is a 33 matrix of this from: This is what PCA performs. It creates new variables (PCs), made by the linear combitions of the correlated initial variables, and those new variables are the vectors (just like the green vector we were seeing previously). The new variables obtained after PCA, named PCs, are thus also referred as eigenvectors. Each of those eigenvectors is associated with an eigenvalue which can be interpreted as the "length" or "magnitude" of the corresponding eigenvector. The eigenvectors and eigenvalues of a covariance (or correlation) matrix represent the "core" of a PCA: The eigenvectors (principal components) determine the directions of the new feature space, and the eigenvalues determine their magnitude. In other words, the eigenvalues explain the variance of the data along the new feature axes.

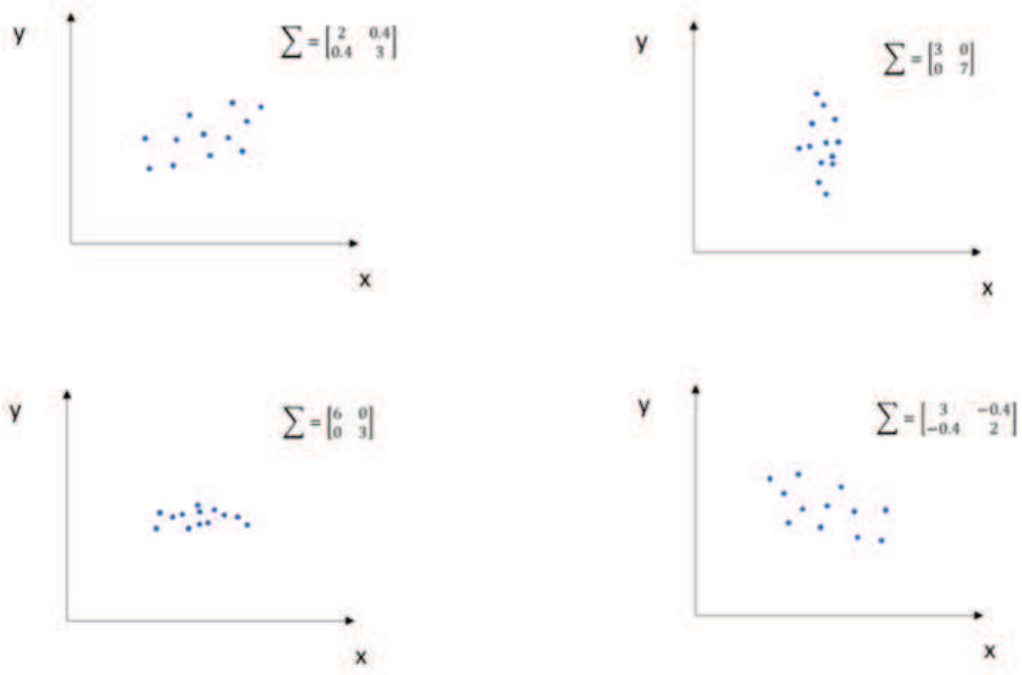

Fig. 14.4 Some examples of the covariance matrices representing the relation occurring between the two variables x and y with different spatial relationships.

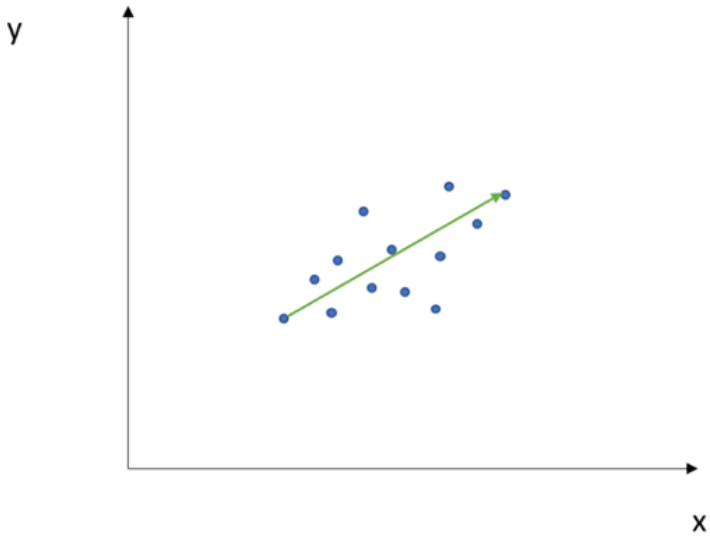

Fig. 14.5 The green vector indicates the direction of the relation between x and y, and is the eigenvector. It has a corresponding value, called eigenvalue, which describes its magnitude.

After that, PCA just reorder the obtained PCs based on the magnitude of their eigenvalues. So, the first PC is the one that explain most part of the total variability of the dataset. If some eigenvalues have a significantly larger magnitude than others, then the reduction of the dataset via PCA onto a smaller dimensional subspace by dropping the "less informative" eigenpairs is reasonable, as only some of them are actually good for explaining the total variance of the dataset. PCA is often used to analyze datasets obtained in biological sciences and by omics analyses (for example, genomics, transcriptomics, lipidomics, proteomics, etc. . .), as in those types of analyses we obtain much more columns (variables; e.g. hundreds of lipids, proteins, genes), than rows (observations, or samples).

