Multivariate Analysis II

Alboukadel Kassambara

Practical Guide To Principal Component Methods in R

PCA, (M)CA, FAMD, MFA, HCPC, factoextra

sthda.com https://www.facebook.com/groups/stats.ebooksandpapers/ Edition 1

Practical Guide to Principal Component Methods in R

Alboukadel KASSAMBARA

Published by STHDA (http://www.sthda.com), Alboukadel Kassambara

Contact: Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

For general information contact Alboukadel Kassambara <alboukadel.kassambara@gmail.com>.

Contents

Preface

0.1 What you will learn

Large data sets containing multiple samples and variables are collected everyday by researchers in various fields, such as in Bio-medical, marketing, and geo-spatial fields.

Discovering knowledge from these data requires specific techniques for analyzing data sets containing multiple variables. **Multivariate analysis** (MVA) refers to a set of techniques used for analyzing a data set containing more than one variable.

Among these techniques, there are:

- Cluster analysis for identifying groups of observations with similar profile according to a specific criteria.
- Principal component methods, which consist of summarizing and visualizing the most important information contained in a multivariate data set.

> Previously, we published a book entitled "Practical Guide To Cluster Analysis in R" (`https://goo.gl/DmJ5y5`). The aim of the current book is to provide a solid practical guidance to principal component methods in R. Additionally, we developed an R package named factoextra to create, easily, a ggplot2-based elegant plots of the results of principal component method. Factoextra official online documentation: `http://www.sthda.com/english/rpkgs/factoextra`

One of the difficulties inherent in multivariate analysis is the problem of visualizing data that has many variables. In R, there are many functions and packages for displaying a graph of the relationship between two variables (`http://www.sthda.com/english/wiki/data-visualization`). There are also commands for displaying different three-dimensional views. But when there are more than three variables, it is more difficult to visualize their relationships.

Fortunately, in data sets with many variables, some variables are often correlated. This can be explained by the fact that, more than one variable might be measuring the same driving principle governing the behavior of the system. Correlation indicates that there is redundancy in the data. When this happens, you can simplify the problem by replacing a group of correlated variables with a single new variable.

Principal component analysis is a rigorous statistical method used for achieving this simplification. The method creates a new set of variables, called principal components. Each principal component is a linear combination of the original variables. All the principal components are orthogonal to each other, so there is no redundant information.

The type of principal component methods to use depends on variable types contained in the data set. This practical guide will describe the following methods:

1. **Principal Component Analysis (PCA)**, which is one of the most popular multivariate analysis method. The goal of PCA is to summarize the information contained in a continuous (i.e, quantitative) multivariate data by reducing the dimensionality of the data without loosing important information.

2. **Correspondence Analysis (CA)**, which is an extension of the principal component analysis for analyzing a large contingency table formed by two *qualitative variables* (or categorical data).

3. **Multiple Correspondence Analysis (MCA)**, which is an adaptation of CA to a data table containing more than two categorical variables.

4. **Factor Analysis of Mixed Data (FAMD)**, dedicated to analyze a data set containing both quantitative and qualitative variables.

5. **Multiple Factor Analysis (MFA)**, dedicated to analyze data sets, in which variables are organized into groups (qualitative and/or quantitative variables).

Additionally, we'll discuss the **HCPC (Hierarchical Clustering on Principal Component)** method. It applies agglomerative hierarchical clustering on the results of principal component methods (PCA, CA, MCA, FAMD, MFA). It allows us, for example, to perform clustering analysis on any type of data (quantitative, qualitative or mixed data).

Figure 1 illustrates the type of analysis to be performed depending on the type of variables contained in the data set.

0.2 Key features of this book

Although there are several good books on principal component methods and related topics, we felt that many of them are either too theoretical or too advanced.

Our goal was to write a practical guide to multivariate analysis, visualization and interpretation, focusing on principal component methods.

The book presents the basic principles of the different methods and provide many examples in R. This book offers solid guidance in data mining for students and researchers.

Key features

- Covers principal component methods and implementation in R
- Short, self-contained chapters with tested examples that allow for flexibility in designing a course and for easy reference

At the end of each chapter, we present R lab sections in which we systematically work through applications of the various methods discussed in that chapter. Additionally, we provide links to other resources and to our hand-curated list of videos on principal component methods for further learning.

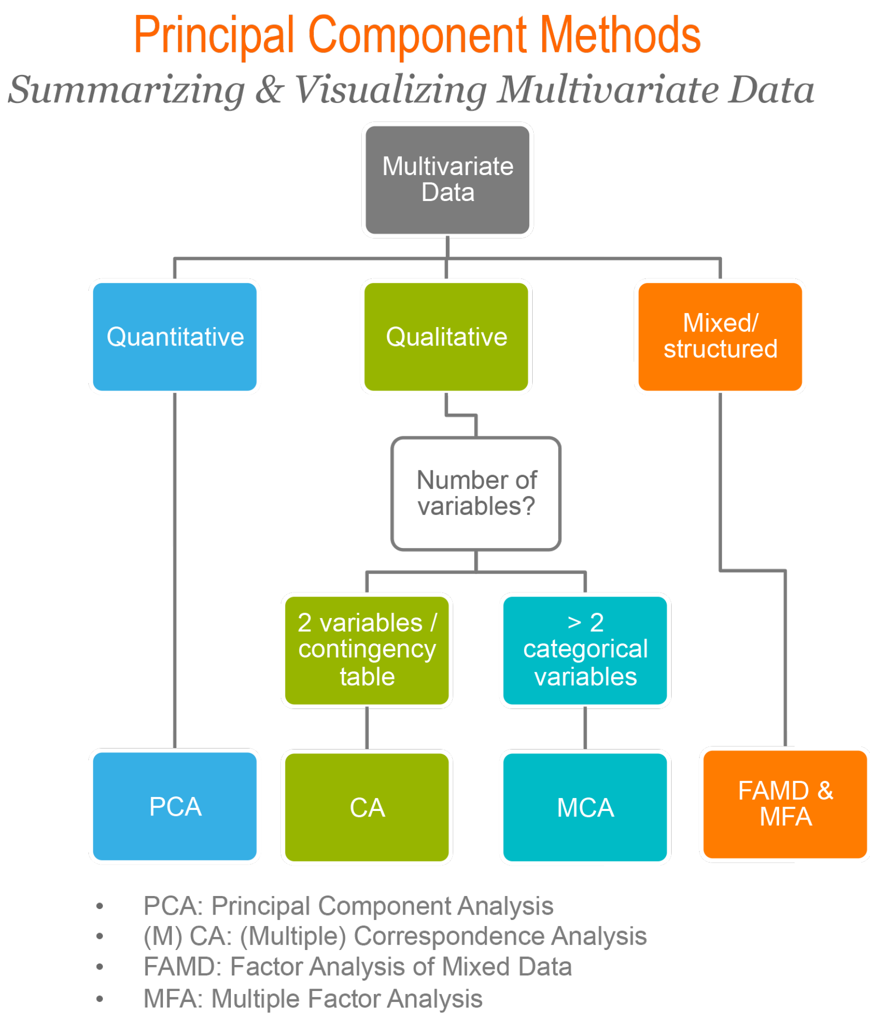


Figure 1: Principal component methods

0.3 How this book is organized

This book is divided into 4 parts and 6 chapters. Part I provides a quick introduction to R (chapter 1) and presents required R packages for the analysis and visualization (chapter 2).

In Part II, we describe classical multivariate analysis methods:

- Principal Component Analysis - PCA (chapter 3)
- Correspondence Analysis - CA (chapter 4)
- Multiple Correspondence Analysis - MCA (chapter 5)

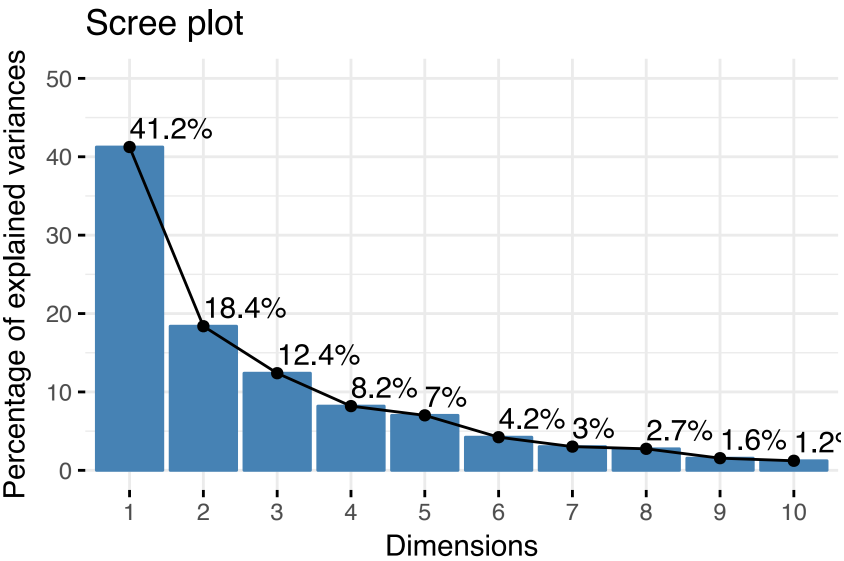
In part III, we continue by discussing advanced methods for analyzing a data set containing a mix of variables (qualitative & quantitative) organized or not into groups:

- Factor Analysis of Mixed Data - FAMD (chapter 6) and,
- Multiple Factor Analysis - MFA (chapter 7).

Finally, we show in Part IV, how to perform hierarchical clustering on principal components (HCPC) (chapter 8), which is useful for performing clustering with a data set containing only qualitative variables or with a mixed data of qualitative and quantitative variables.

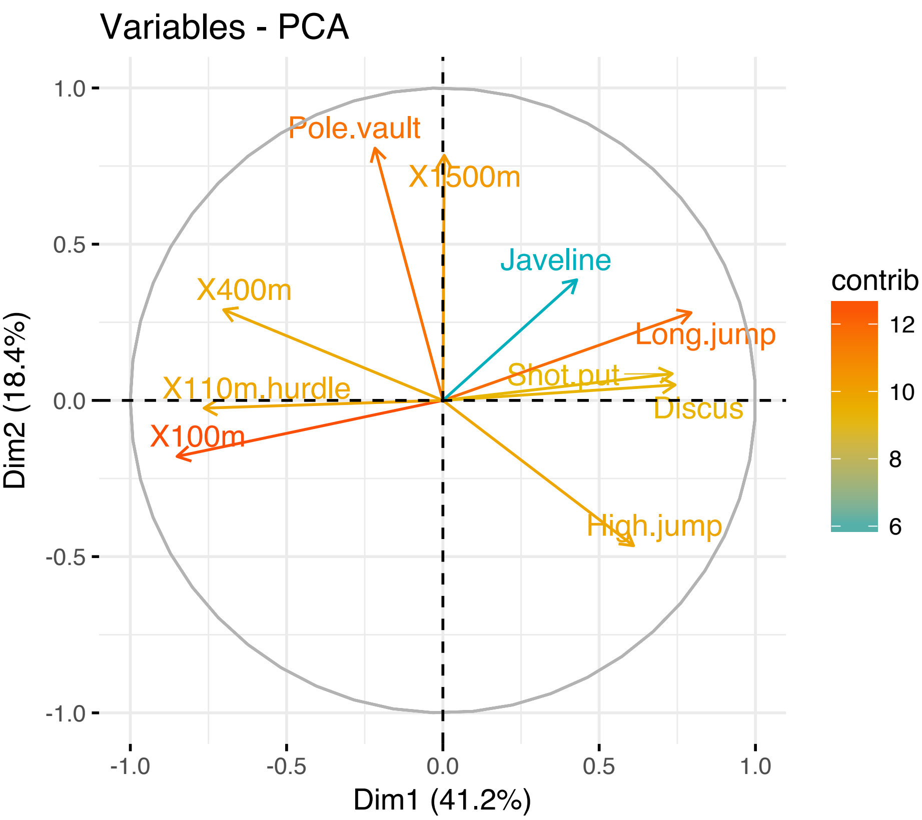
Some examples of plots generated in this book are shown hereafter. You'll learn how to create, customize and interpret these plots.

1) **Eigenvalues/variances of principal components**. Proportion of information retained by each principal component.

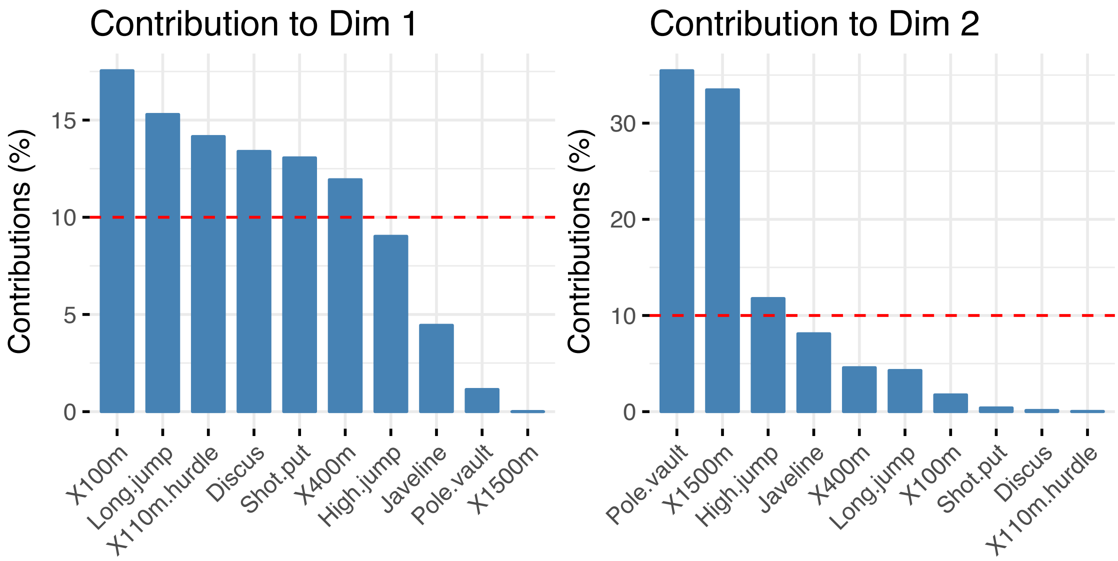


2) **PCA - Graph of variables**:

- Control variable colors using their contributions to the principal components.

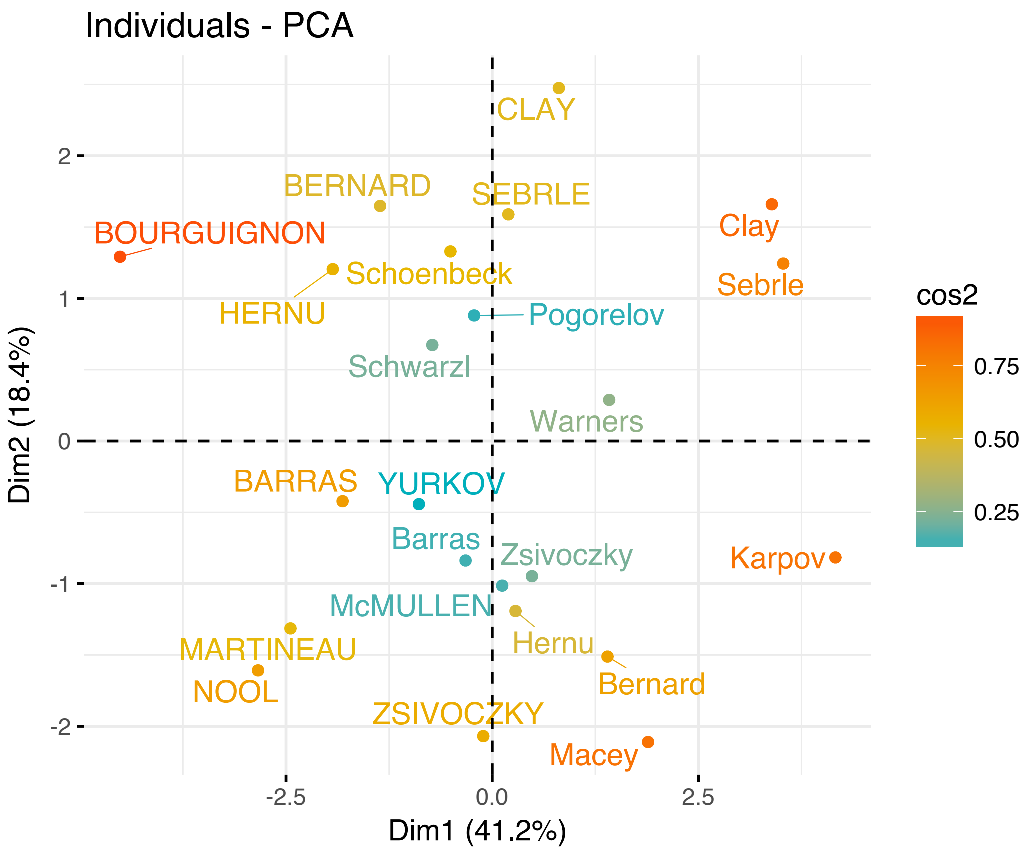


- Highlight the most contributing variables to each principal dimension:

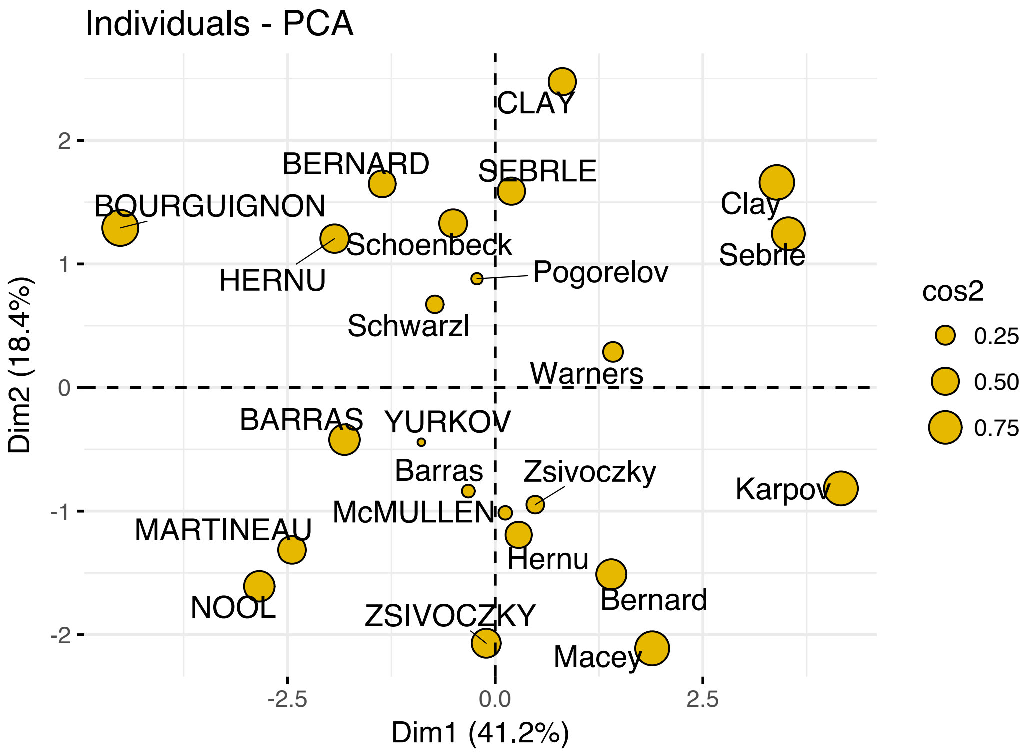


3) **PCA - Graph of individuals**:

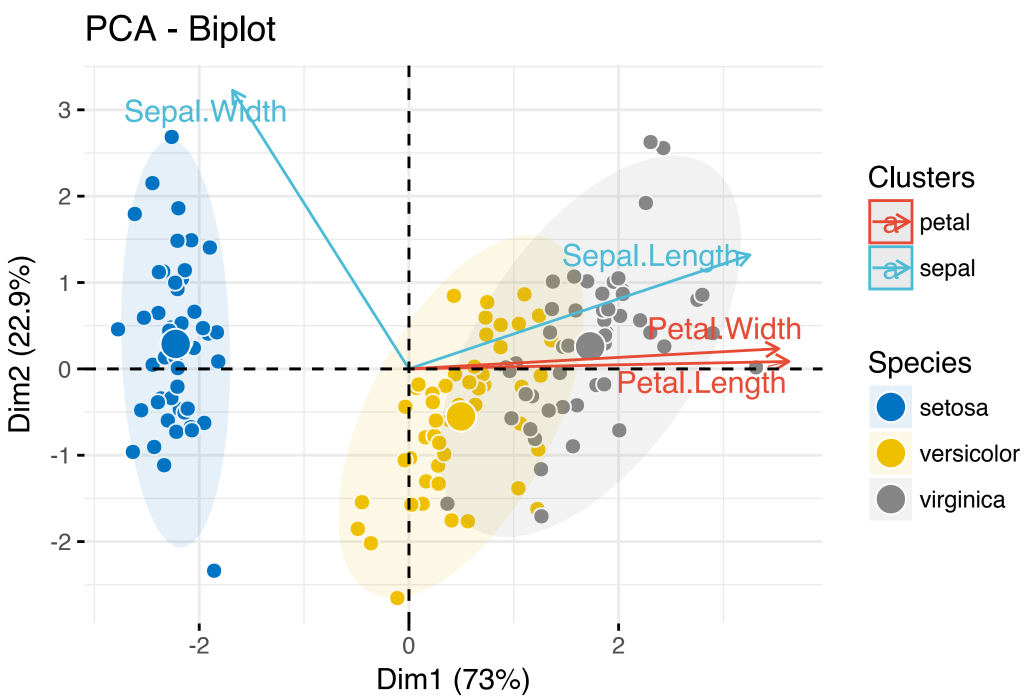
- Control automatically the color of individuals using the cos2 (the quality of the individuals on the factor map)



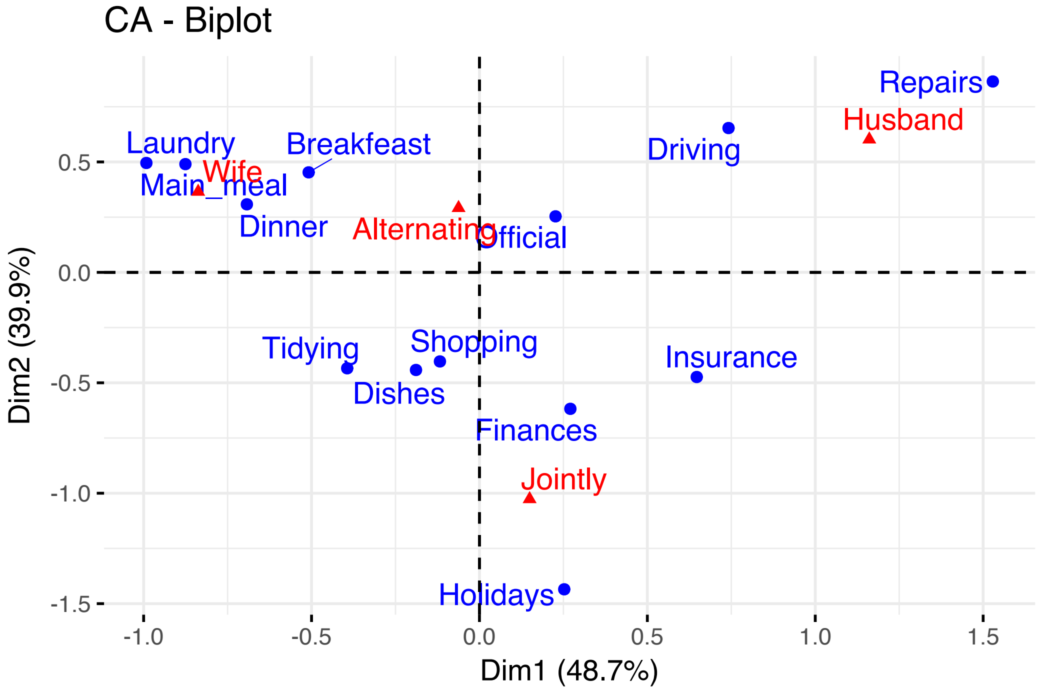
- Change the point size according to the cos2 of the corresponding individuals:



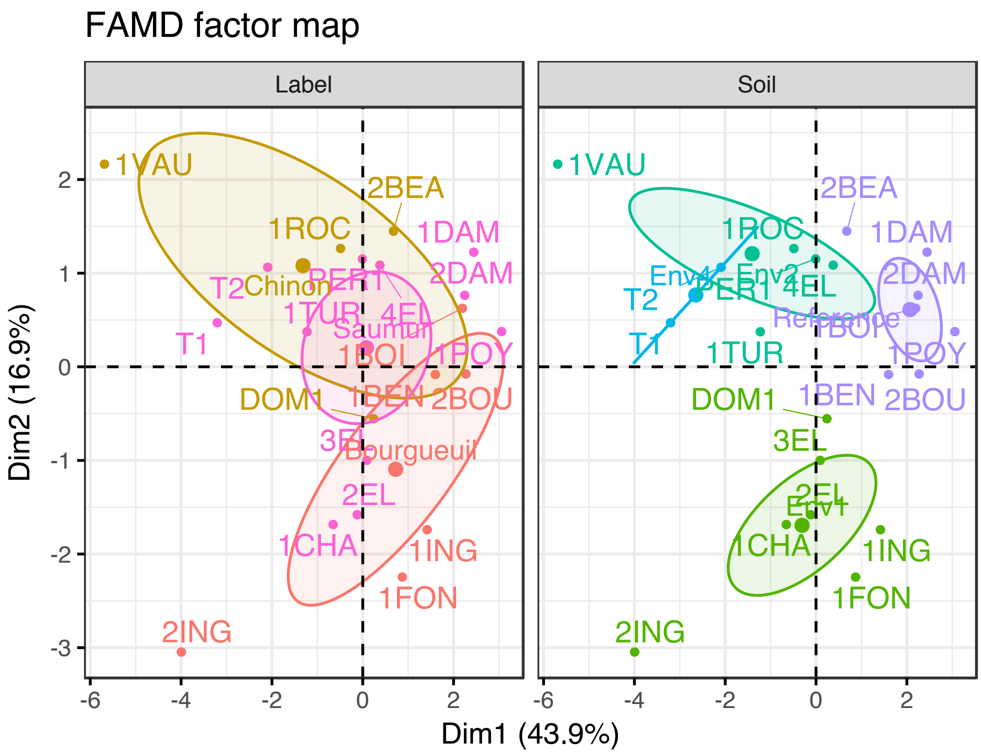
4) **PCA - Biplot of individuals and variables**



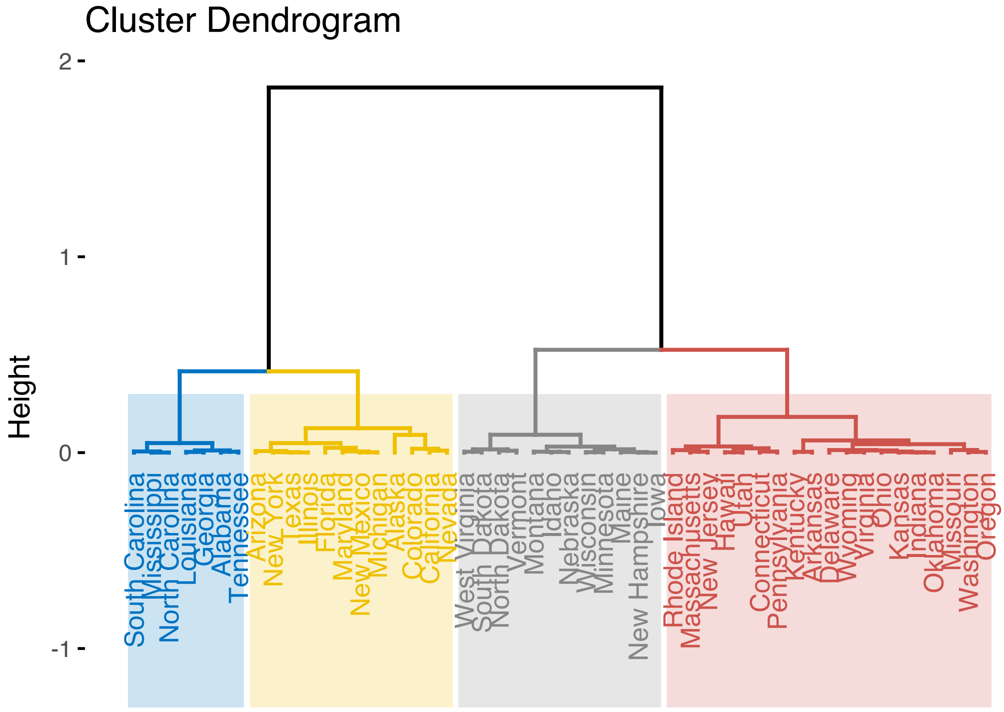
5) **Correspondence analysis**. Association between categorical variables.



6) **FAMD** - Analyzing mixed data



7) **Clustering on principal components**



0.4 Book website

The website for this book is located at : `http://www.sthda.com/english/`. It contains number of resources.

0.5 Executing the R codes from the PDF

For a single line R code, you can just copy the code from the PDF to the R console.

For a multiple-line R codes, an error is generated, sometimes, when you copy and paste directly the R code from the PDF to the R console. If this happens, a solution is to:

- Paste firstly the code in your R code editor or in your text editor
- Copy the code from your text/code editor to the R console

0.6 Acknowledgment

I sincerely thank all developers for their efforts behind the packages that factoextra depends on, namely, ggplot2 (Hadley Wickham, Springer-Verlag New York, 2009), FactoMineR (Sebastien Le et al., Journal of Statistical Software, 2008), dendextend (Tal Galili, Bioinformatics, 2015), cluster (Martin Maechler et al., 2016) and more.

0.7 Colophon

This book was built with:

- R 3.3.2
- factoextra 1.0.5
- FactoMineR 1.36
- ggpubr 0.1.5
- dplyr 0.7.2
- bookdown 0.4.3

About the author

Alboukadel Kassambara is a PhD in Bioinformatics and Cancer Biology. He works since many years on genomic data analysis and visualization (read more: `http://www.alboukadel.com/`).

He has work experiences in statistical and computational methods to identify prognostic and predictive biomarker signatures through integrative analysis of large-scale genomic and clinical data sets.

He created a bioinformatics web-tool named GenomicScape (www.genomicscape.com) which is an easy-to-use web tool for gene expression data analysis and visualization.

He developed also a training website on data science, named STHDA (Statistical Tools for High-throughput Data Analysis, www.sthda.com/english), which contains many tutorials on data analysis and visualization using R software and packages.

He is the author of many popular R packages for:

- multivariate data analysis (**factoextra**, `http://www.sthda.com/english/rpkgs/factoextra`),
- survival analysis (**survminer**, `http://www.sthda.com/english/rpkgs/survminer/`),
- correlation analysis (**ggcorrplot**, `http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2`),
- creating publication ready plots in R (**ggpubr**, `http://www.sthda.com/english/rpkgs/ggpubr`).

Recently, he published three books on data analysis and visualization:

1. Practical Guide to Cluster Analysis in R (`https://goo.gl/DmJ5y5`)
2. Guide to Create Beautiful Graphics in R (`https://goo.gl/vJ0OYb`).
3. Complete Guide to 3D Plots in R (`https://goo.gl/v5gwl0`).

Part I

Basics

Chapter 1

Introduction to R

R is a free and powerful statistical software for **analyzing** and **visualizing** data. If you want to learn easily the essential of R programming, visit our series of tutorials available on STHDA: `http://www.sthda.com/english/wiki/r-basics-quick-and-easy`.

In this chapter, we provide a very brief introduction to **R**, for installing R/RStudio as well as importing your data into R for computing principal component methods.

1.1 Installing R and RStudio

R and RStudio can be installed on Windows, MAC OSX and Linux platforms. RStudio is an integrated development environment for R that makes using R easier. It includes a console, code editor and tools for plotting.

1. R can be downloaded and installed from the Comprehensive R Archive Network (CRAN) webpage (`http://cran.r-project.org/`)
2. After installing R software, install also the RStudio software available at: `http://www.rstudio.com/products/RStudio/`.
3. Launch RStudio and start use R inside R studio.

1.2 Installing and loading R packages

An **R package** is an extension of R containing data sets and specific R functions to solve specific questions.

For example, in this book, you'll learn how to compute and visualize principal component methods using **FactoMineR** and **factoextra** R packages.

There are thousands other R packages available for download and installation from CRAN[1], Bioconductor[2] (biology related R packages) and GitHub[3] repositories.

[1]`https://cran.r-project.org/`
[2]`https://www.bioconductor.org/`
[3]`https://github.com/`

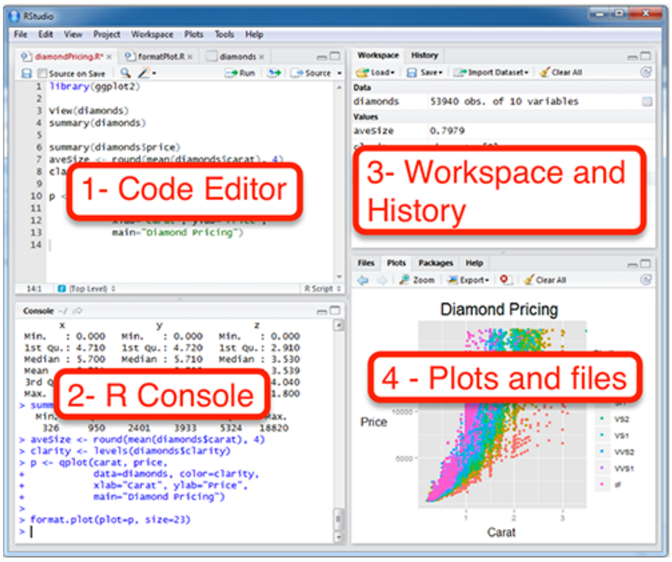


Figure 1.1: Rstudio interface

1. How to install packages from CRAN? Use the function *install.packages*():

```
install.packages("FactoMineR")
install.packages("factoextra")
```

2. How to install packages from GitHub? You should first install *devtools* if you don't have it already installed on your computer:

For example, the following R code installs the latest developmental version of *factoextra* R package developed by A. Kassambara (`https://github.com/kassambara/facoextra`) for multivariate data analysis and elegant visualization.

```
install.packages("devtools")
devtools::install_github("kassambara/factoextra")
```

Note that, GitHub contains the latest developmental version of R packages.

3. After installation, you must first load the package for using the functions in the package. The function *library*() is used for this task.

```
library("FactoMineR")
library("factoextra")
```

Now, we can use R functions, such as *PCA*() [in the FactoMineR package] for performing principal component analysis.

1.3 Getting help with functions in R

If you want to learn more about a given function, say PCA(), type this in R console:

```
?PCA
```

1.4 Importing your data into R

1. **Prepare your file** as follow:

- Use the first row as **column names**. Generally, columns represent **variables**
- Use the first column as **row names**. Generally rows represent **observations** or **individuals**.
- Each row/column name should be unique, so remove duplicated names.
- Avoid names with blank spaces. Good column names: *Long_jump* or *Long.jump*. Bad column name: *Long jump*.
- Avoid names with special symbols: ?, $, *, +, #, (,), -, /, }, {, |, >, < etc. Only underscore can be used.
- Avoid beginning variable names with a number. Use letter instead. Good column names: sport_100m or x100m. Bad column name: 100m
- R is case sensitive. This means that Name is different from Name or NAME.
- Avoid blank rows in your data.
- Delete any comments in your file.
- Replace missing values by **NA** (for not available)
- If you have a column containing date, use the four digit format. Good format: 01/01/2016. Bad format: 01/01/16

2. The **final file** should look like this:

name	x100m	Long.jump	Shot.put	High.jump
SEBRLE	11.04	7.58	14.83	2.07
CLAY	10.76	7.4	14.26	1.86
BERNARD	11.02	7.23	14.25	1.92
YURKOV	11.34	7.09	15.19	2.1
ZSIVOCZKY	11.13	7.3	NA	2.01
McMULLEN	10.83	7.31	13.76	2.13
MARTINEAU	NA	6.81	14.57	1.95
HERNU	NA	7.56	14.41	1.86
BARRAS	11.33	6.97	14.09	1.95
NOOL	11.33	7.27	12.68	1.98
BOURGUIGNON	11.36	6.8	13.46	1.86

Figure 1.2: General data format for importation into R

3. **Save your file**

We recommend to save your file into **.txt** (tab-delimited text file) or **.csv** (comma separated value file) format.

4. **Get your data into R**:

Use the R code below. You will be asked to choose a file:

```r
# .txt file: Read tab separated values
my_data <- read.delim(file.choose(), row.names = 1)
```

```r
# .csv file: Read comma (",") separated values
my_data <- read.csv(file.choose(), row.names = 1)

# .csv file: Read semicolon (";") separated values
my_data <- read.csv2(file.choose(), row.names = 1)
```

Using these functions, the imported data will be of class **data.frame** (R terminology).

You can read more about how to import data into R at this link: http://www.sthda.com/english/wiki/importing-data-into-r

1.5 Demo data sets

R comes with several *built-in data sets*, which are generally used as demo data for playing with R functions. The most used R demo data sets include: **USArrests**, **iris** and **mtcars**. To load a demo data set, use the function **data**() as follow:

```r
data("USArrests")   # Loading
head(USArrests, 3)  # Print the first 3 rows
```

```
##         Murder Assault UrbanPop Rape
## Alabama   13.2     236       58 21.2
## Alaska    10.0     263       48 44.5
## Arizona    8.1     294       80 31.0
```

If you want learn more about USArrests data sets, type this:

```r
?USArrests
```

To select just certain columns from a data frame, you can either refer to the columns by name or by their location (i.e., column 1, 2, 3, etc.).

```r
# Access the data in 'Murder' column
# dollar sign is used
head(USArrests$Murder)
```

```
## [1] 13.2 10.0  8.1  8.8  9.0  7.9
```

```r
# Or use this
USArrests[, 'Murder']
# Or use this
USArrests[, 1] # column number 1
```

1.6 Close your R/RStudio session

Each time you close R/RStudio, you will be asked whether you want to save the data from your R session. If you decide to save, the data will be available in future R sessions.

Chapter 2

Required R packages

2.1 FactoMineR & factoextra

There are a number of R packages implementing principal component methods. These packages include: *FactoMineR*, *ade4*, *stats*, *ca*, *MASS* and *ExPosition*.

However, the result is presented differently depending on the used package.

To help in the interpretation and in the visualization of multivariate analysis - such as cluster analysis and principal component methods - we developed an easy-to-use R package named **factoextra** (official online documentation: `http://www.sthda.com/english/rpkgs/factoextra`)(Kassambara and Mundt, 2017).

> No matter which package you decide to use for computing principal component methods, the factoextra R package can help to extract easily, in a human readable data format, the analysis results from the different packages mentioned above. factoextra provides also convenient solutions to create ggplot2-based beautiful graphs.

In this book, we'll use mainly:

- the **FactoMineR** package (Husson et al., 2017a) to compute principal component methods;
- and the **factoextra** package (Kassambara and Mundt, 2017) for extracting, visualizing and interpreting the results.

The other packages - ade4, ExPosition, etc - will be presented briefly.

The Figure 2.1 illustrates the key functionality of FactoMineR and factoextra.

Methods, which outputs can be visualized using the factoextra package are shown on the Figure 2.2:

2.2 Installation

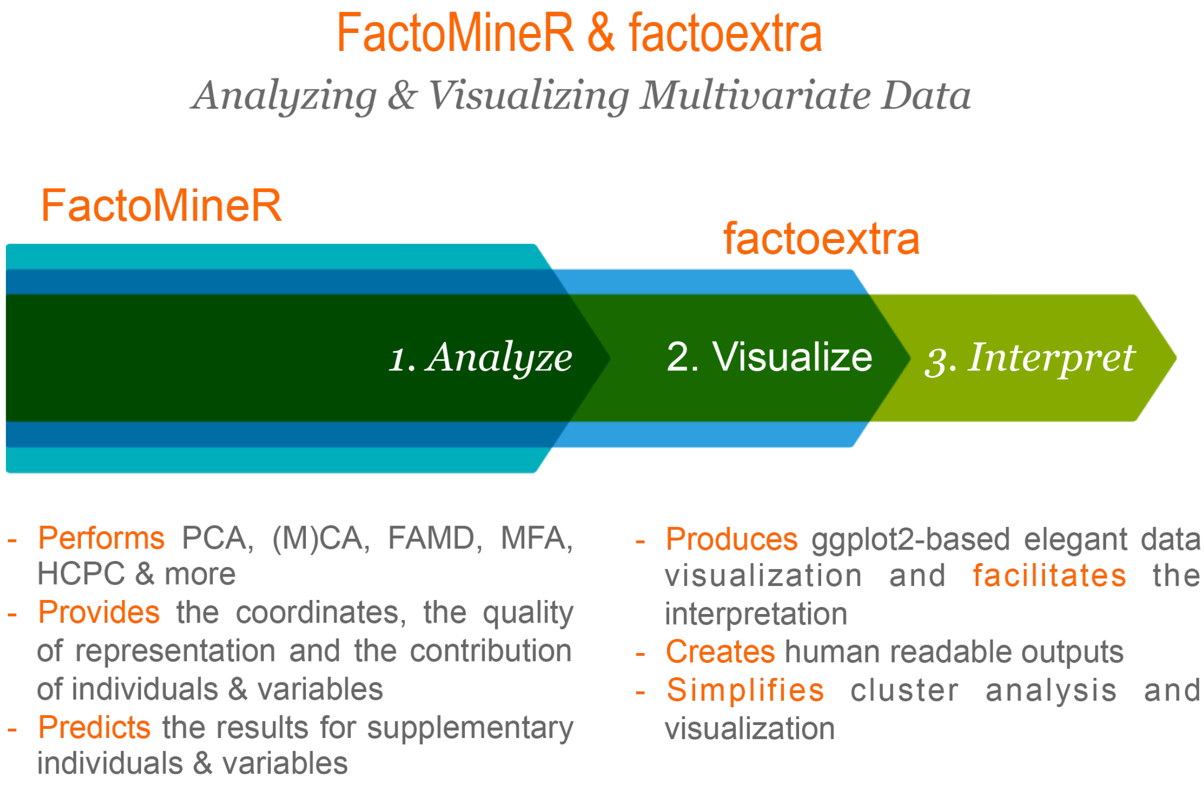


Figure 2.1: Key features of FactoMineR and factoextra for multivariate analysis

2.2.1 Installing FactoMineR

The FactoMineR package can be installed and loaded as follow:

```
# Install
install.packages("FactoMineR")

# Load
library("FactoMineR")
```

2.2.2 Installing factoextra

- factoextra can be installed from CRAN[1] as follow:

```
install.packages("factoextra")
```

- Or, install the latest developmental version from Github[2]

```
if(!require(devtools)) install.packages("devtools")
devtools::install_github("kassambara/factoextra")
```

- Load factoextra as follow :

[1] https://cran.r-project.org/package=factoextra
[2] https://github.com/kassambara/factoextra

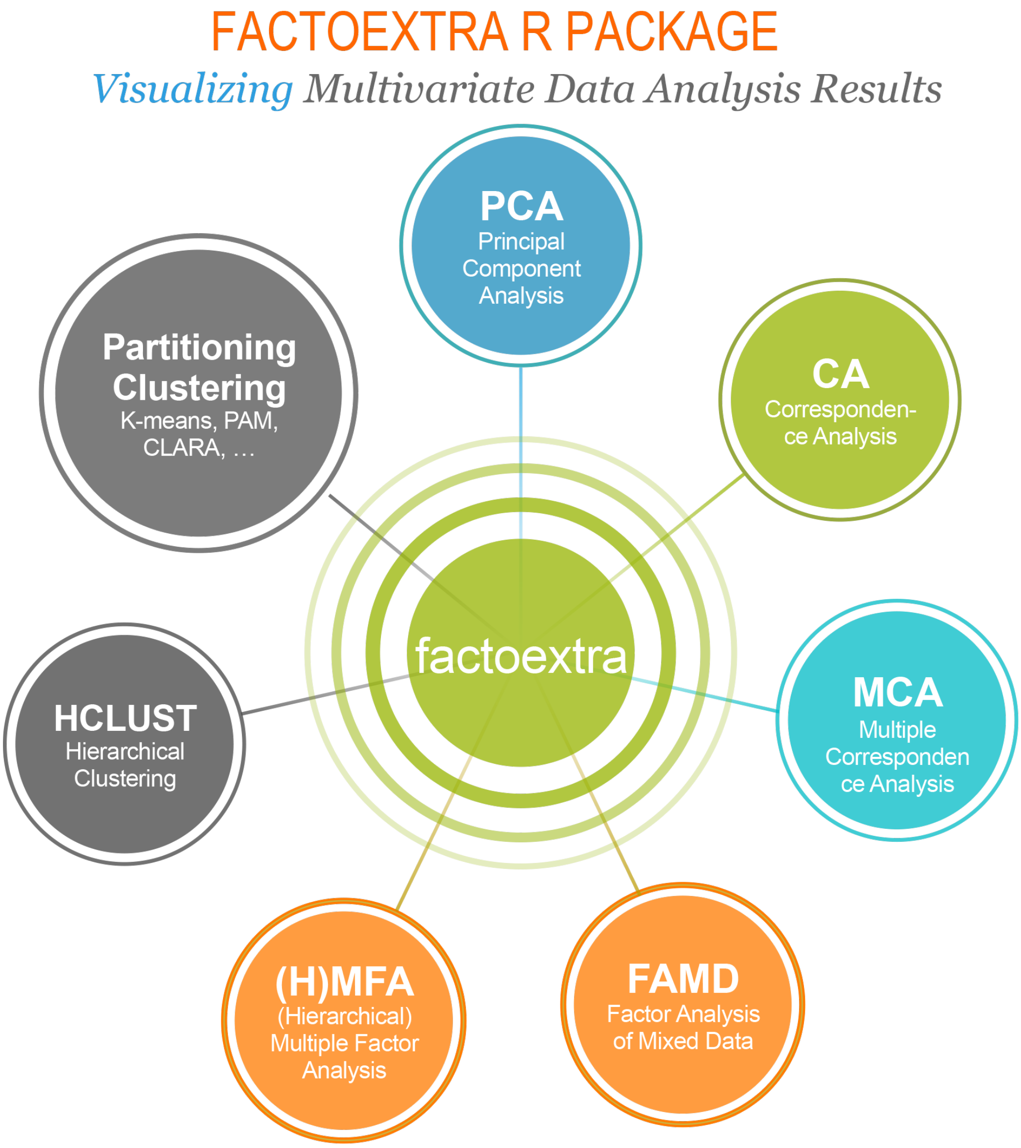


Figure 2.2: Principal component methods and clustering methods supported by the factoextra R package

```
library("factoextra")
```

2.3 Main R functions

2.3.1 Main functions in FactoMineR

Functions for computing principal component methods and clustering:

Functions	Description
PCA	Principal component analysis.
CA	Correspondence analysis.