The example is about a dataset with lipids, which are 34 times 509 observations. The first thing is to eliminate the missing values

```
### Ch14_3_PCA.R
# Principal component
##the example is a dataset with 509 rows (observations).
## For each observation we have measured
##34 lipids. We would like to identify which lipids are
## the most important and contribute the most to the total
## variability of the dataset.

##let's start to import the dataset
setwd("~/Didattica/R_class_4/exercises/Ch14_multivariate")

fenopercorr <- read.csv("data/forcorr.csv" , header=TRUE, sep=";",
na.strings=c("", "NA"), dec=".")
library(factoextra)

##the first step, we must delete the rows with
## missing values because prcomp is not able
##to perform PCA with missing values
fenopercorr1 <- na.omit(fenopercorr)

fatty <- prcomp(fenopercorr1, scale = T)



##let's visualize eigenvalues. Eigenvalues represents
## the magnitude of the eigenvectors (direction cosins of principal components)
##they show the \% of total variance
## explained by each new variable (principal component)
fviz_eig(fatty)

##The function fviz_mca_ind() [in factoextra] is used to visualize only individuals (rows).
##It's also possible to color individuals by their cos2 values:
```

```
fviz_pca_ind(fatty,
col.ind = "cos2", # Color by the quality of representation
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE ) # Avoid text overlapping


##Graph of variables.
##Positive correlated variables point to the same side of the plot.
##Negative correlated variables point to opposite sides of the graph.
##The variable categories with the larger value,
## contribute the most to the definition of the dimensions.
##Variable categories that contribute the most to
## Dim.1 and Dim.2 are the most important in explaining the
##variability in the data set.
fviz_pca_var(fatty,
col.var = "contrib", # Color by contributions to the PC
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE ) # Avoid text overlapping


##Biplot of individuals and variables
fviz_pca_biplot(fatty, repel = TRUE,
col.var = "#2E9FDF", # Variables color
col.ind = "#696969" )# Individuals color

##other ways to plot the biplot with the eigenvectors of the variables
library(ggfortify)
autoplot(fatty)

autoplot(fatty, loadings = TRUE, loadings.colour = 'blue',
loadings.label = T, loadings.label.size = 3)

# Eigenvalues
eig.val <- get_eigenvalue(fatty)
eig.val

write.table(eig.val, "data/varianceexplained.txt",
row.names = TRUE, col.names = TRUE, quote = F)

# Results for Variables
res.var <- get_pca_var(fatty)
res.var$coord # Coordinates
res.var$contrib # Contributions to the PCs
res.var$cos2 # Quality of representation
# Results for individuals
```

```
res.ind <- get_pca_ind(fatty)
res.ind$coord # Coordinates
res.ind$contrib # Contributions to the PCs
res.ind$cos2 # Quality of representation

##we can export the results in files
write.table(eig.val, file = "data/eigenvalues.txt",
row.names = TRUE, col.names = TRUE, quote = F)
write.table(res.var$coord , file = "data/variables coordinates.txt",
row.names = TRUE, col.names = TRUE, quote = F)
write.table(res.var$coord , file = "data/variables coordinates.txt",
row.names = TRUE, col.names = TRUE, quote = F)
```