Functions	Description
MCA	Multiple correspondence analysis.
FAMD	Factor analysis of mixed data.
MFA	Multiple factor analysis.
HCPC	Hierarchical clustering on principal components.
dimdesc	Dimension description.

2.3.2 Main functions in factoextra

factoextra functions covered in this book are listed in the table below. See the online documentation (`http://www.sthda.com/english/rpkgs/factoextra`) for a complete list.

- **Visualizing principal component method outputs**

Functions	Description
fviz_eig (or fviz_eigenvalue)	Visualize eigenvalues.
fviz_pca	Graph of PCA results.
fviz_ca	Graph of CA results.
fviz_mca	Graph of MCA results.
fviz_mfa	Graph of MFA results.
fviz_famd	Graph of FAMD results.
fviz_hmfa	Graph of HMFA results.
fviz_ellipses	Plot ellipses around groups.
fviz_cos2	Visualize element cos2. [3]
fviz_contrib	Visualize element contributions. [4]

- **Extracting data from principal component method outputs**. The following functions extract all the results (coordinates, squared cosine, contributions) for the active individuals/variables from the analysis outputs.

Functions	Description
get_eigenvalue	Access to the dimension eigenvalues.
get_pca	Access to PCA outputs.
get_ca	Access to CA outputs.
get_mca	Access to MCA outputs.
get_mfa	Access to MFA outputs.
get_famd	Access to MFA outputs.
get_hmfa	Access to HMFA outputs.
facto_summarize	Summarize the analysis.

- **Clustering analysis and visualization**

[3] Cos2: quality of representation of the row/column variables on the principal component maps.
[4] This is the contribution of row/column elements to the definition of the principal components.

Functions	Description
fviz_dend	Enhanced Visualization of Dendrogram.
fviz_cluster	Visualize Clustering Results.

Part II

Classical Methods

Chapter 3

Principal Component Analysis

3.1 Introduction

Principal component analysis (**PCA**) allows us to summarize and to visualize the information in a data set containing individuals/observations described by multiple inter-correlated quantitative variables. Each variable could be considered as a different dimension. If you have more than 3 variables in your data sets, it could be very difficult to visualize a multi-dimensional hyperspace.

Principal component analysis is used to extract the important information from a multivariate data table and to express this information as a set of few new variables called **principal components**. These new variables correspond to a linear combination of the originals. The number of principal components is less than or equal to the number of original variables.

The information in a given data set corresponds to the *total variation* it contains. The goal of PCA is to identify directions (or principal components) along which the variation in the data is maximal.

In other words, PCA reduces the dimensionality of a multivariate data to two or three principal components, that can be visualized graphically, with minimal loss of information.

In this chapter, we describe the basic idea of PCA and, demonstrate how to compute and visualize PCA using R software. Additionally, we'll show how to reveal the most important variables that explain the variations in a data set.

3.2 Basics

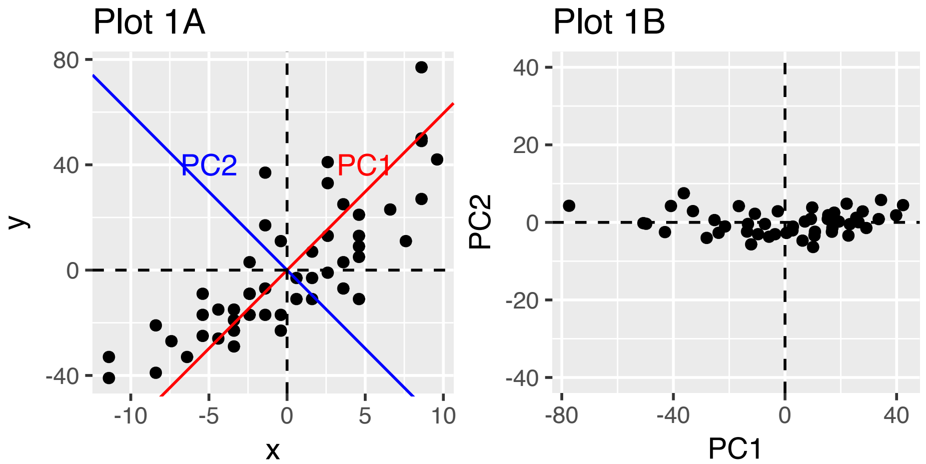
Understanding the details of PCA requires knowledge of linear algebra. Here, we'll explain only the basics with simple graphical representation of the data.

In the Plot 1A below, the data are represented in the X-Y coordinate system. The dimension reduction is achieved by identifying the principal directions, called principal components, in which the data varies.

PCA assumes that the directions with the largest variances are the most "important" (i.e, the most principal).

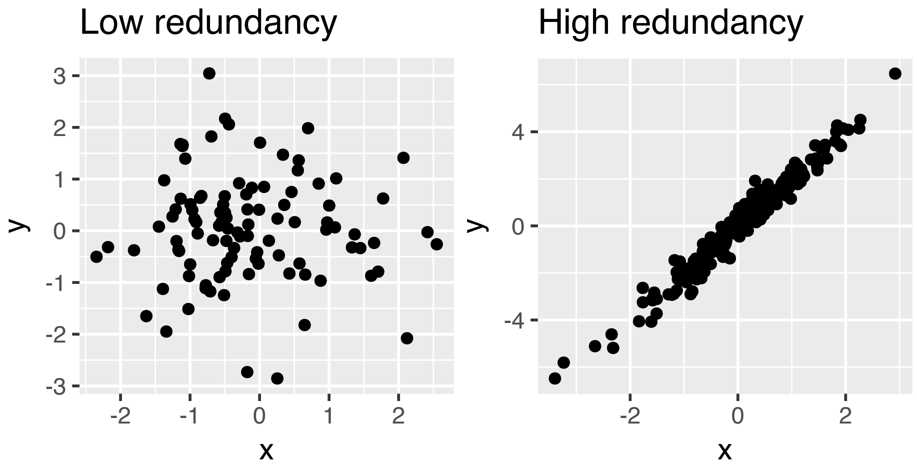
In the figure below, the *PC1 axis* is the **first principal direction** along which the samples show the largest variation. The *PC2 axis* is the **second most important direction** and it is **orthogonal** to the PC1 axis.

The dimensionality of our two-dimensional data can be reduced to a single dimension by projecting each sample onto the first principal component (Plot 1B)



Technically speaking, the amount of variance retained by each principal component is measured by the so-called **eigenvalue**.

Note that, the PCA method is particularly useful when the variables within the data set are highly correlated. Correlation indicates that there is redundancy in the data. Due to this redundancy, PCA can be used to reduce the original variables into a smaller number of new variables (= **principal components**) explaining most of the variance in the original variables.



Taken together, the main purpose of principal component analysis is to:

- identify hidden pattern in a data set,
- reduce the dimensionnality of the data by **removing the noise** and **redundancy** in the data,
- identify correlated variables

3.3 Computation

3.3.1 R packages

Several functions from different packages are available in the *R software* for computing PCA:

- *prcomp*() and *princomp*() [built-in R *stats* package],
- *PCA*() [*FactoMineR* package],
- *dudi.pca*() [*ade4* package],
- and *epPCA*() [*ExPosition* package]

No matter what function you decide to use, you can easily extract and visualize the results of PCA using R functions provided in the *factoextra* R package.

Here, we'll use the two packages FactoMineR (for the analysis) and factoextra (for ggplot2-based visualization).

Install the two packages as follow:

```
install.packages(c("FactoMineR", "factoextra"))
```

Load them in R, by typing this:

```
library("FactoMineR")
library("factoextra")
```

3.3.2 Data format

We'll use the demo data sets *decathlon2* from the *factoextra* package:

```
data(decathlon2)
# head(decathlon2)
```

As illustrated in Figure 3.1, the data used here describes athletes' performance during two sporting events (Desctar and OlympicG). It contains 27 individuals (athletes) described by 13 variables.

Note that, only some of these individuals and variables will be used to perform the principal component analysis. The coordinates of the remaining individuals and variables on the factor map will be predicted after the PCA.

In PCA terminology, our data contains :

- *Active individuals* (in light blue, rows 1:23) : Individuals that are used during the principal component analysis.
- *Supplementary individuals* (in dark blue, rows 24:27) : The coordinates of these individuals will be predicted using the PCA information and parameters obtained with active individuals/variables

name	100m	Long.jump	//	Javeline	1500m	Rank	Points	Competition
SEBRLE	11.04	7.58		63.19	291.7	1	8217	Decastar
CLAY	10.76	7.4		60.15	301.5	2	8122	Decastar
Macey	10.89	7.47		58.46	265.42	4	8414	OlympicG
Warners	10.62	7.74		55.39	278.05	5	8343	OlympicG
\\								
Zsivoczky	10.91	7.14		63.45	269.54	6	8287	OlympicG
Hernu	10.97	7.19		57.76	264.35	7	8237	OlympicG
Pogorelov	10.95	7.31		53.45	287.63	11	8084	OlympicG
Schoenbeck	10.9	7.3		60.89	278.82	12	8077	OlympicG
Barras	11.14	6.99		64.55	267.09	13	8067	OlympicG
KARPOV	11.02	7.3		50.31	300.2	3	8099	Decastar
WARNERS	11.11	7.6		51.77	278.1	6	8030	Decastar
Nool	10.8	7.53		61.33	276.33	8	8235	OlympicG
Drews	10.87	7.38		51.53	274.21	19	7926	OlympicG

Active individuals
Active variables
Supplementary **quantitative** variables
Supplementary **qualitative** variable
Supplementary individuals

Figure 3.1: Principal component analysis data format

- *Active variables* (in pink, columns 1:10) : Variables that are used for the principal component analysis.
- *Supplementary variables*: As supplementary individuals, the coordinates of these variables will be predicted also. These can be:
  - *Supplementary continuous variables* (red): Columns 11 and 12 corresponding respectively to the rank and the points of athletes.
  - *Supplementary qualitative variables* (green): Column 13 corresponding to the two athlete-tic meetings (2004 Olympic Game or 2004 Decastar). This is a categorical (or factor) variable factor. It can be used to color individuals by groups.

We start by subsetting active individuals and active variables for the principal component analysis:

```
decathlon2.active <- decathlon2[1:23, 1:10]
head(decathlon2.active[, 1:6], 4)
```

```
##         X100m Long.jump Shot.put High.jump X400m X110m.hurdle
## SEBRLE   11.0      7.58     14.8      2.07  49.8         14.7
## CLAY     10.8      7.40     14.3      1.86  49.4         14.1
## BERNARD  11.0      7.23     14.2      1.92  48.9         15.0
## YURKOV   11.3      7.09     15.2      2.10  50.4         15.3
```

3.3.3 Data standardization

In principal component analysis, variables are often scaled (i.e. standardized). This is particularly recommended when variables are measured in different scales (e.g: kilograms, kilometers, centimeters, …); otherwise, the PCA outputs obtained will be severely affected.

The goal is to make the variables comparable. Generally variables are scaled to have i) standard deviation one and ii) mean zero.

The standardization of data is an approach widely used in the context of gene expression data analysis before PCA and clustering analysis. We might also want to scale the data when the mean and/or the standard deviation of variables are largely different.

When scaling variables, the data can be transformed as follow:

$$\frac{x_i - mean(x)}{sd(x)}$$

Where $mean(x)$ is the mean of x values, and $sd(x)$ is the standard deviation (SD).

The R base function *scale*() can be used to standardize the data. It takes a numeric matrix as an input and performs the scaling on the columns.

Note that, by default, the function **PCA**() [in *FactoMineR*], standardizes the data automatically during the PCA; so you don't need do this transformation before the PCA.

3.3.4 R code

The function *PCA*() [*FactoMineR* package] can be used. A simplified format is :

```
PCA(X, scale.unit = TRUE, ncp = 5, graph = TRUE)
```

- **X**: a data frame. Rows are individuals and columns are numeric variables
- **scale.unit**: a logical value. If *TRUE*, the data are scaled to unit variance before the analysis. This standardization to the same scale avoids some variables to become dominant just because of their large measurement units. It makes variable comparable.
- **ncp**: number of dimensions kept in the final results.
- **graph**: a logical value. If TRUE a graph is displayed.

The R code below, computes principal component analysis on the active individuals/variables:

```
library("FactoMineR")
res.pca <- PCA(decathlon2.active, graph = FALSE)
```

The output of the function *PCA*() is a list, including the following components :

```r
print(res.pca)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 23 individuals, described by 10 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

The object that is created using the function *PCA*() contains many information found in many different lists and matrices. These values are described in the next section.

3.4 Visualization and Interpretation

We'll use the *factoextra* R package to help in the interpretation of PCA. No matter what function you decide to use [stats::prcomp(), FactoMiner::PCA(), ade4::dudi.pca(), ExPosition::epPCA()], you can easily extract and visualize the results of PCA using R functions provided in the *factoextra* R package.

These functions include:

- *get_eigenvalue*(res.pca): Extract the eigenvalues/variances of principal components
- *fviz_eig*(res.pca): Visualize the eigenvalues
- *get_pca_ind*(res.pca), *get_pca_var*(res.pca): Extract the results for individuals and variables, respectively.
- *fviz_pca_ind*(res.pca), *fviz_pca_var*(res.pca): Visualize the results individuals and variables, respectively.
- *fviz_pca_biplot*(res.pca): Make a biplot of individuals and variables.

In the next sections, we'll illustrate each of these functions.

3.4.1 Eigenvalues / Variances

As described in previous sections, the *eigenvalues* measure the amount of variation retained by each principal component. *Eigenvalues* are large for the first PCs and small for the subsequent PCs. That is, the first PCs corresponds to the directions with the maximum amount of variation in the data set.

We examine the eigenvalues to determine the number of principal components to be considered. The eigenvalues and the proportion of variances (i.e., information) retained by the principal components (PCs) can be extracted using the function *get_eigenvalue*() [*factoextra* package].

```
library("factoextra")
eig.val <- get_eigenvalue(res.pca)
eig.val
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1       4.124            41.24                        41.2
## Dim.2       1.839            18.39                        59.6
## Dim.3       1.239            12.39                        72.0
## Dim.4       0.819             8.19                        80.2
## Dim.5       0.702             7.02                        87.2
## Dim.6       0.423             4.23                        91.5
## Dim.7       0.303             3.03                        94.5
## Dim.8       0.274             2.74                        97.2
## Dim.9       0.155             1.55                        98.8
## Dim.10      0.122             1.22                       100.0
```

The sum of all the eigenvalues give a total variance of 10.

The proportion of variation explained by each eigenvalue is given in the second column. For example, 4.124 divided by 10 equals 0.4124, or, about 41.24% of the variation is explained by this first eigenvalue. The cumulative percentage explained is obtained by adding the successive proportions of variation explained to obtain the running total. For instance, 41.242% plus 18.385% equals 59.627%, and so forth. Therefore, about 59.627% of the variation is explained by the first two eigenvalues together.

Eigenvalues can be used to determine the number of principal components to retain after PCA (Kaiser, 1961):

- An *eigenvalue* > 1 indicates that PCs account for more variance than accounted by one of the original variables in standardized data. This is commonly used as a cutoff point for which PCs are retained. This holds true only when the data are standardized.
- You can also limit the number of component to that number that accounts for a certain fraction of the total variance. For example, if you are satisfied with 70% of the total variance explained then use the number of components to achieve that.

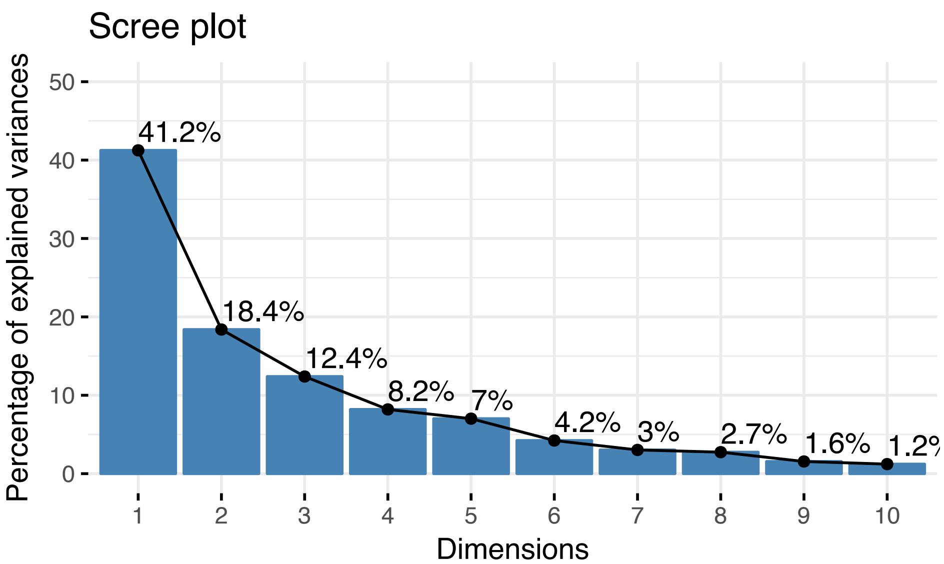
Unfortunately, there is no well-accepted objective way to decide how many principal components are enough. This will depend on the specific field of application and the specific data set. In practice, we tend to look at the first few principal components in order to find interesting patterns in the data.

In our analysis, the first three principal components explain 72% of the variation. This is an acceptably large percentage.

An alternative method to determine the number of principal components is to look at a Scree Plot, which is the plot of eigenvalues ordered from largest to the smallest. The number of component is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size (Jollife, 2002, Peres-Neto et al. (2005)).

The scree plot can be produced using the function *fviz_eig*() or *fviz_screeplot*() [*factoextra* package].

```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```



From the plot above, we might want to stop at the fifth principal component. 87% of the information (variances) contained in the data are retained by the first five principal components.

3.4.2 Graph of variables

3.4.2.1 Results

A simple method to extract the results, for variables, from a PCA output is to use the function *get_pca_var*() [*factoextra* package]. This function provides a list of matrices containing all the results for the active variables (coordinates, correlation between variables and axes, squared cosine and contributions)

```
var <- get_pca_var(res.pca)
var
```

```
## Principal Component Analysis Results for variables
##  ===================================================
```

```
##   Name       Description
## 1 "$coord"   "Coordinates for the variables"
## 2 "$cor"     "Correlations between variables and dimensions"
## 3 "$cos2"    "Cos2 for the variables"
## 4 "$contrib" "contributions of the variables"
```

The components of the *get_pca_var*() can be used in the plot of variables as follow:

- *var$coord*: coordinates of variables to create a scatter plot
- *var$cos2*: represents the quality of representation for variables on the factor map. It's calculated as the squared coordinates: var.cos2 = var.coord * var.coord.
- *var$contrib*: contains the contributions (in percentage) of the variables to the principal components. The contribution of a variable (var) to a given principal component is (in percentage) : (var.cos2 * 100) / (total cos2 of the component).

> Note that, it's possible to plot variables and to color them according to either i) their quality on the factor map (cos2) or ii) their contribution values to the principal components (contrib).

The different components can be accessed as follow:

```
# Coordinates
head(var$coord)

# Cos2: quality on the factore map
head(var$cos2)

# Contributions to the principal components
head(var$contrib)
```

In this section, we describe how to visualize variables and draw conclusions about their correlations. Next, we highlight variables according to either i) their quality of representation on the factor map or ii) their contributions to the principal components.

3.4.2.2 Correlation circle

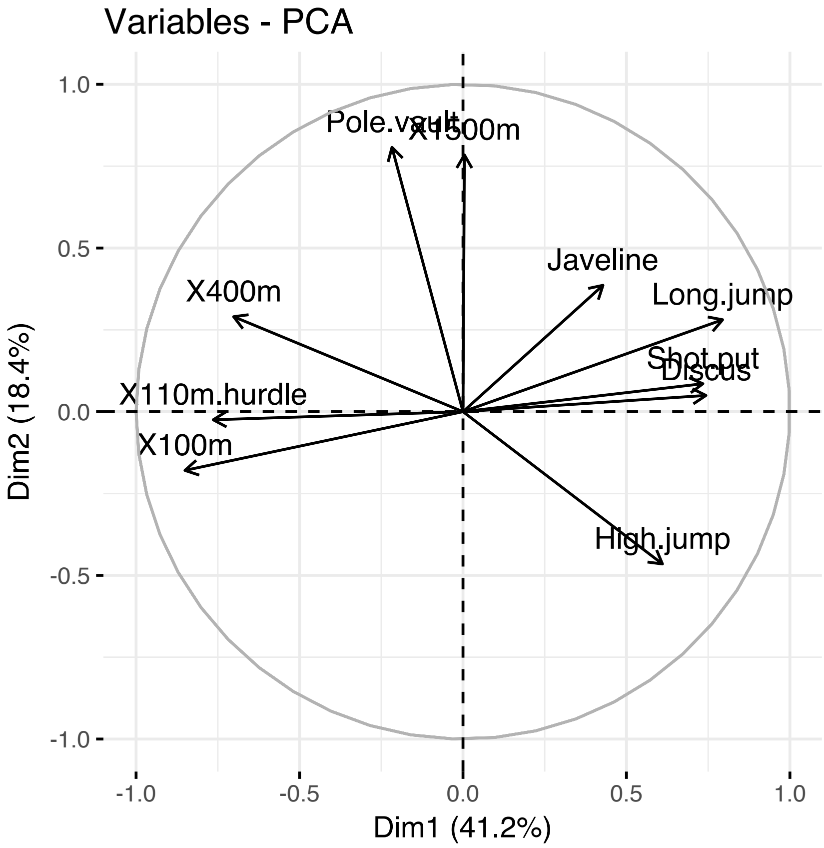
The correlation between a variable and a principal component (PC) is used as the coordinates of the variable on the PC. The representation of variables differs from the plot of the observations: The observations are represented by their projections, but the variables are represented by their correlations (Abdi and Williams, 2010).

```
# Coordinates of variables
head(var$coord, 4)
```

```
##              Dim.1   Dim.2  Dim.3   Dim.4  Dim.5
## X100m       -0.851 -0.1794  0.302  0.0336 -0.194
## Long.jump    0.794  0.2809 -0.191 -0.1154  0.233
## Shot.put     0.734  0.0854  0.518  0.1285 -0.249
## High.jump    0.610 -0.4652  0.330  0.1446  0.403
```

To plot variables, type this:

```r
fviz_pca_var(res.pca, col.var = "black")
```



The plot above is also known as variable correlation plots. It shows the relationships between all variables. It can be interpreted as follow:

- Positively correlated variables are grouped together.
- Negatively correlated variables are positioned on opposite sides of the plot origin (opposed quadrants).

- The distance between variables and the origin measures the quality of the variables on the factor map. Variables that are away from the origin are well represented on the factor map.

3.4.2.3 Quality of representation

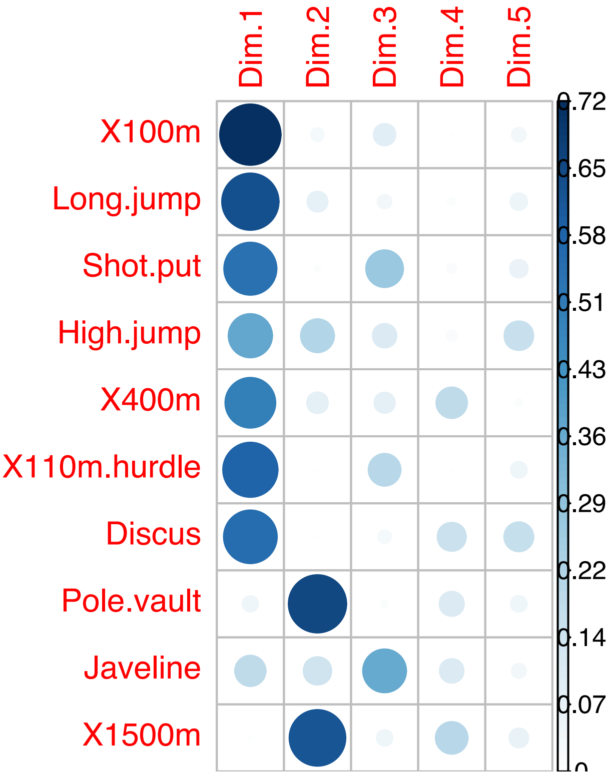
The quality of representation of the variables on factor map is called **cos2** (square cosine, squared coordinates) . You can access to the cos2 as follow:

```r
head(var$cos2, 4)
```

```
##              Dim.1   Dim.2  Dim.3   Dim.4  Dim.5
## X100m        0.724 0.03218 0.0909 0.00113 0.0378
## Long.jump    0.631 0.07888 0.0363 0.01331 0.0544
## Shot.put     0.539 0.00729 0.2679 0.01650 0.0619
## High.jump    0.372 0.21642 0.1090 0.02089 0.1622
```

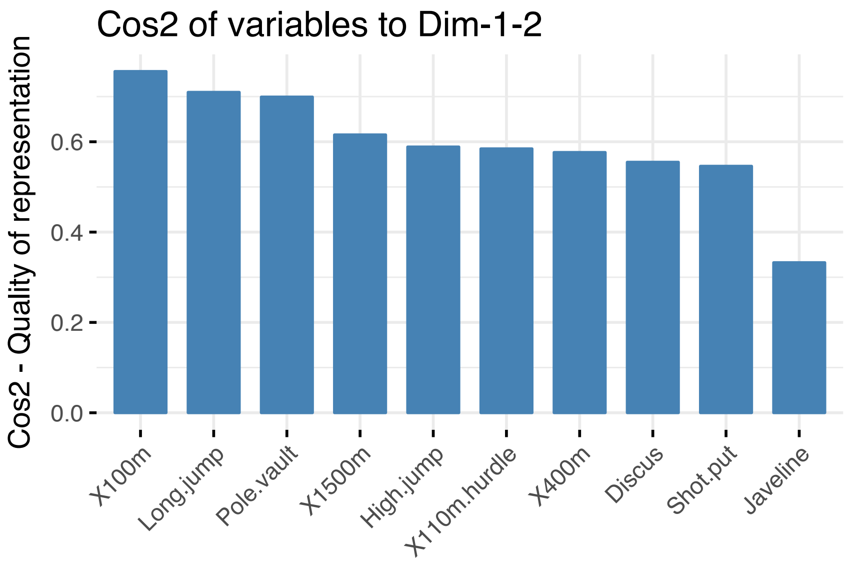
You can visualize the cos2 of variables on all the dimensions using the corrplot package:

```
library("corrplot")
corrplot(var$cos2, is.corr=FALSE)
```



It's also possible to create a bar plot of variables cos2 using the function *fviz_cos2*()[in factoextra]:

```
# Total cos2 of variables on Dim.1 and Dim.2
fviz_cos2(res.pca, choice = "var", axes = 1:2)
```



Note that,

- A high cos2 indicates a good representation of the variable on the principal component. In this case the variable is positioned close to the circumference of the correlation circle.
- A low cos2 indicates that the variable is not perfectly represented by the PCs. In this case the variable is close to the center of the circle.

For a given variable, the sum of the cos2 on all the principal components is equal to one.

If a variable is perfectly represented by only two principal components (Dim.1 & Dim.2), the sum of the cos2 on these two PCs is equal to one. In this case the variables will be positioned on the circle of correlations.

For some of the variables, more than 2 components might be required to perfectly represent the data. In this case the variables are positioned inside the circle of correlations.

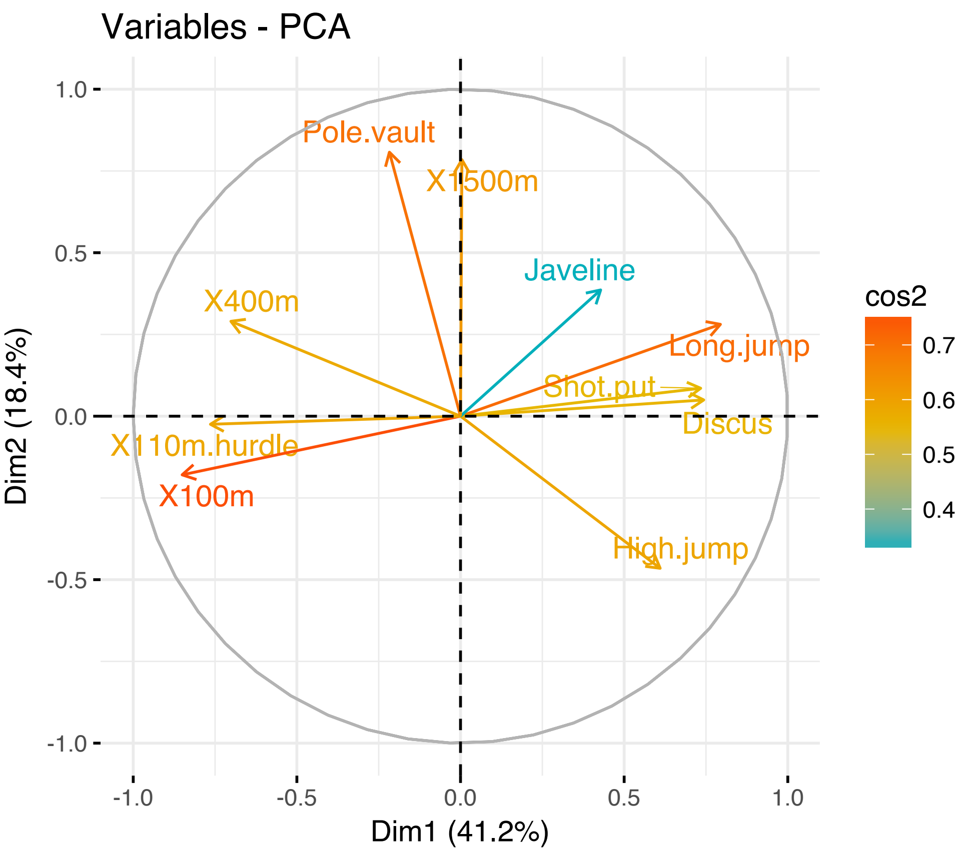
In summary:

- The cos2 values are used to estimate the quality of the representation
- The closer a variable is to the circle of correlations, the better its representation on the factor map (and the more important it is to interpret these components)
- Variables that are closed to the center of the plot are less important for the first components.

It's possible to color variables by their cos2 values using the argument *col.var = "cos2"*. This produces a gradient colors. In this case, the argument *gradient.cols* can be used to provide a custom color. For instance, *gradient.cols = c("white", "blue", "red")* means that:

- variables with low cos2 values will be colored in "white"
- variables with mid cos2 values will be colored in "blue"
- variables with high cos2 values will be colored in red

```
# Color by cos2 values: quality on the factor map
fviz_pca_var(res.pca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping
             )
```



Note that, it's also possible to change the transparency of the variables according to their cos2 values using the option *alpha.var = "cos2"*. For example, type this:

```
# Change the transparency by cos2 values
fviz_pca_var(res.pca, alpha.var = "cos2")
```

3.4.2.4 Contributions of variables to PCs

The contributions of variables in accounting for the variability in a given principal component are expressed in percentage.

- Variables that are correlated with PC1 (i.e., Dim.1) and PC2 (i.e., Dim.2) are the most important in explaining the variability in the data set.
- Variables that do not correlated with any PC or correlated with the last dimensions are variables with low contribution and might be removed to simplify the overall analysis.

The contribution of variables can be extracted as follow :

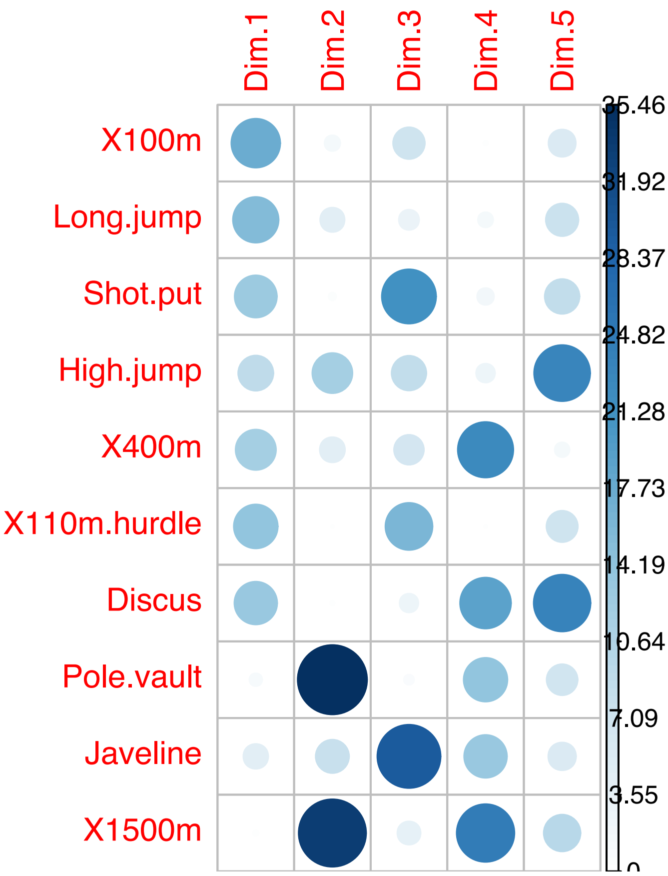
```
head(var$contrib, 4)
```

```
##           Dim.1  Dim.2 Dim.3 Dim.4 Dim.5
## X100m     17.54  1.751  7.34 0.138  5.39
## Long.jump 15.29  4.290  2.93 1.625  7.75
## Shot.put  13.06  0.397 21.62 2.014  8.82
## High.jump  9.02 11.772  8.79 2.550 23.12
```

The larger the value of the contribution, the more the variable contributes to the component.

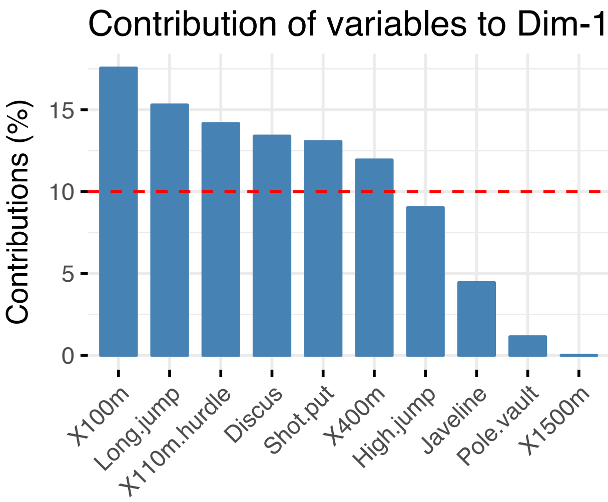
It's possible to use the function *corrplot*() [*corrplot* package] to highlight the most contributing variables for each dimension:

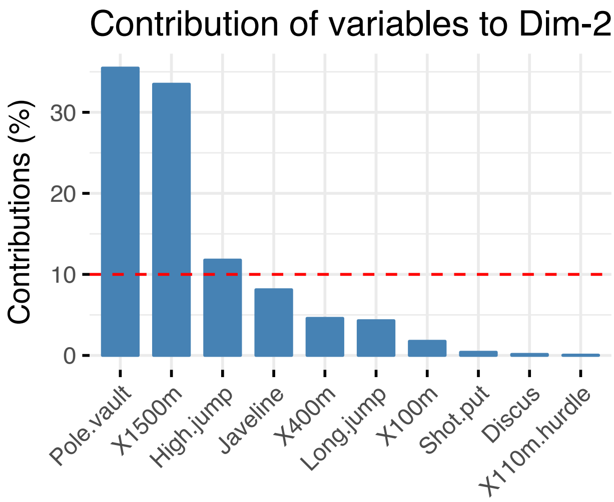
```
library("corrplot")
corrplot(var$contrib, is.corr=FALSE)
```



The function *fviz_contrib*() [factoextra package] can be used to draw a bar plot of variable contributions. If your data contains many variables, you can decide to show only the top contributing variables. The R code below shows the top 10 variables contributing to the principal components:

```
# Contributions of variables to PC1
fviz_contrib(res.pca, choice = "var", axes = 1, top = 10)

# Contributions of variables to PC2
fviz_contrib(res.pca, choice = "var", axes = 2, top = 10)
```





The total contribution to PC1 and PC2 is obtained with the following R code:

```
fviz_contrib(res.pca, choice = "var", axes = 1:2, top = 10)
```

The red dashed line on the graph above indicates the expected average contribution. If the contribution of the variables were uniform, the expected value would be 1/length(variables) = 1/10 = 10%. For a given component, a variable with a contribution larger than this cutoff could be considered as important in contributing to the component.

Note that, the total contribution of a given variable, on explaining the variations retained by two principal components, say PC1 and PC2, is calculated as contrib = [(C1 * Eig1) + (C2 * Eig2)]/(Eig1 + Eig2), where

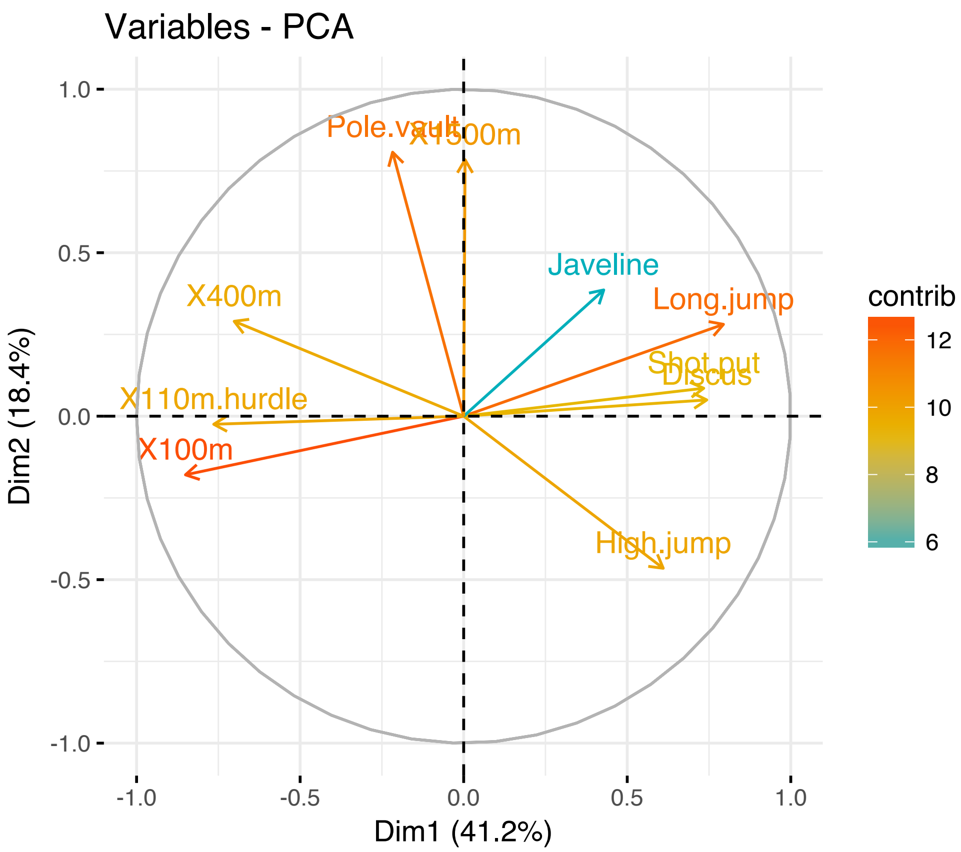
- C1 and C2 are the contributions of the variable on PC1 and PC2, respectively
- Eig1 and Eig2 are the eigenvalues of PC1 and PC2, respectively. Recall that eigenvalues measure the amount of variation retained by each PC.

In this case, the expected average contribution (cutoff) is calculated as follow: As mentioned above, if the contributions of the 10 variables were uniform, the expected average contribution on a given PC would be 1/10 = 10%. The expected average contribution of a variable for PC1 and PC2 is : [(10* Eig1) + (10 * Eig2)]/(Eig1 + Eig2)

It can be seen that the variables - X100m, Long.jump and Pole.vault - contribute the most to the dimensions 1 and 2.

The most important (or, contributing) variables can be highlighted on the correlation plot as follow:

```
fviz_pca_var(res.pca, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
             )
```



Note that, it's also possible to change the transparency of variables according to their contrib values using the option *alpha.var = "contrib"*. For example, type this:

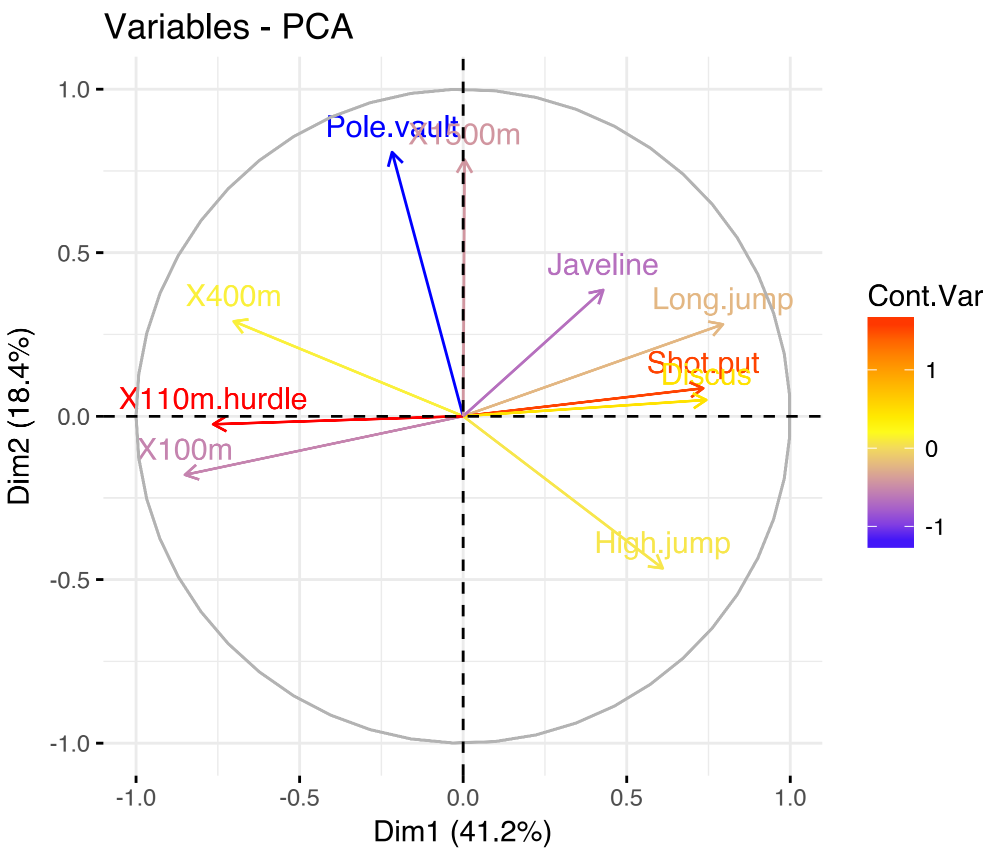
```
# Change the transparency by contrib values
fviz_pca_var(res.pca, alpha.var = "contrib")
```

3.4.2.5 Color by a custom continuous variable

In the previous sections, we showed how to color variables by their contributions and their cos2. Note that, it's possible to color variables by any custom continuous variable. The coloring variable should have the same length as the number of active variables in the PCA (here n = 10).

For example, type this:

```
# Create a random continuous variable of length 10
set.seed(123)
my.cont.var <- rnorm(10)

# Color variables by the continuous variable
fviz_pca_var(res.pca, col.var = my.cont.var,
             gradient.cols = c("blue", "yellow", "red"),
             legend.title = "Cont.Var")
```



3.4.2.6 Color by groups

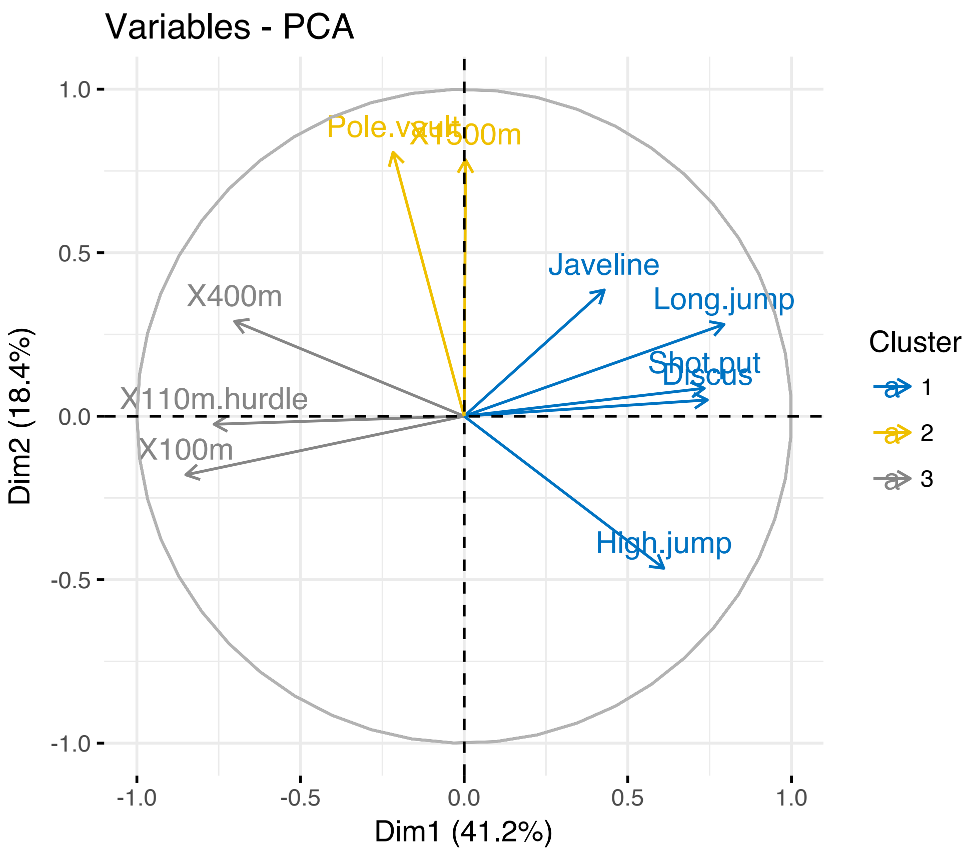
It's also possible to change the color of variables by groups defined by a qualitative/categorical variable, also called *factor* in R terminology.

As we don't have any grouping variable in our data sets for classifying variables, we'll create it.

In the following demo example, we start by classifying the variables into 3 groups using the **kmeans** clustering algorithm. Next, we use the clusters returned by the kmeans algorithm to color variables.

> Note that, if you are interested in learning clustering, we previously published a book named "Practical Guide To Cluster Analysis in R" (`https://goo.gl/DmJ5y5`).

```
# Create a grouping variable using kmeans
# Create 3 groups of variables (centers = 3)
set.seed(123)
res.km <- kmeans(var$coord, centers = 3, nstart = 25)
grp <- as.factor(res.km$cluster)

# Color variables by groups
fviz_pca_var(res.pca, col.var = grp,
             palette = c("#0073C2FF", "#EFC000FF", "#868686FF"),
             legend.title = "Cluster")
```



> Note that, to change the color of groups the argument **palette** should be used. To change gradient colors, the argument **gradient.cols** should be used.

3.4.3 Dimension description

In the section 3.4.2.4, we described how to highlight variables according to their contributions to the principal components.

Note also that, the function *dimdesc*() [in FactoMineR], for dimension description, can be used to identify the most significantly associated variables with a given principal component . It can be used as follow:

```
res.desc <- dimdesc(res.pca, axes = c(1,2), proba = 0.05)
# Description of dimension 1
res.desc$Dim.1
```

```
## $quanti
##              correlation  p.value
## Long.jump          0.794 6.06e-06
## Discus             0.743 4.84e-05
## Shot.put           0.734 6.72e-05
## High.jump          0.610 1.99e-03
## Javeline           0.428 4.15e-02
## X400m             -0.702 1.91e-04
## X110m.hurdle      -0.764 2.20e-05
## X100m             -0.851 2.73e-07
```

```
# Description of dimension 2
res.desc$Dim.2
```

```
## $quanti
##              correlation  p.value
## Pole.vault         0.807 3.21e-06
## X1500m             0.784 9.38e-06
## High.jump         -0.465 2.53e-02
```

In the output above, *$quanti* means results for quantitative variables. Note that, variables are sorted by the p-value of the correlation.

3.4.4 Graph of individuals

3.4.4.1 Results

The results, for individuals can be extracted using the function *get_pca_ind*() [*factoextra* package]. Similarly to the *get_pca_var*(), the function *get_pca_ind*() provides a list of matrices containing all the results for the individuals (coordinates, correlation between variables and axes, squared cosine and contributions)

```
ind <- get_pca_ind(res.pca)
ind
```

```
## Principal Component Analysis Results for individuals
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the individuals"
## 2 "$cos2"    "Cos2 for the individuals"
## 3 "$contrib" "contributions of the individuals"
```

To get access to the different components, use this:

```
# Coordinates of individuals
head(ind$coord)

# Quality of individuals
head(ind$cos2)

# Contributions of individuals
head(ind$contrib)
```

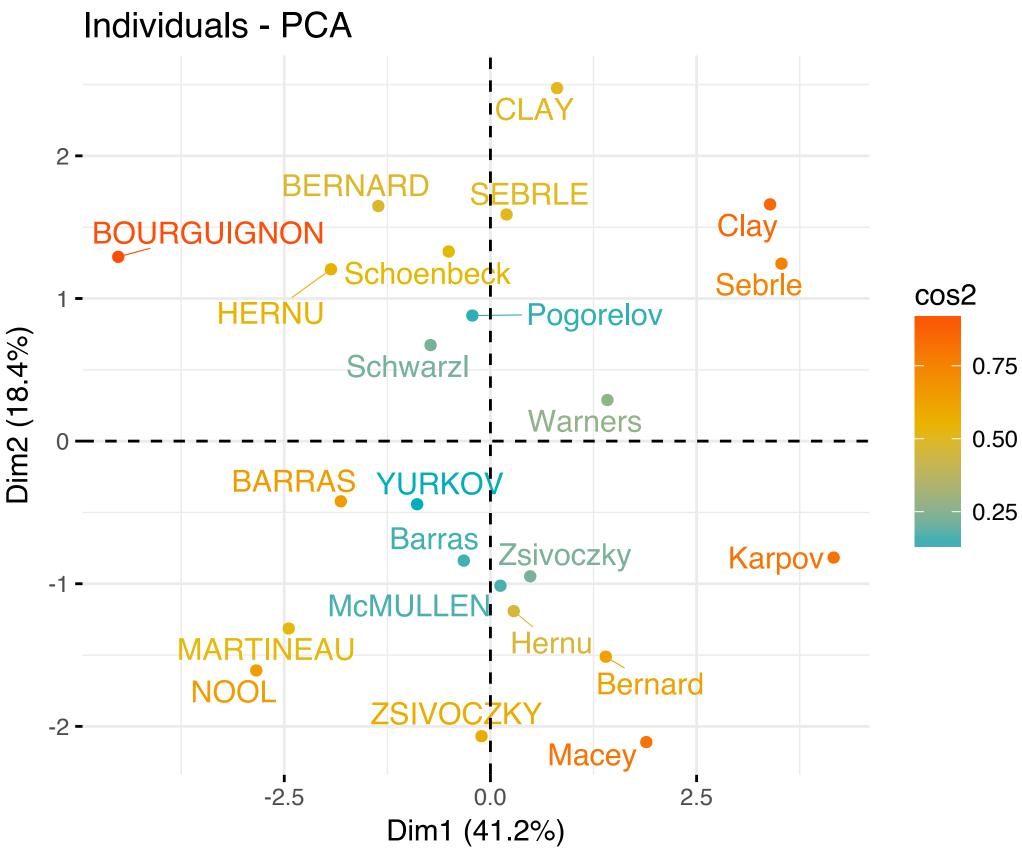
3.4.4.2 Plots: quality and contribution

The *fviz_pca_ind*() is used to produce the graph of individuals. To create a simple plot, type this:

```
fviz_pca_ind(res.pca)
```

Like variables, it's also possible to color individuals by their cos2 values:

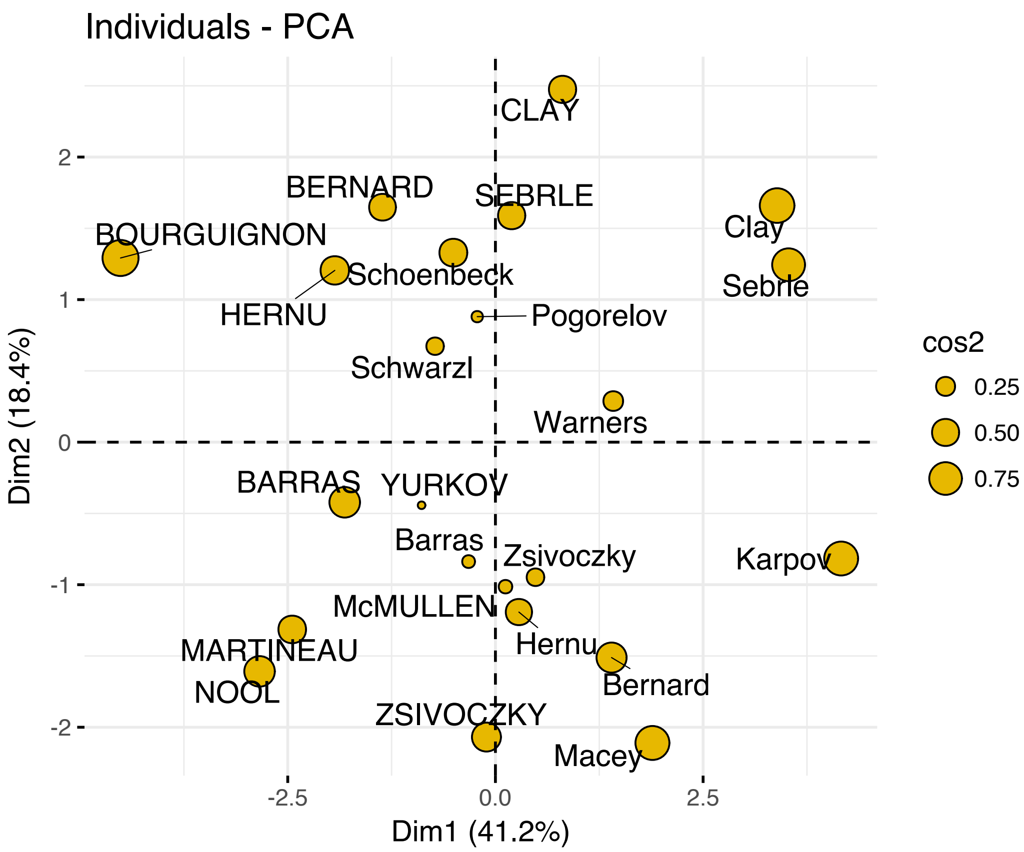
```
fviz_pca_ind(res.pca, col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping (slow if many points)
             )
```



> Note that, individuals that are similar are grouped together on the plot.

You can also change the point size according the cos2 of the corresponding individuals:

```
fviz_pca_ind(res.pca, pointsize = "cos2",
             pointshape = 21, fill = "#E7B800",
             repel = TRUE # Avoid text overlapping (slow if many points)
             )
```



To change both point size and color by cos2, try this:

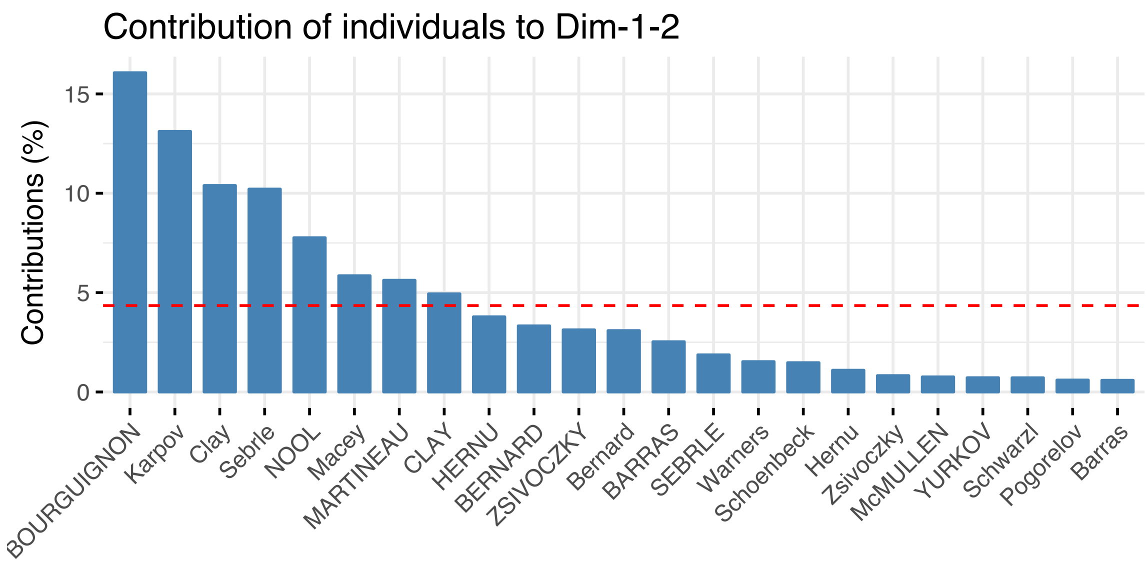
```
fviz_pca_ind(res.pca, col.ind = "cos2", pointsize = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping (slow if many points)
             )
```

To create a bar plot of the quality of representation (cos2) of individuals on the factor map, you can use the function *fviz_cos2*() as previously described for variables:

```
fviz_cos2(res.pca, choice = "ind")
```

To visualize the contribution of individuals to the first two principal components, type this:

```
# Total contribution on PC1 and PC2
fviz_contrib(res.pca, choice = "ind", axes = 1:2)
```



3.4.4.3 Color by a custom continuous variable

As for variables, individuals can be colored by any custom continuous variable by specifying the argument *col.ind*.

For example, type this:

```
# Create a random continuous variable of length 23,
# Same length as the number of active individuals in the PCA
set.seed(123)
my.cont.var <- rnorm(23)

# Color variables by the continuous variable
fviz_pca_ind(res.pca, col.ind = my.cont.var,
             gradient.cols = c("blue", "yellow", "red"),
             legend.title = "Cont.Var")
```

3.4.4.4 Color by groups

Here, we describe how to color individuals by group. Additionally, we show how to add *concentration ellipses* and *confidence ellipses* by groups. For this, we'll use the iris data as demo data sets.

Iris data sets look like this:

```
head(iris, 3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

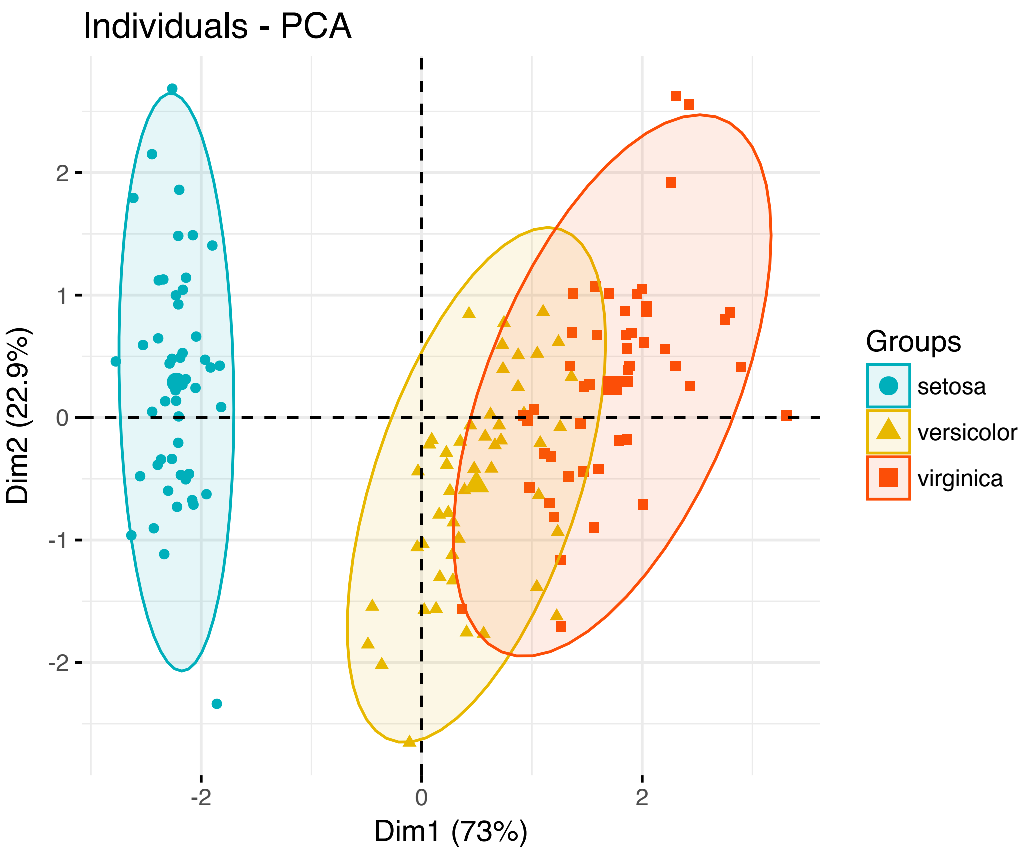
The column "Species" will be used as grouping variable. We start by computing principal component analysis as follow:

```
# The variable Species (index = 5) is removed
# before PCA analysis
iris.pca <- PCA(iris[,-5], graph = FALSE)
```

In the R code below: the argument *habillage* or *col.ind* can be used to specify the factor variable for coloring the individuals by groups.

To add a concentration ellipse around each group, specify the argument *addEllipses = TRUE*. The argument *palette* can be used to change group colors.

```
fviz_pca_ind(iris.pca,
             geom.ind = "point", # show points only (nbut not "text")
             col.ind = iris$Species, # color by groups
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE, # Concentration ellipses
             legend.title = "Groups"
             )
```



To remove the group mean point, specify the argument *mean.point = FALSE*.

If you want confidence ellipses instead of concentration ellipses, use *ellipse.type = "confidence"*.

```
# Add confidence ellipses
fviz_pca_ind(iris.pca, geom.ind = "point", col.ind = iris$Species,
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE, ellipse.type = "confidence",
             legend.title = "Groups"
             )
```

Note that, allowed values for palette include:

- "grey" for grey color palettes;
- brewer palettes e.g. "RdBu", "Blues", …; To view all, type this in R: *RColorBrewer::display.brewer.all()*.
- custom color palette e.g. c("blue", "red");
- and scientific journal palettes from *ggsci R package*, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".

For example, to use the jco (journal of clinical oncology) color palette, type this:

```
fviz_pca_ind(iris.pca,
             label = "none", # hide individual labels
             habillage = iris$Species, # color by groups
             addEllipses = TRUE, # Concentration ellipses
             palette = "jco"
             )
```

3.4.5 Graph customization

Note that, *fviz_pca_ind*() and *fviz_pca_var*() and related functions are wrapper around the core function *fviz*() [in *factoextra*]. *fviz*() is a wrapper around the function *ggscatter*() [in *ggpubr*]. Therefore, further arguments, to be passed to the function *fviz*() and *ggscatter*(), can be specified in *fviz_pca_ind*() and *fviz_pca_var*().

Here, we present some of these additional arguments to customize the PCA graph of variables and individuals.

3.4.5.1 Dimensions

By default, variables/individuals are represented on dimensions 1 and 2. If you want to visualize them on dimensions 2 and 3, for example, you should specify the argument *axes = c(2, 3)*.

```
# Variables on dimensions 2 and 3
fviz_pca_var(res.pca, axes = c(2, 3))

# Individuals on dimensions 2 and 3
fviz_pca_ind(res.pca, axes = c(2, 3))
```

3.4.5.2 Plot elements: point, text, arrow

The argument *geom* (for geometry) and derivatives are used to specify the geometry elements or graphical elements to be used for plotting.

1) **geom.var**: a text specifying the geometry to be used for plotting variables. Allowed values are the combination of c("point", "arrow", "text").

- Use *geom.var = "point"*, to show only points;
- Use *geom.var = "text"* to show only text labels;
- Use *geom.var = c("point", "text")* to show both points and text labels
- Use *geom.var = c("arrow", "text")* to show arrows and labels (default).

For example, type this:

```
# Show variable points and text labels
fviz_pca_var(res.pca, geom.var = c("point", "text"))
```

2) **geom.ind**: a text specifying the geometry to be used for plotting individuals. Allowed values are the combination of c("point", "text").

- Use *geom.ind = "point"*, to show only points;
- Use *geom.ind = "text"* to show only text labels;
- Use *geom.ind = c("point", "text")* to show both point and text labels (default)

For example, type this:

```
# Show individuals text labels only
fviz_pca_ind(res.pca, geom.ind =  "text")
```

3.4.5.3 Size and shape of plot elements

1. **labelsize**: font size for the text labels, e.g.: *labelsize = 4*.
2. **pointsize**: the size of points, e.g.: *pointsize = 1.5*.
3. **arrowsize**: the size of arrows. Controls the thickness of arrows, e.g.: *arrowsize = 0.5*.
4. **pointshape**: the shape of points, *pointshape = 21*. Type *ggpubr::show_point_shapes()* to see available point shapes.

```
# Change the size of arrows an labels
fviz_pca_var(res.pca, arrowsize = 1, labelsize = 5,
             repel = TRUE)

# Change points size, shape and fill color
# Change labelsize
fviz_pca_ind(res.pca,
             pointsize = 3, pointshape = 21, fill = "lightblue",
             labelsize = 5, repel = TRUE)
```

3.4.5.4 Ellipses

As we described in the previous section 3.4.4.4, when coloring individuals by groups, you can add point concentration ellipses using the argument *addEllipses = TRUE*.

Note that, the argument *ellipse.type* can be used to change the type of ellipses. Possible values are:

- *"convex"*: plot convex hull of a set o points.
- *"confidence"*: plot confidence ellipses around group mean points as the function *coord.ellipse*()[in FactoMineR].
- *"t"*: assumes a multivariate t-distribution.
- *"norm"*: assumes a multivariate normal distribution.
- *"euclid"*: draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless *coord_fixed*() is applied.

The argument *ellipse.level* is also available to change the size of the concentration ellipse in normal probability. For example, specify ellipse.level = 0.95 or ellipse.level = 0.66.

```
# Add confidence ellipses
fviz_pca_ind(iris.pca, geom.ind = "point",
             col.ind = iris$Species, # color by groups
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE, ellipse.type = "confidence",
             legend.title = "Groups"
             )

# Convex hull
fviz_pca_ind(iris.pca, geom.ind = "point",
             col.ind = iris$Species, # color by groups
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE, ellipse.type = "convex",
             legend.title = "Groups"
             )
```

3.4.5.5 Group mean points

When coloring individuals by groups (section 3.4.4.4), the mean points of groups (barycenters) are also displayed by default.

To remove the mean points, use the argument *mean.point = FALSE*.

```
fviz_pca_ind(iris.pca,
             geom.ind = "point", # show points only (but not "text")
             group.ind = iris$Species, # color by groups
             legend.title = "Groups",
             mean.point = FALSE)
```

3.4.5.6 Axis lines

The argument **axes.linetype** can be used to specify the line type of axes. Default is "dashed". Allowed values include "blank", "solid", "dotted", etc. To see all possible values type *ggpubr::show_line_types()* in R.

To remove axis lines, use axes.linetype = "blank":

```
fviz_pca_var(res.pca, axes.linetype = "blank")
```

3.4.5.7 Graphical parameters

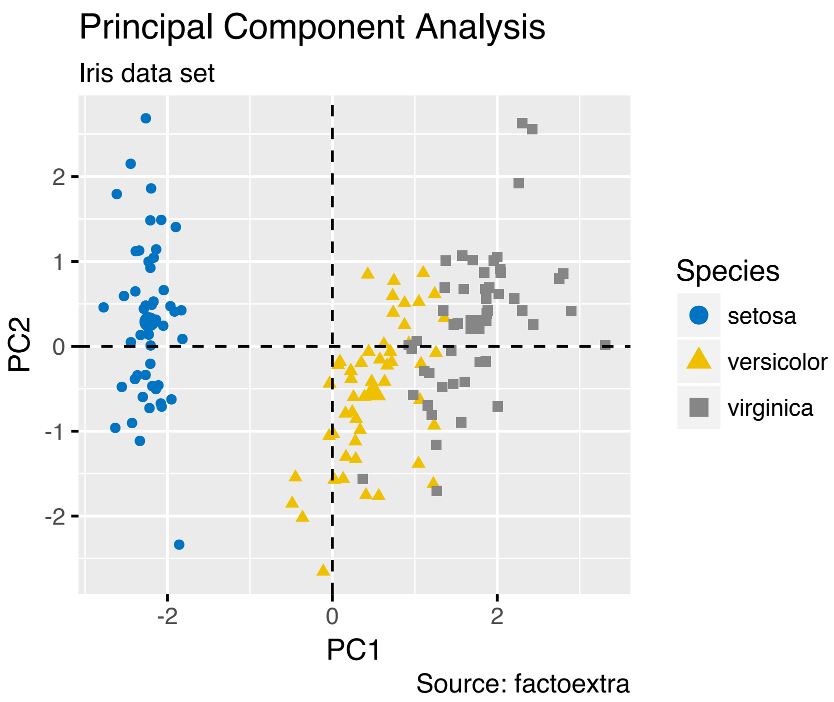
To change easily the graphical of any ggplots, you can use the function ggpar()[1] [ggpubr package]

The graphical parameters that can be changed using ggpar() include:

- Main titles, axis labels and legend titles
- Legend position. Possible values: "top", "bottom", "left", "right", "none".
- Color palette.
- Themes. Allowed values include: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void().

```
ind.p <- fviz_pca_ind(iris.pca, geom = "point", col.ind = iris$Species)
ggpubr::ggpar(ind.p,
              title = "Principal Component Analysis",
              subtitle = "Iris data set",
              caption = "Source: factoextra",
              xlab = "PC1", ylab = "PC2",
              legend.title = "Species", legend.position = "top",
              ggtheme = theme_gray(), palette = "jco"
              )
```

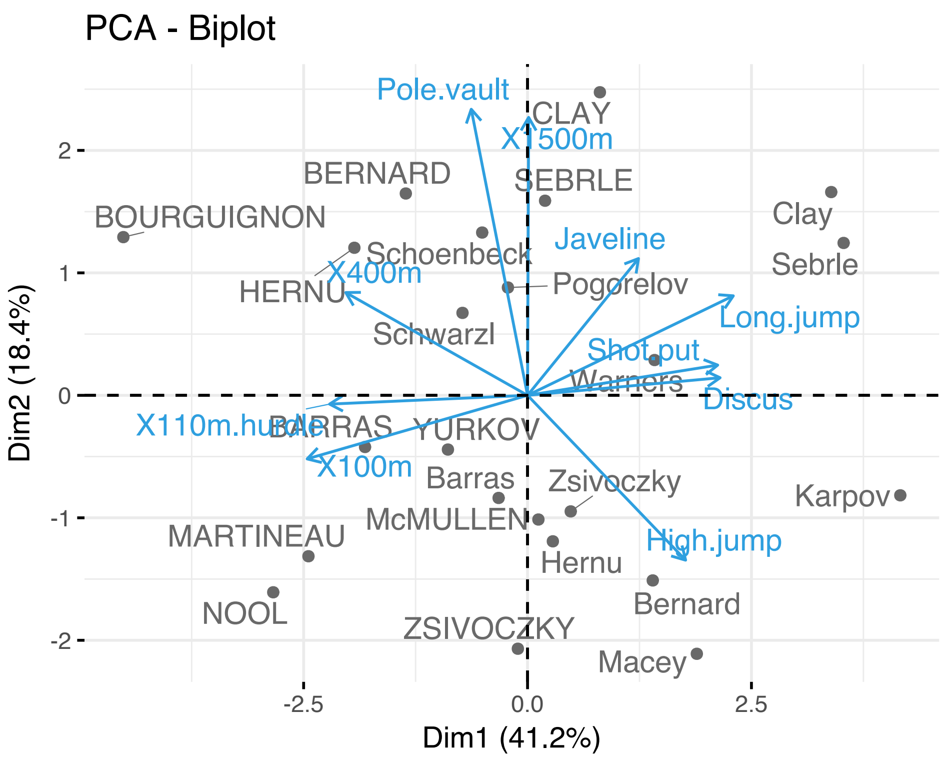
[1] http://www.sthda.com/english/rpkgs/ggpubr/reference/ggpar.html



3.4.6 Biplot

To make a simple biplot of individuals and variables, type this:

```
fviz_pca_biplot(res.pca, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969"  # Individuals color
                )
```



Note that, the biplot might be only useful when there is a low number of variables and individuals in the data set; otherwise the final plot would be unreadable.

Note also that, the coordinate of individuals and variables are note constructed on the same space. Therefore, on biplot, you should mainly focus on the direction of variables but not on their absolute positions on the plot.

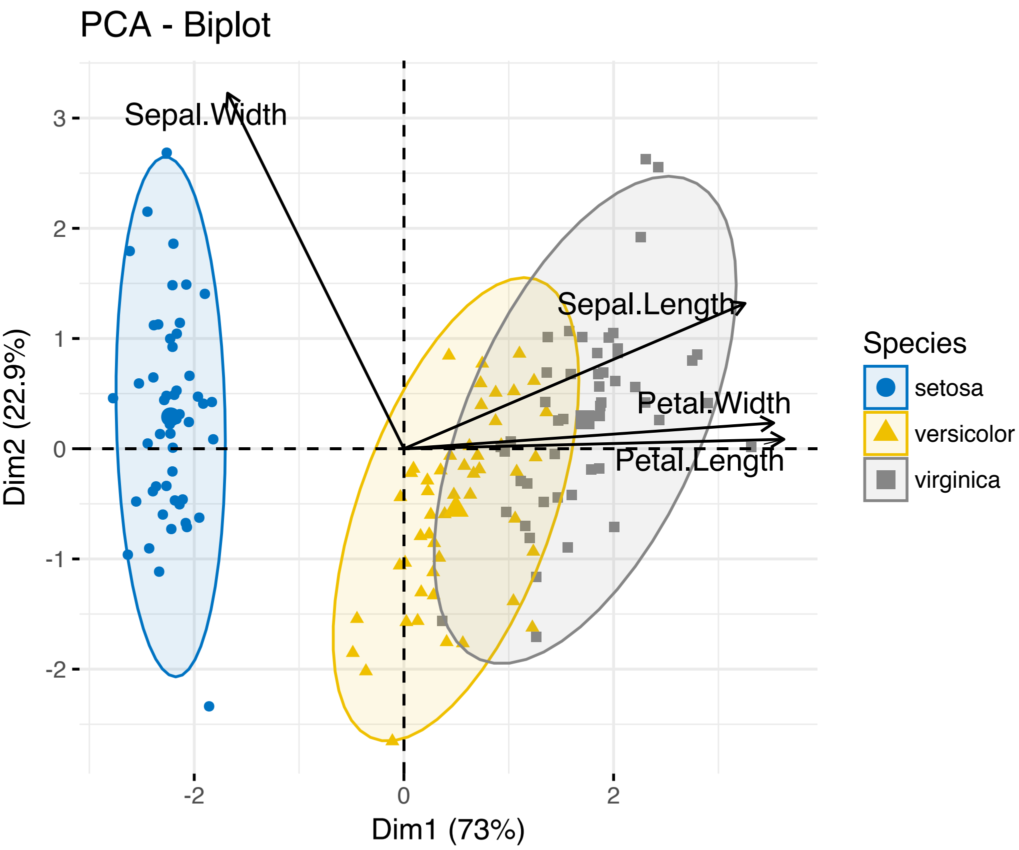
Roughly speaking a biplot can be interpreted as follow:

- an individual that is on the same side of a given variable has a high value for this variable;
- an individual that is on the opposite side of a given variable has a low value for this variable.

Now, using the *iris.pca* output, let's :

- make a biplot of individuals and variables
- change the color of individuals by groups: col.ind = iris$Species
- show only the labels for variables: *label* = *"var"* or use *geom.ind* = *"point"*

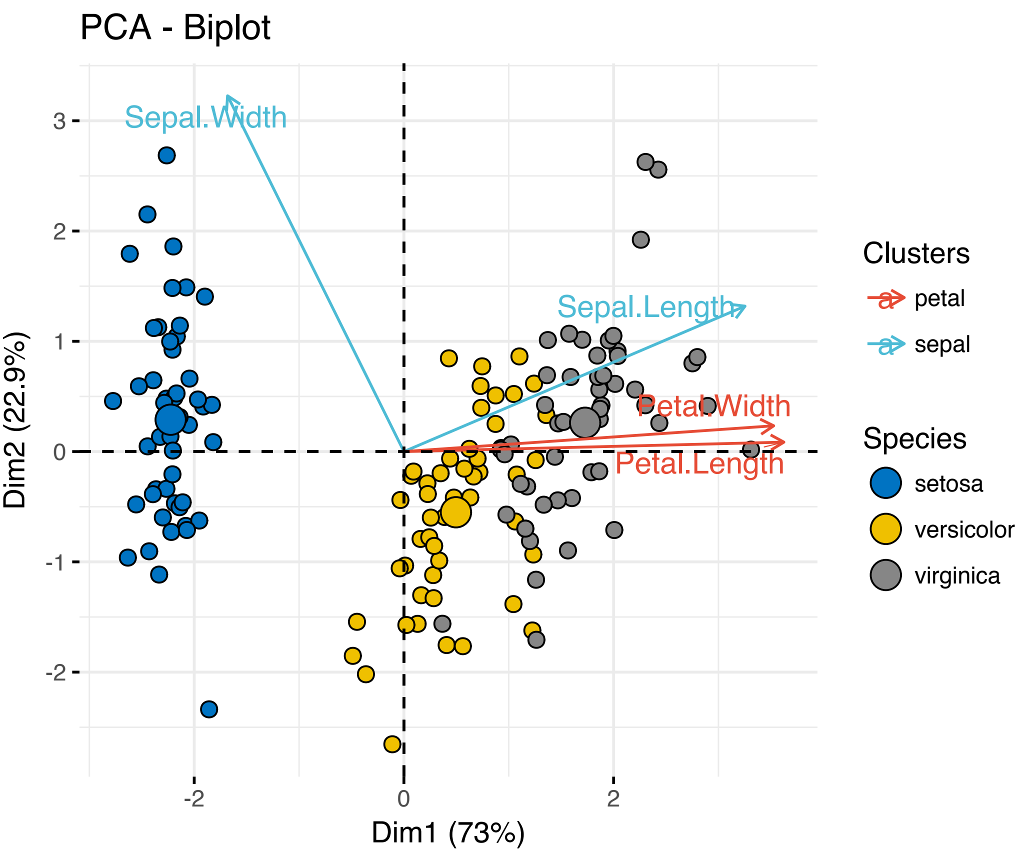
```
fviz_pca_biplot(iris.pca,
                col.ind = iris$Species, palette = "jco",
                addEllipses = TRUE, label = "var",
                col.var = "black", repel = TRUE,
                legend.title = "Species")
```



In the following example, we want to color both individuals and variables by groups. The trick is to use pointshape = 21 for individual points. This particular point shape can be filled by a color using the argument *fill.ind*. The border line color of individual points is set to "black" using *col.ind*. To color variable by groups, the argument *col.var* will be used.

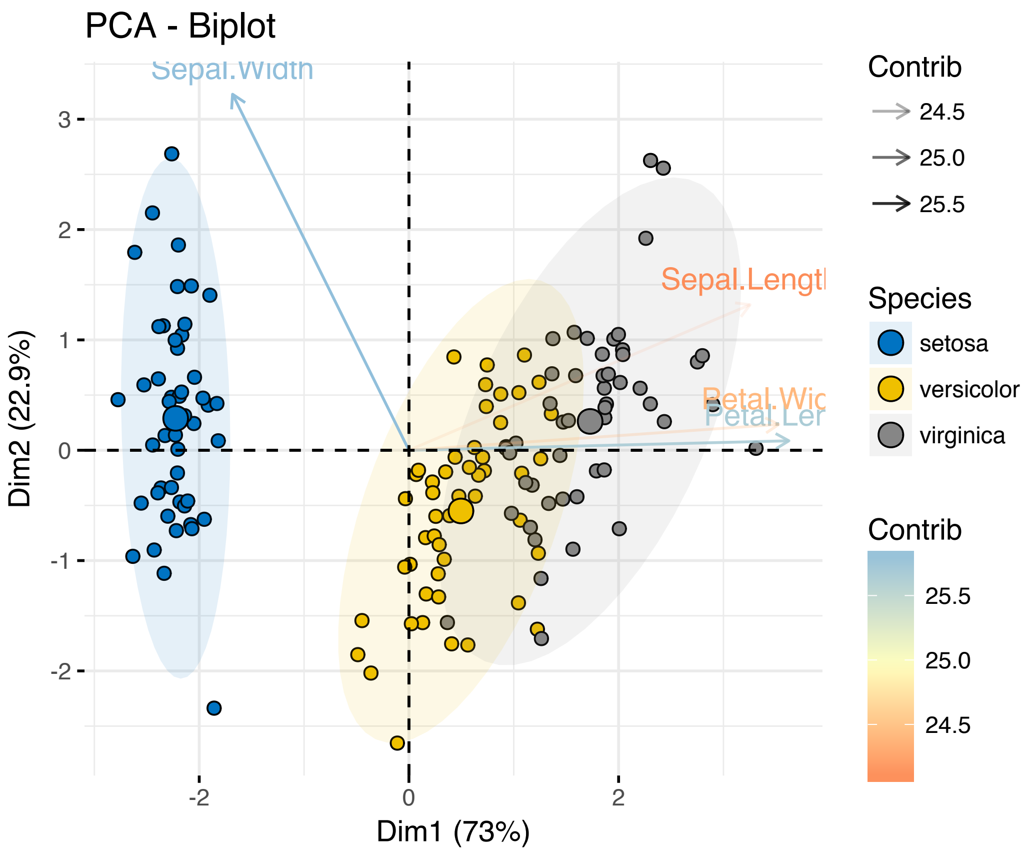
To customize individuals and variable colors, we use the helper functions *fill_palette*() and *color_palette*() [in ggpubr package].

```
fviz_pca_biplot(iris.pca,
                # Fill individuals by groups
                geom.ind = "point",
                pointshape = 21,
                pointsize = 2.5,
                fill.ind = iris$Species,
                col.ind = "black",
                # Color variable by groups
                col.var = factor(c("sepal", "sepal", "petal", "petal")),

                legend.title = list(fill = "Species", color = "Clusters"),
                repel = TRUE        # Avoid label overplotting
             )+
  ggpubr::fill_palette("jco")+      # Indiviual fill color
  ggpubr::color_palette("npg")      # Variable colors
```



Another complex example is to color individuals by groups (discrete color) and variables by their contributions to the principal components (gradient colors). Additionally, we'll change the transparency of variables by their contributions using the argument *alpha.var*.

```
fviz_pca_biplot(iris.pca,
                # Individuals
                geom.ind = "point",
                fill.ind = iris$Species, col.ind = "black",
                pointshape = 21, pointsize = 2,
                palette = "jco",
                addEllipses = TRUE,
                # Variables
                alpha.var ="contrib", col.var = "contrib",
                gradient.cols = "RdYlBu",

                legend.title = list(fill = "Species", color = "Contrib",
                                    alpha = "Contrib")
                )
```



3.5 Supplementary elements

3.5.1 Definition and types

As described above (section 3.3.2), the *decathlon2* data sets contain **supplementary continuous variables** (quanti.sup, columns 11:12), **supplementary qualitative variables** (quali.sup, column 13) and **supplementary individuals** (ind.sup, rows 24:27).

Supplementary variables and individuals are not used for the determination of the principal components. Their coordinates are predicted using only the information provided by the performed principal component analysis on active variables/individuals.

3.5.2 Specification in PCA

To specify supplementary individuals and variables, the function *PCA*() can be used as follow:

```
PCA(X, ind.sup = NULL,
    quanti.sup = NULL, quali.sup = NULL, graph = TRUE)
```

- **X** : a data frame. Rows are individuals and columns are numeric variables.
- **ind.sup** : a numeric vector specifying the indexes of the supplementary individuals
- **quanti.sup**, **quali.sup** : a numeric vector specifying, respectively, the indexes of the quantitative and qualitative variables
- **graph** : a logical value. If TRUE a graph is displayed.

For example, type this:

```
res.pca <- PCA(decathlon2, ind.sup = 24:27,
               quanti.sup = 11:12, quali.sup = 13, graph=FALSE)
```

3.5.3 Quantitative variables

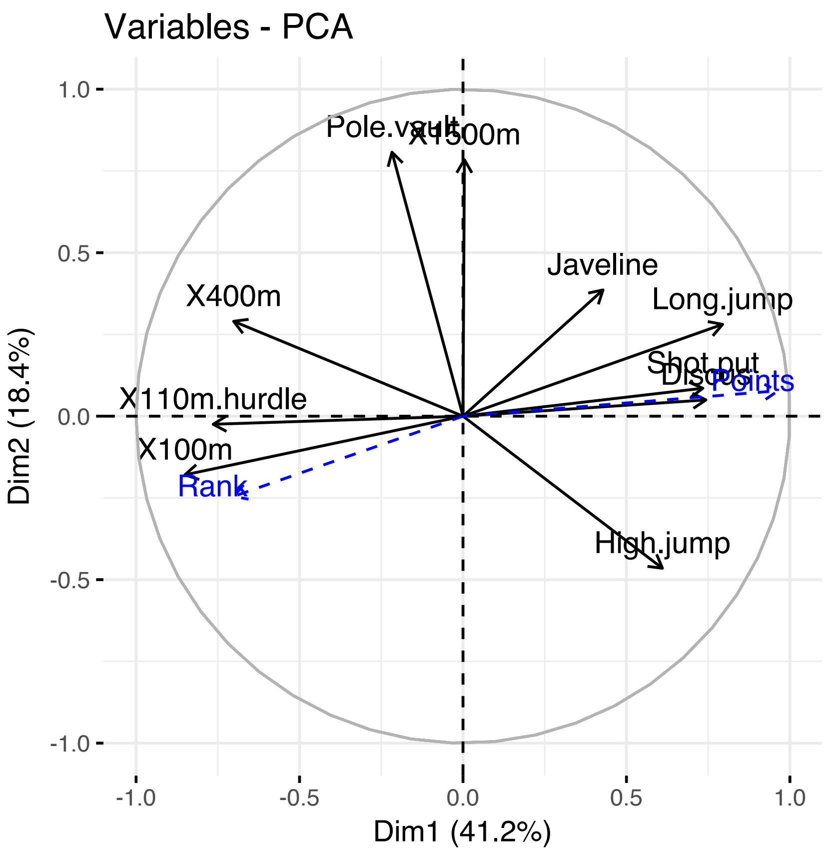
- Predicted results (coordinates, correlation and cos2) for the supplementary quantitative variables:

```
res.pca$quanti.sup
```

```
## $coord
##         Dim.1   Dim.2  Dim.3   Dim.4   Dim.5
## Rank   -0.701 -0.2452 -0.183  0.0558 -0.0738
## Points  0.964  0.0777  0.158 -0.1662 -0.0311
##
## $cor
##         Dim.1   Dim.2  Dim.3   Dim.4   Dim.5
## Rank   -0.701 -0.2452 -0.183  0.0558 -0.0738
## Points  0.964  0.0777  0.158 -0.1662 -0.0311
##
## $cos2
##        Dim.1   Dim.2  Dim.3   Dim.4   Dim.5
## Rank   0.492 0.06012 0.0336 0.00311 0.00545
## Points 0.929 0.00603 0.0250 0.02763 0.00097
```

- Visualize all variables (active and supplementary ones):

```
fviz_pca_var(res.pca)
```



Note that, by default, supplementary quantitative variables are shown in blue color and dashed lines.

Further arguments to customize the plot:

```
# Change color of variables
fviz_pca_var(res.pca,
             col.var = "black",      # Active variables
             col.quanti.sup = "red" # Suppl. quantitative variables
             )

# Hide active variables on the plot,
# show only supplementary variables
fviz_pca_var(res.pca, invisible = "var")

# Hide supplementary variables
fviz_pca_var(res.pca, invisible = "quanti.sup")
```

Using the *fviz_pca_var*(), the quantitative supplementary variables are displayed automatically on the correlation circle plot. Note that, you can add the *quanti.sup* variables manually, using the *fviz_add*() function, for further customization. An example is shown below.

```
# Plot of active variables
p <- fviz_pca_var(res.pca, invisible = "quanti.sup")
```

```
# Add supplementary active variables
fviz_add(p, res.pca$quanti.sup$coord,
         geom = c("arrow", "text"),
         color = "red")
```

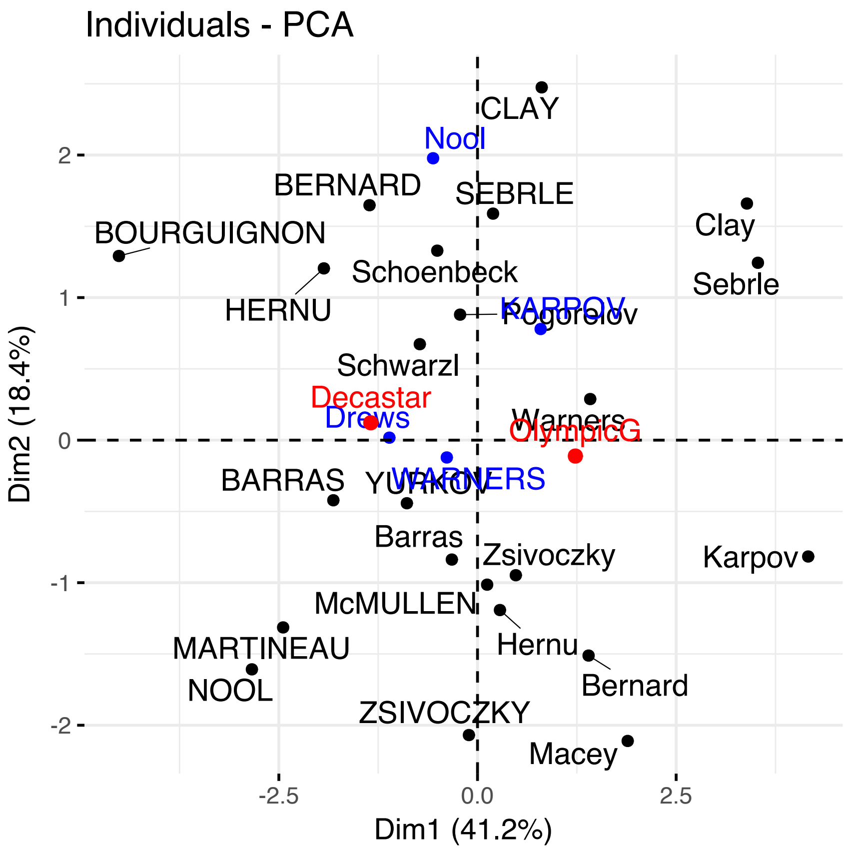
3.5.4 Individuals

- Predicted results for the supplementary individuals (**ind.sup**):

```
res.pca$ind.sup
```

- Visualize all individuals (active and supplementary ones). On the graph, you can add also the supplementary qualitative variables (**quali.sup**), which coordinates is accessible using *res.pca$quali.supp$coord*.