14.4 Multiple Correspondence Analysis

Multiple Correspondence Analysis (MCA) is an extension to PCA conceived to explore relationships among qualitative (or categorical) variables. Like PCA, MCA provides a way to summarize and visualize the structure of the dataset into two−dimensions. This type of analysis permit to study and visualize the simultaneous correspondences of rows and columns of a contingency table in order to highlight the connections between the set of variables. The dataset that must be used is a contingency table. Contingency tables are tables in a matrix format containing frequency distribution of the variables. Those types of variables and tables are quite common in social research, survey experiments, economics, clinical studies and all studies where we can have factors and disease incidence.

```
###Ch14_4.R
## MCA
## an example of a Multiple Correspondence Analysis
## we can simply upload a dataset from MASS package.
## The farms data frame has 20 rows and 4 columns.
## The rows are farms on the Dutch island of Terschelling
## and the columns are factors describing
## the management of grassland.
## This data frame contains the following columns:
## Mois = Five levels of soil moisture - level 3 does not
## occur at these 20 farms.
## Manag = Grassland management type
## (SF = standard, BF = biological, HF = hobby farming,
## NM = nature conservation).
## Use = Grassland use (U1=hay production, U2=intermediate, U3=grazing).
## Manure = Manure usage - classes C0 to C4.

library(MASS)
library(FactoMineR)
```

```
library(factoextra)

esempio <- farms
res.mca <- mca(farms, abbrev = TRUE) # Use levels as names
eqscplot(res.mca$cs, type = "n")
text(res.mca$rs, cex = 0.7)
text(res.mca$cs, labels = dimnames(res.mca$cs)[[1]], cex = 0.7)



res.mca <- MCA(esempio)
get_mca_ind(res.mca)

# Extract the results for variable categories
## with get_eig we are obtaining the
## eigenvalues of the eigenvectors, and the variance explained
eig <- get_eig(res.mca)
View(eig)

##we can plot those explained variances with fviz_screeplot function
fviz_screeplot(res.mca, addlabels = TRUE, ylim = c(0, 30))


## The function get_mca_var() [in factoextra]
## is used to extract the results for
## variable categories.
## This function returns a list containing the coordinates, the cos2 and the
## contribution of variable categories:
## The components of the get_mca_var()
## can be used in the plot of rows as follow:
## a$coord: coordinates of variables to create a scatter plot
## a$cos2: represents the quality of the representation
## for variables on the factor map.
## a$contrib: contains the contributions
## (in percentage) of the variables to the
## definition of the dimensions.

a <- get_mca_var(res.mca)

## however the same results are also reported
## in the list res.mca that we have computed as the initial step
res.var <- res.mca$var
res.var$coord # Coordinates
options(digits=2)
write.table(res.var$coord, "data/coordinates MCA.txt", row.names = TRUE,
```

```
col.names = TRUE)
res.var$contrib # Contributions to the PCs
write.table(res.var$contrib, "data/contribution MCA.txt", row.names = TRUE,
col.names = TRUE)
res.var$cos2 # Quality of representation

res.mca[["var"]][["coord"]]

##To visualize the correlation between variables
## and MCA principal dimensions, use fviz_mca_var
## the variables that have the highest coordinates
## for the two axes are those that are the most
## correlated with x and y (dim 1 and dim2)

fviz_mca_var(res.mca, choice = "mca.cor",
repel = TRUE, # Avoid text overlapping (slow)
ggtheme = theme_minimal())


## To plot each observation with its coordinates
## for the new dimensions we can use the same function,
## without expressing the choice

fviz_mca_var(res.mca,
repel = TRUE, # Avoid text overlapping (slow)
ggtheme = theme_minimal())

## It's possible to change the color and the
## shape of the variable points using the arguments
## col.var and shape.var as follow:

fviz_mca_var(res.mca, col.var="blue", shape.var = 15,
repel = TRUE)

## what can we observe from the plot?
## The plot above shows the relationships
## between variable categories. It can be interpreted as follow:
## Variable categories with a similar profile are grouped together.
## Negatively correlated variable categories are positioned
## on opposite sides of the plot origin (opposed quadrants).
##The distance between category points and the origin
## measures the quality of the variable category
##on the factor map.
```

```
##The two dimensions 1 and 2 are sufficient
## to retain 40\% of the total inertia (variation)
## contained in the data.
##Not all the points are equally well displayed
## in the two dimensions. The quality of
## the representation is
##called the squared cosine (cos2), which measures
## the degree of association between variable categories and
##a particular axis. The cos2 of variable categories
## can be extracted as follows:
res.var$cos2


##If a variable category is well represented by two dimensions,
## the sum of the cos2 is close to one.
##For some of the row items, more than 2 dimensions
## are required to perfectly represent the data.
##It's possible to color variable categories by
## their cos2 values using the argument col.var = "cos2".
##This produces a gradient colors, which can be
## customized using the argument gradient.cols.
# Color by cos2 values: quality on the factor map

fviz_mca_var(res.mca, col.var = "cos2",
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE, # Avoid text overlapping
ggtheme = theme_minimal())

fviz_mca_var(res.mca, col.var = "cos2",
gradient.cols = c("red", "white", "blue"),
repel = TRUE, # Avoid text overlapping
ggtheme = theme_minimal())



fviz_mca_var(res.mca, alpha.var = "cos2",
repel = TRUE, # Avoid text overlapping
ggtheme = theme_minimal())

# Cos2 of variable categories on Dim.1 and Dim.2
fviz_cos2(res.mca, choice = "var", axes = 1:2)

##The contribution of the variable categories
## (in \%) to the definition of the dimensions
```

```
## can be extracted as follow:
res.var$contrib

## The variable categories with the larger value,
## contribute the most to the definition of the dimensions.
## Variable categories that contribute the
## most to Dim.1 and Dim.2 are the most important
## in explaining the variability in the data set.

# Contributions of rows to dimension 1
fviz_contrib(res.mca, choice = "var", axes = 1, top = 15)

# Contributions of rows to dimension 2
fviz_contrib(res.mca, choice = "var", axes = 2, top = 15)

# Total contribution to dimension 1 and 2
fviz_contrib(res.mca, choice = "var", axes = 1:2, top = 15)

## The red dashed line on the graph above
## indicates the expected average value
## if the contributions were uniform.
##It can be seen that:
## the categories NM, C0,C4, SF, U2, M5
## are the most important in the definition
## of the first dimension.
## The categories C4, SF, C1, HF contribute the most to dimension 2.

fviz_mca_var(res.mca, col.var = "contrib",
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE, # avoid text overlapping (slow)
ggtheme = theme_minimal())

# Change the transparency by contrib values
fviz_mca_var(res.mca, alpha.var="contrib",
repel = TRUE,
ggtheme = theme_minimal())

## The function fviz_mca_ind()
## [in factoextra] is used to visualize only individuals (rows).
##Like variable categories, it's also
## possible to color individuals by their cos2 values:

fviz_mca_ind(res.mca, col.ind = "cos2",
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE, # Avoid text overlapping (slow if many points)
```

```
ggtheme = theme_minimal())

fviz_mca_ind(res.mca,
label = "none", # hide individual labels
habillage = "Manag", # color by groups
palette = c("yellow", "blue", "green", "black"),
addEllipses = TRUE, ellipse.type = "confidence",
ggtheme = theme_minimal())

fviz_ellipses(res.mca, c("Manag", "Manure"),
geom = "point")
```

14.5 Factor Analysis

Similar to PCA, Factor Analysis (FA) is used to identify patterns in the correlations between variables in a dataset. While PCA seeks to identify variables that are combinations of the initial variables, FA assumes the existence of latent variables (factors) underlying the structure of the dataset. FA therefore can be used to identify groups of variables (or factors) that act in parallel (being highly correlated, with positive or negative correlations) to determine the structure of the dataset. PCA creates a new set of independent variables that can be found in a number of p-1 (p is the number of the initial variables) and that might or might not represent latent variables. FA finds instead linear combinations that represent a specified number of latent variables (the user can specify them). The R function to perform FA is factanal (package stats). This package returns a chi-square statistic and a p-value that test the hypothesis that the model fits the data perfectly. When the p value is low (let's say statistically significant) we can reject this hypothesis, and thus the number of factor is not sufficient to fit the data perfectly. However, we have to be careful, as this output can not be taken as it is. For example, we can have some cases where adding another factor helps explaining little variance and is mainly related to some background noise. The obtained output needs therefore to be always accompanied by a deeper interpretation of the results.

```
###Ch14_5.R
## Factor Analysis

setwd("~/Didattica/R_class_4/exercises/Ch14_multivariate")
fenopercorr <- read.csv("data/forcorr.csv" ,
header=TRUE, sep=";", na.strings=c("", "NA"), dec=".")

fenopercorr1 <- na.omit(fenopercorr)

hope <- factanal(fenopercorr1, factors = 2)

hope
```

```
##The model output starts with the function call
##to remind us on the specifications of our function call.
##The first chunk provides the uniquenesses, which range from 0 to 1.
##The uniqueness, sometimes referred to as noise, corresponds
## to the proportion of variability,
## which can not be explained by a linear combination
## of the factors. This is the in the equation above.
## A high uniqueness for a variable indicates that the
## factors do not account well for its variance.
## Before we interpret the results of the factor
## analysis recall the basic idea behind it.
## Factor analysis creates linear combinations
## of factors to abstract the variable's underlying communality.
## To the extent that the
## variables have an underlying communality,
## they can be linearly combined allowing
## however to explain the total variance of the dataset.
## The variability in our data is given by
## the sum of the variability explained by a ## linear combination
##of the factors (communality) and of the variability
## that can not be explained by a linear combination of the factors (uniqueness).



hope$uniquenesses

## By squaring the loading we compute the fraction
## of the variable's total variance explained by the factor
## This proportion of the variability is denoted as communality.
## Another way to calculate the communality is to subtract
## the uniquenesses from 1.
## An appropriate factor model results in low values
## for uniqueness and high values for communality.

apply(hope$loadings^2,1,sum) # communality

1 - apply(hope$loadings^2,1,sum) # uniqueness


##The purpose of a rotation is to produce
## factors with a mix of high and low loadings
## and few moderate-sized loadings.
## The idea is to give meaning to the factors, which helps interpret them.
## From a mathematical viewpoint, there is no difference between
## a rotated and unrotated matrix.
```

```
## The fitted model is the same, the uniquenesses are the same,
## and the proportion of ##variance explained is the same.
## Let's try to fit three factor models, one with no rotation,
## one with varimax rotation, and one with promax rotation,
##and make a scatter plot of the first and second loadings.
##depending on the authors, rotation is defined
##as a procedure in which the eigenvectors (factors)
## are rotated in an attempt to achieve a simpler structure.

hope.none <- factanal(fenopercorr1, factors = 2, rotation = "none")
hope.varimax <- factanal(fenopercorr1, factors = 2, rotation = "varimax")
hope.promax <- factanal(fenopercorr1, factors = 2, rotation = "promax")

par(mfrow = c(1,3))
plot(hope.none$loadings[,1],
hope.none$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "No rotation")
abline(h = 0, v = 0)

plot(hope.varimax$loadings[,1],
hope.varimax$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "Varimax rotation")

text(hope.varimax$loadings[,1]-0.08,
hope.varimax$loadings[,2]+0.08,
colnames(fenopercorr1),
col="blue")
abline(h = 0, v = 0)

plot(hope.promax$loadings[,1],
hope.promax$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "Promax rotation")
abline(h = 0, v = 0)
```

```
hope <- factanal(fenopercorr1, factors = 3)
hope
hope.none <- factanal(fenopercorr1, factors = 3, rotation = "none")
hope.varimax <- factanal(fenopercorr1, factors = 3, rotation = "varimax")
hope.promax <- factanal(fenopercorr1, factors = 3, rotation = "promax")

par(mfrow = c(1,3))
plot(hope.none$loadings[,1],
hope.none$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "No rotation")
abline(h = 0, v = 0)

plot(hope.varimax$loadings[,1],
hope.varimax$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "Varimax rotation")

text(hope.varimax$loadings[,1]-0.08,
hope.varimax$loadings[,2]+0.08,
colnames(fenopercorr1),
col="blue")
abline(h = 0, v = 0)

plot(hope.promax$loadings[,1],
hope.promax$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "Promax rotation")
abline(h = 0, v = 0)


hope1 <- factanal(fenopercorr1, factors = 4)
hope1

hope.none <- factanal(fenopercorr1, factors = 4, rotation = "none")
```

```
hope.varimax <- factanal(fenopercorr1, factors = 4, rotation = "varimax")
hope.promax <- factanal(fenopercorr1, factors = 4, rotation = "promax")

par(mfrow = c(1,3))
plot(hope.none$loadings[,1],
hope.none$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "No rotation")
abline(h = 0, v = 0)

plot(hope.varimax$loadings[,1],
hope.varimax$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "Varimax rotation")

text(hope.varimax$loadings[,1]-0.08,
hope.varimax$loadings[,2]+0.08,
colnames(fenopercorr1),
col="blue")
abline(h = 0, v = 0)

plot(hope.promax$loadings[,1],
hope.promax$loadings[,2],
xlab = "Factor 1",
ylab = "Factor 2",
ylim = c(-1,1),
xlim = c(-1,1),
main = "Promax rotation")
abline(h = 0, v = 0)

##which rotation should we use? goal of
## rotation and of choosing a particular type
## of rotation as seeking
##something called simple structure,
## or put another way, one way we
## know if we have selected an
##adequate rotation method is if the
## results achieve simple structure.
##what is a simple structure?
```

```
## A simple structure is a structure
## A condition in which
##variables load at near 1 (in absolute value)
## or at near 0 on an eigenvector (factor).
##Variables that load near 1 are clearly
## important in the interpretation of the factor,
##and variables that load near 0 are
## clearly unimportant.
##Simple structure thus simplifies
## the task of interpreting the factors.
##among the proposed ways to understand
## whether simple structure has been reached
##Thurstone (1947) first proposed and
## argued for five criteria that needed to be met
##1. Each variable should produce at
## least one zero loading on some factor.
##2. Each factor should have at
## least as many zero loadings as there are factors.
##3. Each pair of factors should
## have variables with significant
## loadings on one and zero loadings on the other.
##4. if there are say four or more
## factors total each pair of factors
## should have a large proportion
## of zero loadings on both factors
##5. Each pair of factors should
## have only a few complex variables.
```