```
p <- fviz_pca_ind(res.pca, col.ind.sup = "blue", repel = TRUE)
p <- fviz_add(p, res.pca$quali.sup$coord, color = "red")
p
```



Supplementary individuals are shown in blue. The levels of the supplementary qualitative variable are shown in red color.

3.5.5 Qualitative variables

In the previous section, we showed that you can add the supplementary qualitative variables on individuals plot using *fviz_add*().

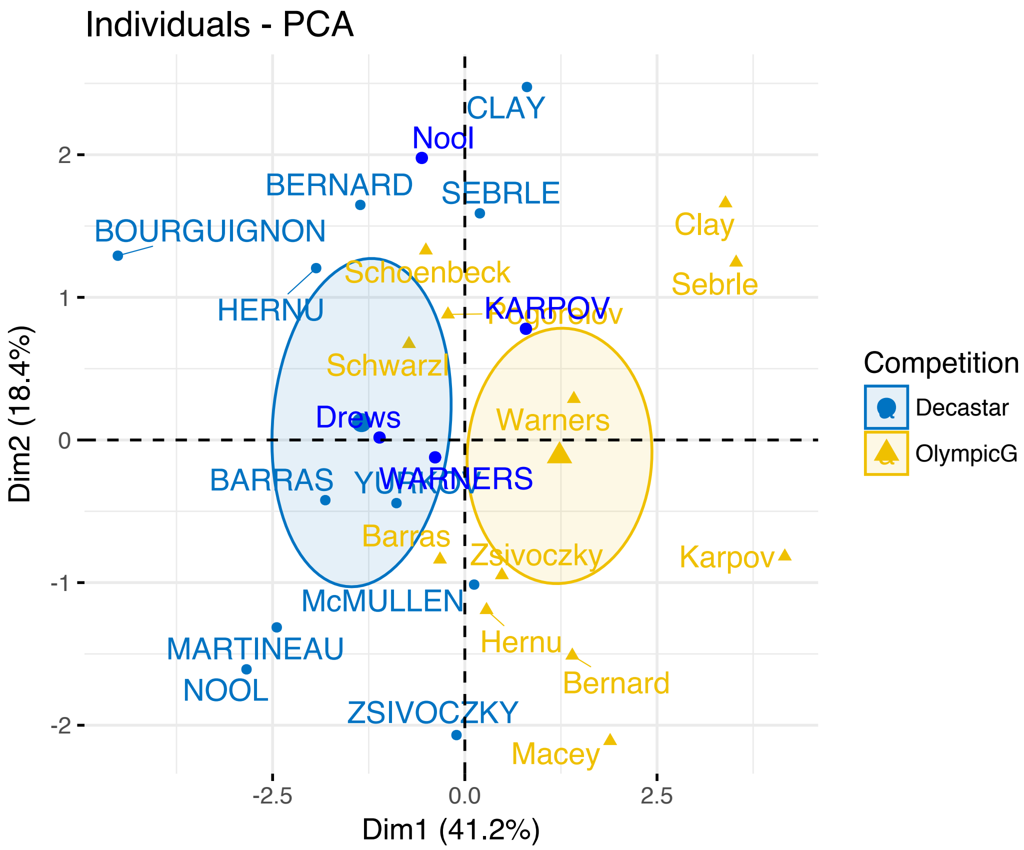
Note that, the supplementary qualitative variables can be also used for coloring individuals by groups. This can help to interpret the data. The data sets *decathlon2* contain a *supplementary qualitative variable* at columns 13 corresponding to the type of competitions.

The results concerning the supplementary qualitative variable are:

```
res.pca$quali
```

To color individuals by a supplementary qualitative variable, the argument *habillage* is used to specify the index of the supplementary qualitative variable. Historically, this argument name comes from the FactoMineR package. It's a french word meaning "dressing" in english. To keep consistency between FactoMineR and factoextra, we decided to keep the same argument name

```
fviz_pca_ind(res.pca, habillage = 13,
             addEllipses =TRUE, ellipse.type = "confidence",
             palette = "jco", repel = TRUE)
```



Recall that, to remove the mean points of groups, specify the argument *mean.point = FALSE*.

3.6 Filtering results

If you have many individuals/variable, it's possible to visualize only some of them using the arguments *select.ind* and *select.var*.

select.ind, select.var: a selection of individuals/variable to be plotted. Allowed values are *NULL* or a *list* containing the arguments name, cos2 or contrib:

- *name*: is a character vector containing individuals/variable names to be plotted
- *cos2*: if cos2 is in [0, 1], ex: 0.6, then individuals/variables with a cos2 > 0.6 are plotted
- *if cos2 > 1*, ex: 5, then the top 5 active individuals/variables and top 5 supplementary columns/rows with the highest cos2 are plotted
- *contrib*: if contrib > 1, ex: 5, then the top 5 individuals/variables with the highest contributions are plotted

```
# Visualize variable with cos2 >= 0.6
fviz_pca_var(res.pca, select.var = list(cos2 = 0.6))

# Top 5 active variables with the highest cos2
fviz_pca_var(res.pca, select.var= list(cos2 = 5))

# Select by names
name <- list(name = c("Long.jump", "High.jump", "X100m"))
fviz_pca_var(res.pca, select.var = name)

# top 5 contributing individuals and variable
fviz_pca_biplot(res.pca, select.ind = list(contrib = 5),
               select.var = list(contrib = 5),
               ggtheme = theme_minimal())
```

When the selection is done according to the contribution values, supplementary individuals/variables are not shown because they don't contribute to the construction of the axes.

3.7 Exporting results

3.7.1 Export plots to PDF/PNG files

The **factoextra** package produces a ggplot2-based graphs. To save any ggplots, the standard R code is as follow:

```
# Print the plot to a pdf file
pdf("myplot.pdf")
print(myplot)
dev.off()
```

In the following examples, we'll show you how to save the different graphs into pdf or png files.

The first step is to create the plots you want as an R object:

```
# Scree plot
scree.plot <- fviz_eig(res.pca)

# Plot of individuals
ind.plot <- fviz_pca_ind(res.pca)

# Plot of variables
var.plot <- fviz_pca_var(res.pca)
```

Next, the plots can be exported into a single pdf file as follow:

```
pdf("PCA.pdf") # Create a new pdf device

print(scree.plot)
print(ind.plot)
print(var.plot)

dev.off() # Close the pdf device
```

> Note that, using the above R code will create the PDF file into your current working directory. To see the path of your current working directory, type *getwd()* in the R console.

To print each plot to specific png file, the R code looks like this:

```
# Print scree plot to a png file
png("pca-scree-plot.png")
print(scree.plot)
dev.off()

# Print individuals plot to a png file
png("pca-variables.png")
print(var.plot)
dev.off()

# Print variables plot to a png file
png("pca-individuals.png")
print(ind.plot)
dev.off()
```

Another alternative, to export ggplots, is to use the function **ggexport**() [in ggpubr package]. We like **ggexport**(), because it's very simple. With one line R code, it allows us to export individual plots to a file (pdf, eps or png) (one plot per page). It can also arrange the plots (2 plot per page, for example) before exporting them. The examples below demonstrates how to export ggplots using **ggexport**().

Export individual plots to a pdf file (one plot per page):

```r
library(ggpubr)
ggexport(plotlist = list(scree.plot, ind.plot, var.plot),
         filename = "PCA.pdf")
```

Arrange and export. Specify *nrow* and *ncol* to display multiple plots on the same page:

```r
ggexport(plotlist = list(scree.plot, ind.plot, var.plot),
         nrow = 2, ncol = 2,
         filename = "PCA.pdf")
```

Export plots to png files. If you specify a list of plots, then multiple png files will be automatically created to hold each plot.

```r
ggexport(plotlist = list(scree.plot, ind.plot, var.plot),
         filename = "PCA.png")
```

3.7.2 Export results to txt/csv files

All the outputs of the PCA (individuals/variables coordinates, contributions, etc) can be exported at once, into a TXT/CSV file, using the function **write.infile**() [in *FactoMineR*] package:

```r
# Export into a TXT file
write.infile(res.pca, "pca.txt", sep = "\t")

# Export into a CSV file
write.infile(res.pca, "pca.csv", sep = ";")
```

3.8 Summary

In conclusion, we described how to perform and interpret principal component analysis (PCA). We computed PCA using the **PCA**() function [FactoMineR]. Next, we used the **factoextra** R package to produce ggplot2-based visualization of the PCA results.

There are other functions [packages] to compute PCA in R:

1) Using **prcomp**() [stats]

```r
res.pca <- prcomp(iris[, -5], scale. = TRUE)
```

Read more: `http://www.sthda.com/english/wiki/pca-using-prcomp-and-princomp`

2) Using **princomp**() [stats]

```r
res.pca <- princomp(iris[, -5], cor = TRUE)
```

Read more: `http://www.sthda.com/english/wiki/pca-using-prcomp-and-princomp`

3) Using **dudi.pca**() [ade4]

```
library("ade4")
res.pca <- dudi.pca(iris[, -5], scannf = FALSE, nf = 5)
```

Read more: http://www.sthda.com/english/wiki/pca-using-ade4-and-factoextra

4) Using **epPCA**() [ExPosition]

```
library("ExPosition")
res.pca <- epPCA(iris[, -5], graph = FALSE)
```

No matter what functions you decide to use, in the list above, the factoextra package can handle the output for creating beautiful plots similar to what we described in the previous sections for FactoMineR:

```
fviz_eig(res.pca)     # Scree plot

fviz_pca_ind(res.pca) # Graph of individuals

fviz_pca_var(res.pca) # Graph of variables
```

3.9 Further reading

For the mathematical background behind CA, refer to the following video courses, articles and books:

- Principal component analysis (article) (Abdi and Williams, 2010). https://goo.gl/1Vtwq1.
- Principal Component Analysis Course Using FactoMineR (Video courses). https://goo.gl/VZJsnM
- Exploratory Multivariate Analysis by Example Using R (book) (Husson et al., 2017b).
- Principal Component Analysis (book) (Jollife, 2002).

See also:

- PCA using **prcomp**() and **princomp**() (tutorial). http://www.sthda.com/english/wiki/pca-using-prcomp-and-princomp
- PCA using **ade4** and **factoextra** (tutorial). http://www.sthda.com/english/wiki/pca-using-ade4-and-factoextra

Chapter 4

Correspondence Analysis

4.1 Introduction

Correspondence analysis (**CA**) is an extension of principal component analysis (Chapter 3) suited to explore relationships among qualitative variables (or categorical data). Like principal component analysis, it provides a solution for summarizing and visualizing data set in two-dimension plots.

Here, we describe the simple correspondence analysis, which is used to analyze frequencies formed by two categorical data, a data table known as *contengency table*. It provides factor scores (coordinates) for both row and column points of contingency table. These coordinates are used to visualize graphically the association between row and column elements in the contingency table.

When analyzing a two-way contingency table, a typical question is whether certain row elements are associated with some elements of column elements. Correspondence analysis is a geometric approach for visualizing the rows and columns of a two-way contingency table as points in a low-dimensional space, such that the positions of the row and column points are consistent with their associations in the table. The aim is to have a global view of the data that is useful for interpretation.

In the current chapter, we'll show how to compute and interpret correspondence analysis using two R packages: i) *FactoMineR* for the analysis and ii) *factoextra* for data visualization. Additionally, we'll show how to reveal the most important variables that explain the variations in a data set. We continue by explaining how to apply correspondence analysis using supplementary rows and columns. This is important, if you want to make predictions with CA. The last sections of this guide describe also how to filter CA result in order to keep only the most contributing variables. Finally, we'll see how to deal with outliers.

4.2 Computation

4.2.1 R packages

Several functions from different packages are available in the *R software* for computing correspondence analysis:

- *CA*() [*FactoMineR* package],
- *ca*() [*ca* package],
- *dudi.coa*() [*ade4* package],
- *corresp*() [*MASS* package],
- and *epCA*() [*ExPosition* package]

No matter what function you decide to use, you can easily extract and visualize the results of correspondence analysis using R functions provided in the *factoextra* R package.

Here, we'll use FactoMineR (for the analysis) and factoextra (for ggplot2-based elegant visualization). To install the two packages, type this:

```
install.packages(c("FactoMineR", "factoextra"))
```

Load the packages:

```
library("FactoMineR")
library("factoextra")
```

4.2.2 Data format

The data should be a contingency table. We'll use the demo data sets *housetasks* available in the *factoextra* R package

```
data(housetasks)
# head(housetasks)
```

The data is a contingency table containing 13 housetasks and their repartition in the couple:

- rows are the different tasks
- values are the frequencies of the tasks done :
  - by the wife only
  - alternatively
  - by the husband only
  - or jointly

The data is illustrated in the following image:

	Wife	Alternating	Husband	Jointly
Laundry	156	14	2	4
Main_meal	124	20	5	4
Dinner	77	11	7	13
Breakfeast	82	36	15	7
Tidying	53	11	1	57
Dishes	32	24	4	53
Shopping	33	23	9	55
Official	12	46	23	15
Driving	10	51	75	3
Finances	13	13	21	66
Insurance	8	1	53	77
Repairs	0	3	160	2
Holidays	0	1	6	153

4.2.3 Graph of contingency tables and chi-square test

The above *contingency table* is not very large. Therefore, it's easy to visually inspect and interpret row and column profiles:

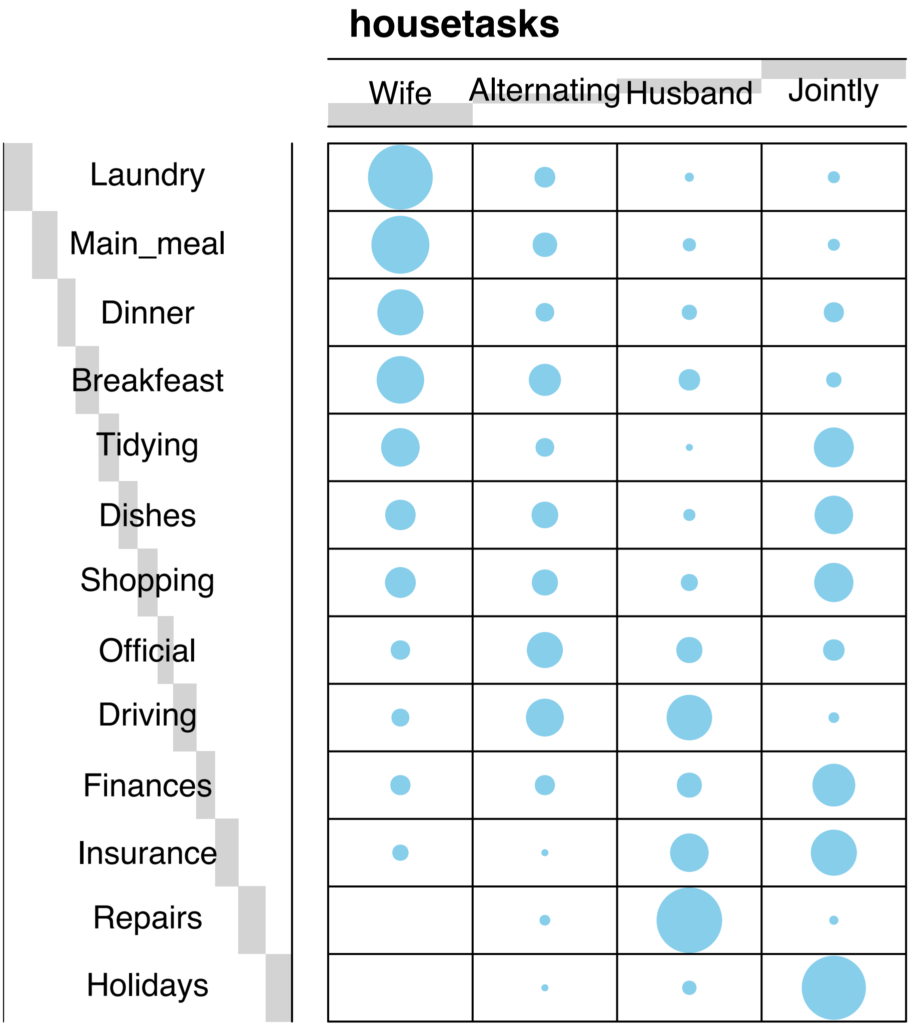
- It's evident that, the housetasks - *Laundry, Main_Meal and Dinner* - are more frequently done by the "Wife".

- Repairs and driving are dominantly done by the husband
- Holidays are frequently associated with the column "jointly"

Exploratory data analysis and visualization of contingency tables have been covered in our previous article: Chi-Square test of independence in R[1]. Briefly, contingency table can be visualized using the functions *balloonplot*() [*gplots* package] and *mosaicplot*() [*garphics* package]:

```
library("gplots")
# 1. convert the data as a table
dt <- as.table(as.matrix(housetasks))
# 2. Graph
balloonplot(t(dt), main ="housetasks", xlab ="", ylab="",
            label = FALSE, show.margins = FALSE)
```

[1] http://www.sthda.com/english/wiki/chi-square-test-of-independence-in-r



Note that, row and column sums are printed by default in the bottom and right margins, respectively. These values are hidden, in the above plot, using the argument *show.margins = FALSE*.

For a small contingency table, you can use the Chi-square test to evaluate whether there is a significant dependence between row and column categories:

```
chisq <- chisq.test(housetasks)
chisq
```

```
##
##  Pearson's Chi-squared test
##
## data:  housetasks
## X-squared = 2000, df = 40, p-value <2e-16
```

In our example, the row and the column variables are statistically significantly associated (*p-value* = `r chisq$p.value`).

4.2.4 R code to compute CA

The function *CA*()[*FactoMiner* package] can be used. A simplified format is :

```
CA(X, ncp = 5, graph = TRUE)
```

- **X** : a data frame (contingency table)
- **ncp** : number of dimensions kept in the final results.
- **graph** : a logical value. If TRUE a graph is displayed.

To compute correspondence analysis, type this:

```
library("FactoMineR")
res.ca <- CA(housetasks, graph = FALSE)
```

The output of the function *CA*() is a list including :

```
print(res.ca)
```

```
## **Results of the Correspondence Analysis (CA)**
## The row variable has  13  categories; the column variable has 4 categories
## The chi square of independence between the two variables is equal to 1944 (p-value
## *The results are available in the following objects:
##
##    name              description
## 1  "$eig"            "eigenvalues"
## 2  "$col"            "results for the columns"
## 3  "$col$coord"      "coord. for the columns"
## 4  "$col$cos2"       "cos2 for the columns"
## 5  "$col$contrib"    "contributions of the columns"
## 6  "$row"            "results for the rows"
## 7  "$row$coord"      "coord. for the rows"
## 8  "$row$cos2"       "cos2 for the rows"
## 9  "$row$contrib"    "contributions of the rows"
## 10 "$call"           "summary called parameters"
## 11 "$call$marge.col" "weights of the columns"
## 12 "$call$marge.row" "weights of the rows"
```

The object that is created using the function *CA*() contains many information found in many different lists and matrices. These values are described in the next section.

4.3 Visualization and interpretation

We'll use the following functions [in *factoextra*] to help in the interpretation and the visualization of the correspondence analysis:

- *get_eigenvalue*(res.ca): Extract the eigenvalues/variances retained by each dimension (axis)
- *fviz_eig*(res.ca): Visualize the eigenvalues

- *get_ca_row*(res.ca), *get_ca_col*(res.ca): Extract the results for rows and columns, respectively.
- *fviz_ca_row*(res.ca), *fviz_ca_col*(res.ca): Visualize the results for rows and columns, respectively.
- *fviz_ca_biplot*(res.ca): Make a biplot of rows and columns.

In the next sections, we'll illustrate each of these functions.

4.3.1 Statistical significance

To interpret correspondence analysis, the first step is to evaluate whether there is a significant dependency between the rows and columns.

A rigorous method is to use the *chi-square statistic* for examining the association between row and column variables. This appears at the top of the report generated by the function *summary(res.ca)* or *print(res.ca)*, see section 4.2.4. A high chi-square statistic means strong link between row and column variables.

In our example, the association is highly significant (*chi-square: 1944.456*, *p = 0*).

```
# Chi-square statistics
chi2 <- 1944.456
# Degree of freedom
df <- (nrow(housetasks) - 1) * (ncol(housetasks) - 1)
# P-value
pval <- pchisq(chi2, df = df, lower.tail = FALSE)
pval
```

```
## [1] 0
```

4.3.2 Eigenvalues / Variances

Recall that, we examine the eigenvalues to determine the number of axis to be considered. The eigenvalues and the proportion of variances retained by the different axes can be extracted using the function *get_eigenvalue*() [*factoextra* package]. *Eigenvalues* are large for the first axis and small for the subsequent axis.

```
library("factoextra")
eig.val <- get_eigenvalue(res.ca)
eig.val
```

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1      0.543             48.7                        48.7
## Dim.2      0.445             39.9                        88.6
## Dim.3      0.127             11.4                       100.0
```

Eigenvalues correspond to the amount of information retained by each axis. Dimensions are ordered decreasingly and listed according to the amount of variance explained in the solution. Dimension 1 explains the most variance in the solution, followed by dimension 2 and so on.

The cumulative percentage explained is obtained by adding the successive proportions of variation explained to obtain the running total. For instance, 48.69% plus 39.91% equals 88.6%, and so forth. Therefore, about 88.6% of the variation is explained by the first two dimensions.

Eigenvalues can be used to determine the number of axes to retain. There is no "rule of thumb" to choose the number of dimensions to keep for the data interpretation. It depends on the research question and the researcher's need. For example, if you are satisfied with 80% of the total variances explained then use the number of dimensions necessary to achieve that.

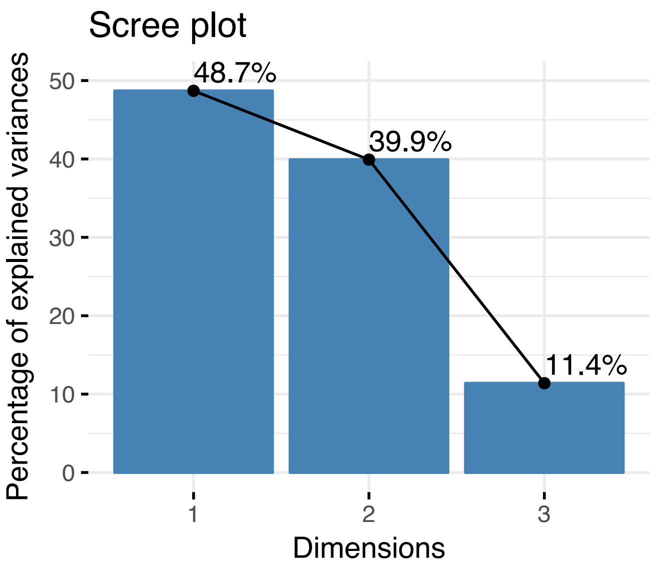
Note that, a good dimension reduction is achieved when the the first few dimensions account for a large proportion of the variability.

In our analysis, the first two axes explain 88.6% of the variation. This is an acceptably large percentage.

An alternative method to determine the number of dimensions is to look at a Scree Plot, which is the plot of eigenvalues/variances ordered from largest to the smallest. The number of component is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size.

The scree plot can be produced using the function *fviz_eig*() or *fviz_screeplot*() [*factoextra* package].

```
fviz_screeplot(res.ca, addlabels = TRUE, ylim = c(0, 50))
```



The point at which the *scree plot* shows a bend (so called "elbow") can be considered as indicating an optimal dimensionality.

It's also possible to calculate an average eigenvalue above which the axis should be kept in the solution.

Our data contains 13 rows and 4 columns.

If the data were random, the expected value of the eigenvalue for each axis would be 1/(nrow(housetasks)-1) = 1/12 = 8.33% in terms of rows.

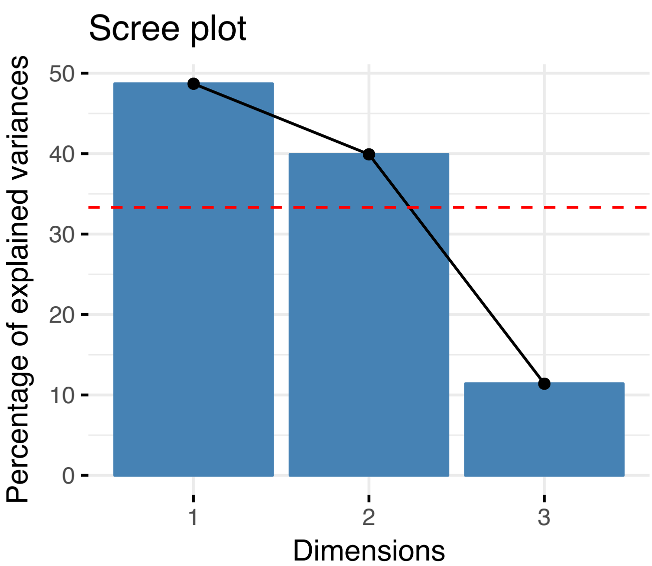
Likewise, the average axis should account for 1/(ncol(housetasks)-1) = 1/3 = 33.33% in terms of the 4 columns.

According to (Bendixen, 1995):

Any axis with a contribution larger than the maximum of these two percentages should be considered as important and included in the solution for the interpretation of the data.

The R code below, draws the scree plot with a red dashed line specifying the average eigenvalue:

```
fviz_screeplot(res.ca) +
 geom_hline(yintercept=33.33, linetype=2, color="red")
```



According to the graph above, only dimensions 1 and 2 should be used in the solution. The dimension 3 explains only 11.4% of the total inertia which is below the average eigeinvalue (33.33%) and too little to be kept for further analysis.

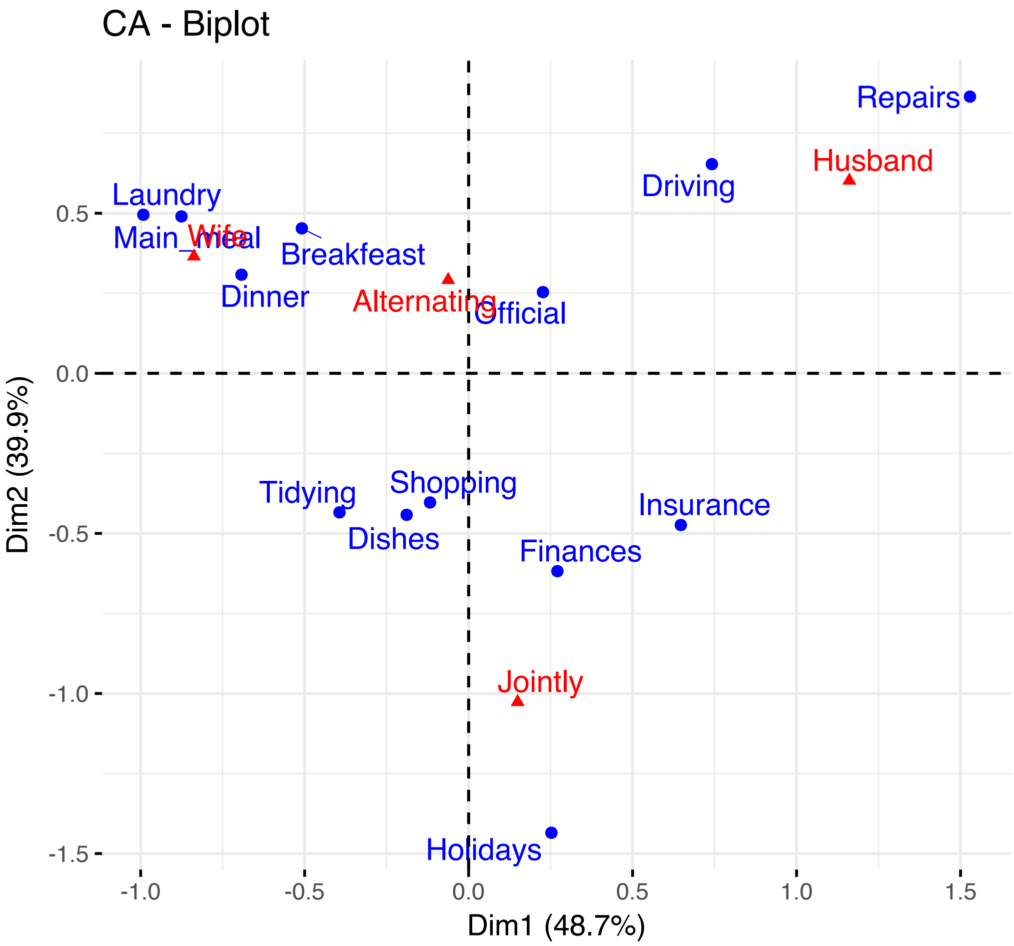
Note that, you can use more than 2 dimensions. However, the supplementary dimensions are unlikely to contribute significantly to the interpretation of nature of the association between the rows and columns.

Dimensions 1 and 2 explain approximately 48.7% and 39.9% of the total inertia respectively. This corresponds to a cumulative total of 88.6% of total inertia retained by the 2 dimensions. The higher the retention, the more subtlety in the original data is retained in the low-dimensional solution (Bendixen, 2003).

4.3.3 Biplot

The function *fviz_ca_biplot()* [*factoextra* package] can be used to draw the biplot of rows and columns variables.

```
# repel= TRUE to avoid text overlapping (slow if many point)
fviz_ca_biplot(res.ca, repel = TRUE)
```



The graph above is called *symetric plot* and shows a global pattern within the data. Rows are represented by blue points and columns by red triangles.

The distance between any row points or column points gives a measure of their similarity (or dissimilarity). Row points with similar profile are closed on the factor map. The same holds true for column points.

This graph shows that :

- housetasks such as dinner, breakfeast, laundry are done more often by the wife
- Driving and repairs are done by the husband
- ……

- *Symetric plot* represents the row and column profiles simultaneously in a common space. In this case, only the distance between row points or the distance between column points can be really interpreted.

- The distance between any row and column items is not meaningful! You can only make a general statements about the observed pattern.
- In order to interpret the distance between column and row points, the column profiles must be presented in row space or vice-versa. This type of map is called *asymmetric biplot* and is discussed at the end of this article.

The next step for the interpretation is to determine which row and column variables contribute the most in the definition of the different dimensions retained in the model.

4.3.4 Graph of row variables

4.3.4.1 Results

The function *get_ca_row*() [in *factoextra*] is used to extract the results for row variables. This function returns a list containing the coordinates, the cos2, the contribution and the inertia of row variables:

```
row <- get_ca_row(res.ca)
row
```

```
## Correspondence Analysis - Results for rows
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the rows"
## 2 "$cos2"    "Cos2 for the rows"
## 3 "$contrib" "contributions of the rows"
## 4 "$inertia" "Inertia of the rows"
```

The components of the *get_ca_row*() function can be used in the plot of rows as follow:

- *row$coord*: coordinates of each row point in each dimension (1, 2 and 3). Used to create the scatter plot.
- *row$cos2*: quality of representation of rows.
- *var$contrib*: contribution of rows (in %) to the definition of the dimensions.

Note that, it's possible to plot row points and to color them according to either i) their quality on the factor map (cos2) or ii) their contribution values to the definition of dimensions (contrib).

The different components can be accessed as follow:

```
# Coordinates
head(row$coord)

# Cos2: quality on the factore map
head(row$cos2)

# Contributions to the principal components
head(row$contrib)
```

In this section, we describe how to visualize row points only. Next, we highlight rows according to either i) their quality of representation on the factor map or ii) their contributions to the dimensions.

4.3.4.2 Coordinates of row points

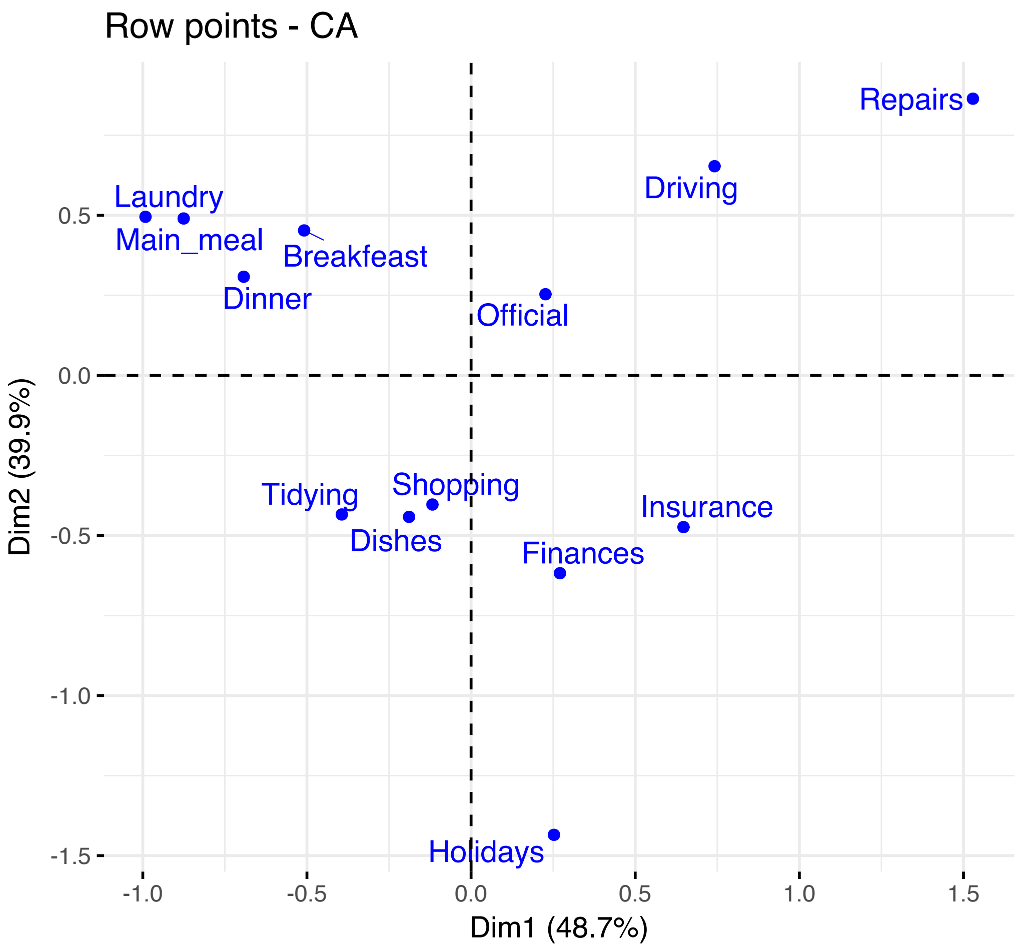
The R code below displays the coordinates of each row point in each dimension (1, 2 and 3):

```
head(row$coord)
```

```
##              Dim 1  Dim 2   Dim 3
## Laundry     -0.992  0.495 -0.3167
## Main_meal   -0.876  0.490 -0.1641
## Dinner      -0.693  0.308 -0.2074
## Breakfeast -0.509  0.453  0.2204
## Tidying     -0.394 -0.434 -0.0942
## Dishes      -0.189 -0.442  0.2669
```

Use the function *fviz_ca_row*() [in factoextra] to visualize only row points:

```
fviz_ca_row(res.ca, repel = TRUE)
```



It's possible to change the color and the shape of the row points using the arguments *col.row* and *shape.row* as follow:

```r
fviz_ca_row(res.ca, col.row="steelblue", shape.row = 15)
```

The plot above shows the relationships between row points:

- Rows with a similar profile are grouped together.
- Negatively correlated rows are positioned on opposite sides of the plot origin (opposed quadrants).

- The distance between row points and the origin measures the quality of the row points on the factor map. Row points that are away from the origin are well represented on the factor map.

4.3.4.3 Quality of representation of rows

The result of the analysis shows that, the contingency table has been successfully represented in low dimension space using correspondence analysis. The two dimensions 1 and 2 are sufficient to retain 88.6% of the total inertia (variation) contained in the data.

However, not all the points are equally well displayed in the two dimensions.

> Recall that, the *quality of representation* of the rows on the factor map is called the *squared cosine* (cos2) or the *squared correlations*.

The cos2 measures the degree of association between rows/columns and a particular axis. The cos2 of row points can be extracted as follow:

```r
head(row$cos2, 4)
```

```
##            Dim 1 Dim 2  Dim 3
## Laundry    0.740 0.185 0.0755
## Main_meal  0.742 0.232 0.0260
## Dinner     0.777 0.154 0.0697
## Breakfeast 0.505 0.400 0.0948
```

The values of the cos2 are comprised between 0 and 1. The sum of the cos2 for rows on all the CA dimensions is equal to one.

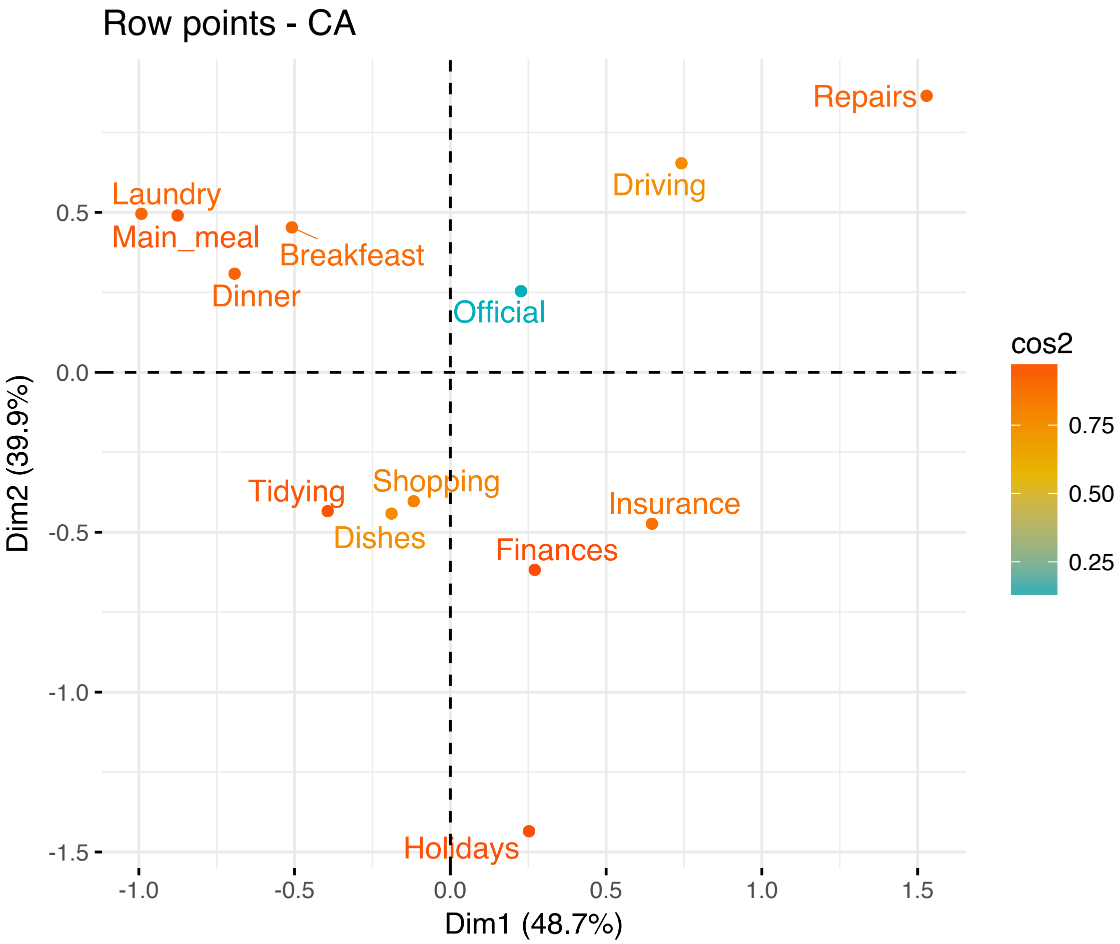
The quality of representation of a row or column in n dimensions is simply the sum of the squared cosine of that row or column over the n dimensions.

If a row item is well represented by two dimensions, the sum of the cos2 is closed to one. For some of the row items, more than 2 dimensions are required to perfectly represent the data.

It's possible to color row points by their cos2 values using the argument *col.row = "cos2"*. This produces a gradient colors, which can be customized using the argument *gradient.cols*. For instance, *gradient.cols = c("white", "blue", "red")* means that:

- variables with low cos2 values will be colored in "white"
- variables with mid cos2 values will be colored in "blue"
- variables with high cos2 values will be colored in red

```r
# Color by cos2 values: quality on the factor map
fviz_ca_row(res.ca, col.row = "cos2",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
            repel = TRUE)
```

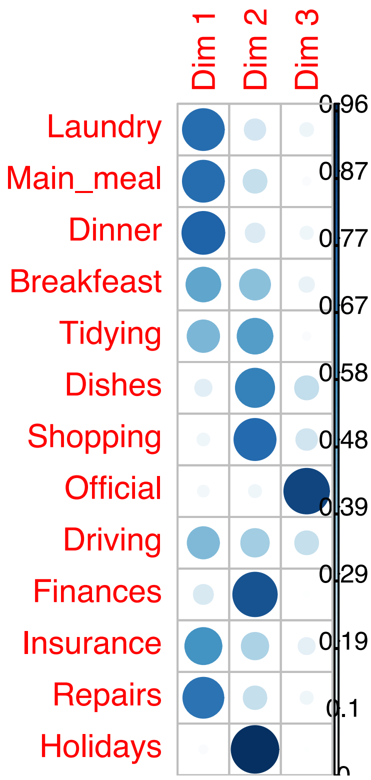


Note that, it's also possible to change the transparency of the row points according to their cos2 values using the option *alpha.row = "cos2"*. For example, type this:

```r
# Change the transparency by cos2 values
fviz_ca_row(res.ca, alpha.row="cos2")
```

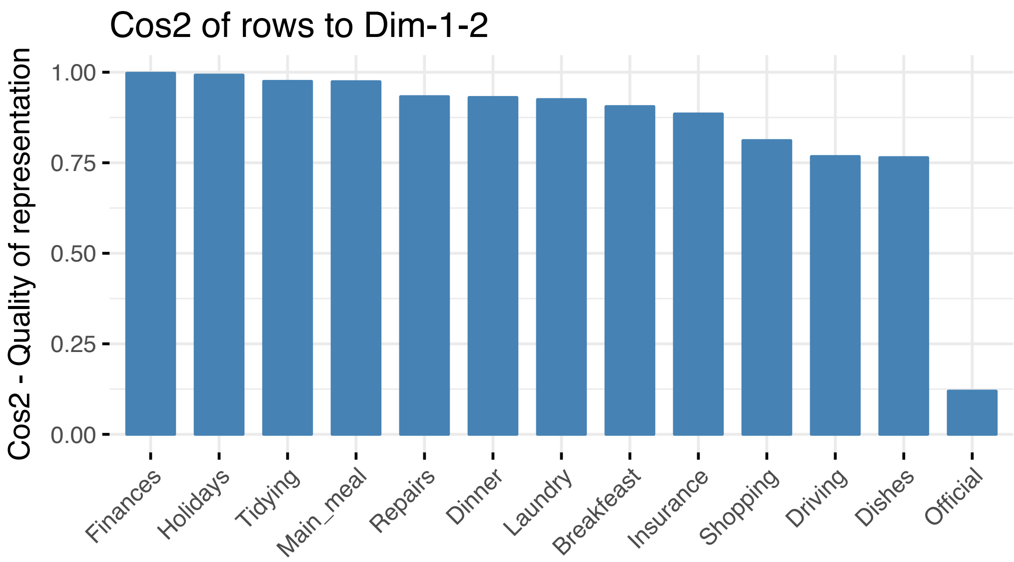
You can visualize the cos2 of row points on all the dimensions using the corrplot package:

```r
library("corrplot")
corrplot(row$cos2, is.corr=FALSE)
```



It's also possible to create a bar plot of rows cos2 using the function *fviz_cos2()*[in *factoextra*]:

```
# Cos2 of rows on Dim.1 and Dim.2
fviz_cos2(res.ca, choice = "row", axes = 1:2)
```



Note that, all row points except *Official* are well represented by the first two dimensions. This implies that the position of the point corresponding the item *Official* on the scatter plot should be interpreted with some caution. A higher dimensional solution is probably necessary for the item *Official*.

4.3.4.4 Contributions of rows to the dimensions

The contribution of rows (in %) to the definition of the dimensions can be extracted as follow:

```r
head(row$contrib)
```

```
##             Dim 1 Dim 2 Dim 3
## Laundry    18.287  5.56 7.968
## Main_meal  12.389  4.74 1.859
## Dinner      5.471  1.32 2.097
## Breakfeast  3.825  3.70 3.069
## Tidying     1.998  2.97 0.489
## Dishes      0.426  2.84 3.634
```

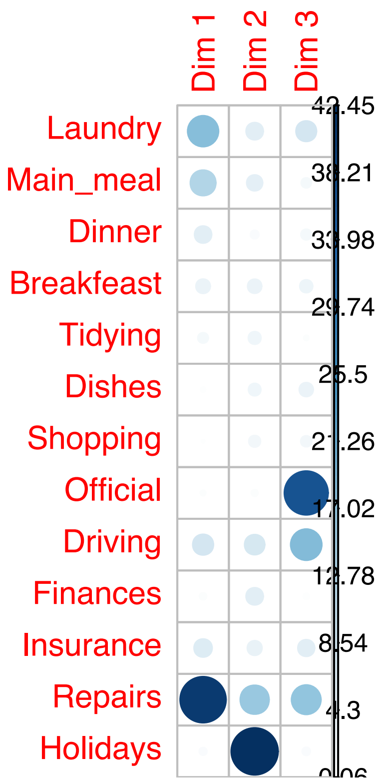
The row variables with the larger value, contribute the most to the definition of the dimensions.