14.6 Canonical Correlation Analysis

Canonical Correlation Analysis is a statistical method that can be used to correlate two sets of linear variables. Thus, CCA allows for the identification of linear relation or symmetry between two datasets. This means that CCA is used to find the maximal correlations between a linear combination of independent variables with a linear combination of dependent variables. Canonical correlation may be used in an appropriate way when we are dealing with multiple intercorrelated outcome variables and we want to identify the linear relations occurring between those independent variables and the dependent ones. Canonical correlation analysis determines a set of canonical variates, orthogonal linear combinations of the variables within each set that best explain the variability both within and between sets. This type of analysis can be used for example to relate plant characters [X set $\smile$ fruit length, fruit diameter, flesh thickness, fiber weight per fruit, length of seed cavity, etc...], and yield components [Y set $-$ total fruit weight per plant, average fruit weight and number of fruits per plant].

14.7 Discriminant Analysis

Discriminant Analysis is a Multivariate statistics technique that can be used to classify an observation into one or several groups. To distinguish among the groups, the researcher selects a collection of discriminating variables that measure characteristics on which the groups are expected to differ. In other words, DA is a technique that reverses the statistical assumptions of MANOVA. In MANOVA, the independent variables are made of qualitative variables (with classes/groups) and the dependent variables are the continuous variables, while DA has the continuous variables and predictors as independent variables and the dependent variables are the groups. DA is therefore useful in the situations where we want to build a predictive model of group membership based on the observed characteristics of each observation. The question we are asking is thus: "can a combination of the variables that I am measuring be used to predict group membership of my observations?". The outcome of DA generates one or more functions based on linear combinations of the predictor variables, and providing the best discrimination between the groups. The functions are generated from a sample of observations whose group membership is known. The obtained functions can then be applied to the observations that have no assigned group membership, and for future cases. Those functions are estimated basing on a set of variables that are the best predictors: DA indeed selects from the available variables the ones that discriminate better and thus account for most of the variability between groups, causing the smallest error of classification when we are using the function as an instrument of discrimination. The function(s) are formed in such a way to maximize the separation of the groups.

The linear combination of discriminant function is usually in the following form:

$$D_i = a + b_1 X_1 + b_2 X_2 + \ldots + b_n X_n \tag{14.9}$$

where: D_i is the discriminate function or the predicted score (also named discriminant score); a is a constant; b is the discriminant coefficient or weight for that variable (predictor); X is the predictor; n is the number of predictor variables. The weighting/discriminant coefficients indicate the variables that contribute the most to the differentiation along the respective dimension. The best predictors tend in fact to have the largest weights in the respective linear combination.

With DA we are however using some assumptions:first of all we are assuming the observations are a random sample of the population; groups of the dependent variable are mutually exclusive and collectively exhaustive; the observations with known classification are a random sample of the population and were correctly classified; the groups were defined before data collection. Which purposes we can use DA for? First of all, we can use DA to classify cases into groups; to test a theory by observing wether observations are classified as predicted; to investigate differences among groups; the determine the most parsimonious (thus selecting the best predictors) way to distinguish among groups; to estimate the percentage variance explained by the predictors; to identify the most important variables explaining intergroup variability; to discard variables that are poorly related to the variability of the dependent variable DA can be successfully applied also when we are dealing with i) the dependent variable is categorical with the predictors (independent variables) having interval levels, and/or