- Rows that contribute the most to Dim.1 and Dim.2 are the most important in explaining the variability in the data set.

- Rows that do not contribute much to any dimension or that contribute to the last dimensions are less important.

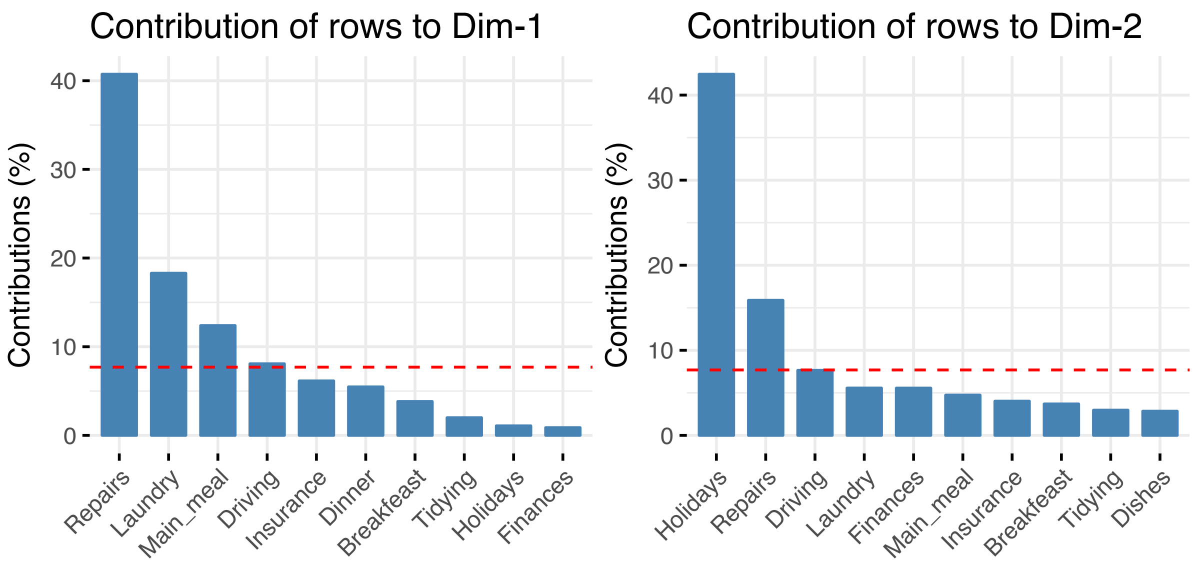
It's possible to use the function *corrplot*() [*corrplot* package] to highlight the most contributing row points for each dimension:

```r
library("corrplot")
corrplot(row$contrib, is.corr=FALSE)
```



The function *fviz_contrib*() [factoextra package] can be used to draw a bar plot of row contributions. If your data contains many rows, you can decide to show only the top contributing rows. The R code below shows the top 10 rows contributing to the dimensions:

```
# Contributions of rows to dimension 1
fviz_contrib(res.ca, choice = "row", axes = 1, top = 10)

# Contributions of rows to dimension 2
fviz_contrib(res.ca, choice = "row", axes = 2, top = 10)
```



The total contribution to dimension 1 and 2 can be obtained as follow:

```
# Total contribution to dimension 1 and 2
fviz_contrib(res.ca, choice = "row", axes = 1:2, top = 10)
```

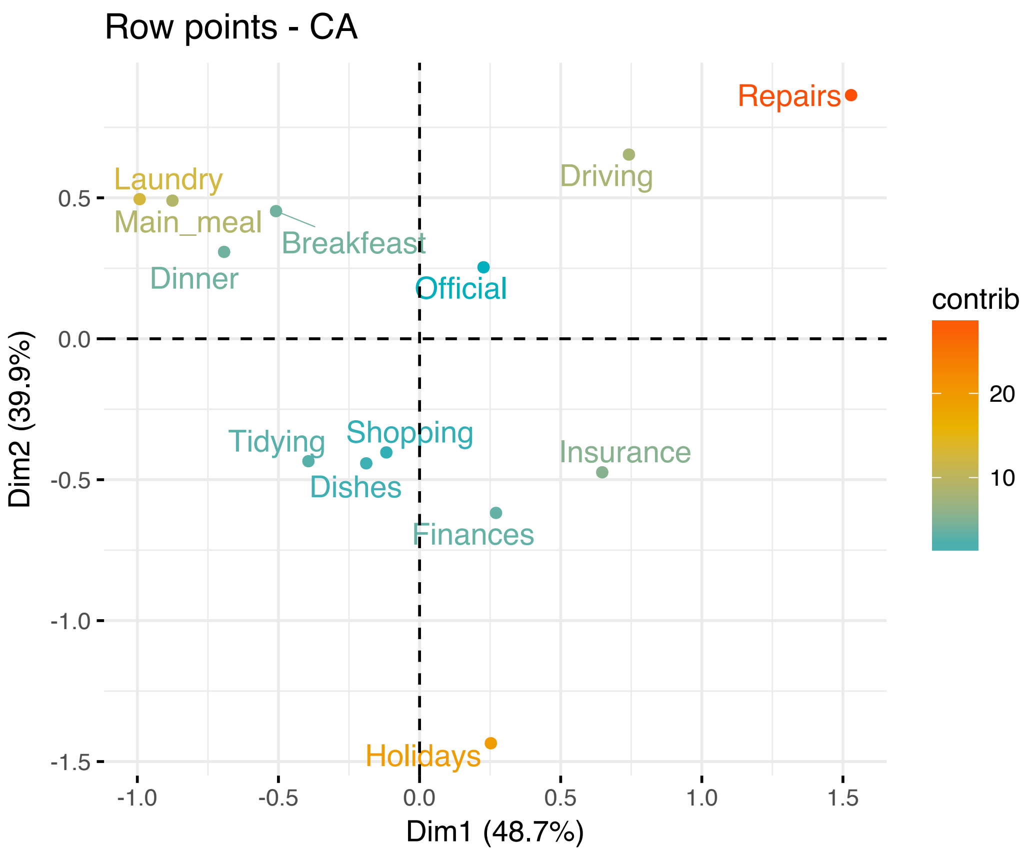
The red dashed line on the graph above indicates the expected average value, If the contributions were uniform. The calculation of the expected contribution value, under null hypothesis, has been detailed in the principal component analysis chapter (3).

It can be seen that:

- the row items *Repairs, Laundry, Main_meal and Driving* are the most important in the definition of the first dimension.
- the row items *Holidays and Repairs* contribute the most to the dimension 2.

The most important (or, contributing) row points can be highlighted on the scatter plot as follow:

```
fviz_ca_row(res.ca, col.row = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```



> The scatter plot gives an idea of what pole of the dimensions the row categories are actually contributing to.
> It is evident that row categories *Repair and Driving* have an important contribution to the positive pole of the first dimension, while the categories *Laundry and Main_meal* have a major contribution to the negative pole of the first dimension; etc, ….
>
> In other words, dimension 1 is mainly defined by the opposition of *Repair and Driving* (positive pole), and *Laundry and Main_meal* (negative pole).

Note that, it's also possible to control the transparency of row points according to their contribution values using the option *alpha.row = "contrib"*. For example, type this:

```
# Change the transparency by contrib values
fviz_ca_row(res.ca, alpha.row="contrib",
            repel = TRUE)
```

4.3.5 Graph of column variables

4.3.5.1 Results

The function *get_ca_col()* [in *factoextra*] is used to extract the results for column variables. This function returns a list containing the coordinates, the cos2, the contribution and the inertia of columns variables:

```
col <- get_ca_col(res.ca)
col
```

```
## Correspondence Analysis - Results for columns
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the columns"
## 2 "$cos2"    "Cos2 for the columns"
## 3 "$contrib" "contributions of the columns"
## 4 "$inertia" "Inertia of the columns"
```

The result for columns gives the same information as described for rows. For this reason, we'll just displayed the result for columns in this section with only a very brief comment.

To get access to the different components, use this:

```
# Coordinates of column points
head(col$coord)

# Quality of representation
head(col$cos2)

# Contributions
head(col$contrib)
```

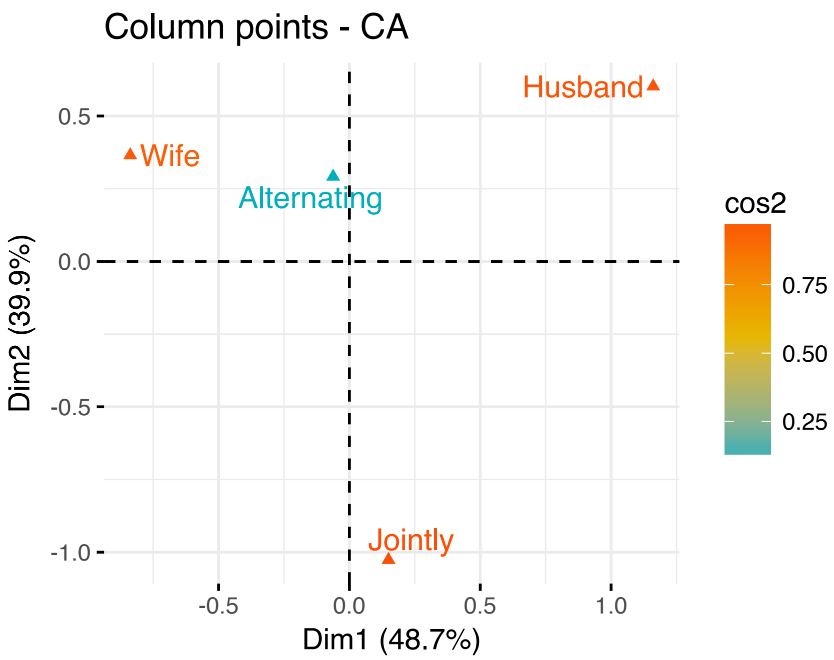
4.3.5.2 Plots: quality and contribution

The *fviz_ca_col*() is used to produce the graph of column points. To create a simple plot, type this:

```
fviz_ca_col(res.ca)
```

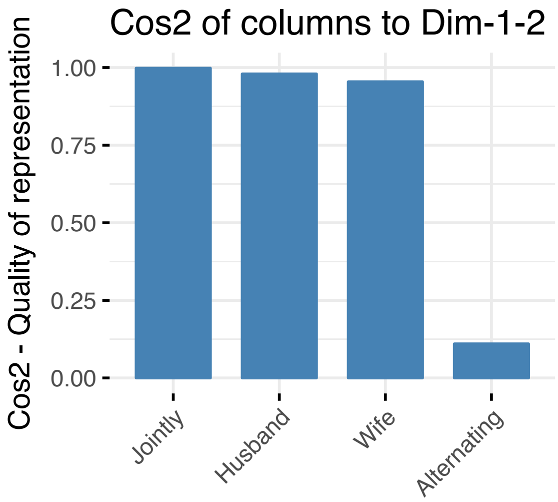
Like row points, it's also possible to color column points by their cos2 values:

```
fviz_ca_col(res.ca, col.col = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```



The R code below creates a barplot of columns cos2:

```
fviz_cos2(res.ca, choice = "col", axes = 1:2)
```

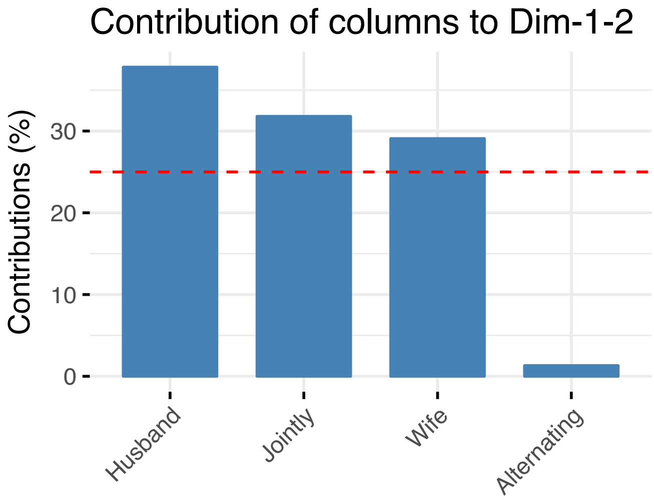


Recall that, the value of the cos2 is between 0 and 1. A cos2 closed to 1 corresponds to a column/row variables that are well represented on the factor map.

Note that, only the column item *Alternating* is not very well displayed on the first two dimensions. The position of this item must be interpreted with caution in the space formed by dimensions 1 and 2.

To visualize the contribution of rows to the first two dimensions, type this:

```
fviz_contrib(res.ca, choice = "col", axes = 1:2)
```



4.3.6 Biplot options

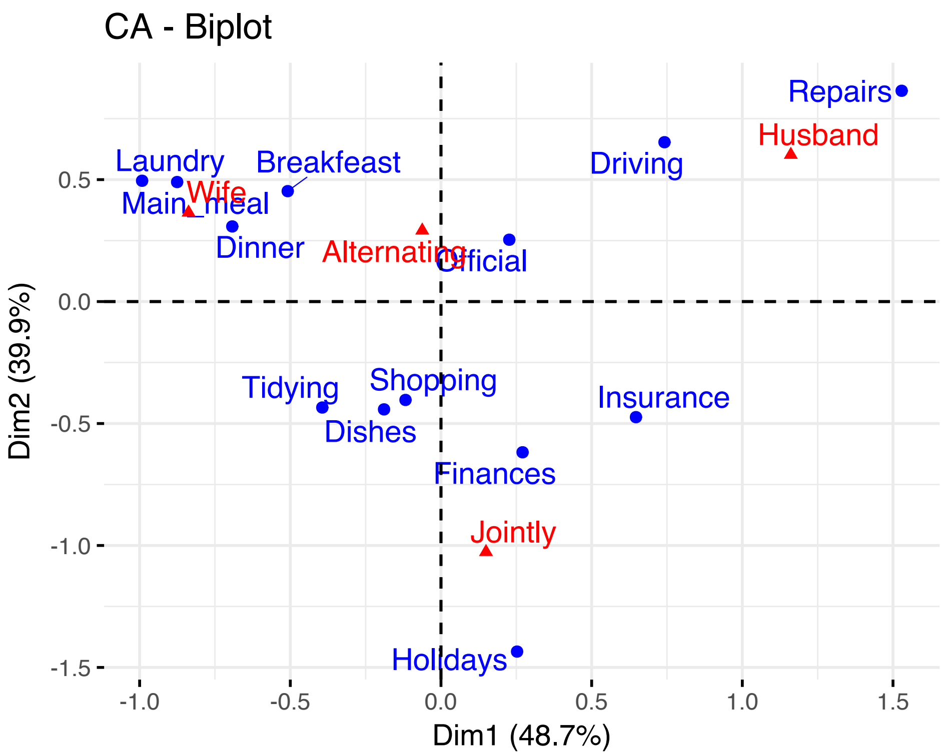
Biplot is a graphical display of rows and columns in 2 or 3 dimensions. We have already described how to create CA biplots in section 4.3.3. Here, we'll describe different types of CA biplots.

4.3.6.1 Symmetric biplot

As mentioned above, the standard plot of correspondence analysis is a *symmetric biplot* in which both rows (blue points) and columns (red triangles) are represented in the same space using the *principal coordinates*. These coordinates represent the row and column profiles. In this case, only the distance between row points or the distance between column points can be really interpreted.

> With symmetric plot, the inter-distance between rows and columns can't be interpreted. Only a general statements can be made about the pattern.

```
fviz_ca_biplot(res.ca, repel = TRUE)
```



Note that, in order to interpret the distance between column points and row points, the simplest way is to make an *asymmetric plot*. This means that, the column profiles must be presented in row space or vice-versa.

4.3.6.2 Asymmetric biplot

To make an *asymetric biplot*, rows (or columns) points are plotted from the *standard co-ordinates* (S) and the profiles of the columns (or the rows) are plotted from the *principale coordinates* (P) (Bendixen, 2003).

For a given axis, the standard and principle co-ordinates are related as follows:

$P = sqrt(eigenvalue)\ X\ S$

- P: the principal coordinate of a row (or a column) on the axis
- *eigenvalue*: the eigenvalue of the axis

Depending on the situation, other types of display can be set using the argument *map* (Nenadic and Greenacre, 2007) in the function *fviz_ca_biplot*() [in *factoextra*].

The allowed options for the argument *map* are:

1. **"rowprincipal"** or **"colprincipal"** - these are the so-called *asymmetric biplots*, with either rows in principal coordinates and columns in standard coordinates, or vice versa (also known as row-metric-preserving or column-metric-preserving, respectively).

- **"rowprincipal"**: columns are represented in row space
- **"colprincipal"**: rows are represented in column space

2. **"symbiplot"** - both rows and columns are scaled to have variances equal to the singular values (square roots of eigenvalues), which gives a *symmetric biplot* but does not preserve row or column metrics.

3. **"rowgab"** or **"colgab"**: *Asymetric maps* proposed by Gabriel & Odoroff (Gabriel and Odoroff, 1990):

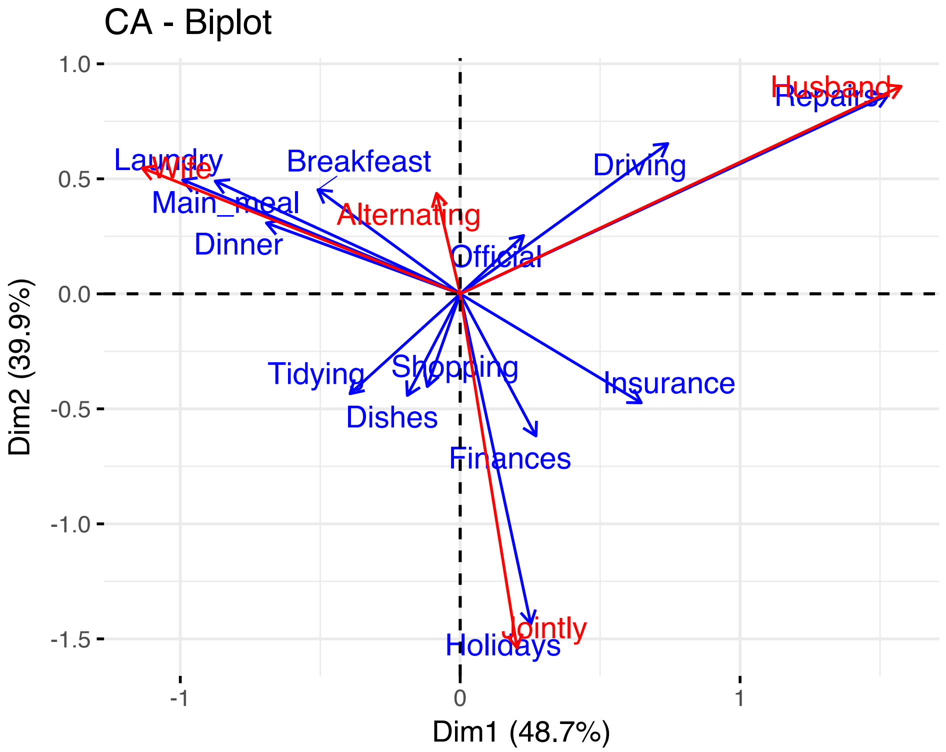
- "*rowgab*": rows in principal coordinates and columns in standard coordinates multiplied by the mass.
- "*colgab*": columns in principal coordinates and rows in standard coordinates multiplied by the mass.

4. **"rowgreen"** or **"colgreen"**: The so-called *contribution biplots* showing visually the most contributing points (Greenacre 2006b).

- "*rowgreen*": rows in principal coordinates and columns in standard coordinates multiplied by square root of the mass.
- "*colgreen*": columns in principal coordinates and rows in standard coordinates multiplied by the square root of the mass.

The R code below draws a standard *asymetric biplot*:

```
fviz_ca_biplot(res.ca,
               map ="rowprincipal", arrow = c(TRUE, TRUE),
               repel = TRUE)
```



We used, the argument *arrows*, which is a vector of two logicals specifying if the plot should contain points (FALSE, default) or arrows (TRUE). The first value sets the rows and the second value sets the columns.

If the angle between two arrows is acute, then their is a strong association between the corresponding row and column.

To interpret the distance between rows and and a column you should perpendicularly project row points on the column arrow.

4.3.6.3 Contribution biplot

In the standard *symmetric biplot* (mentioned in the previous section), it's difficult to know the most contributing points to the solution of the CA.

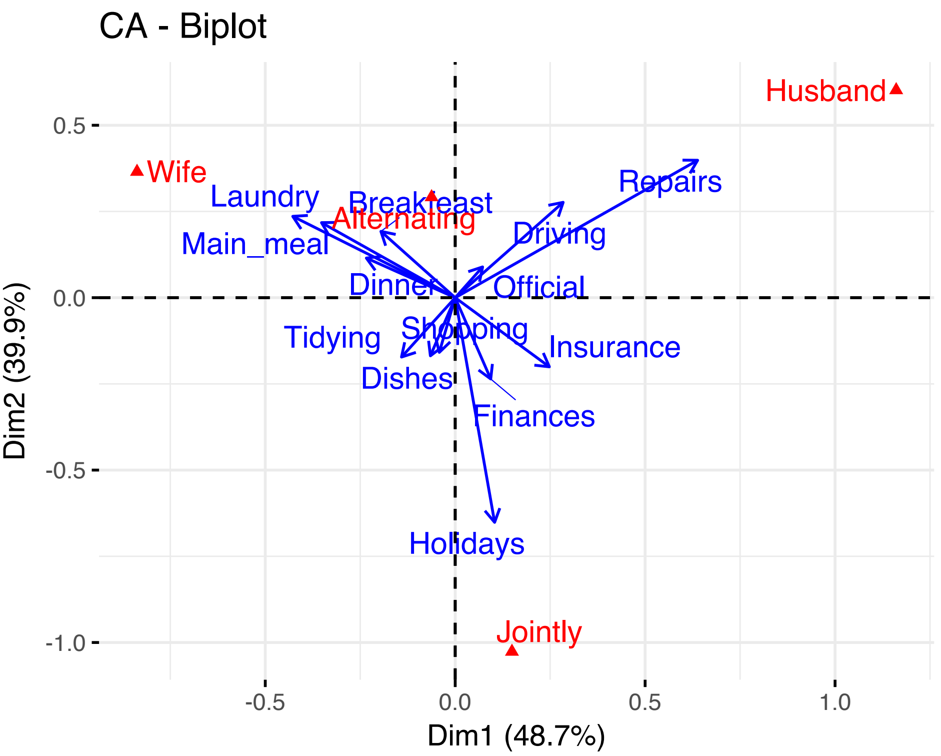
Michael Greenacre proposed a new scaling displayed (called contribution biplot) which incorporates the contribution of points (Greenacre, 2013). In this display, points that contribute very little to the solution, are close to the center of the biplot and are relatively unimportant to the interpretation.

A *contribution biplot* can be drawn using the argument *map = "rowgreen"* or *map = "colgreen"*.

Firstly, you have to decide whether to analyse the contributions of rows or columns to the definition of the axes.

In our example we'll interpret the contribution of rows to the axes. The argument *map ="colgreen"* is used. In this case, recall that columns are in principal coordinates and rows in standard coordinates multiplied by the square root of the mass. For a given row, the square of the new coordinate on an axis i is exactly the contribution of this row to the inertia of the axis i.

```
fviz_ca_biplot(res.ca, map ="colgreen", arrow = c(TRUE, FALSE),
               repel = TRUE)
```



In the graph above, the position of the column profile points is unchanged relative to that in the conventional biplot. However, the distances of the row points from the plot origin are related to their contributions to the two-dimensional factor map.

The closer an arrow is (in terms of angular distance) to an axis the greater is the contribution of the row category on that axis relative to the other axis. If the arrow is halfway between the two, its row category contributes to the two axes to the same extent.

- It is evident that row category *Repairs* have an important contribution to the positive pole of the first dimension, while the categories *Laundry* and *Main_meal* have a major contribution to the negative pole of the first dimension;
- Dimension 2 is mainly defined by the row category *Holidays*.
- The row category *Driving* contributes to the two axes to the same extent.

4.3.7 Dimension description

To easily identify row and column points that are the most associated with the principal dimensions, you can use the function *dimdesc*() [in FactoMineR]. Row/column variables are sorted by their coordinates in the *dimdesc*() output.

```
# Dimension description
res.desc <- dimdesc(res.ca, axes = c(1,2))
```

Description of dimension 1:

```
# Description of dimension 1 by row points
head(res.desc[[1]]$row, 4)
```

```
##              coord
## Laundry    -0.992
## Main_meal  -0.876
## Dinner     -0.693
## Breakfeast -0.509
```

```
# Description of dimension 1 by column points
head(res.desc[[1]]$col, 4)
```

```
##               coord
## Wife        -0.8376
## Alternating -0.0622
## Jointly      0.1494
## Husband      1.1609
```

Description of dimension 2:

```
# Description of dimension 2 by row points
res.desc[[2]]$row

# Description of dimension 1 by column points
res.desc[[2]]$col
```

4.4 Supplementary elements

4.4.1 Data format

We'll use the data set *children* [in *FactoMineR* package]. It contains 18 rows and 8 columns:

```
data(children)
# head(children)
```

	unqualified	cep	bepc	high_school_diploma	university	thirty	fifty	more_fifty
money	51	64	32	29	17	59	66	70
future	53	90	78	75	22	115	117	86
unemployment	71	111	50	40	11	79	88	177
circumstances	1	7	5	5	4	9	8	5
hard	7	11	4	3	2	2	17	18
economic	7	13	12	11	11	18	19	17
egoism	21	37	14	26	9	14	34	61
employment	12	35	19	6	7	21	30	28
finances	10	7	7	3	1	8	12	8
war	4	7	7	6	2	7	6	13
housing	8	22	7	10	5	10	27	17
fear	25	45	38	38	13	48	59	52
health	18	27	20	19	9	13	29	53
work	35	61	29	14	12	30	63	58
comfort	2	4	3	1	4	NA	NA	NA
disagreement	2	8	2	5	2	NA	NA	NA
world	1	5	4	6	3	NA	NA	NA
to_live	3	3	1	3	4	NA	NA	NA

Figure 4.1: Data format for correspondence analysis with supplementary elements

The data used here is a contingency table describing the answers given by different categories of people to the following question: What are the reasons that can make hesitate a woman or a couple to have children?

Only some of the rows and columns will be used to perform the correspondence analysis (CA). The coordinates of the remaining (supplementary) rows/columns on the factor map will be **predicted** after the CA.

In CA terminology, our data contains :

- *Active rows* (rows 1:14) : Rows that are used during the correspondence analysis.
- *Supplementary rows* (row.sup 15:18) : The coordinates of these rows will be predicted using the CA information and parameters obtained with active rows/columns
- *Active columns* (columns 1:5) : Columns that are used for the correspondence analysis.
- *Supplementary columns* (col.sup 6:8) : As supplementary rows, the coordinates of these columns will be predicted also.

4.4.2 Specification in CA

As mentioned above, supplementary rows and columns are not used for the definition of the principal dimensions. Their coordinates are predicted using only the information provided by the performed CA on active rows/columns.

To specify supplementary rows/columns, the function *CA*()[in *FactoMineR*] can be used as follow :

```
CA(X,  ncp = 5, row.sup = NULL, col.sup = NULL,
   graph = TRUE)
```

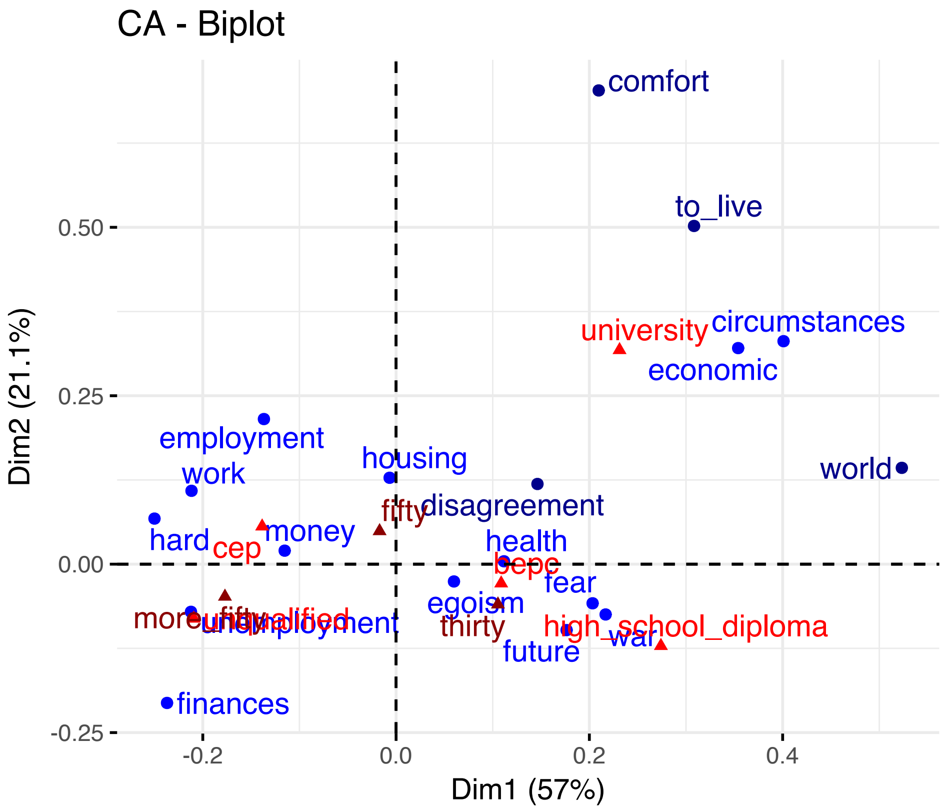
- **X** : a data frame (contingency table)
- **row.sup** : a numeric vector specifying the indexes of the supplementary rows
- **col.sup** : a numeric vector specifying the indexes of the supplementary columns
- **ncp** : number of dimensions kept in the final results.
- **graph** : a logical value. If TRUE a graph is displayed.

For example, type this:

```
res.ca <- CA (children, row.sup = 15:18, col.sup = 6:8,
              graph = FALSE)
```

4.4.3 Biplot of rows and columns

```
fviz_ca_biplot(res.ca, repel = TRUE)
```



- Active rows are in blue
- Supplementary rows are in darkblue
- Columns are in red
- Supplementary columns are in darkred

It's also possible to hide supplementary rows and columns using the argument *invisible*:

```
fviz_ca_biplot(res.ca, repel = TRUE,
               invisible = c("row.sup", "col.sup"))
```

4.4.4 Supplementary rows

Predicted results (coordinates and cos2) for the supplementary rows:

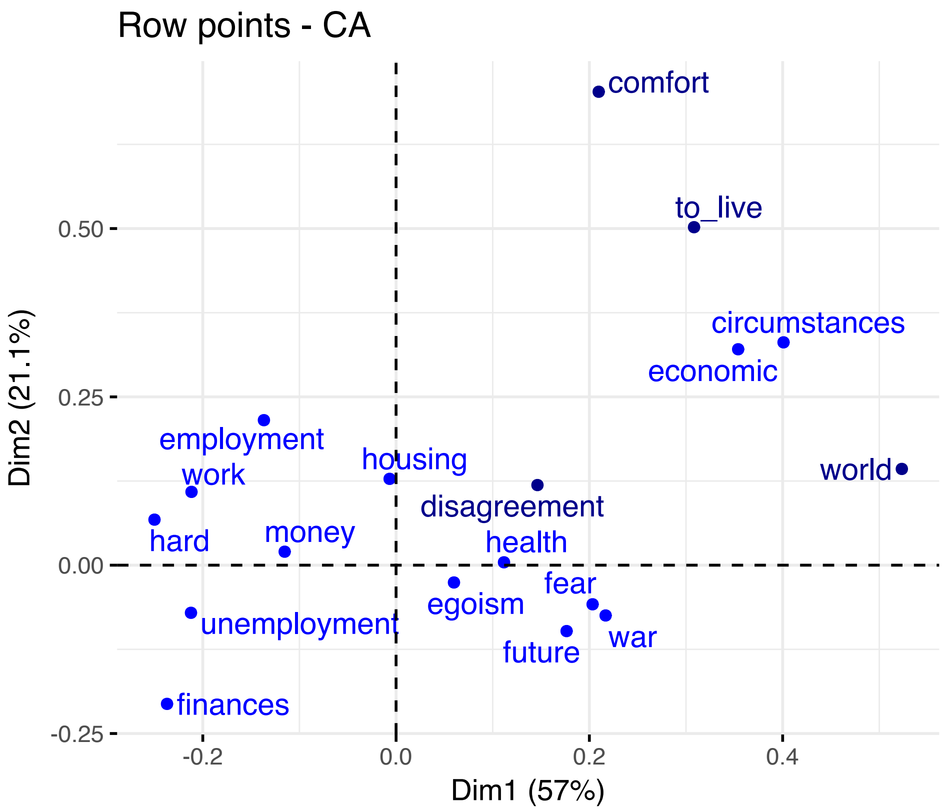
```
res.ca$row.sup
```

```
## $coord
##                Dim 1 Dim 2  Dim 3  Dim 4
## comfort        0.210 0.703 0.0711  0.307
## disagreement 0.146 0.119 0.1711 -0.313
## world          0.523 0.143 0.0840 -0.106
## to_live        0.308 0.502 0.5209  0.256
##
## $cos2
##                 Dim 1  Dim 2   Dim 3  Dim 4
## comfort        0.0689 0.7752 0.00793 0.1479
## disagreement 0.1313 0.0869 0.17965 0.6021
```

```
## world        0.8759 0.0654 0.02256 0.0362
## to_live      0.1390 0.3685 0.39683 0.0956
```

Plot of active and supplementary row points:

```
fviz_ca_row(res.ca, repel = TRUE)
```



Supplementary rows are shown in darkblue color.

4.4.5 Supplementary columns

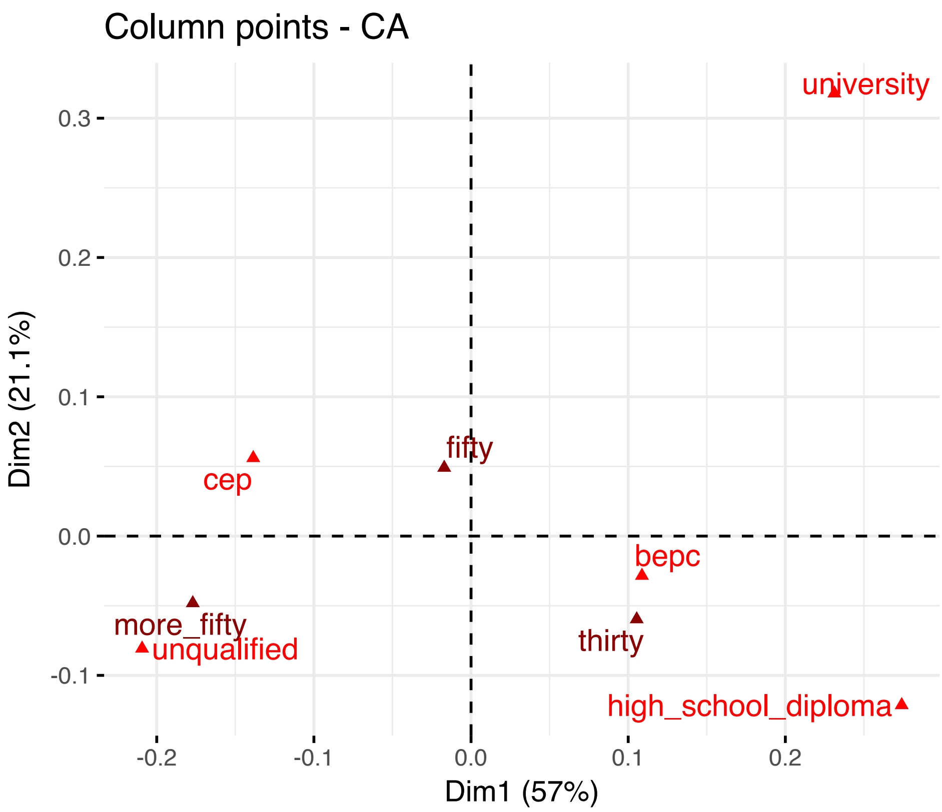
Predicted results (coordinates and cos2) for the supplementary columns:

```
res.ca$col.sup
```

```
## $coord
##                 Dim 1   Dim 2   Dim 3   Dim 4
## thirty       0.1054 -0.0597 -0.1032  0.0698
## fifty       -0.0171  0.0491 -0.0157 -0.0131
## more_fifty -0.1771 -0.0481  0.1008 -0.0852
##
## $cos2
##              Dim 1  Dim 2   Dim 3   Dim 4
## thirty     0.1376 0.0441 0.13191 0.06028
## fifty      0.0109 0.0899 0.00919 0.00637
## more_fifty 0.2861 0.0211 0.09267 0.06620
```

Plot of active and supplementary column points:

```
fviz_ca_col(res.ca, repel = TRUE)
```



Supplementary columns are shown in darkred.

4.5 Filtering results

If you have many row/column variables, it's possible to visualize only some of them using the arguments *select.row* and *select.col*.

select.col, select.row: a selection of columns/rows to be drawn. Allowed values are *NULL* or a *list* containing the arguments name, cos2 or contrib:

- *name*: is a character vector containing column/row names to be drawn
- *cos2*: if cos2 is in [0, 1], ex: 0.6, then columns/rows with a cos2 > 0.6 are drawn
- *if cos2 > 1*, ex: 5, then the top 5 active columns/rows and top 5 supplementary columns/rows with the highest cos2 are drawn
- *contrib*: if contrib > 1, ex: 5, then the top 5 columns/rows with the highest contributions are drawn

```
# Visualize rows with cos2 >= 0.8
fviz_ca_row(res.ca, select.row = list(cos2 = 0.8))

# Top 5 active rows and 5 suppl. rows with the highest cos2
fviz_ca_row(res.ca, select.row = list(cos2 = 5))
```

```
# Select by names
name <- list(name = c("employment", "fear", "future"))
fviz_ca_row(res.ca, select.row = name)

# Top 5 contributing rows and columns
fviz_ca_biplot(res.ca, select.row = list(contrib = 5),
               select.col = list(contrib = 5)) +
  theme_minimal()
```

4.6 Outliers

If one or more "outliers" are present in the contingency table, they can dominate the interpretation the axes (Bendixen, 2003).

Outliers are points that have high absolute co-ordinate values and high contributions. They are represented, on the graph, very far from the centroïd. In this case, the remaining row/column points tend to be tightly clustered in the graph which become difficult to interpret.

In the CA output, the coordinates of row/column points represent the number of standard deviations the row/column is away from the barycentre (Bendixen, 2003).

According to (Bendixen, 2003):

> Outliers are points that are are at least *one standard deviation away from the barycentre*. They contribute also, significantly to the interpretation to one pole of an axis.

> There are no apparent outliers in our data. If there were outliers in the data, they must be suppressed or treated as supplementary points when re-running the correspondence analysis.

4.7 Exporting results

4.7.1 Export plots to PDF/PNG files

To save the different graphs into pdf or png files, we start by creating the plot of interest as an R object:

```
# Scree plot
scree.plot <- fviz_eig(res.ca)

# Biplot of row and column variables
biplot.ca <- fviz_ca_biplot(res.ca)
```

Next, the plots can be exported into a single pdf file as follow (one plot per page):

```r
library(ggpubr)
ggexport(plotlist = list(scree.plot, biplot.ca),
         filename = "CA.pdf")
```

More options at: Chapter 3 (section: Exporting results).

4.7.2 Export results to txt/csv files

Easy to use R function: **write.infile**() [in *FactoMineR*] package:

```r
# Export into a TXT file
write.infile(res.ca, "ca.txt", sep = "\t")

# Export into a CSV file
write.infile(res.ca, "ca.csv", sep = ";")
```

4.8 Summary

In conclusion, we described how to perform and interpret correspondence analysis (CA). We computed CA using the *CA*() function [FactoMineR package]. Next, we used the *factoextra* R package to produce ggplot2-based visualization of the CA results.

Other functions [packages] to compute CA in R, include:

1) Using **dudi.coa**() [ade4]

```r
library("ade4")
res.ca <- dudi.coa(housetasks, scannf = FALSE, nf = 5)
```

Read more: `http://www.sthda.com/english/wiki/ca-using-ade4`

2) Using **ca**() [ca]

```r
library(ca)
res.ca <- ca(housetasks)
```

Read more: `http://www.sthda.com/english/wiki/ca-using-ca-package`

3) Using **corresp**() [MASS]

```r
library(MASS)
res.ca <- corresp(housetasks, nf = 3)
```

Read more: `http://www.sthda.com/english/wiki/ca-using-mass`

4) Using **epCA**() [ExPosition]

```r
library("ExPosition")
res.ca <- epCA(housetasks, graph = FALSE)
```

No matter what functions you decide to use, in the list above, the factoextra package can handle the output.

```
fviz_eig(res.ca)     # Scree plot

fviz_ca_biplot(res.ca) # Biplot of rows and columns
```

4.9 Further reading

For the mathematical background behind CA, refer to the following video courses, articles and books:

- Correspondence Analysis Course Using FactoMineR (Video courses). `https://goo.gl/Hhh6hC`
- Exploratory Multivariate Analysis by Example Using R (book) (Husson et al., 2017b).
- Principal component analysis (article). (Abdi and Williams, 2010). `https://goo.gl/1Vtwq1`.
- Correspondence analysis basics (blog post). `https://goo.gl/Xyk8KT`.
- Understanding the Math of Correspondence Analysis with Examples in R (blog post). `https://goo.gl/H9hxf9`

Chapter 5

Multiple Correspondence Analysis

5.1 Introduction

The **Multiple correspondence analysis** (**MCA**) is an extension of the simple correspondence analysis (chapter 4) for summarizing and visualizing a data table containing more than two categorical variables. It can also be seen as a generalization of principal component analysis when the variables to be analyzed are categorical instead of quantitative (Abdi and Williams, 2010).

MCA is generally used to analyse a data set from survey. The goal is to identify:

- A group of individuals with similar profile in their answers to the questions
- The associations between variable categories

Previously, we described how to compute and interpret the simple correspondence analysis (chapter 4). In the current chapter, we demonstrate how to compute and visualize multiple correspondence analysis in R software using *FactoMineR* (for the analysis) and *factoextra* (for data visualization). Additionally, we'll show how to reveal the most important variables that contribute the most in explaining the variations in the data set. We continue by explaining how to predict the results for supplementary individuals and variables. Finally, we'll demonstrate how to filter MCA results in order to keep only the most contributing variables.

5.2 Computation

5.2.1 R packages

Several functions from different packages are available in the *R software* for computing multiple correspondence analysis. These functions/packages include:

- *MCA*() function [*FactoMineR* package]
- *dudi.mca()* function [*ade4* package]
- and *epMCA*() [*ExPosition* package]

No matter what function you decide to use, you can easily extract and visualize the MCA results using R functions provided in the *factoextra* R package.

Here, we'll use FactoMineR (for the analysis) and factoextra (for ggplot2-based elegant visualization). To install the two packages, type this:

```
install.packages(c("FactoMineR", "factoextra"))
```

Load the packages:

```
library("FactoMineR")
library("factoextra")
```

5.2.2 Data format

We'll use the demo data sets *poison* available in *FactoMineR* package:

```
data(poison)
head(poison[, 1:7], 3)
```

```
##   Age Time   Sick Sex   Nausea Vomiting Abdominals
## 1   9   22 Sick_y   F Nausea_y  Vomit_n     Abdo_y
## 2   5    0 Sick_n   F Nausea_n  Vomit_n     Abdo_n
## 3   6   16 Sick_y   F Nausea_n  Vomit_y     Abdo_y
```

This data is a result from a survey carried out on children of primary school who suffered from food poisoning. They were asked about their symptoms and about what they ate.

The data contains 55 rows (individuals) and 15 columns (variables). We'll use only some of these individuals (children) and variables to perform the multiple correspondence analysis. The coordinates of the remaining individuals and variables on the factor map will be predicted from the previous MCA results.

In **MCA** terminology, our data contains :

- *Active individuals* (rows 1:55): Individuals that are used in the multiple correspondence analysis.
- *Active variables* (columns 5:15) : Variables that are used in the MCA.
- *Supplementary variables*: They don't participate to the MCA. The coordinates of these variables will be predicted.
  - *Supplementary quantitative variables* (quanti.sup): Columns 1 and 2 corresponding to the columns *age* and *time*, respectively.
  - *Supplementary qualitative variables* (quali.sup: Columns 3 and 4 corresponding to the columns *Sick* and *Sex*, respectively. This factor variables will be used to color individuals by groups.

Subset only active individuals and variables for multiple correspondence analysis:

```
poison.active <- poison[1:55, 5:15]
head(poison.active[, 1:6], 3)
```

```
##     Nausea Vomiting Abdominals   Fever   Diarrhae   Potato
```

```
## 1 Nausea_y  Vomit_n     Abdo_y Fever_y Diarrhea_y Potato_y
## 2 Nausea_n  Vomit_n     Abdo_n Fever_n Diarrhea_n Potato_y
## 3 Nausea_n  Vomit_y     Abdo_y Fever_y Diarrhea_y Potato_y
```

5.2.3 Data summary

The R base function *summary*() can be used to compute the frequency of variable categories. As the data table contains a large number of variables, we'll display only the results for the first 4 variables.

Statistical summaries:

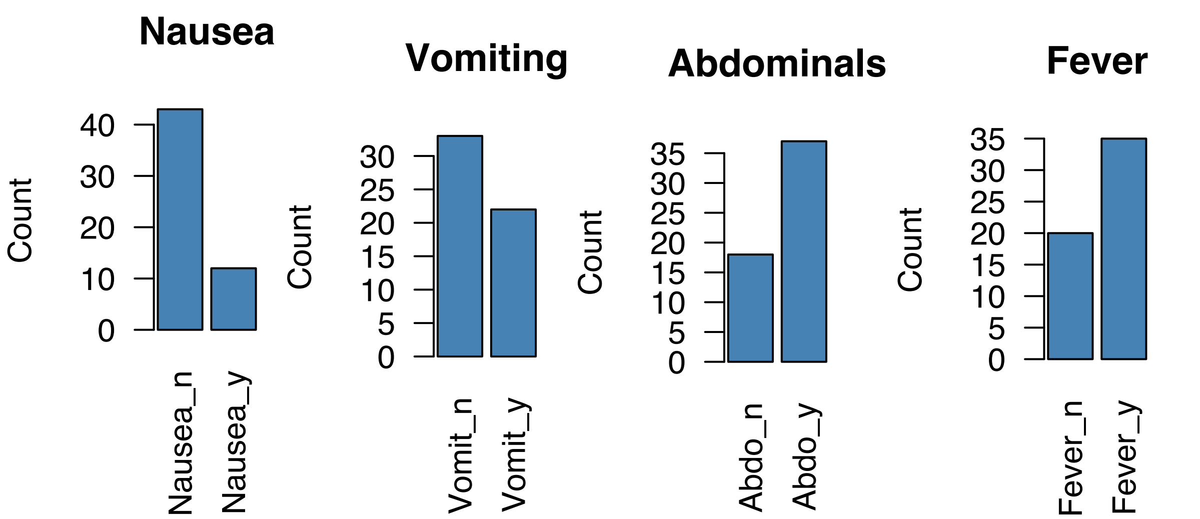
```
# Summary of the 4 first variables
summary(poison.active)[, 1:4]
```

```
##       Nausea      Vomiting   Abdominals     Fever
##  Nausea_n:43   Vomit_n:33   Abdo_n:18   Fever_n:20
##  Nausea_y:12   Vomit_y:22   Abdo_y:37   Fever_y:35
```

The summary() functions return the size of each variable category.

It's also possible to plot the frequency of variable categories. The R code below, plots the first 4 columns:

```
for (i in 1:4) {
  plot(poison.active[,i], main=colnames(poison.active)[i],
       ylab = "Count", col="steelblue", las = 2)
  }
```



The graphs above can be used to identify variable categories with a very low frequency. These types of variables can distort the analysis and should be removed.

5.2.4 R code

The function *MCA*()[*FactoMiner* package] can be used. A simplified format is :

```r
MCA(X, ncp = 5, graph = TRUE)
```

- **X** : a data frame with n rows (individuals) and p columns (categorical variables)
- **ncp** : number of dimensions kept in the final results.
- **graph** : a logical value. If TRUE a graph is displayed.

In the R code below, the MCA is performed only on the active individuals/variables :

```r
res.mca <- MCA(poison.active, graph = FALSE)
```

The output of the *MCA*() function is a list including :

```r
print(res.mca)
```

```
## **Results of the Multiple Correspondence Analysis (MCA)**
## The analysis was performed on 55 individuals, described by 11 variables
## *The results are available in the following objects:
##
##    name                 description
## 1  "$eig"               "eigenvalues"
## 2  "$var"               "results for the variables"
## 3  "$var$coord"         "coord. of the categories"
## 4  "$var$cos2"          "cos2 for the categories"
## 5  "$var$contrib"       "contributions of the categories"
## 6  "$var$v.test"        "v-test for the categories"
## 7  "$ind"               "results for the individuals"
## 8  "$ind$coord"         "coord. for the individuals"
## 9  "$ind$cos2"          "cos2 for the individuals"
## 10 "$ind$contrib"       "contributions of the individuals"
## 11 "$call"              "intermediate results"
## 12 "$call$marge.col"    "weights of columns"
## 13 "$call$marge.li"     "weights of rows"
```

The object that is created using the function *MCA*() contains many information found in many different lists and matrices. These values are described in the next section.

5.3 Visualization and interpretation

We'll use the *factoextra* R package to help in the interpretation and the visualization of the multiple correspondence analysis. No matter what function you decide to use [FactoMiner::MCA(), ade4::dudi.mca()], you can easily extract and visualize the results of multiple correspondence analysis using R functions provided in the *factoextra* R package.

These *factoextra* functions include:

- *get_eigenvalue*(res.mca): Extract the eigenvalues/variances retained by each dimension (axis)
- *fviz_eig*(res.mca): Visualize the eigenvalues/variances

- *get_mca_ind*(res.mca), *get_mca_var*(res.mca): Extract the results for individuals and variables, respectively.
- *fviz_mca_ind*(res.mca), *fviz_mca_var*(res.mca): Visualize the results for individuals and variables, respectively.
- *fviz_mca_biplot*(res.mca): Make a biplot of rows and columns.

In the next sections, we'll illustrate each of these functions.

Note that, the MCA results is interpreted as the results from a simple correspondence analysis (CA). Therefore, it's strongly recommended to read the interpretation of simple CA which has been comprehensively described in the Chapter 4.

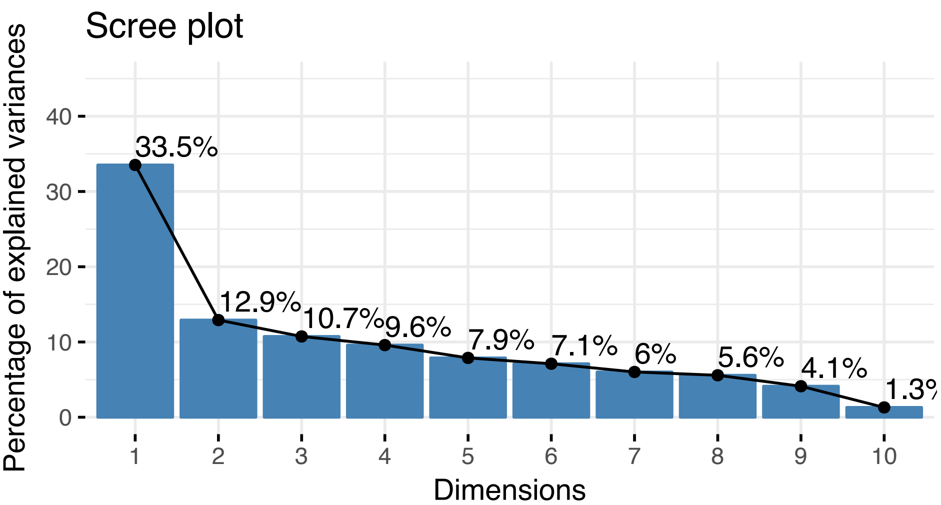
5.3.1 Eigenvalues / Variances

The proportion of variances retained by the different dimensions (axes) can be extracted using the function *get_eigenvalue*() [*factoextra* package] as follow:

```
library("factoextra")
eig.val <- get_eigenvalue(res.mca)
 # head(eig.val)
```

To visualize the percentages of inertia explained by each MCA dimensions, use the function *fviz_eig*() or *fviz_screeplot*() [*factoextra* package]:

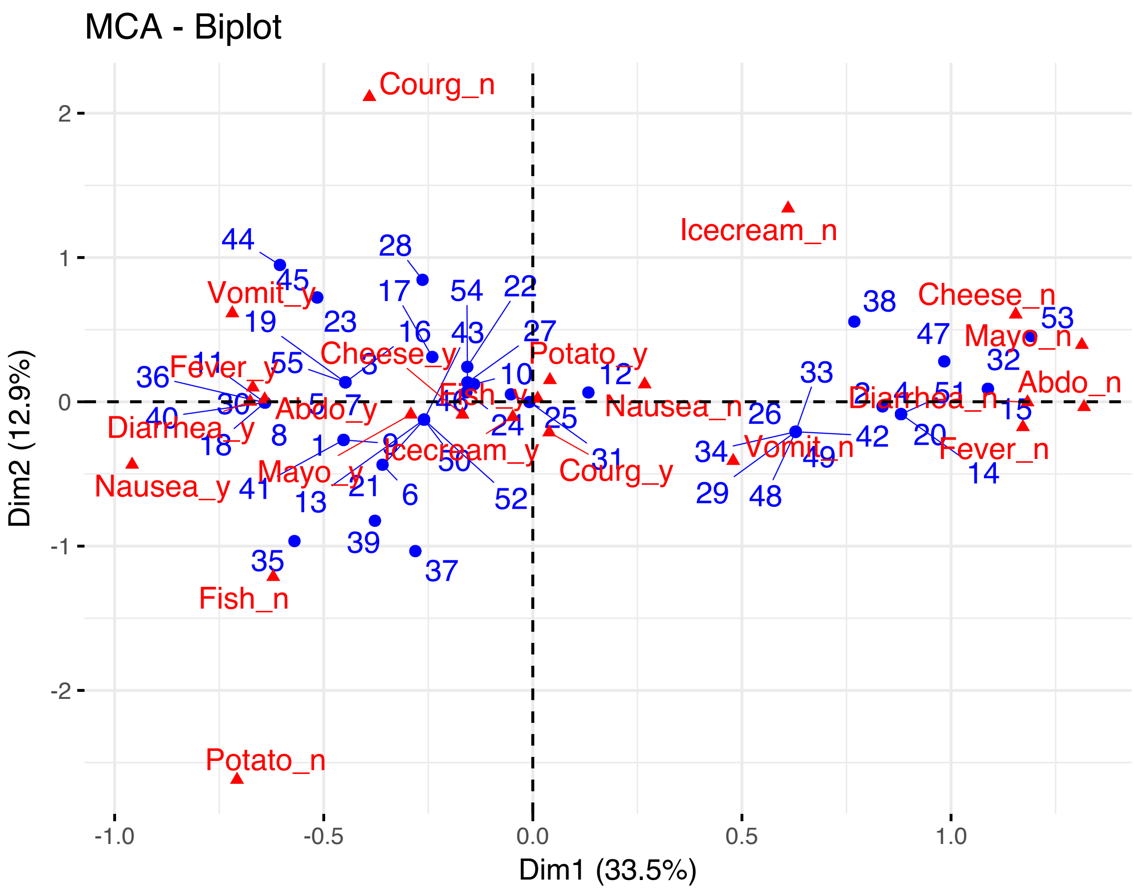
```
fviz_screeplot(res.mca, addlabels = TRUE, ylim = c(0, 45))
```



5.3.2 Biplot

The function *fviz_mca_biplot*() [*factoextra* package] is used to draw the biplot of individuals and variable categories:

```
fviz_mca_biplot(res.mca,
               repel = TRUE, # Avoid text overlapping (slow if many point)
               ggtheme = theme_minimal())
```



The plot above shows a global pattern within the data. Rows (individuals) are represented by blue points and columns (variable categories) by red triangles.

The distance between any row points or column points gives a measure of their similarity (or dissimilarity). Row points with similar profile are closed on the factor map. The same holds true for column points.

5.3.3 Graph of variables

5.3.3.1 Results

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:

```
var <- get_mca_var(res.mca)
var
```

```
## Multiple Correspondence Analysis Results for variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for categories"
## 2 "$cos2"    "Cos2 for categories"
## 3 "$contrib" "contributions of categories"
```

The components of the *get_mca_var*() can be used in the plot of rows as follow:

- *var$coord*: coordinates of variables to create a scatter plot
- *var$cos2*: represents the quality of the representation for variables on the factor map.
- *var$contrib*: contains the contributions (in percentage) of the variables to the definition of the dimensions.

Note that, it's possible to plot variable categories and to color them according to either i) their quality on the factor map (cos2) or ii) their contribution values to the definition of dimensions (contrib).

The different components can be accessed as follow:

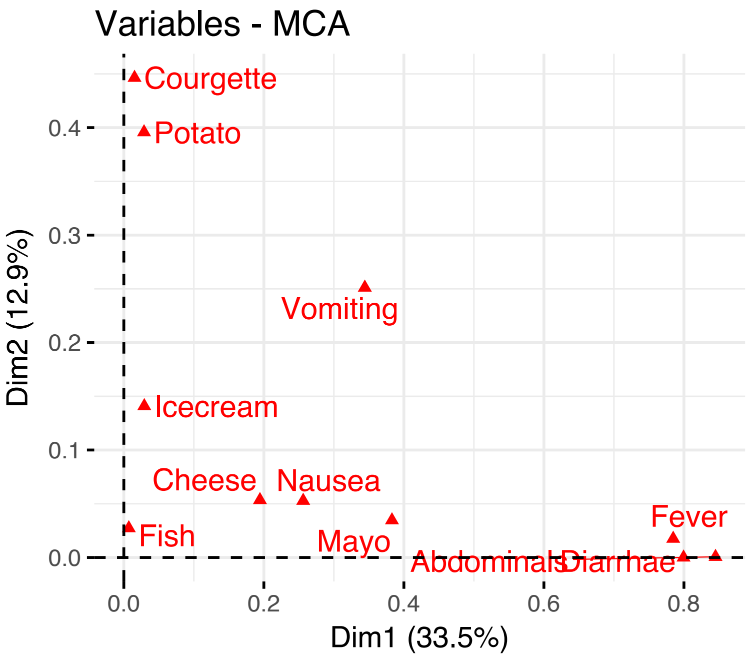
```
# Coordinates
head(var$coord)

# Cos2: quality on the factore map
head(var$cos2)

# Contributions to the principal components
head(var$contrib)
```

In this section, we'll describe how to visualize variable categories only. Next, we'll highlight variable categories according to either i) their quality of representation on the factor map or ii) their contributions to the dimensions.

5.3.3.2 Correlation between variables and principal dimensions

To visualize the correlation between variables and MCA principal dimensions, type this:

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



- The plot above helps to identify variables that are the most correlated with each dimension. The squared correlations between variables and the dimensions are used as coordinates.
- It can be seen that, the variables *Diarrhae, Abdominals and Fever* are the most correlated with dimension 1. Similarly, the variables *Courgette and Potato* are the most correlated with dimension 2.

5.3.3.3 Coordinates of variable categories

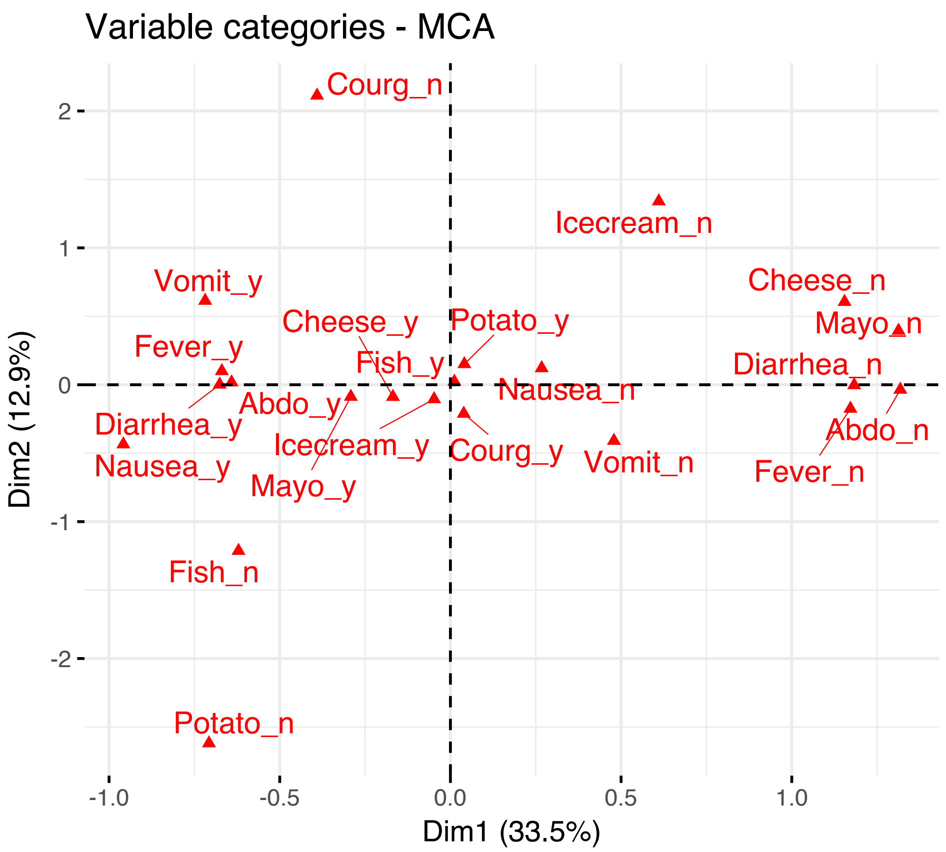
The R code below displays the coordinates of each variable categories in each dimension (1, 2 and 3):

```r
head(round(var$coord, 2), 4)
```

```
##          Dim 1 Dim 2 Dim 3 Dim 4 Dim 5
## Nausea_n  0.27  0.12 -0.27  0.03  0.07
## Nausea_y -0.96 -0.43  0.95 -0.12 -0.26
## Vomit_n   0.48 -0.41  0.08  0.27  0.05
## Vomit_y  -0.72  0.61 -0.13 -0.41 -0.08
```

Use the function *fviz_mca_var*() [in *factoextra*] to visualize only variable categories:

```r
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="black", shape.var = 15,
             repel = TRUE)
```

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. Category points that are away from the origin are well represented on the factor map.

5.3.3.4 Quality of representation of variable categories

The two dimensions 1 and 2 are sufficient to retain 46% 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 follow:

```r
head(var$cos2, 4)
```

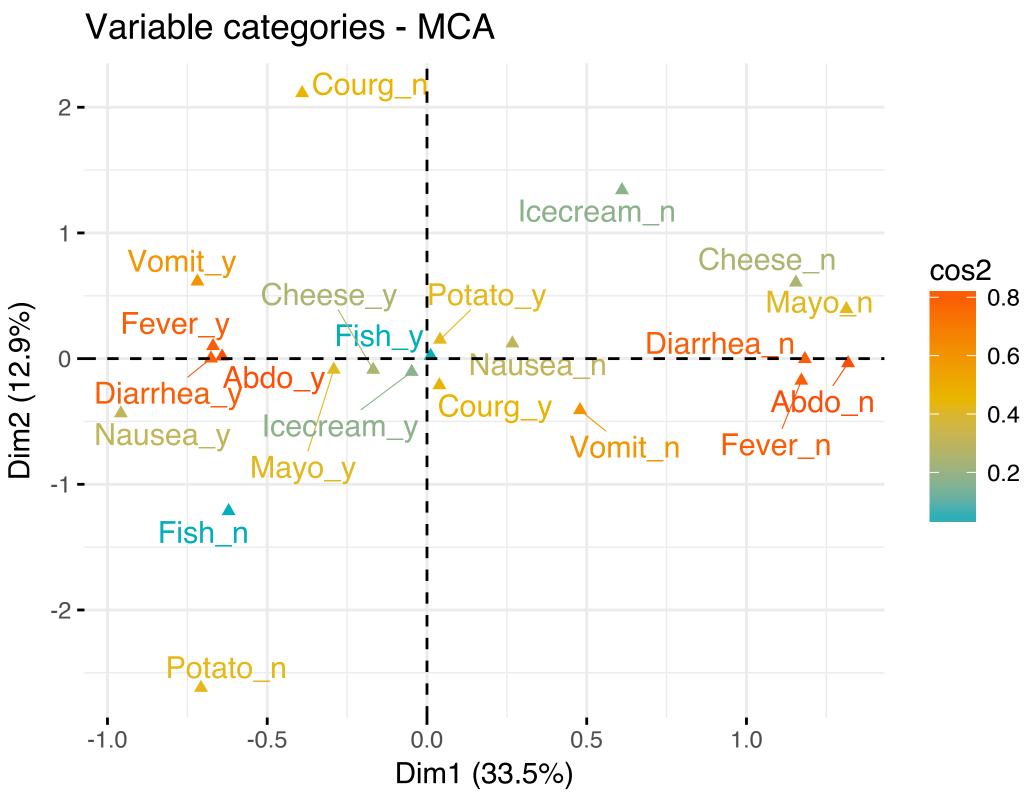
```
##           Dim 1  Dim 2  Dim 3   Dim 4   Dim 5
## Nausea_n 0.256 0.0528 0.2527 0.00408 0.01947
## Nausea_y 0.256 0.0528 0.2527 0.00408 0.01947
## Vomit_n  0.344 0.2512 0.0107 0.11229 0.00413
## Vomit_y  0.344 0.2512 0.0107 0.11229 0.00413
```

If a variable category is well represented by two dimensions, the sum of the cos2 is closed 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*. For instance, *gradient.cols = c("white", "blue", "red")* means that:

- variable categories with low cos2 values will be colored in "white"
- variable categories with mid cos2 values will be colored in "blue"
- variable categories with high cos2 values will be colored in "red"

```r
# 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())
```



Note that, it's also possible to change the transparency of the variable categories according to their cos2 values using the option *alpha.var = "cos2"*. For example, type this:

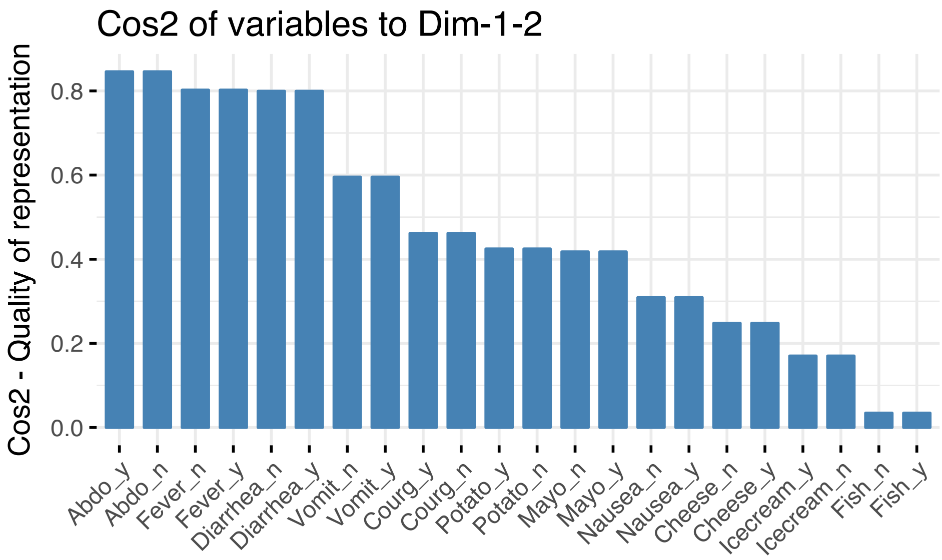
```
# Change the transparency by cos2 values
fviz_mca_var(res.mca, alpha.var="cos2",
             repel = TRUE,
             ggtheme = theme_minimal())
```

You can visualize the cos2 of row categories on all the dimensions using the corrplot package:

```
library("corrplot")
corrplot(var$cos2, is.corr=FALSE)
```

It's also possible to create a bar plot of variable cos2 using the function *fviz_cos2()*[in *factoextra*]:

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



Note that, variable categories *Fish_n, Fish_y, Icecream_n and Icecream_y* are not very well represented by the first two dimensions. This implies that the position of the corresponding points on the scatter plot should be interpreted with some caution. A higher dimensional solution is probably necessary.

5.3.3.5 Contribution of variable categories to the dimensions

The contribution of the variable categories (in %) to the definition of the dimensions can be extracted as follow:

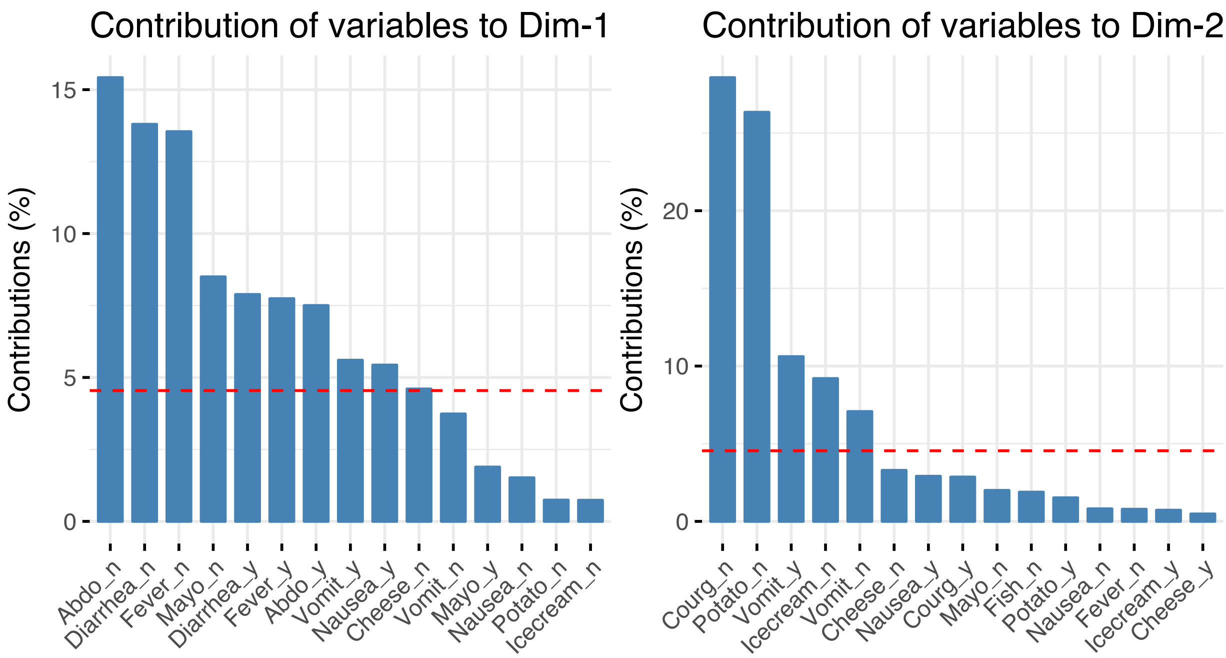
```
head(round(var$contrib,2), 4)
```

```
##           Dim 1 Dim 2 Dim 3 Dim 4 Dim 5
## Nausea_n   1.52  0.81  4.67  0.08  0.49
## Nausea_y   5.43  2.91 16.73  0.30  1.76
## Vomit_n    3.73  7.07  0.36  4.26  0.19
## Vomit_y    5.60 10.61  0.54  6.39  0.29
```

> 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.

The function *fviz_contrib*() [factoextra package] can be used to draw a bar plot of the contribution of variable categories. The R code below shows the top 15 variable categories contributing to the dimensions:

```
# 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)
```



The total contributions to dimension 1 and 2 are obtained as follow:

```
# 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. The calculation of the expected contribution value, under null hypothesis, has been detailed in the principal component analysis[1] chapter.

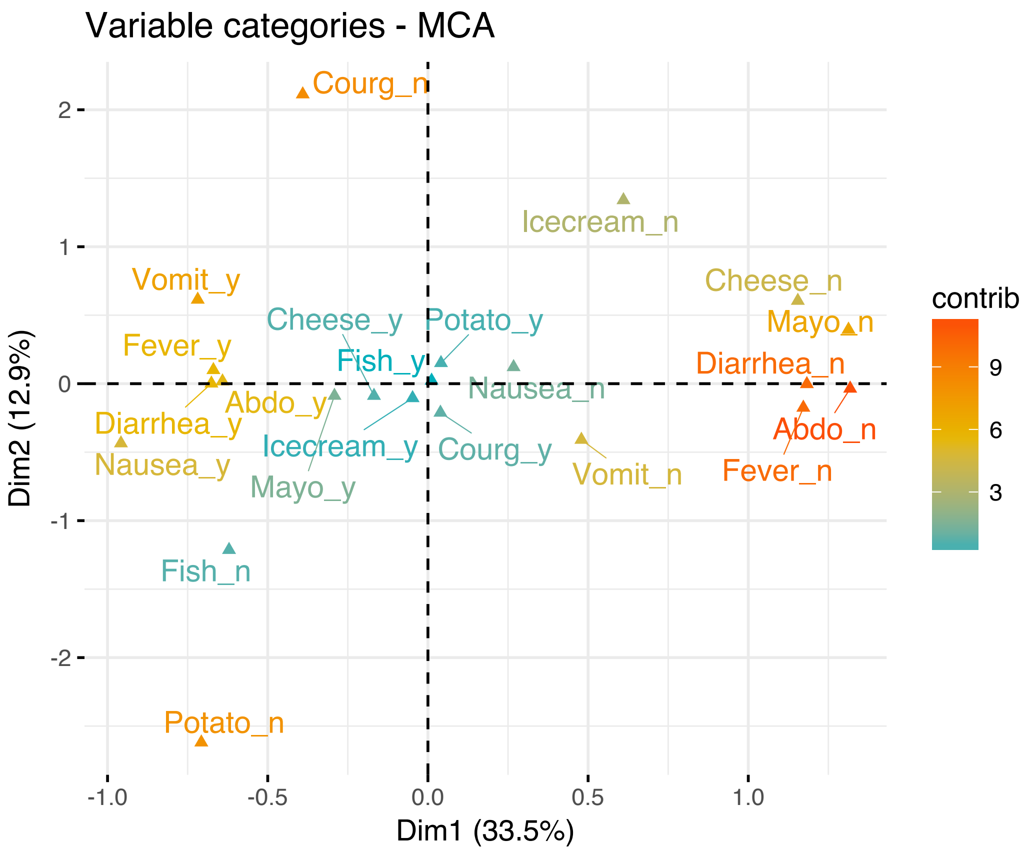
It can be seen that:

- the categories *Abdo_n, Diarrhea_n, Fever_n and Mayo_n* are the most important in the definition of the first dimension.
- The categories *Courg_n, Potato_n, Vomit_y and Icecream_n* contribute the most to the dimension 2

The most important (or, contributing) variable categories can be highlighted on the scatter plot as follow:

[1] `principal-component-analysis`

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



The plot above gives an idea of what pole of the dimensions the categories are actually contributing to.

It is evident that the categories *Abdo_n, Diarrhea_n, Fever_n and Mayo_n* have an important contribution to the positive pole of the first dimension, while the categories *Fever_y and Diarrhea_y* have a major contribution to the negative pole of the first dimension; etc, ...

Note that, it's also possible to control the transparency of variable categories according to their contribution values using the option *alpha.var = "contrib"*. For example, type this:

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

5.3.4 Graph of individuals

5.3.4.1 Results

The function **get_mca_ind()**[in *factoextra*] is used to extract the results for individuals. This function returns a list containing the coordinates, the cos2 and the contributions of individuals:

```
ind <- get_mca_ind(res.mca)
ind
```

```
## Multiple Correspondence Analysis Results for individuals
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the individuals"
## 2 "$cos2"    "Cos2 for the individuals"
## 3 "$contrib" "contributions of the individuals"
```

> The result for *individuals* gives the same information as described for variable categories. For this reason, I'll just displayed the result for individuals in this section without commenting.

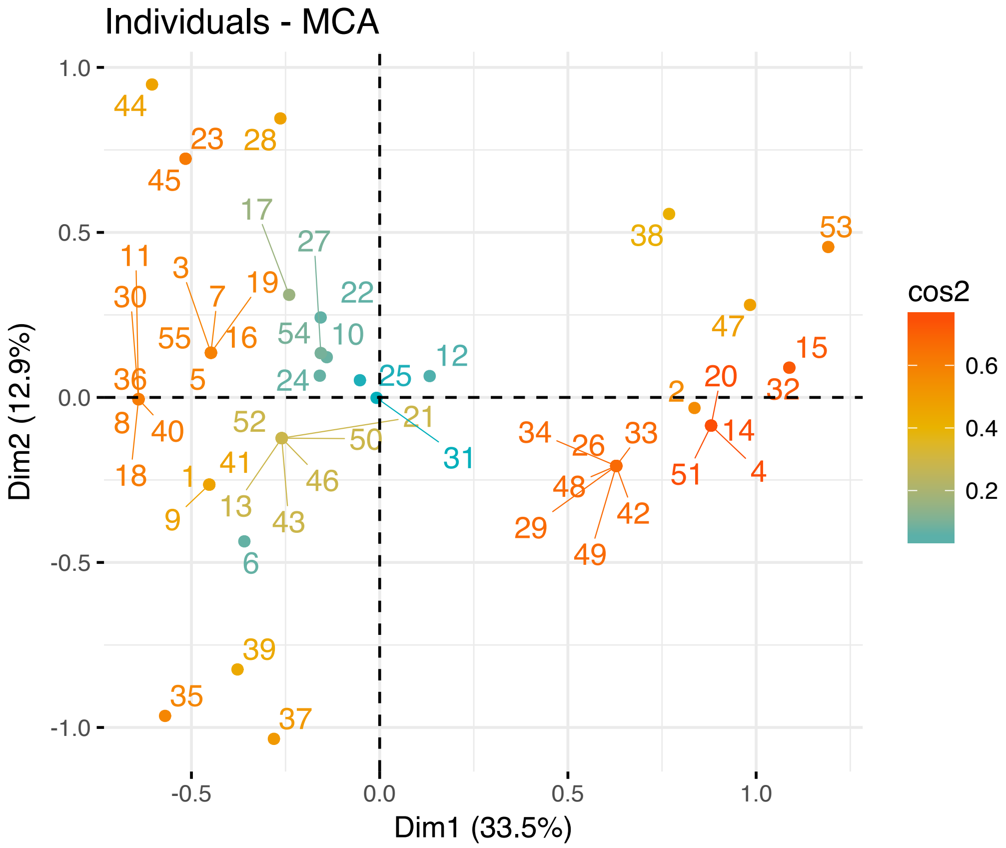
To get access to the different components, use this:

```
# Coordinates of column points
head(ind$coord)

# Quality of representation
head(ind$cos2)

# Contributions
head(ind$contrib)
```

5.3.4.2 Plots: quality and contribution

The function *fviz_mca_ind*() [in *factoextra*] is used to visualize only individuals. 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())
```



The R code below creates a bar plots of individuals cos2 and contributions:

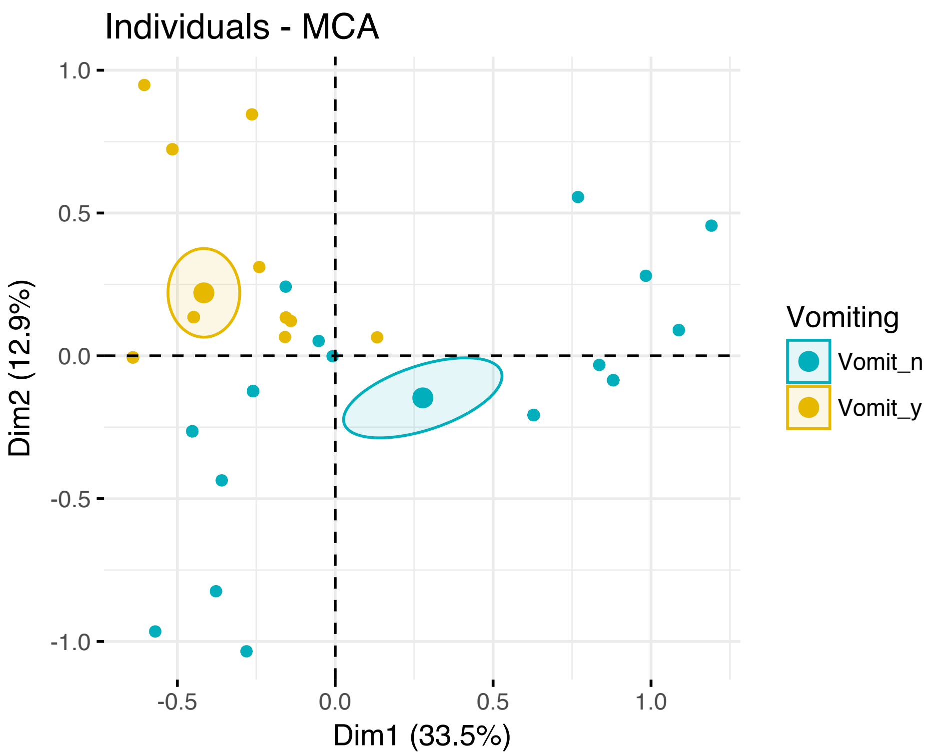
```
# Cos2 of individuals
fviz_cos2(res.mca, choice = "ind", axes = 1:2, top = 20)

# Contribution of individuals to the dimensions
fviz_contrib(res.mca, choice = "ind", axes = 1:2, top = 20)
```

5.3.5 Color individuals by groups

Note that, it's possible to color the individuals using any of the qualitative variables in the initial data table (poison)

The R code below colors the individuals by groups using the levels of the variable *Vomiting*. The argument *habillage* is used to specify the factor variable for coloring the individuals by groups. A concentration ellipse can be also added around each group using the argument *addEllipses = TRUE*. If you want a confidence ellipse around the mean point of categories, use *ellipse.type = "confidence"* The argument *palette* is used to change group colors.

```
fviz_mca_ind(res.mca,
             label = "none", # hide individual labels
             habillage = "Vomiting", # color by groups
             palette = c("#00AFBB", "#E7B800"),
             addEllipses = TRUE, ellipse.type = "confidence",
             ggtheme = theme_minimal())
```

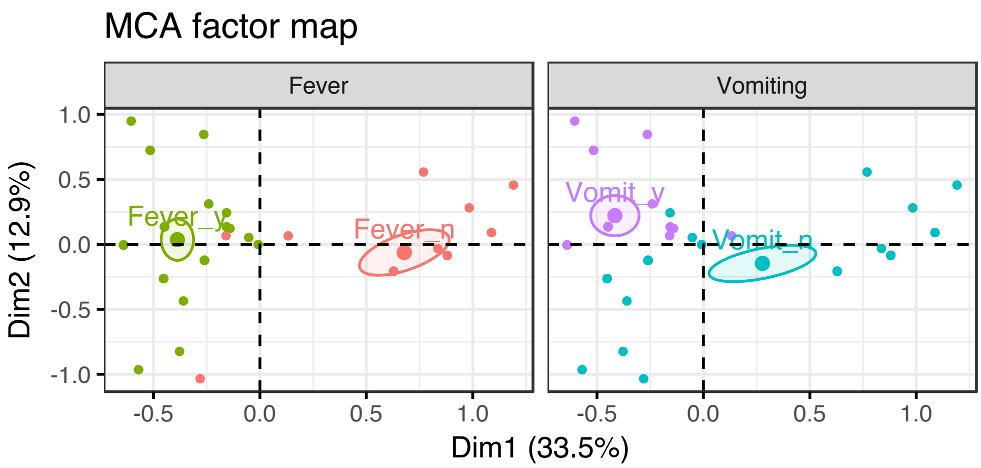


Note that, to specify the value of the argument *habillage*, it's also possible to use the index of the column as follow (*habillage = 2*). Additionally, you can provide an external grouping variable as follow: *habillage = poison$Vomiting*. For example:

```
# habillage = index of the column to be used as grouping variable
fviz_mca_ind(res.mca, habillage = 2, addEllipses = TRUE)

# habillage = external grouping variable
fviz_mca_ind(res.mca, habillage = poison$Vomiting, addEllipses = TRUE)
```

If you want to color individuals using multiple categorical variables at the same time, use the function fviz_ellipses() [in *factoextra*] as follow:

```
fviz_ellipses(res.mca, c("Vomiting", "Fever"),
              geom = "point")
```



Alternatively, you can specify categorical variable indices:

```
fviz_ellipses(res.mca, 1:4, geom = "point")
```

5.3.6 Dimension description

The function **dimdesc()** [in FactoMineR] can be used to identify the most correlated variables with a given dimension:

```
res.desc <- dimdesc(res.mca, axes = c(1,2))
# Description of dimension 1
res.desc[[1]]

# Description of dimension 2
res.desc[[2]]
```

5.4 Supplementary elements

5.4.1 Definition and types

As described above (section 5.2.2), the data set *poison* contains:

- *supplementary continuous variables* (quanti.sup = 1:2, columns 1 and 2 corresponding to the columns *age* and *time*, respectively)
- *supplementary qualitative variables* (quali.sup = 3:4, corresponding to the columns *Sick* and *Sex*, respectively). This factor variables are used to color individuals by groups

The data doesn't contain *supplementary individuals*. However, for demonstration, we'll use the individuals 53:55 as supplementary individuals.

Supplementary variables and individuals are not used for the determination of the principal dimensions. Their coordinates are predicted using only the information provided by the performed multiple correspondence analysis on active variables/individuals.

5.4.2 Specification in MCA

To specify supplementary individuals and variables, the function *MCA*() can be used as follow :

```
MCA(X,  ind.sup = NULL, quanti.sup = NULL, quali.sup=NULL,
    graph = TRUE, axes = c(1,2))
```

- **X** : a data frame. Rows are individuals and columns are variables.
- **ind.sup** : a numeric vector specifying the indexes of the supplementary individuals.

- **quanti.sup**, **quali.sup** : a numeric vector specifying, respectively, the indexes of the quantitative and qualitative variables.
- **graph** : a logical value. If TRUE a graph is displayed.
- **axes** : a vector of length 2 specifying the components to be plotted.