being continuous or dummy variables; the dependent variable has more than 2 levels (and thus we can not apply logistic regression models).

```
## Code Ch14_7.R
# Discriminant Analysis
library(readxl)
setwd("~/Didattica/R_class_4/exercises/Ch14_multivariate")
data <- read_excel("data/example per DA.xlsx")
data1 <- na.omit(data)

library(caret)
library(dplyr)
# Split the data into training (80%) and test set (20%)
set.seed(123)
training.samples <- data1$eur %>%
createDataPartition(p = 0.8, list = FALSE)
train.data <- data1[training.samples, ]
test.data <- data1[-training.samples, ]


##now let's normalize our data.
##Categorical variables will be ignored automatically
# Estimate preprocessing parameters
preproc.param <- train.data %>%
preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformed <- preproc.param %>% predict(train.data)
test.transformed <- preproc.param %>% predict(test.data)

##linear discriminant analysis can be applied to predictors
##that are normally distributed (Gaussian distribution)
##and that the different classes have class-specific means and equal
###variance/covariance.
##Before performing LDA, consider:
##Inspecting the univariate distributions of each variable and
## make sure that they are normally distribute.
##If not, you can transform them using log and root for exponential
## distributions and Box-Cox for skewed distributions.
##removing outliers from your data and standardize the variables
## to make their scale comparable.

library(MASS)
# Fit the model
model <- lda(eur~., data = train.transformed)
model
```

```
##LDA determines group means and computes, for each individual,
## the probability of belonging
##to the different groups. The individual is then affected to the group
## with the highest probability score.
##The lda() outputs contain the following elements:
##Prior probabilities of groups: the proportion of training observations
## in each group. For example,
##there are 57% of the training observations in the R group
##Group means: group center of gravity. Shows the mean of each variable
## in each group.
##Coefficients of linear discriminants: Shows the linear combination
## of predictor variables that are
##used to form the LDA decision rule. for example,
## LD1 = -0.32*C14 + 1.41*C16 + 0.44*C16m1c9 etc...

plot(model)

# Make predictions
predictions <- model %>% predict(test.transformed)
names(predictions)
##The predict() function returns the following elements:
##class: predicted classes of observations.
##posterior: is a matrix whose columns are the groups,
## rows are the individuals
##and values are the posterior probability that the
## corresponding observation belongs to the groups.
##x: contains the linear discriminants, described above


##let's inspect the outputs
# Predicted classes
head(predictions$class, 6)
# Predicted probabilities of class memebership.
head(predictions$posterior, 6)
# Linear discriminants
head(predictions$x, 3)


lda.data <- cbind(train.transformed, predict(model)$x)
ggplot(lda.data, aes(LD1, LD2)) +
geom_point(aes(color = eur))
```

```
lda.data <- cbind(train.transformed, predict(model)$x)
ggplot(lda.data, aes(LD2, LD3)) +
geom_point(aes(color = eur))


lda.data <- cbind(train.transformed, predict(model)$x)
ggplot(lda.data, aes(LD1, LD3)) +
geom_point(aes(color = eur))

# Model accuracy
mean(predictions$class==test.transformed$eur)

##this means that we are correctly classifying a
##little more than 60\% of the test observations


##depending on the type of predictors we can also apply other types of DA:

##Quadratic discriminant analysis (QDA):
## More flexible than LDA.
##There is no assumption that the covariance matrix
## of classes is the same.
##we can apply qda() function from MASS package

##Mixture discriminant analysis (MDA):
## Each class is assumed to be a Gaussian mixture of subclasses.
##we can apply mda() function from library(mda)

##Flexible Discriminant Analysis (FDA):
##Non-linear combinations of predictors is used such as splines.
####we can apply fda() function from library(mda)

##Regularized discriminant anlysis (RDA):
## Regularization (or shrinkage) improves the estimate of the
##covariance matrices in situations where
## the number of predictors is larger than the number of samples
##in the training data. This leads to an improvement
## of the discriminant analysis.
####we can apply rda() function from library(klaR)
```

14.8 Cluster Analysis

Cluster Analysis (CA) is a multivariate method that can be applied to classify a sample of observations (raws, subjects) on a basis of a set of measured variables. The number of different groups (i.e. clusters) depends on the structural variability of the sample,

and inside each group are placed the observations that are similar. Cluster analysis is concerned with investigating a set of data to discover whether or not it consists of relatively distinct groups of observations. This method is therefore an approach that groups samples based on their multivariate structure and variability, but has a completely different meaning from DA. In DA we have indeed a definite number of groups that we are setting before measuring the predictors, and we want to find a function that is able to classify some observations of unknown group membership. In CA we have instead a sample of observations and we would like to identify clusters of observations that are similar. CA, therefore, creates clusters on the basis of all the variables considered and has no mechanism for differentiating between relevant and irrelevant variables. The choice of variables to be included in a CA must be underpinned by conceptual considerations. This point os of great importance as the formed clusters are strongly dependent on the variables included. There are different approaches that can be used in CA to find clusters:

- we can have hierarchical methods
- and non-hierarchical methods (also known as k-means clustering methods)

14.8.1 Hierarchical methods for Cluster Analysis

In a hierarchical classification the data are not partitioned into a particular number of groups or clusters at a single step. This partition is made in several steps. There are two types of hierarchical methods: agglomerative and divisive methods. In agglomerative methods, the analysis starts with each observation in its own separate cluster. The two closest clusters (the most similar) are then combined and this step is performed repeatedly until all observations belong to one cluster. At the end, the best combination of cluster is chosen out of all cluster solutions identified during the analysis. Divisive methods act instead in the opposite way: all subjects are in the same cluster when the analysis starts, and at each step they are divided into smaller clusters based on the observations dissimilarities, until at the end each subject is in a separate cluster. At the end, the best combination of clusters is chosen. Agglomerative methods are more commonly used in statistics. Figure 14.6 graphically presents the difference in how hierarchical agglomerative and divisive methods work. The type of graph presented is a dendrogram, a graphical representation of how much clusters are similar (or dissimilar). This type of representation is quite common in Evolution trees.

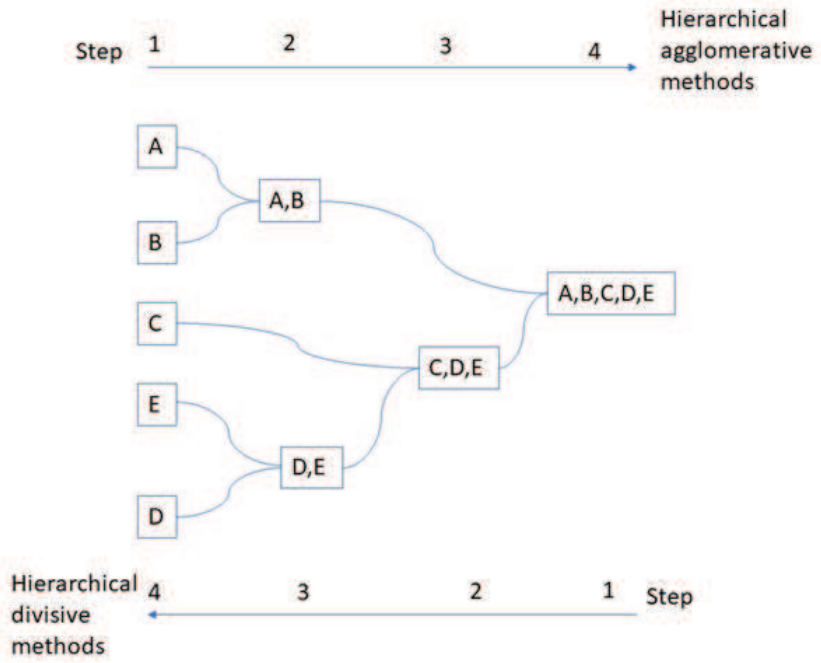

Fig. 14.6 Graphical representation of how hierarchical agglomerative (upper part) and divisive (bottom part) methods work.