For example, type this:

```
res.mca <- MCA(poison, ind.sup = 53:55,
               quanti.sup = 1:2, quali.sup = 3:4,  graph=FALSE)
```

5.4.3 Results

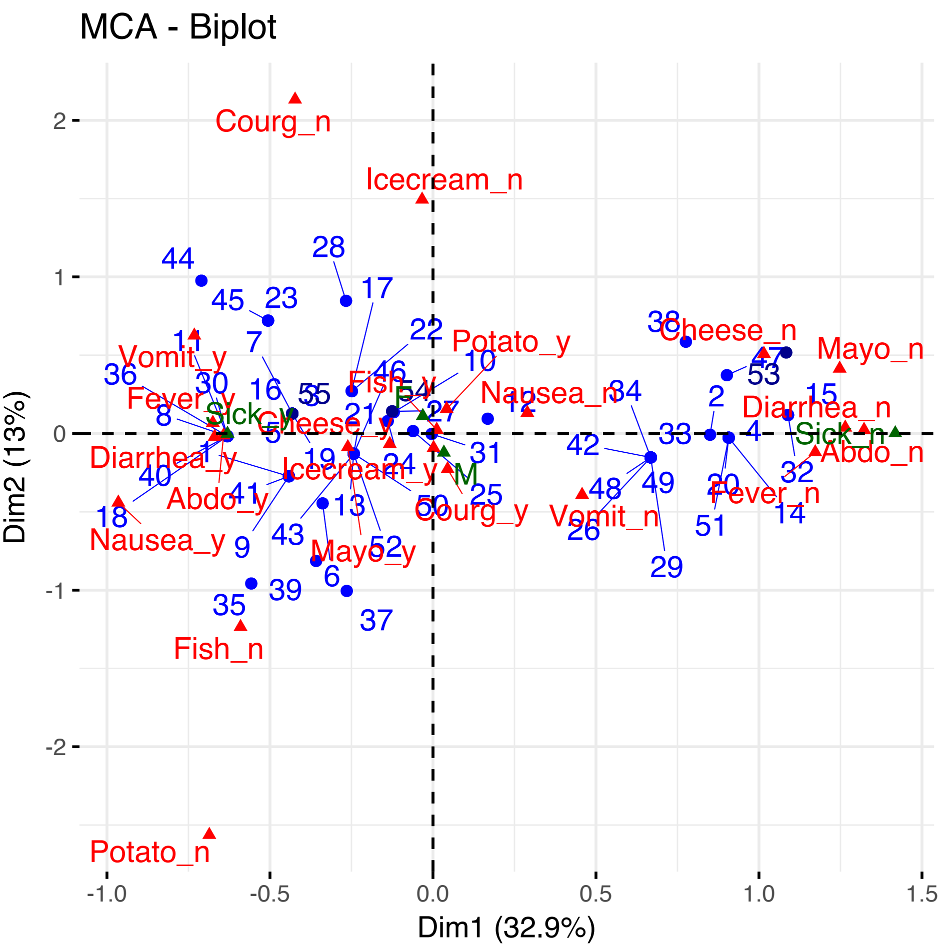
The predicted results for supplementary individuals/variables can be extracted as follow:

```
# Supplementary qualitative variable categories
res.mca$quali.sup

# Supplementary quantitative variables
res.mca$quanti

# Supplementary individuals
res.mca$ind.sup
```

5.4.4 Plots

To make a biplot of individuals and variable categories, type this:

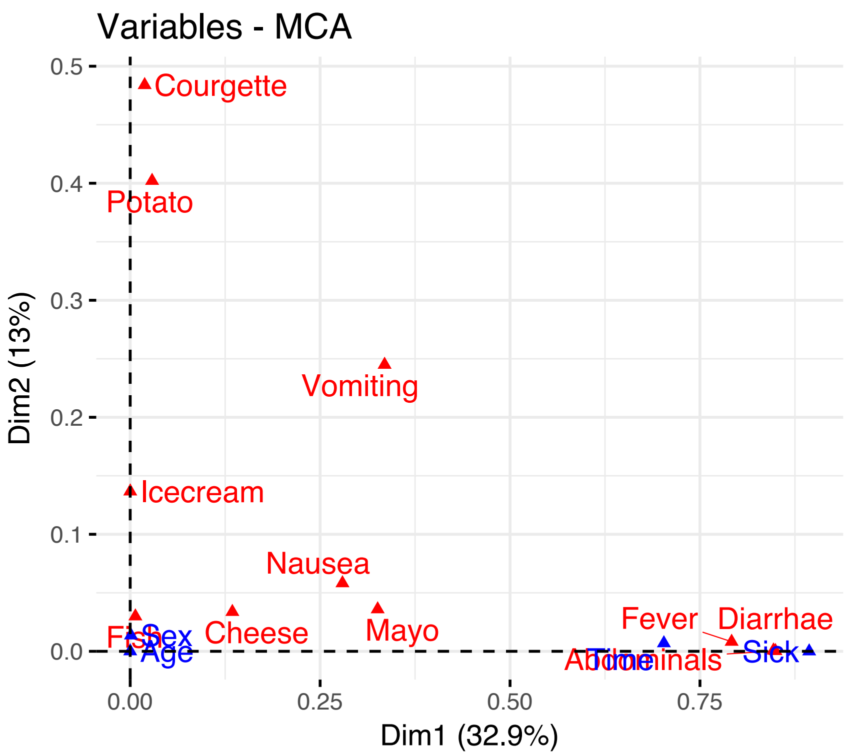
```
# Biplot of individuals and variable categories
fviz_mca_biplot(res.mca, repel = TRUE,
                ggtheme = theme_minimal())
```



- Active individuals are in blue
- Supplementary individuals are in darkblue
- Active variable categories are in red
- Supplementary variable categories are in darkgreen

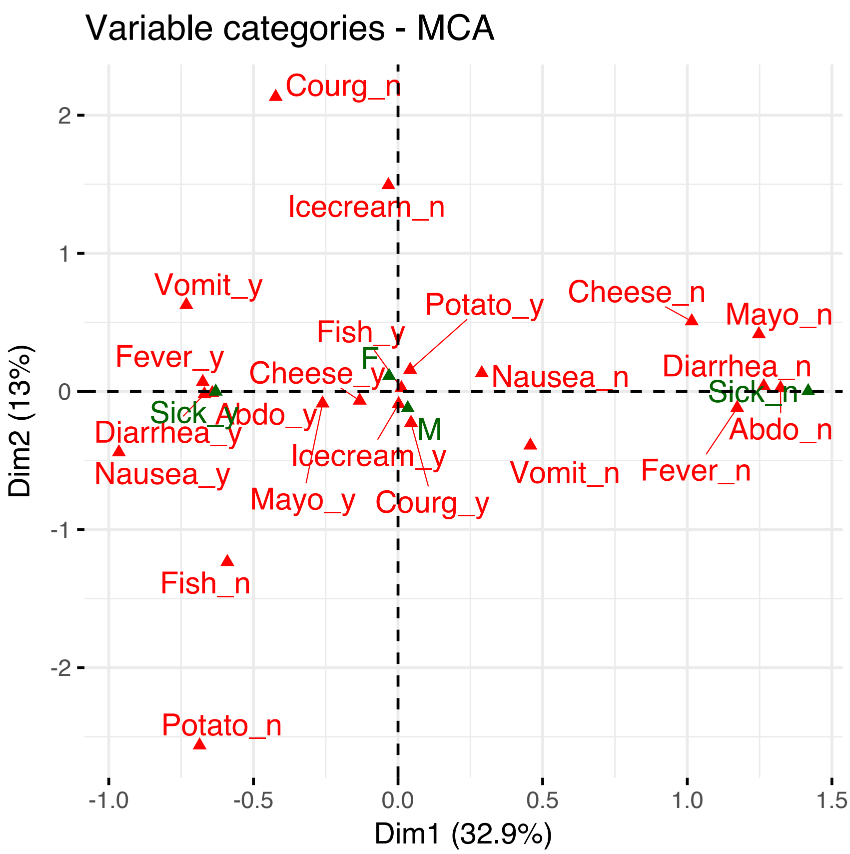
If you want to highlight the correlation between variables (active & supplementary) and dimensions, use the function *fviz_mca_var*() with the argument choice = "mca.cor":

```
fviz_mca_var(res.mca, choice = "mca.cor",
            repel = TRUE)
```



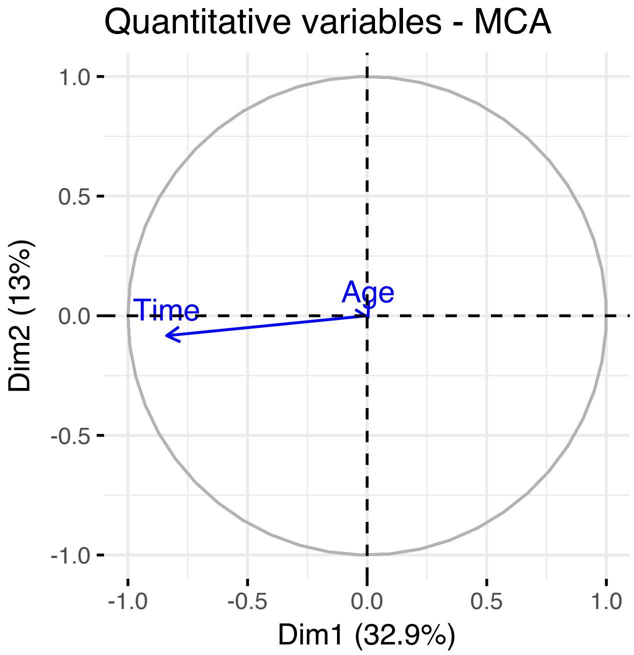
The R code below plots qualitative variable categories (active & supplementary variables):

```
fviz_mca_var(res.mca, repel = TRUE,
             ggtheme= theme_minimal())
```



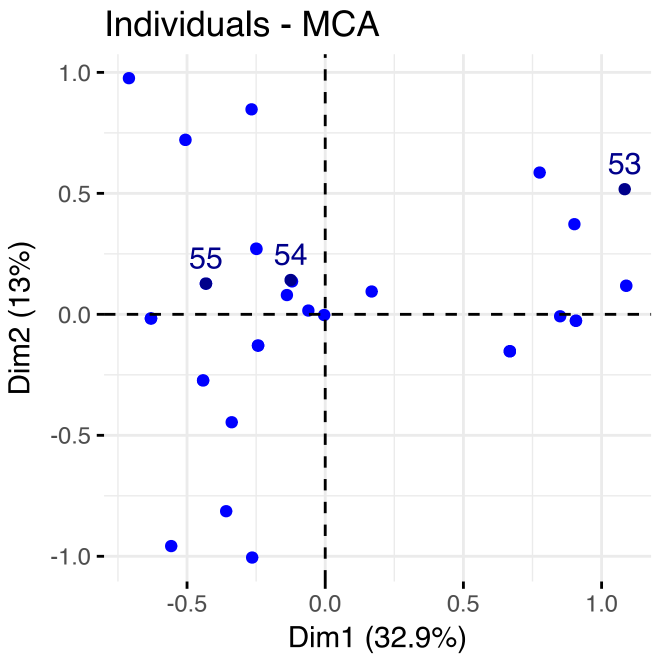
For supplementary quantitative variables, type this:

```r
fviz_mca_var(res.mca, choice = "quanti.sup",
             ggtheme = theme_minimal())
```



To visualize supplementary individuals, type this:

```r
fviz_mca_ind(res.mca,
             label = "ind.sup", #Show the label of ind.sup only
             ggtheme = theme_minimal())
```



5.5 Filtering results

If you have many individuals/variable categories, it's possible to visualize only some of them using the arguments *select.ind* and *select.var*.

select.ind, select.var: a selection of individuals/variable categories to be drawn. Allowed values are *NULL* or a *list* containing the arguments name, cos2 or contrib:

- *name*: is a character vector containing individuals/variable category names to be plotted
- *cos2*: if cos2 is in [0, 1], ex: 0.6, then individuals/variable categories with a cos2 > 0.6 are plotted
- *if cos2 > 1*, ex: 5, then the top 5 active individuals/variable categories and top 5 supplementary columns/rows with the highest cos2 are plotted
- *contrib*: if contrib > 1, ex: 5, then the top 5 individuals/variable categories with the highest contributions are plotted

```
# Visualize variable categories with cos2 >= 0.4
fviz_mca_var(res.mca, select.var = list(cos2 = 0.4))

# Top 10 active variables with the highest cos2
fviz_mca_var(res.mca, select.var= list(cos2 = 10))

# Select by names
name <- list(name = c("Fever_n", "Abdo_y", "Diarrhea_n",
                      "Fever_Y", "Vomit_y", "Vomit_n"))
fviz_mca_var(res.mca, select.var = name)

# top 5 contributing individuals and variable categories
fviz_mca_biplot(res.mca, select.ind = list(contrib = 5),
               select.var = list(contrib = 5),
               ggtheme = theme_minimal())
```

When the selection is done according to the contribution values, supplementary individuals/variable categories are not shown because they don't contribute to the construction of the axes.

5.6 Exporting results

5.6.1 Export plots to PDF/PNG files

Two steps:

1) Create the plot of interest as an R object:

```
# Scree plot
scree.plot <- fviz_eig(res.mca)
```

```r
# Biplot of row and column variables
biplot.mca <- fviz_mca_biplot(res.mca)
```

2) Export the plots into a single pdf file as follow (one plot per page):

```r
library(ggpubr)
ggexport(plotlist = list(scree.plot, biplot.mca),
         filename = "MCA.pdf")
```

More options at: Chapter 3 (section: Exporting results).

5.6.2 Export results to txt/csv files

Easy to use R function: **write.infile**() [in *FactoMineR*] package.

```r
# Export into a TXT file
write.infile(res.mca, "mca.txt", sep = "\t")

# Export into a CSV file
write.infile(res.mca, "mca.csv", sep = ";")
```

5.7 Summary

In conclusion, we described how to perform and interpret multiple correspondence analysis (CA). We computed MCA using the *MCA*() function [FactoMineR package]. Next, we used the *factoextra* R package to produce ggplot2-based visualization of the CA results.

Other functions [packages] to compute MCA in R, include:

1) Using **dudi.acm**() [ade4]

```r
library("ade4")
res.mca <- dudi.acm(poison.active, scannf = FALSE, nf = 5)
```

4) Using **epMCA**() [ExPosition]

```r
library("ExPosition")
res.mca <- epMCA(poison.active, graph = FALSE, correction = "bg")
```

No matter what functions you decide to use, in the list above, the factoextra package can handle the output.

```r
fviz_eig(res.mca)      # Scree plot

fviz_mca_biplot(res.mca) # Biplot of rows and columns
```

5.8 Further reading

For the mathematical background behind MCA, refer to the following video courses, articles and books:

- Correspondence Analysis Course Using FactoMineR (Video courses). `https://goo.gl/Hhh6hC`
- Exploratory Multivariate Analysis by Example Using R (book) (Husson et al., 2017b).
- Principal component analysis (article) (Abdi and Williams, 2010). `https://goo.gl/1Vtwq1`.
- Correspondence analysis basics (blog post). `https://goo.gl/Xyk8KT`.

Part III

Advanced Methods

Chapter 6

Factor Analysis of Mixed Data

6.1 Introduction

Factor analysis of mixed data (**FAMD**) is a principal component method dedicated to analyze a data set containing both quantitative and qualitative variables (Pagès, 2004). It makes it possible to analyze the similarity between individuals by taking into account a mixed types of variables. Additionally, one can explore the association between all variables, both quantitative and qualitative variables.

Roughly speaking, the FAMD algorithm can be seen as a mixed between principal component analysis (PCA) (Chapter 3) and multiple correspondence analysis (MCA) (Chapter 5). In other words, it acts as PCA quantitative variables and as MCA for qualitative variables.

Quantitative and qualitative variables are normalized during the analysis in order to balance the influence of each set of variables.

In the current chapter, we demonstrate how to compute and visualize factor analysis of mixed data using *FactoMineR* (for the analysis) and *factoextra* (for data visualization) R packages.

6.2 Computation

6.2.1 R packages

Install required packages as follow:

```
install.packages(c("FactoMineR", "factoextra"))
```

Load the packages:

```
library("FactoMineR")
library("factoextra")
```

6.2.2 Data format

We'll use a subset of the *wine* data set available in *FactoMineR* package:

```
library("FactoMineR")
data(wine)
df <- wine[,c(1,2, 16, 22, 29, 28, 30,31)]
head(df[, 1:7], 4)
```

```
##           Label Soil Plante Acidity Harmony Intensity Overall.quality
## 2EL      Saumur Env1   2.00    2.11    3.14      2.86            3.39
## 1CHA     Saumur Env1   2.00    2.11    2.96      2.89            3.21
## 1FON Bourgueuil Env1   1.75    2.18    3.14      3.07            3.54
## 1VAU     Chinon Env2   2.30    3.18    2.04      2.46            2.46
```

To see the structure of the data, type this:

```
str(df)
```

The data contains 21 rows (wines, *individuals*) and 8 columns (*variables*):

- The first two columns are factors (*categorical variables*): *label* (Saumur, Bourgueil or Chinon) and *soil* (Reference, Env1, Env2 or Env4).
- The remaining columns are numeric (*continuous variables*).

The goal of this study is to analyze the characteristics of the wines.

6.2.3 R code

The function *FAMD*() [*FactoMiner* package] can be used to compute FAMD. A simplified format is :

```
FAMD (base, ncp = 5, sup.var = NULL, ind.sup = NULL, graph = TRUE)
```

- **base** : a data frame with n rows (individuals) and p columns (variables).
- **ncp**: the number of dimensions kept in the results (by default 5)
- **sup.var**: a vector indicating the indexes of the supplementary variables.
- **ind.sup**: a vector indicating the indexes of the supplementary individuals.
- **graph** : a logical value. If TRUE a graph is displayed.

To compute FAMD, type this:

```
library(FactoMineR)
res.famd <- FAMD(df, graph = FALSE)
```

The output of the *FAMD*() function is a list including :

```
print(res.famd)
```

```
## *The results are available in the following objects:
##
##   name          description
```

```
## 1 "$eig"        "eigenvalues and inertia"
## 2 "$var"        "Results for the variables"
## 3 "$ind"        "results for the individuals"
## 4 "$quali.var"  "Results for the qualitative variables"
## 5 "$quanti.var" "Results for the quantitative variables"
```

6.3 Visualization and interpretation

We'll use the following *factoextra* functions:

- *get_eigenvalue*(res.famd): Extract the eigenvalues/variances retained by each dimension (axis).
- *fviz_eig*(res.famd): Visualize the eigenvalues/variances.
- *get_famd_ind*(res.famd): Extract the results for individuals.
- *get_famd_var*(res.famd): Extract the results for quantitative and qualitative variables.
- *fviz_famd_ind*(res.famd), *fviz_famd_var*(res.famd): Visualize the results for individuals and variables, respectively.

In the next sections, we'll illustrate each of these functions.

To help in the interpretation of FAMD, we highly recommend to read the interpretation of principal component analysis (Chapter 3) and multiple correspondence analysis (Chapter 5). Many of the graphs presented here have been already described in our previous chapters.

6.3.1 Eigenvalues / Variances

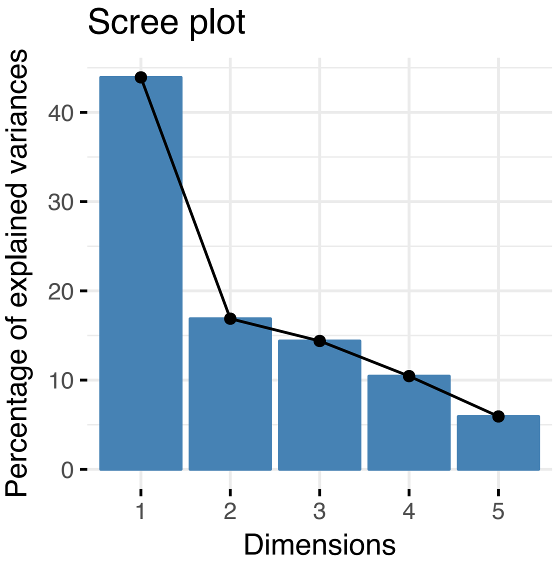
The proportion of variances retained by the different dimensions (axes) can be extracted using the function *get_eigenvalue*() [*factoextra* package] as follow:

```
library("factoextra")
eig.val <- get_eigenvalue(res.famd)
head(eig.val)
```

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1      4.832            43.92                        43.9
## Dim.2      1.857            16.88                        60.8
## Dim.3      1.582            14.39                        75.2
## Dim.4      1.149            10.45                        85.6
## Dim.5      0.652             5.93                        91.6
```

The function *fviz_eig*() or *fviz_screeplot*() [*factoextra* package] can be used to draw the scree plot (the percentages of inertia explained by each FAMD dimensions):

```
fviz_screeplot(res.famd)
```



6.3.2 Graph of variables

6.3.2.1 All variables

The function *get_mfa_var*()[in *factoextra*] is used to extract the results for variables. By default, this function returns a list containing the coordinates, the cos2 and the contribution of all variables:

```
var <- get_famd_var(res.famd)
var
```

```
## FAMD results for variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates"
## 2 "$cos2"    "Cos2, quality of representation"
## 3 "$contrib" "Contributions"
```

The different components can be accessed as follow:

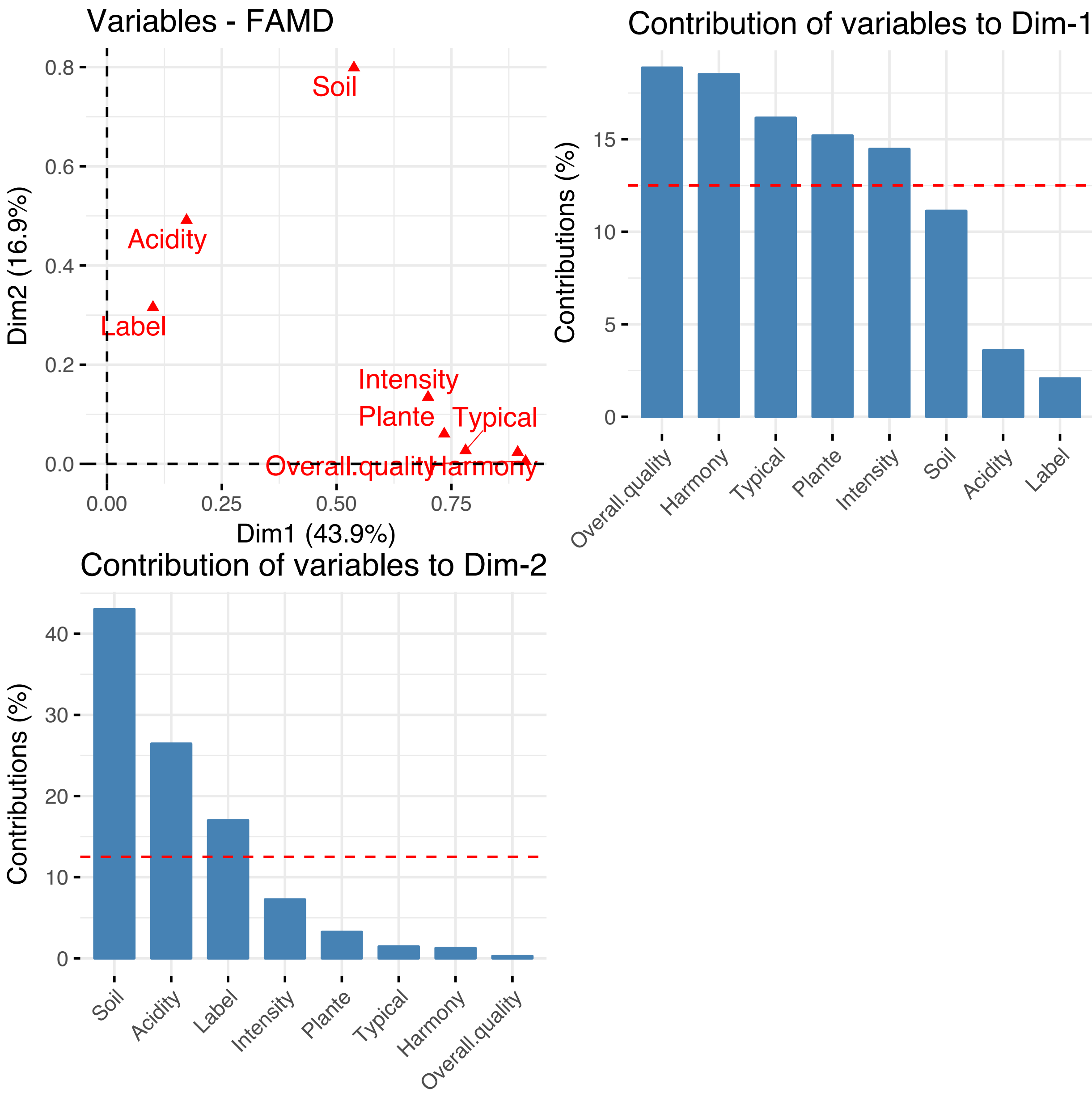
```
# Coordinates of variables
head(var$coord)

# Cos2: quality of representation on the factore map
head(var$cos2)

# Contributions to the  dimensions
head(var$contrib)
```

The following figure shows the correlation between variables - both quantitative and qualitative variables - and the principal dimensions, as well as, the contribution of variables to the dimensions 1 and 2. The following functions [in the *factoextra* package] are used:

- *fviz_famd_var*() to plot both quantitative and qualitative variables

- *fviz_contrib*() to visualize the contribution of variables to the principal dimensions

```
# Plot of variables
fviz_famd_var(res.famd, repel = TRUE)

# Contribution to the first dimension
fviz_contrib(res.famd, "var", axes = 1)

# Contribution to the second dimension
fviz_contrib(res.famd, "var", axes = 2)
```



The red dashed line on the graph above indicates the expected average value, If the contributions were uniform. Read more in chapter (Chapter 3).

From the plots above, it can be seen that:

- variables that contribute the most to the first dimension are: Overall.quality and Harmony.

- variables that contribute the most to the second dimension are: Soil and Acidity.

6.3.2.2 Quantitative variables

To extract the results for quantitative variables, type this:

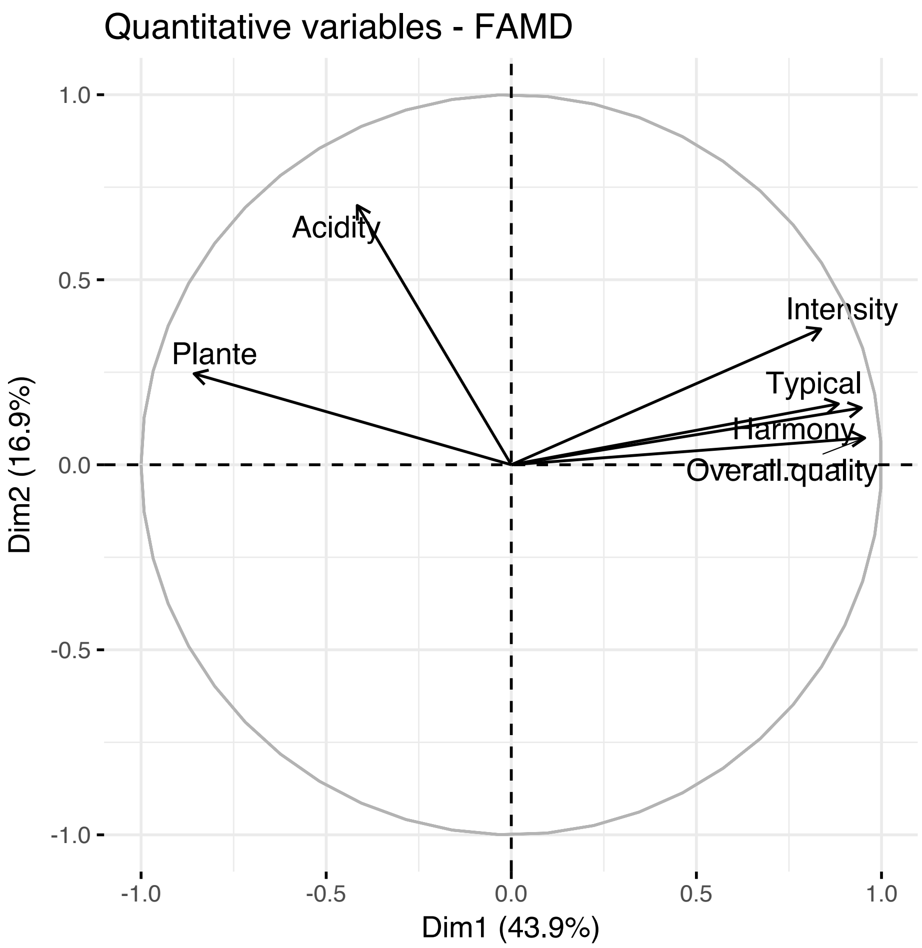
```
quanti.var <- get_famd_var(res.famd, "quanti.var")
quanti.var
```

```
## FAMD results for quantitative variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates"
## 2 "$cos2"    "Cos2, quality of representation"
## 3 "$contrib" "Contributions"
```

In this section, we'll describe how to visualize quantitative variables. Additionally, we'll show how to highlight variables according to either i) their quality of representation on the factor map or ii) their contributions to the dimensions.

The R code below plots quantitative variables. We use repel = TRUE, to avoid text overlapping.

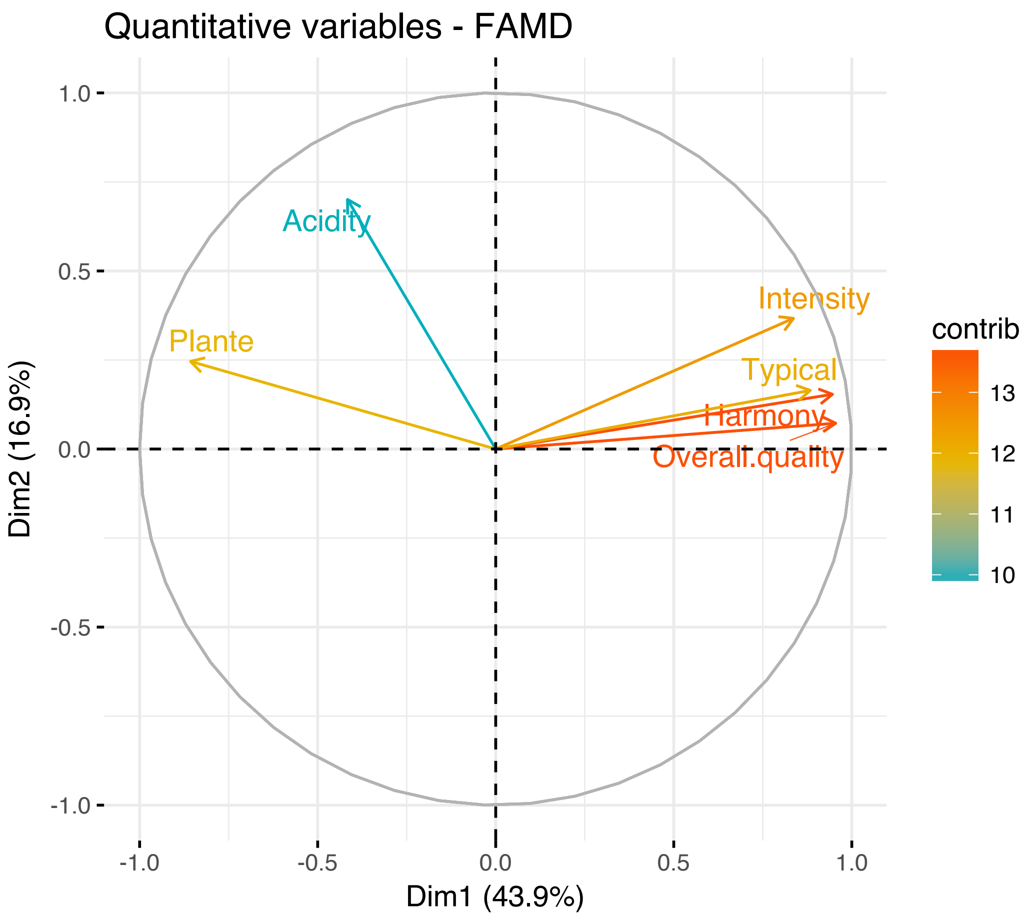
```
fviz_famd_var(res.famd, "quanti.var", repel = TRUE,
              col.var = "black")
```



Briefly, the graph of variables (correlation circle) shows the relationship between variables, the quality of the representation of variables, as well as, the correlation between variables and the dimensions. Read more at PCA (Chapter 3), MCA (Chapter 5) and MFA (Chapter 7).

The most contributing quantitative variables can be highlighted on the scatter plot using the argument col.var = "contrib". This produces a gradient colors, which can be customized using the argument *gradient.cols*.

```
fviz_famd_var(res.famd, "quanti.var", col.var = "contrib",
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE)
```



Similarly, you can highlight quantitative variables using their cos2 values representing the quality of representation on the factor map. If a variable is well represented by two dimensions, the sum of the cos2 is closed to one. For some of the items, more than 2 dimensions might be required to perfectly represent the data.

```
# Color by cos2 values: quality on the factor map
fviz_famd_var(res.famd, "quanti.var", col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

6.3.2.3 Graph of qualitative variables

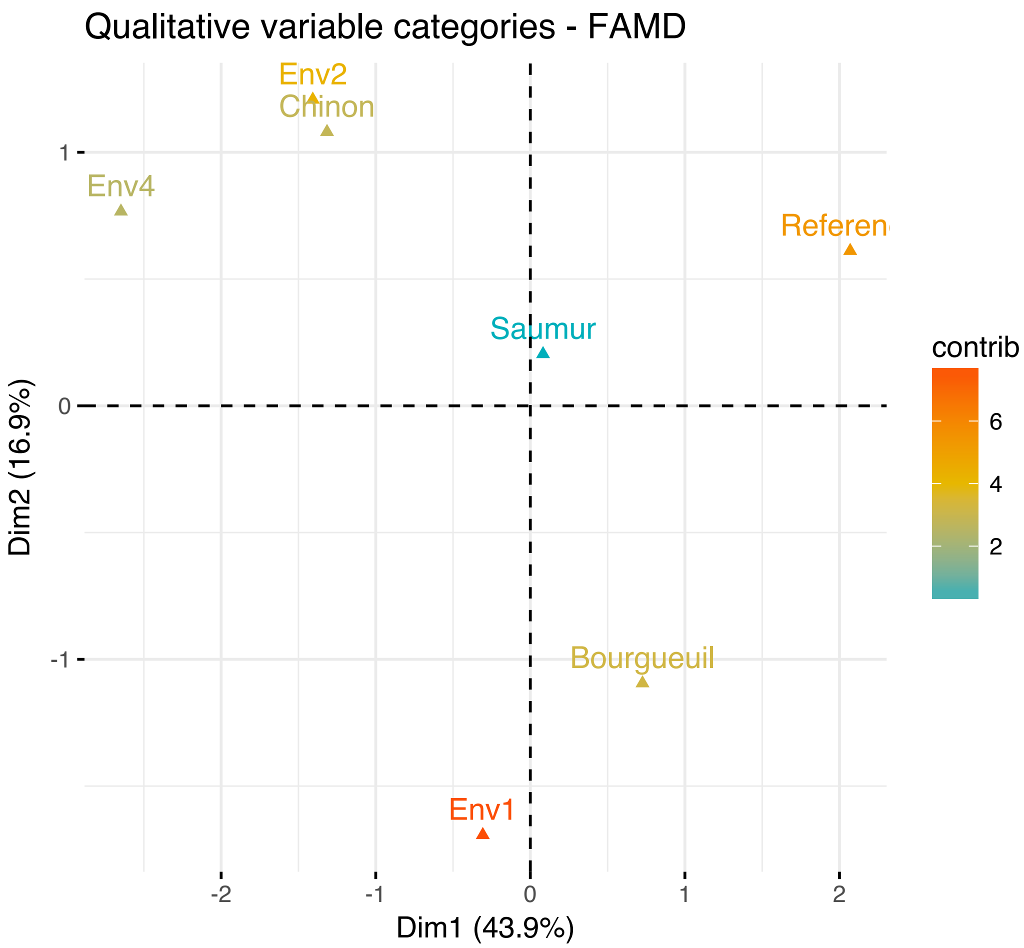
Like quantitative variables, the results for qualitative variables can be extracted as follow:

```
quali.var <- get_famd_var(res.famd, "quali.var")
quali.var
```

```
## FAMD results for qualitative variable categories
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates"
## 2 "$cos2"    "Cos2, quality of representation"
## 3 "$contrib" "Contributions"
```

To visualize qualitative variables, type this:

```
fviz_famd_var(res.famd, "quali.var", col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
             )
```



The plot above shows the categories of the categorical variables.

6.3.3 Graph of individuals

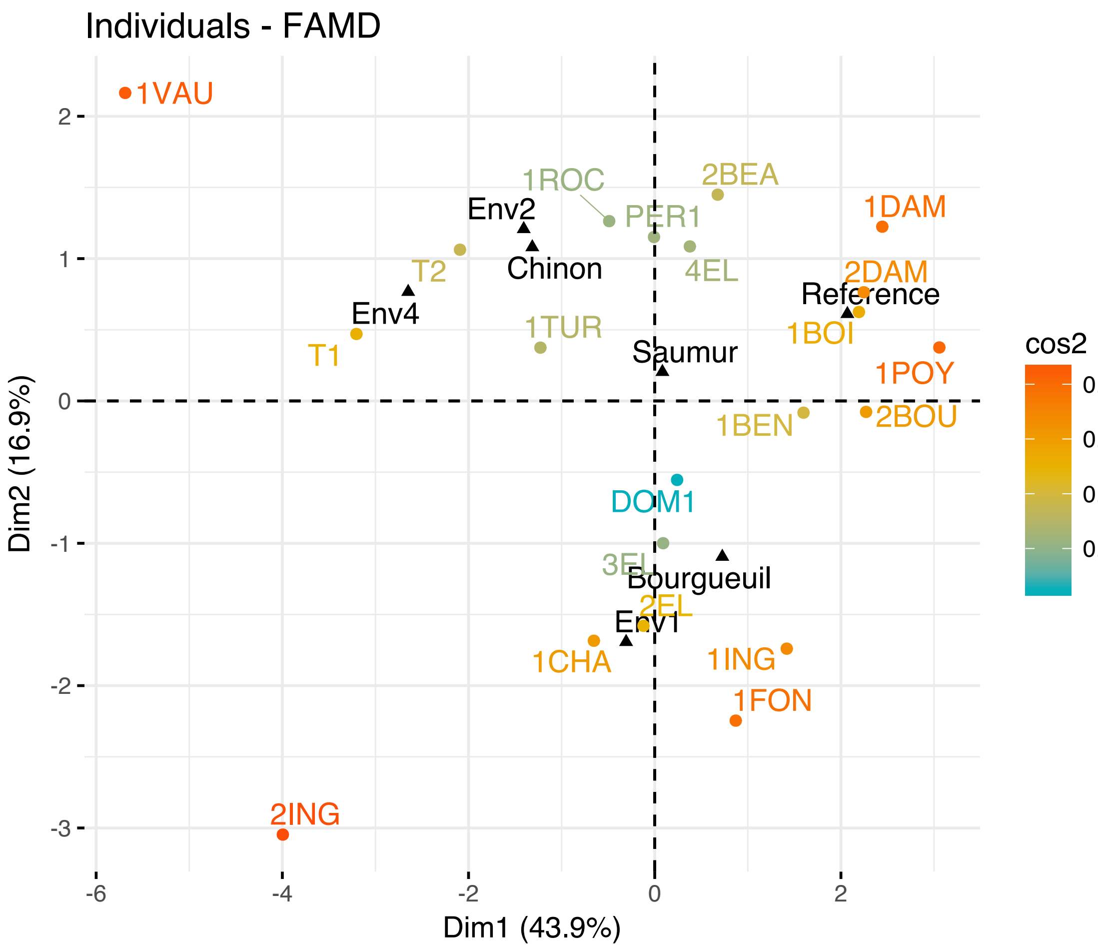
To get the results for individuals, type this:

```
ind <- get_famd_ind(res.famd)
ind
```

```
## FAMD results for individuals
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates"
## 2 "$cos2"    "Cos2, quality of representation"
## 3 "$contrib" "Contributions"
```

To plot individuals, use the function *fviz_mfa_ind*() [in *factoextra*]. By default, individuals are colored in blue. However, like variables, it's also possible to color individuals by their cos2 and contribution values:

```
fviz_famd_ind(res.famd, col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

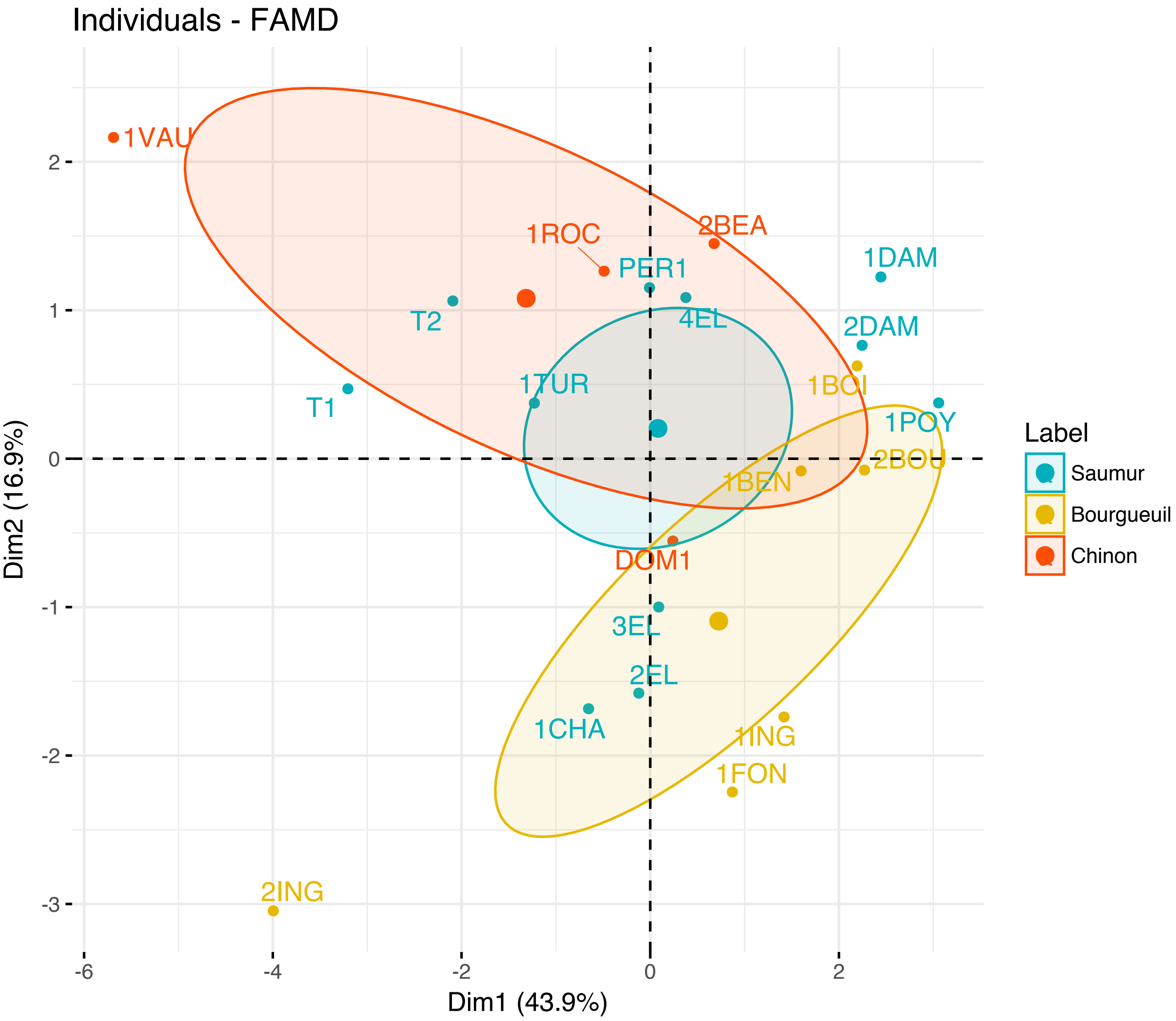


In the plot above, the qualitative variable categories are shown in black. Env1, Env2, Env3 are the categories of the soil. Saumur, Bourgueuil and Chinon are the categories of the wine Label. If you don't want to show them on the plot, use the argument *invisible = "quali.var"*.

Individuals with similar profiles are close to each other on the factor map. For the interpretation, read more at Chapter 5 (MCA) and Chapter 7 (MFA).

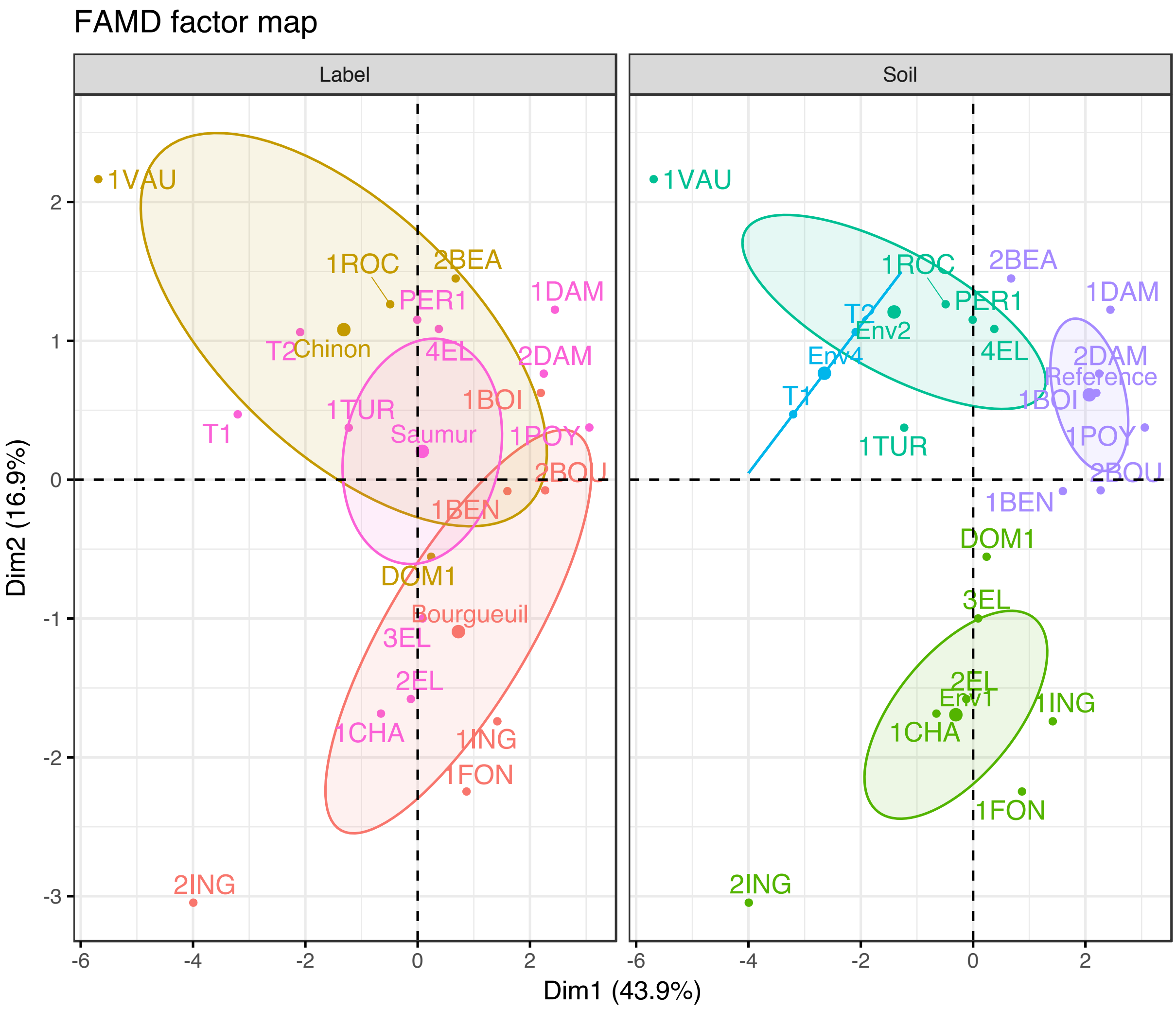
Note that, it's possible to color the individuals using any of the qualitative variables in the initial data table. To do this, the argument *habillage* is used in the *fviz_famd_ind*() function. For example, if you want to color the wines according to the supplementary qualitative variable "Label", type this:

```
fviz_mfa_ind(res.famd,
             habillage = "Label", # color by groups
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE, ellipse.type = "confidence",
             repel = TRUE # Avoid text overlapping
             )
```



If you want to color individuals using multiple categorical variables at the same time, use the function fviz_ellipses() [in *factoextra*] as follow:

```
fviz_ellipses(res.famd, c("Label", "Soil"), repel = TRUE)
```



Alternatively, you can specify categorical variable indices:

```
fviz_ellipses(res.famd, 1:2, geom = "point")
```

6.4 Summary

The factor analysis of mixed data (FAMD) makes it possible to analyze a data set, in which individuals are described by both qualitative and quantitative variables. In this article, we described how to perform and interpret FAMD using FactoMineR and factoextra R packages.

6.5 Further reading

Factor Analysis of Mixed Data Using FactoMineR (video course). `https://goo.gl/64gY3R`

Chapter 7

Multiple Factor Analysis

7.1 Introduction

Multiple factor analysis (**MFA**) (Pagès, 2002) is a multivariate data analysis method for summarizing and visualizing a complex data table in which individuals are described by several sets of variables (quantitative and /or qualitative) structured into groups. It takes into account the contribution of all active groups of variables to define the distance between individuals.
The number of variables in each group may differ and the nature of the variables (qualitative or quantitative) can vary from one group to the other but the variables should be of the same nature in a given group (Abdi and Williams, 2010).

MFA may be considered as a general factor analysis. Roughly, the core of MFA is based on:

- Principal component analysis (PCA) (Chapter 3) when variables are quantitative,
- Multiple correspondence analysis (MCA) (Chapter 5) when variables are qualitative.

This global analysis, where multiple sets of variables are simultaneously considered, requires to balance the influences of each set of variables. Therefore, in MFA, the variables are weighted during the analysis. Variables in the same group are normalized using the same weighting value, which can vary from one group to another. Technically, MFA assigns to each variable of group j, a weight equal to the inverse of the first eigenvalue of the analysis (PCA or MCA according to the type of variable) of the group j.

Multiple factor analysis can be used in a variety of fields (Pagès, 2002), where the variables are organized into groups:

1. *Survey analysis*, where an individual is a person; a variable is a question. Questions are organized by themes (groups of questions).
2. *Sensory analysis*, where an individual is a food product. A first set of variables includes *sensory variables* (sweetness, bitterness, etc.); a second one includes chemical variables (pH, glucose rate, etc.).

3. *Ecology*, where an individual is an observation place. A first set of variables describes soil characteristics ; a second one describes flora.

4. *Times series*, where several individuals are observed at different dates. In this situation, there is commonly two ways of defining groups of variables:
   - generally, variables observed at the same time (date) are gathered together.
   - When variables are the same from one date to the others, each set can gather the different dates for one variable.

In the current chapter, we show how to compute and visualize multiple factor analysis in R software using *FactoMineR* (for the analysis) and *factoextra* (for data visualization). Additional, we'll show how to reveal the most important variables that contribute the most in explaining the variations in the data set.

7.2 Computation

7.2.1 R packages

Install FactoMineR and factoextra as follow:

```
install.packages(c("FactoMineR", "factoextra"))
```

Load the packages:

```
library("FactoMineR")
library("factoextra")
```

7.2.2 Data format

We'll use the demo data sets *wine* available in *FactoMineR* package. This data set is about a sensory evaluation of wines by different judges.