Often an investigator will not be interested in the complete hierarchy but only in a single partition of the data into a particular number of groups, say g. Deciding on an appropriate value of g from a hierarchy is one of the problems that a researcher should deal with.

14.8.2 Hierarchical Agglomerative methods for Cluster Analysis

Those methods are similar, as at each step the individuals or groups of individuals that are the most similar are fused together. Each agglomerative method has however a different way in which distance (or similarity) between two groups can be defined. In **single linkage clustering** (or nearest neighbour method), the distance between two clusters (intergroup distance) is defined to be the distance between the two closest members (or neighbours) from the two clusters. This method is relatively simple but may have some problems as it does not take into account the cluster structure and can result in a problem called chaining whereby clusters end up being long and straggly. Chaining cause this method to cluster together groups of asamples also if the two clusters have a relatively small number of subjects that are intermediates between the two clusters. Due to chaining it could be hard to single linkage clustering to divide the

two clusters. However, this method is better than the other methods when the natural clusters are not spherical or elliptical in shape.

In **complete linkage method** (or furthest neighbour method) the distance between two clusters is defined to be the maximum distance between members — i.e. the distance between the two subjects of the two clusters that are the furthest. This method tends to produce compact clusters of similar size but, as for the nearest neighbour method, does not take account of cluster structure. As can be easily imagined, it is also quite sensitive to outliers. In average (between groups) linkage method (also called UPGMA- unweighted pair group method with arithmetic mean), the distance between two clusters is calculated as the average distance between all pairs of subjects in the two clusters. This is considered to be a fairly robust method and is quite used in practice. Figure CA2 graphically shows how the distance between two clusters is calcoluated in average linkage method.

```
#Code Ch14_8.R
#Cluster Analysis
library(readxl)
setwd("~/Didattica/R_class_4/exercises/Ch14_multivariate")
data <- read_excel("data/example seeds hierarchical.xlsx")

##let's explore our data
str(data)
summary(data)

##let's control whether we have missing values
any(is.na(data))
##if true you'll have to remove the missing values
data1 <- na.omit(data)

##We are now going to store the last column (type of seeds)
##in a separate vector, in order to perform the
##hierarchical clustering from our dataset.
##Later you will use the true labels to check
##how good your clustering turned out to be.

seeds_label <- data$type.of.seed
data$type.of.seed <- NULL
str(data)

##as each column is a variable that has been
##measured with different units of measures we need to rescale them
seeds_df_sc <- as.data.frame(scale(data))
summary(seeds_df_sc)
##as you can see we have now standardized variables with mean 0.

##Now we evaluate the distance between the observations and
```

```
## create a matrix of distances.
##As all measures are numerical we can use Euclidean distance

dist_mat <- dist(seeds_df_sc, method = 'euclidean')

##now we perform hierarchical cluster. we can change the type of method
hclust_avg <- hclust(dist_mat, method = 'average')
plot(hclust_avg)

##now we have to cut the dendrogram
cut_avg <- cutree(hclust_avg, k = 3)
plot(hclust_avg)
rect.hclust(hclust_avg , k = 3, border = 2:6)
abline(h = 3, col = 'red')


##You can also use the color_branches() function from the dendextend library
##to visualize your tree with different colored branches.
library(dendextend)
avg_dend_obj <- as.dendrogram(hclust_avg)
avg_col_dend <- color_branches(avg_dend_obj, h = 3)
plot(avg_col_dend)


##now with mutate() function we export the results of the cluster analysis
library(dplyr)
seeds_df_cl <- mutate(data, cluster = cut_avg)
count(seeds_df_cl,cluster)

##plotting the cluster factor feature
##for the observations represented for
##their area and perimeter (not rescaled)
##we can notice that there is a strong relation
## between area and perimeter in differentiating
##the different seeds (and clusters in our dataset)
library(ggplot2)
ggplot(seeds_df_cl, aes(x=area, y = perimeter, color = factor(cluster)))
+ geom_point()

##now let's create a contingency table to
##compare our results and the real observations
##we had for the type of seeds
##as you can see cluster analysis was
##quite good at classifying them
```

```
table(seeds_df_cl$cluster,seeds_label)




##in that case we had the real classification
## labels but what about the evaluation
## if we do not have them?

library(clValid)
dunn(distance = dist_mat, clusters = cut_avg)

##now let's use other methods to cluster the
## data and compare them with Dunn's index

##now we perform hierarchical cluster.
## we can change the type of method to centroid
hclust_cent <- hclust(dist_mat, method = 'centroid')
plot(hclust_cent)

##now we have to cut the dendrogram
cut_cent <- cutree(hclust_cent, k = 2)
plot(hclust_cent)
rect.hclust(hclust_cent , k = 2, border = 1:6)
abline(h = 2.2, col = 'red')

dunn(distance = dist_mat, clusters = cut_cent)
##so it's better the average method


##now we perform hierarchical cluster.
## we can change the type of method to centroid
hclust_ward <- hclust(dist_mat, method ="ward.D")
plot(hclust_ward)

##now we have to cut the dendrogram
cut_ward <- cutree(hclust_ward, k = 3)
plot(hclust_ward)
rect.hclust(hclust_ward , k = 3, border = 1:200)
abline(h = 50, col = 'red')

dunn(distance = dist_mat, clusters = cut_ward)
##not bad. let's compare it with the real clusters

seeds_df_clward <- mutate(data, cluster = cut_ward)
```

```
count(seeds_df_clward,cluster)

##plotting the cluster factor feature for
## the observations represented for their
## area and perimeter (not rescaled)
##we can notice that there is a strong relation
## between area and perimeter in differentiating
## the different seeds (and clusters in our dataset)

ggplot(seeds_df_clward, aes(x=area, y = perimeter, color = factor(cluster)))
+ geom_point()

##now let's create a contingency table to compare
## our results and the real observations
## we had for the type of seeds
##as you can see cluster analysis was quite good at classifying them
table(seeds_df_clward$cluster,seeds_label)
##come si vede, ha identificato bene i cluster 2 e 3, mentre sul primo cluster molti sono sta

#---------------- k-means-------------
##now we pass to an example with k-means clustering
library(readxl)
data <- read_excel("D:/statistics/example per DA.xlsx")
data1 <- na.omit(data)

eur_label <- data1$eur
data1$eur<- NULL
str(data1)


data2 <- as.data.frame(scale(data1))
summary(data2)
##now we have to determine the right number of clusters,
#how can we do that?
library(NbClust)
library(factoextra)

# Elbow method
fviz_nbclust(data2,kmeans,method = "wss") +
geom_vline(xintercept = 3, linetype = 2)+
labs(subtitle = "Elbow method")
# Silhouette method
fviz_nbclust(data2, kmeans, method = "silhouette")+
labs(subtitle = "Silhouette method")
# Gap statistic
```

```
# nboot = 50 to keep the function speedy.
# recommended value: nboot= 500 for your analysis.
# Use verbose = FALSE to hide computing progression.
set.seed(123)
fviz_nbclust(data2, kmeans, nstart = 25, method = "gap_stat", nboot = 50)+
labs(subtitle = "Gap statistic method")

##or, even better, we can use NBClust function to
## compare the suggested number of indices
## among the different methods

nb <- NbClust(data2, distance = "euclidean", min.nc = 2,
max.nc = 10, method = "kmeans")

fviz_nbclust(nb)

# Compute k-means with k = 2
set.seed(123)
fatty.res <- kmeans(data2, 2, nstart = 25)
##As the final result of k-means clustering
## result is sensitive to the random starting assignments,
##we specify nstart = 25. This means that R
## will try 25 different random starting assignments and then
##select the best results corresponding to
## the one with the lowest within cluster variation.
##The default value of nstart in R is one.
## But, it's strongly recommended to compute k-means clustering
##with a large value of nstart such as 25 or 50,
## in order to have a more stable result.
print(fatty.res)
fviz_cluster(fatty.res, data2, ellipse.type = "norm")



##------------------Hierarchical clustering applied after PCA----------------------------
##we can also apply hierarchical clustering
##after PCA to identify clusters of observations in our dataset
##let's take our dataset with fatty acids
##the example is a dataset with 509 rows (observations).
## For each observation we have measured
##34 lipids. We would like to identify which lipids
## are the most important and contribute the
##most to the total variability of
##the dataset.
```

```
##let's start to import the dataset
fenopercorr <- read.csv("D:/statistics/forcorr.csv" ,
header=TRUE, sep=";", na.strings=c("", "NA"), dec=".")

##the first step, we must delete the rows
## with missing values because prcomp is not able
##to perform PCA with missing values
fenopercorr1 <- na.omit(fenopercorr)

library(FactoMineR)
library(factoextra)


res.pca <- PCA(fenopercorr1, scale.unit = TRUE, graph = FALSE)

##let's visualize eigenvalues. Eigenvalues
## represents the magnitude of the eigenvectors
## (direction cosins of principal components)
##they show the \% of total variance explained by
## each new variable (principal component)
fviz_eig(res.pca)

# Compute PCA with ncp = 2, the number of PC we
## want to retain and use for clustering
res.pca <- PCA(fenopercorr1, ncp = 2, graph = FALSE)


# Compute hierarchical clustering on principal components
res.hcpc <- HCPC(res.pca, graph = FALSE)


##now let's visualize the dendrogram
## with observations and clusters
fviz_dend(res.hcpc,
cex = 0.7, # Label size
palette = "jco", # Color palette see ?ggpubr::ggpar
rect = TRUE, rect_fill = TRUE, # Add rectangle around groups
rect_border = "jco", # Rectangle color
labels_track_height = 0.8 # Augment the room for labels
)

##we can also plot the clusters on the PCA biplot

fviz_cluster(res.hcpc,
repel = TRUE, # Avoid label overlapping
```

```
show.clust.cent = TRUE, # Show cluster centers
palette = "jco", # Color palette see ?ggpubr::ggpar
ggtheme = theme_minimal(),
main = "Factor map"
)

# Principal components + tree
plot(res.hcpc, choice = "3D.map")


##The function HCPC() returns a list containing:
##data.clust: The original data
## with a supplementary column called
## class containing the partition.
##desc.var: The variables describing clusters
##desc.ind: The more typical individuals of each cluster
##desc.axes: The axes describing clusters

head(res.hcpc$data.clust, 10)

##the variables that are the most
## important in defining the variability
## inside each cluster are those that have the
##highest v.test and the lowest p-value
res.hcpc$desc.var$quanti

res.hcpc$desc.axes$quanti

res.hcpc$desc.ind$para

##now you can export and save the clusters
##and the variables having the most importance
## in explaining the variability inside
##each cluster

##remember you can use hcpc with the same
## scripts and functions also for MCA results. You just need to apply the
##scripts to a list of objects obtained from MCA() instead that PCA()
```