```
library("FactoMineR")
data(wine)
colnames(wine)
```

```
##  [1] "Label"                          "Soil"
##  [3] "Odor.Intensity.before.shaking" "Aroma.quality.before.shaking"
##  [5] "Fruity.before.shaking"          "Flower.before.shaking"
##  [7] "Spice.before.shaking"           "Visual.intensity"
##  [9] "Nuance"                         "Surface.feeling"
## [11] "Odor.Intensity"                 "Quality.of.odour"
## [13] "Fruity"                         "Flower"
## [15] "Spice"                          "Plante"
## [17] "Phenolic"                       "Aroma.intensity"
## [19] "Aroma.persistency"              "Aroma.quality"
## [21] "Attack.intensity"               "Acidity"
## [23] "Astringency"                    "Alcohol"
```

```
## [25] "Balance"                  "Smooth"
## [27] "Bitterness"               "Intensity"
## [29] "Harmony"                  "Overall.quality"
## [31] "Typical"
```

An image of the data is shown below:

	Label	Soil	Odor.Intensity. before.shaking	Aroma.quality. before.shaking	...	Visual. intensity	Nuance	...	Odor. Intensity	Quality. of.odour	...	Attack. intensity	Acidity	...	Overall. quality	Typical
2EL	Saumur	Env1	3.074	3	...	4.321	4	...	3.407	3.308	...	2.963	2.107	...	3.393	3.25
1CHA	Saumur	Env1	2.964	2.821	...	3.222	3	...	3.37	3	...	3.036	2.107	...	3.214	3.036
1FON	Bourgueuil	Env1	2.857	2.929	...	3.536	3.393	...	3.25	2.929	...	3.222	2.179	...	3.536	3.179
1VAU	Chinon	Env2	2.808	2.593	...	2.893	2.786	...	3.16	2.88	...	2.704	3.179	...	2.464	2.25
1DAM	Saumur	Reference	3.607	3.429	...	4.393	4.036	...	3.536	3.36	...	3.464	2.571	...	3.741	3.444
2BOU	Bourgueuil	Reference	2.857	3.111	...	4.464	4.259	...	3.179	3.385	...	3.286	2.393	...	3.643	3.393
1BOI	Bourgueuil	Reference	3.214	3.222	...	4.143	3.929	...	3.429	3.5	...	3.393	2.607	...	3.714	3.357
3EL	Saumur	Env1	3.12	2.852	...	4.214	3.857	...	3.654	3.077	...	3.25	2.179	...	3.393	3.071
DOM1	Chinon	Env1	2.857	2.815	...	4.037	3.893	...	3.357	3.346	...	3.286	2.286	...	3.2	3.5
1TUR	Saumur	Env2	2.893	3	...	3.704	3.407	...	3.222	3.259	...	2.893	2.357	...	3.179	2.964
4EL	Saumur	Env2	3.25	3.286	...	3.857	3.643	...	3.607	3.385	...	3.321	2.429	...	3.571	3.5
PER1	Saumur	Env2	3.393	3.179	...	4.714	4.5	...	3.481	3.385	...	3.357	2.429	...	3.148	3.556
2DAM	Saumur	Reference	3.179	3.286	...	4.222	4.071	...	3.481	3.423	...	3.393	2.286	...	3.571	3.929
1POY	Saumur	Reference	3.071	3.107	...	4.714	4.536	...	3.357	3.444	...	3.519	2.111	...	3.929	3.481
1ING	Bourgueuil	Env1	3.107	3.143	...	4.071	3.893	...	3.357	3.37	...	3.185	2.286	...	3.643	3.296
1BEN	Bourgueuil	Reference	2.929	3.179	...	3.889	3.429	...	3.286	3.308	...	3.393	2.393	...	3.75	3.571
2BEA	Chinon	Reference	3.036	3.179	...	3.786	3.607	...	3.444	3.5	...	3.071	2.571	...	3.536	3.269
1ROC	Chinon	Env2	3.071	2.926	...	3.679	3.393	...	3.37	3.36	...	3.071	2.393	...	3.464	3.444
2ING	Bourgueuil	Env1	2.643	2.786	...	2.607	2.536	...	2.889	2.8	...	2.179	2.25	...	2.37	2.321
T1	Saumur	Env4	3.696	3.192	...	4.321	4	...	3.737	3.08	...	2.963	2.407	...	2.643	2.571
T2	Saumur	Env4	3.708	2.926	...	4.321	4.107	...	3.727	2.885	...	3.333	2.571	...	2.852	2.75

Figure 7.1: Data format for Multiple Factor analysis

(Image source, FactoMineR, `http://factominer.free.fr`)

The data contains 21 rows (wines, *individuals*) and 31 columns (*variables*):

- The first two columns are *categorical variables*: *label* (Saumur, Bourgueil or Chinon) and *soil* (Reference, Env1, Env2 or Env4).
- The 29 next columns are *continuous sensory variables*. For each wine, the value is the mean score for all the judges.

> The goal of this study is to analyze the characteristics of the wines.

The variables are organized in groups as follow:

1. First group - A group of categorical variables specifying the *origin of the wines*, including the variables *label* and *soil* corresponding to the *first 2 columns* in the data table. In FactoMineR terminology, the arguments *group = 2* is used to define the first 2 columns as a group.

2. Second group - A group of continuous variables, describing the *odor of the wines before shaking*, including the variables: Odor.Intensity.before.shaking, Aroma.quality.before.shaking, Fruity.before.shaking, Flower.before.shaking and Spice.before.shaking. These variables corresponds to the *next 5 columns* after the first group. FactoMineR terminology: *group = 5*.

3. Third group - A group of continuous variables quantifying the *visual inspection of the wines*, including the variables: Visual.intensity, Nuance and Surface.feeling. These variables corresponds to the *next 3 columns* after the second group. FactoMineR terminology: *group = 3*.

4. Fourth group - A group of continuous variables concerning the *odor of the wines after shaking*, including the variables: Odor.Intensity, Quality.of.odour, Fruity, Flower, Spice, Plante, Phenolic, Aroma.intensity, Aroma.persistency and Aroma.quality. These variables corresponds to the *next 10 columns* after the third group. FactoMineR terminology: *group = 10*.

5. Fith group - A group of continuous variables evaluating the *taste of the wines*, including the variables Attack.intensity, Acidity, Astringency, Alcohol, Balance, Smooth, Bitterness, Intensity and Harmony. These variables corresponds to the *next 9 columns* after the fourth group. FactoMineR terminology: *group = 9*.

6. Sixth group - A group of continuous variables concerning the *overall judgement of the wines*, including the variables Overall.quality and Typical. These variables corresponds to the *next 2 columns* after the fith group. FactoMineR terminology: *group = 2*.

In summary:

- We have 6 groups of variables, which can be specified to the FactoMineR as follow: *group = c(2, 5, 3, 10, 9, 2)*.
- These groups can be named as follow: *name.group = c("origin", "odor", "visual", "odor.after.shaking", "taste", "overall")*.
- Among the 6 groups of variables, one is categorical and five groups contain continuous variables. It's recommended, to standardize the continuous variables during the analysis. Standardization makes variables comparable, in the situation where the variables are measured in different units. In FactoMineR, the argument *type = "s"* specifies that a given group of variables should be standardized. If you don't want standardization, use *type = "c"*. To specify categorical variables, *type = "n"* is used. In our example, we'll use *type = c("n", "s", "s", "s", "s", "s")*.

7.2.3 R code

The function *MFA*()[*FactoMiner* package] can be used. A simplified format is :

```
MFA (base, group, type = rep("s",length(group)), ind.sup = NULL,
    name.group = NULL, num.group.sup = NULL, graph = TRUE)
```

- **base** : a data frame with n rows (individuals) and p columns (variables)
- **group**: a vector with the number of variables in each group.
- **type**: the type of variables in each group. By default, all variables are quantitative and scaled to unit variance. Allowed values include:
  - "c" or "s" for quantitative variables. If "s", the variables are scaled to unit variance.
  - "n" for categorical variables.
  - "f" for frequencies (from a contingency tables).
- **ind.sup**: a vector indicating the indexes of the supplementary individuals.
- **name.group**: a vector containing the name of the groups (by default, NULL

and the group are named group.1, group.2 and so on).
- **num.group.sup**: the indexes of the illustrative groups (by default, NULL and no group are illustrative).
- **graph** : a logical value. If TRUE a graph is displayed.

The R code below performs the MFA on the wines data using the groups: odor, visual, odor after shaking and taste. These groups are named **active groups**. The remaining group of variables - origin (the first group) and overall judgement (the sixth group) - are named **supplementary groups**; *num.group.sup = c(1, 6)*:

```
library(FactoMineR)
data(wine)
res.mfa <- MFA(wine,
               group = c(2, 5, 3, 10, 9, 2),
               type = c("n", "s", "s", "s", "s", "s"),
               name.group = c("origin","odor","visual",
                              "odor.after.shaking", "taste","overall"),
               num.group.sup = c(1, 6),
               graph = FALSE)
```

The output of the *MFA*() function is a list including :

```
print(res.mfa)
```

```
## **Results of the Multiple Factor Analysis (MFA)**
## The analysis was performed on 21 individuals, described by 31 variables
## *Results are available in the following objects :
##
##    name
## 1  "$eig"
## 2  "$separate.analyses"
## 3  "$group"
## 4  "$partial.axes"
## 5  "$inertia.ratio"
## 6  "$ind"
## 7  "$quanti.var"
## 8  "$quanti.var.sup"
## 9  "$quali.var.sup"
## 10 "$summary.quanti"
## 11 "$summary.quali"
## 12 "$global.pca"
##    description
## 1  "eigenvalues"
## 2  "separate analyses for each group of variables"
## 3  "results for all the groups"
## 4  "results for the partial axes"
## 5  "inertia ratio"
## 6  "results for the individuals"
## 7  "results for the quantitative variables"
```

```
## 8  "results for the quantitative supplementary variables"
## 9  "results for the categorical supplementary variables"
## 10 "summary for the quantitative variables"
## 11 "summary for the categorical variables"
## 12 "results for the global PCA"
```

7.3 Visualization and interpretation

We'll use the *factoextra* R package to help in the interpretation and the visualization of the multiple factor analysis.

The functions below [in *factoextra* package] will be used:

- *get_eigenvalue*(res.mfa): Extract the eigenvalues/variances retained by each dimension (axis).
- *fviz_eig*(res.mfa): Visualize the eigenvalues/variances.
- *get_mfa_ind*(res.mfa): Extract the results for individuals.
- *get_mfa_var*(res.mfa): Extract the results for quantitative and qualitative variables, as well as, for groups of variables.
- *fviz_mfa_ind*(res.mfa), *fviz_mfa_var*(res.mfa): Visualize the results for individuals and variables, respectively.

In the next sections, we'll illustrate each of these functions.

To help in the interpretation of MFA, we highly recommend to read the interpretation of principal component analysis (Chapter 3), simple (Chapter 4) and multiple correspondence analysis (Chapter 5). Many of the graphs presented here have been already described in previous chapter.

7.3.1 Eigenvalues / Variances

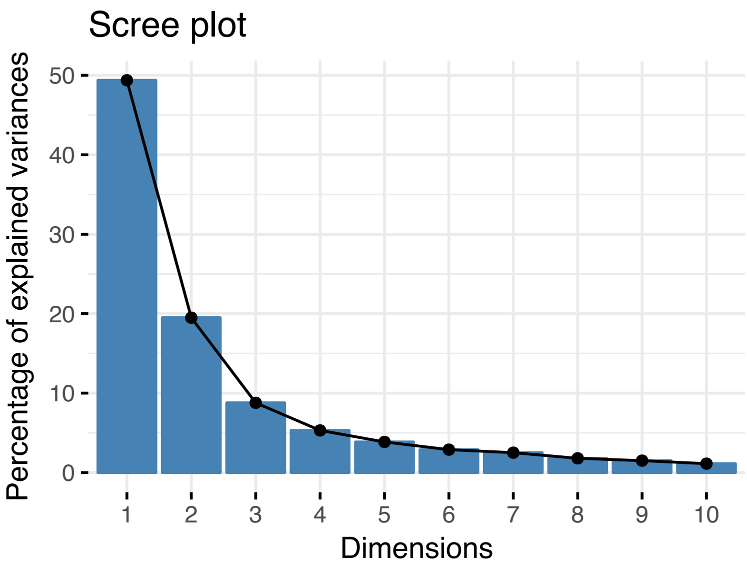
The proportion of variances retained by the different dimensions (axes) can be extracted using the function *get_eigenvalue*() [*factoextra* package] as follow:

```
library("factoextra")
eig.val <- get_eigenvalue(res.mfa)
head(eig.val)
```

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1      3.462            49.38                        49.4
## Dim.2      1.367            19.49                        68.9
## Dim.3      0.615             8.78                        77.7
## Dim.4      0.372             5.31                        83.0
## Dim.5      0.270             3.86                        86.8
## Dim.6      0.202             2.89                        89.7
```

The function *fviz_eig*() or *fviz_screeplot*() [*factoextra* package] can be used to draw the scree plot:

```
fviz_screeplot(res.mfa)
```



7.3.2 Graph of variables

7.3.2.1 Groups of variables

The function *get_mfa_var*() [in *factoextra*] is used to extract the results for groups of variables. This function returns a list containing the coordinates, the cos2 and the contribution of groups, as well as, the

```
group <- get_mfa_var(res.mfa, "group")
group
```

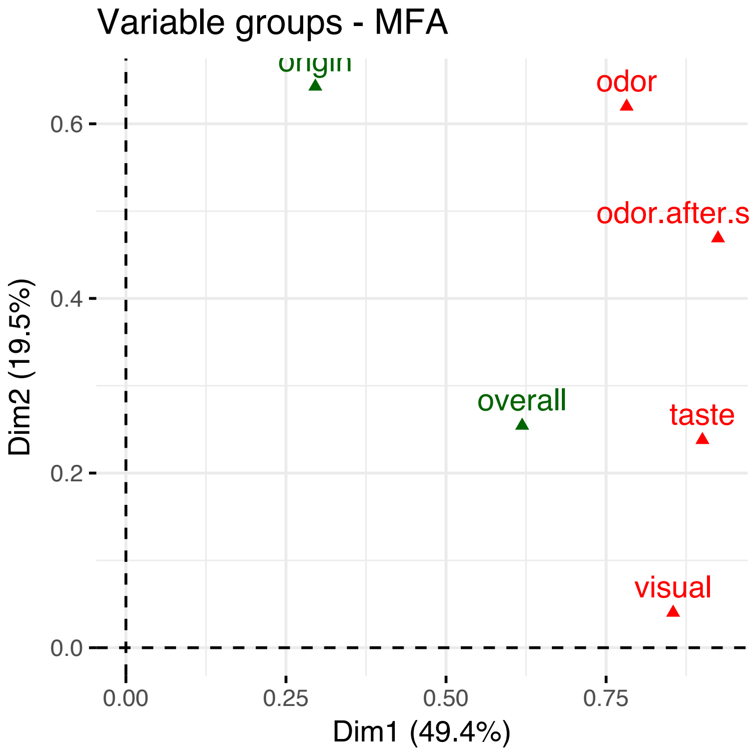
```
## Multiple Factor Analysis results for variable groups
##  ===================================================
##   Name           Description
## 1 "$coord"       "Coordinates"
## 2 "$cos2"        "Cos2, quality of representation"
## 3 "$contrib"     "Contributions"
## 4 "$correlation" "Correlation between groups and principal dimensions"
```

The different components can be accessed as follow:

```
# Coordinates of groups
head(group$coord)

# Cos2: quality of representation on the factore map
head(group$cos2)

# Contributions to the  dimensions
head(group$contrib)
```

To plot the groups of variables, type this:

```
fviz_mfa_var(res.mfa, "group")
```

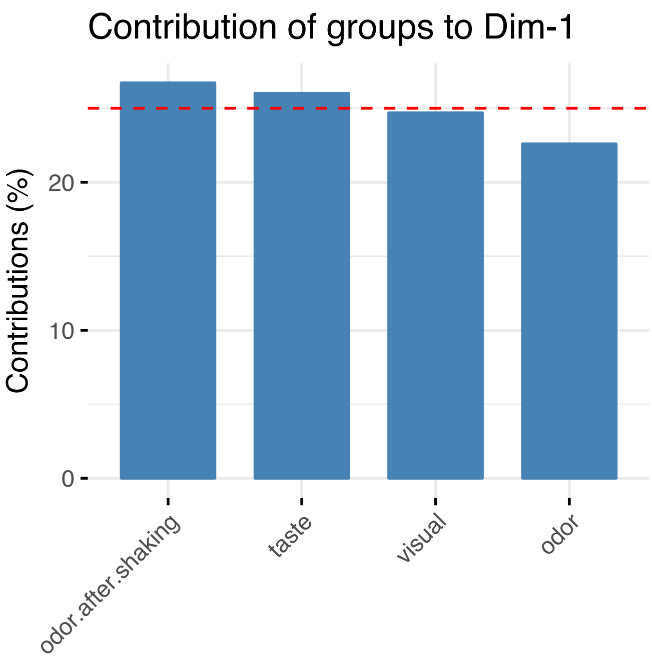


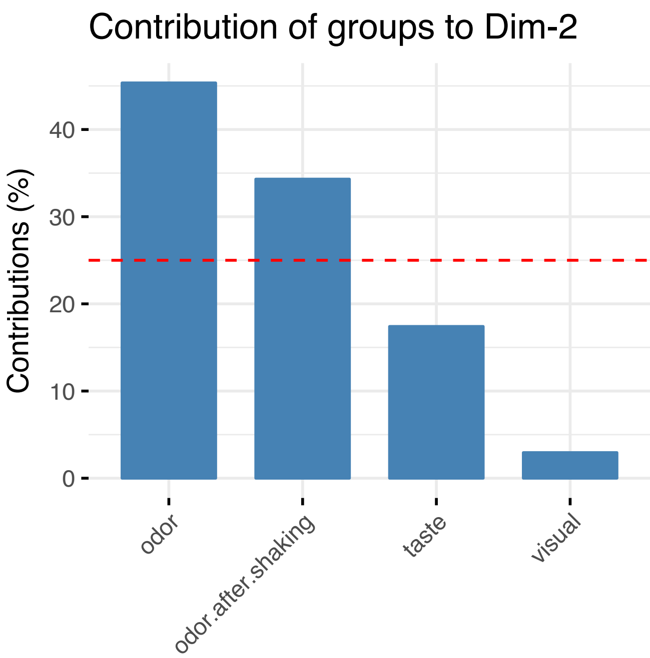
- red color = active groups of variables
- green color = supplementary groups of variables

The plot above illustrates the correlation between groups and dimensions. The coordinates of the four active groups on the first dimension are almost identical. This means that they contribute similarly to the first dimension. Concerning the second dimension, the two groups - odor and odor.after.shake - have the highest coordinates indicating a highest contribution to the second dimension.

To draw a bar plot of groups contribution to the dimensions, use the function *fviz_contrib*():

```
# Contribution to the first dimension
fviz_contrib(res.mfa, "group", axes = 1)

# Contribution to the second dimension
fviz_contrib(res.mfa, "group", axes = 2)
```





7.3.2.2 Quantitative variables

The function *get_mfa_var*() [in *factoextra*] is used to extract the results for quantitative variables. This function returns a list containing the coordinates, the cos2 and the contribution of variables:

```
quanti.var <- get_mfa_var(res.mfa, "quanti.var")
quanti.var
```

```
## Multiple Factor Analysis results for quantitative variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates"
## 2 "$cos2"    "Cos2, quality of representation"
## 3 "$contrib" "Contributions"
```

The different components can be accessed as follow:

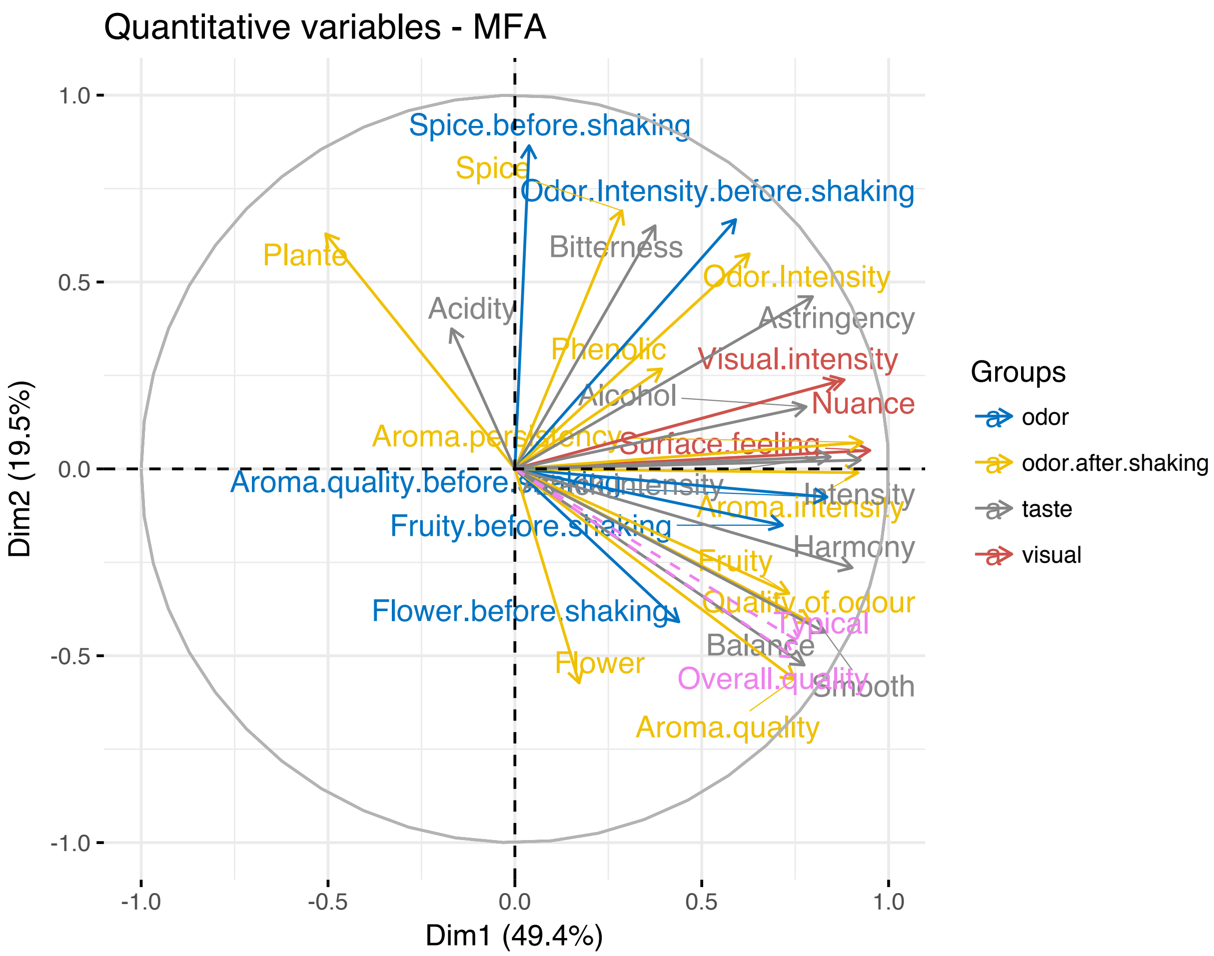
```
# Coordinates
head(quanti.var$coord)

# Cos2: quality on the factore map
head(quanti.var$cos2)

# Contributions to the dimensions
head(quanti.var$contrib)
```

In this section, we'll describe how to visualize quantitative variables colored by groups. Next, we'll highlight variables according to either i) their quality of representation on the factor map or ii) their contributions to the dimensions.

To interpret the graphs presented here, read the chapter on PCA (Chapter 3) and MCA (Chapter 5).

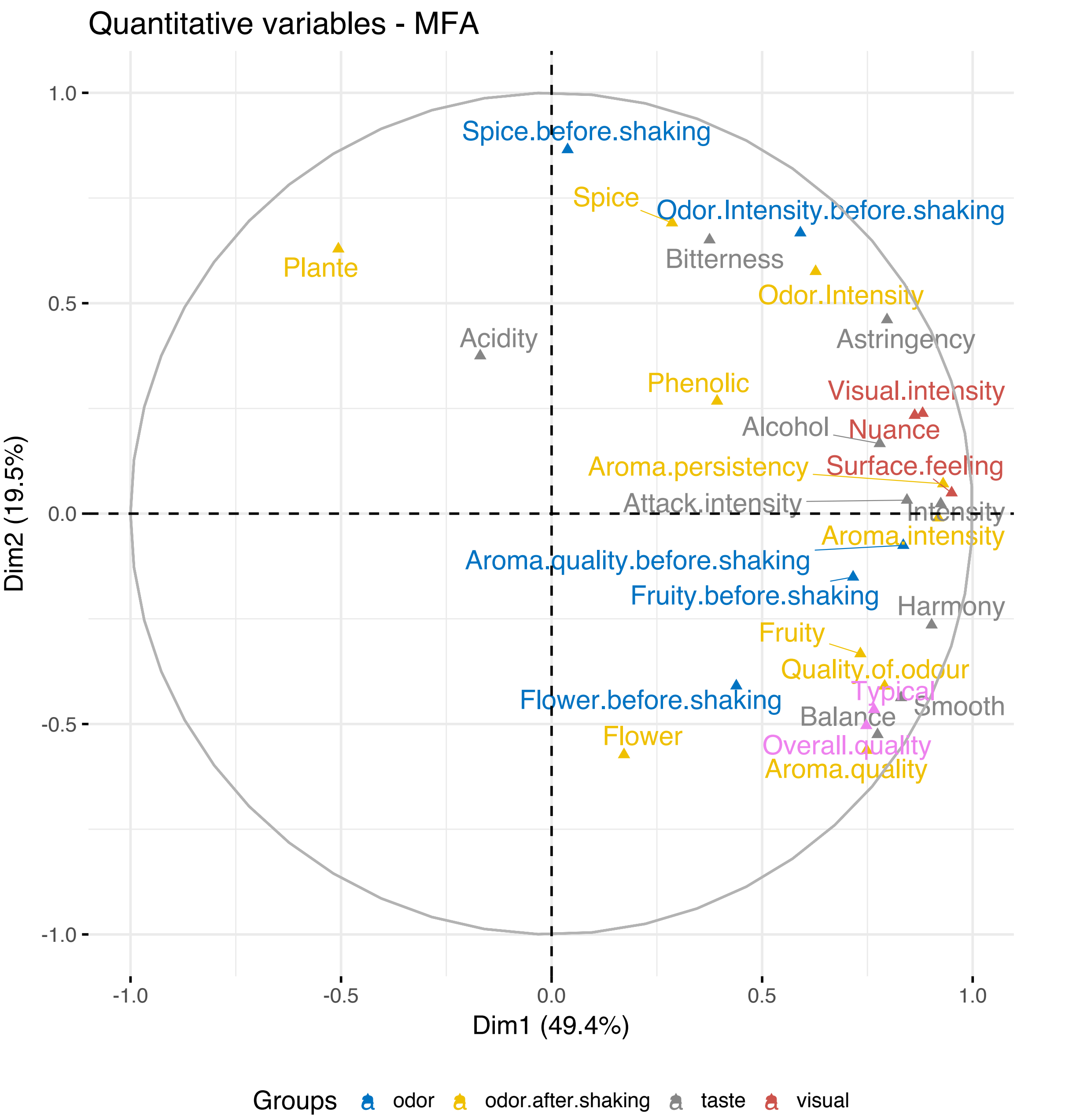
Correlation between quantitative variables and dimensions. The R code below plots quantitative variables colored by groups. The argument palette is used to change group colors (see ?ggpubr::ggpar for more information about palette). Supplementary quantitative variables are in dashed arrow and violet color. We use repel = TRUE, to avoid text overlapping.

```
fviz_mfa_var(res.mfa, "quanti.var", palette = "jco",
             col.var.sup = "violet", repel = TRUE)
```



To make the plot more readable, we can use geom = c("point", "text") instead of geom = c("arrow", "text"). We'll change also the legend position from "right" to "bottom", using the argument legend = "bottom":

```
fviz_mfa_var(res.mfa, "quanti.var", palette = "jco",
             col.var.sup = "violet", repel = TRUE,
             geom = c("point", "text"), legend = "bottom")
```



Briefly, the graph of variables (correlation circle) shows the relationship between variables, the quality of the representation of variables, as well as, the correlation between variables and the dimensions:

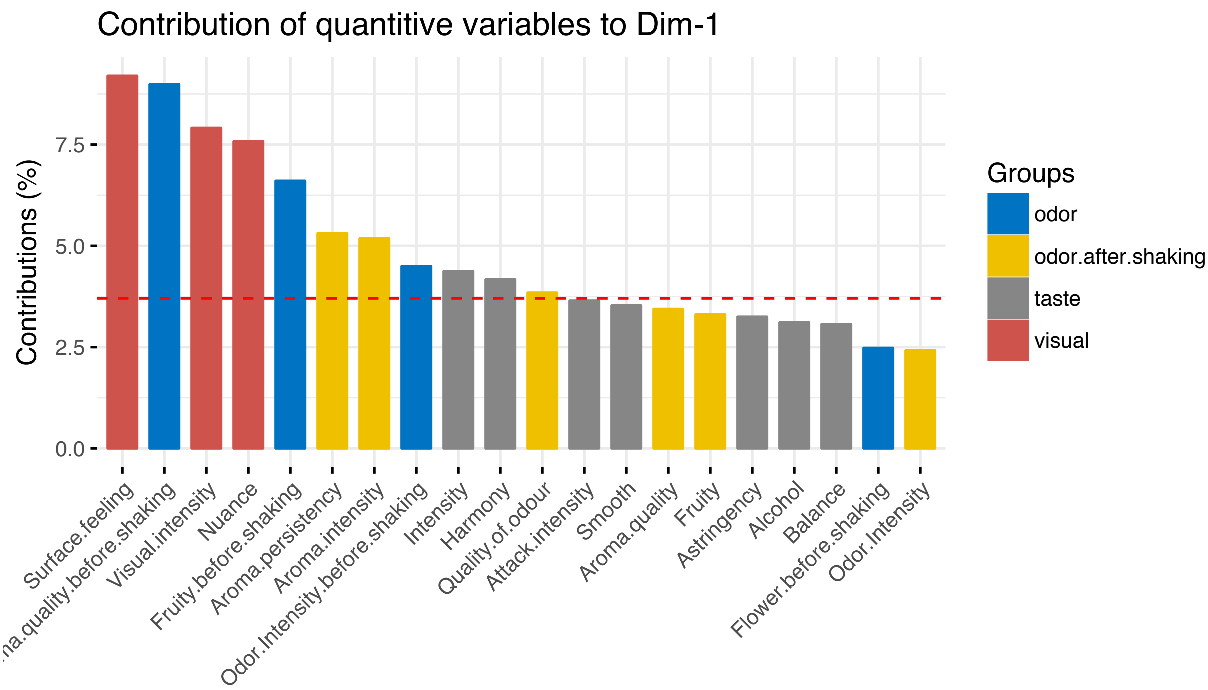
- Positive correlated variables are grouped together, whereas negative ones are positioned on opposite sides of the plot origin (opposed quadrants).
- The distance between variable points and the origin measures the quality of the variable on the factor map. Variable points that are away from the origin are well represented on the factor map.
- For a given dimension, the most correlated variables to the dimension are close to the dimension.

For example, the first dimension represents the positive sentiments about wines: "intensity" and "harmony". The most correlated variables to the second dimension are: i) Spice before shaking and Odor intensity before shaking for the odor group; ii) Spice, Plant and

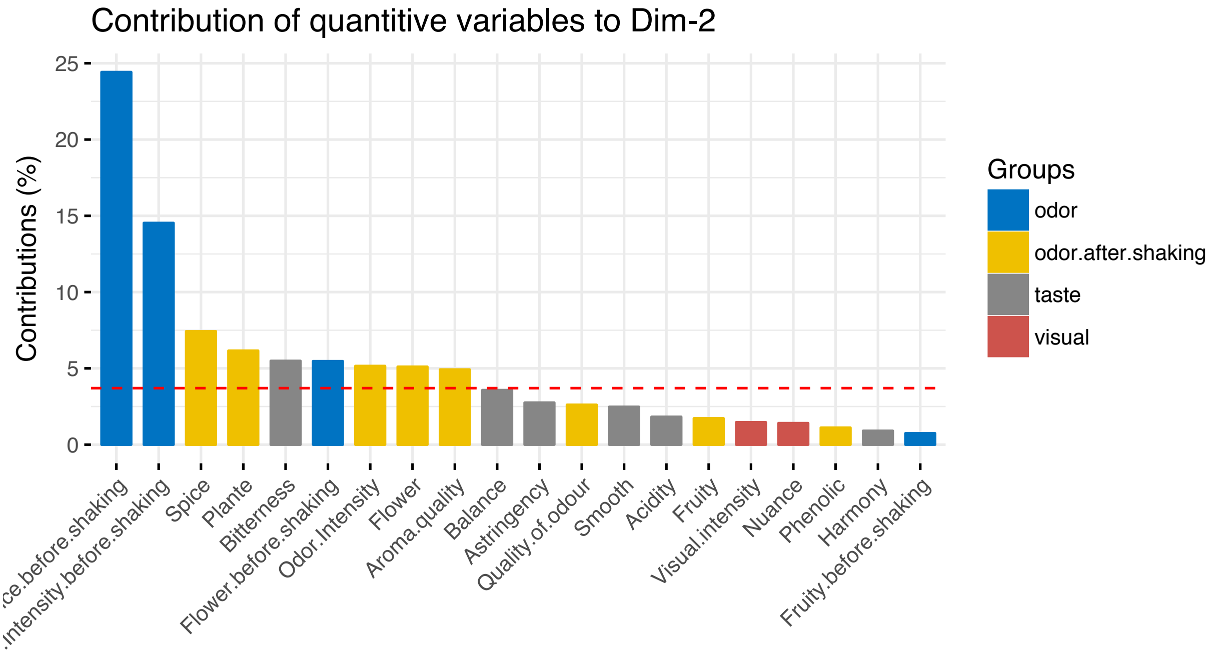
Odor intensity for the odor after shaking group and iii) Bitterness for the taste group. This dimension represents essentially the "spicyness" and the vegetal characteristic due to olfaction.

The contribution of quantitative variables (in %) to the definition of the dimensions can be visualized using the function *fviz_contrib*() [factoextra package]. Variables are colored by groups. The R code below shows the top 20 variable categories contributing to the dimensions:

```
# Contributions to dimension 1
fviz_contrib(res.mfa, choice = "quanti.var", axes = 1, top = 20,
             palette = "jco")
```




```
# Contributions to dimension 2
fviz_contrib(res.mfa, choice = "quanti.var", axes = 2, top = 20,
             palette = "jco")
```

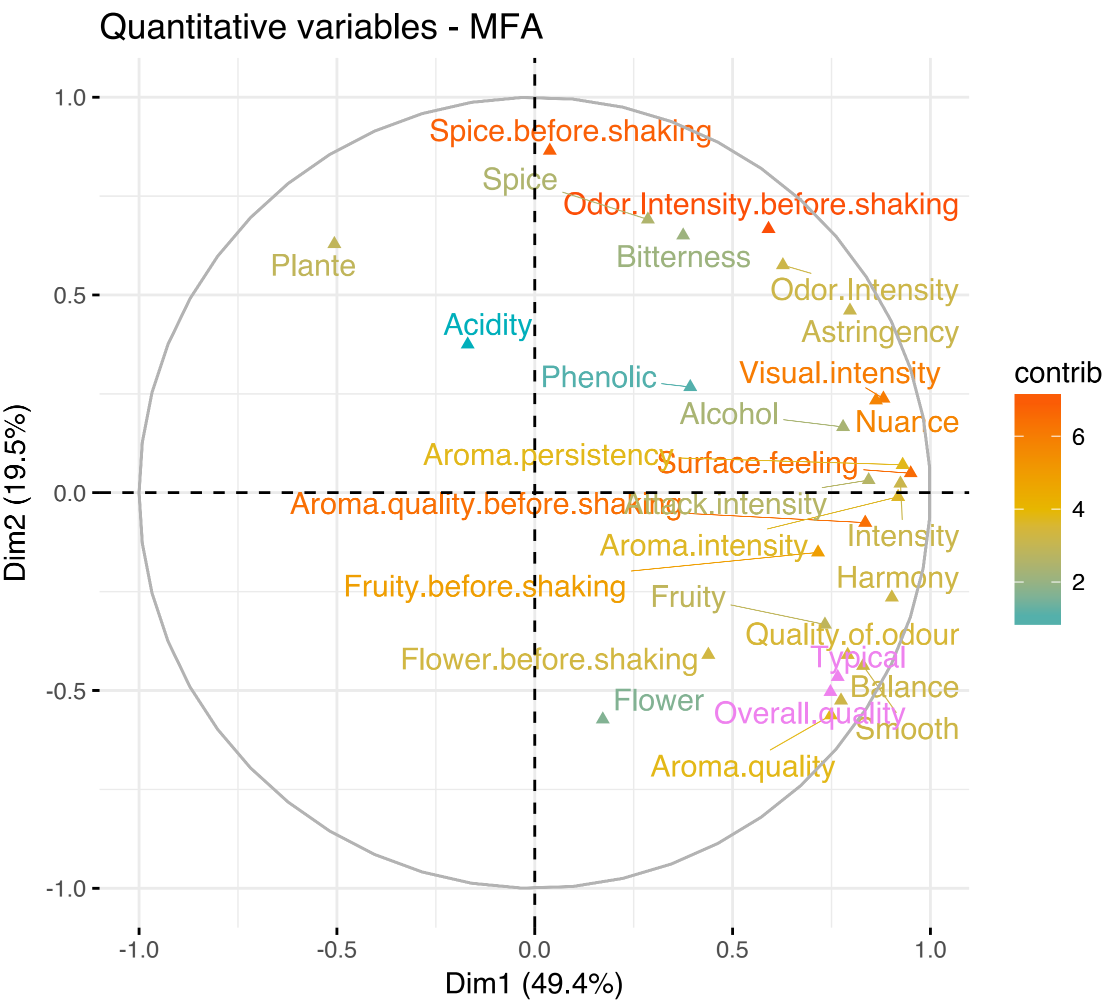


The red dashed line on the graph above indicates the expected average value, If the contributions were uniform. The calculation of the expected contribution value, under null hypothesis, has been detailed in the principal component analysis chapter (Chapter 3).

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

The most contributing quantitative variables can be highlighted on the scatter plot using the argument col.var = "contrib". This produces a gradient colors, which can be customized using the argument *gradient.cols*.

```
fviz_mfa_var(res.mfa, "quanti.var", col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             col.var.sup = "violet", repel = TRUE,
             geom = c("point", "text"))
```



Similarly, you can highlight quantitative variables using their cos2 values representing the quality of representation on the factor map. If a variable is well represented by two dimensions, the sum of the cos2 is closed to one. For some of the row items, more than 2 dimensions might be required to perfectly represent the data.

```
# Color by cos2 values: quality on the factor map
fviz_mfa_var(res.mfa, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             col.var.sup = "violet", repel = TRUE)
```

To create a bar plot of variables cos2, type this:

```
fviz_cos2(res.mfa, choice = "quanti.var", axes = 1)
```

7.3.3 Graph of individuals

To get the results for individuals, type this:

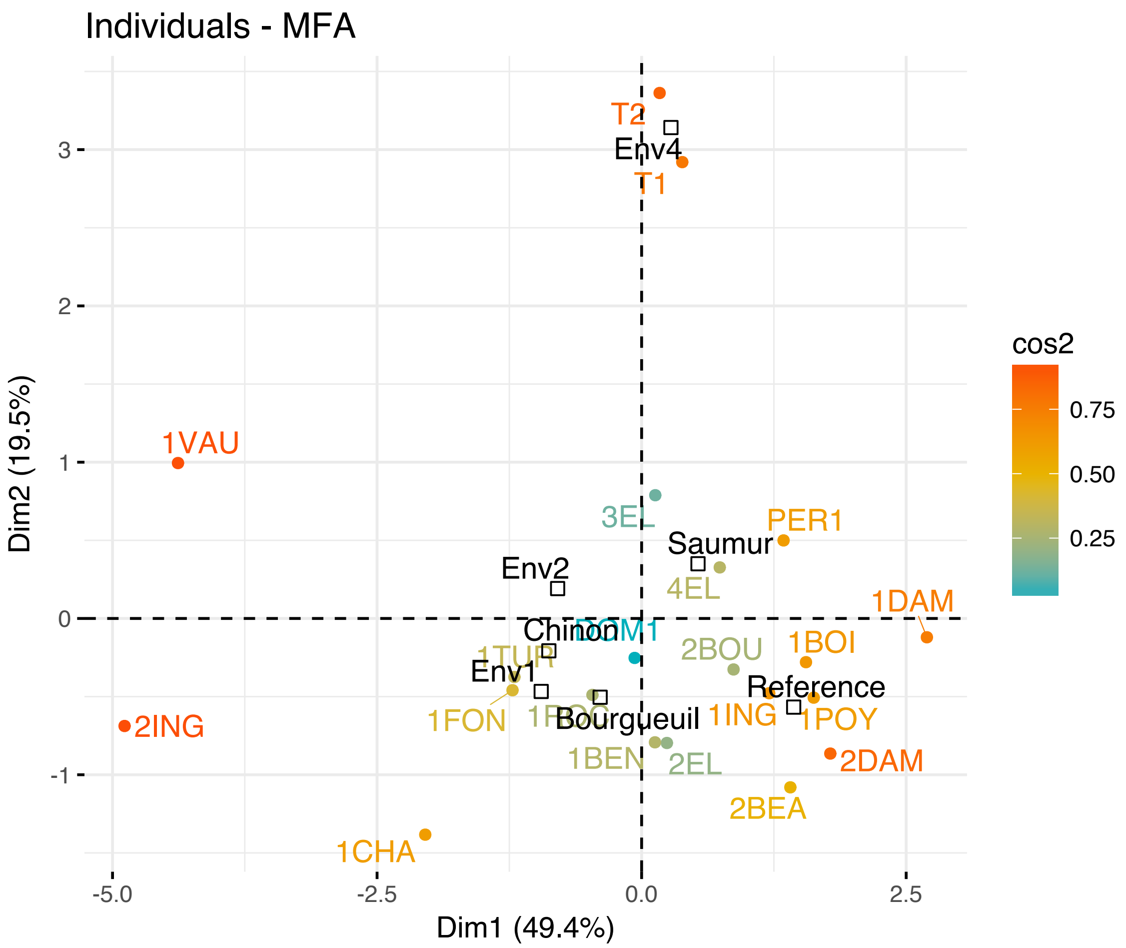
```
ind <- get_mfa_ind(res.mfa)
ind
```

```
## Multiple Factor Analysis results for individuals
##  ===================================================
```

```
##   Name                          Description
## 1 "$coord"                     "Coordinates"
## 2 "$cos2"                      "Cos2, quality of representation"
## 3 "$contrib"                   "Contributions"
## 4 "$coord.partiel"             "Partial coordinates"
## 5 "$within.inertia"            "Within inertia"
## 6 "$within.partial.inertia"    "Within partial inertia"
```

To plot individuals, use the function *fviz_mfa_ind*() [in *factoextra*]. By default, individuals are colored in blue. However, like variables, it's also possible to color individuals by their cos2 values:

```
fviz_mfa_ind(res.mfa, col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```



In the plot above, the supplementary qualitative variable categories are shown in black. Env1, Env2, Env3 are the categories of the soil. Saumur, Bourgueuil and Chinon are the categories of the wine Label. If you don't want to show them on the plot, use the argument *invisible = "quali.var"*.

Individuals with similar profiles are close to each other on the factor map. The first axis, mainly opposes the wine *1DAM* and, the wines *1VAU* and *2ING*. As described in the previous section, the first dimension represents the harmony and the intensity of wines.

Thus, the wine 1DAM (positive coordinates) was evaluated as the most "intense" and "harmonious" contrary to wines 1VAU and 2ING (negative coordinates) which are the least "intense" and "harmonious". The second axis is essentially associated with the two wines T1 and T2 characterized by a strong value of the variables Spice.before.shaking and Odor.intensity.before.shaking.

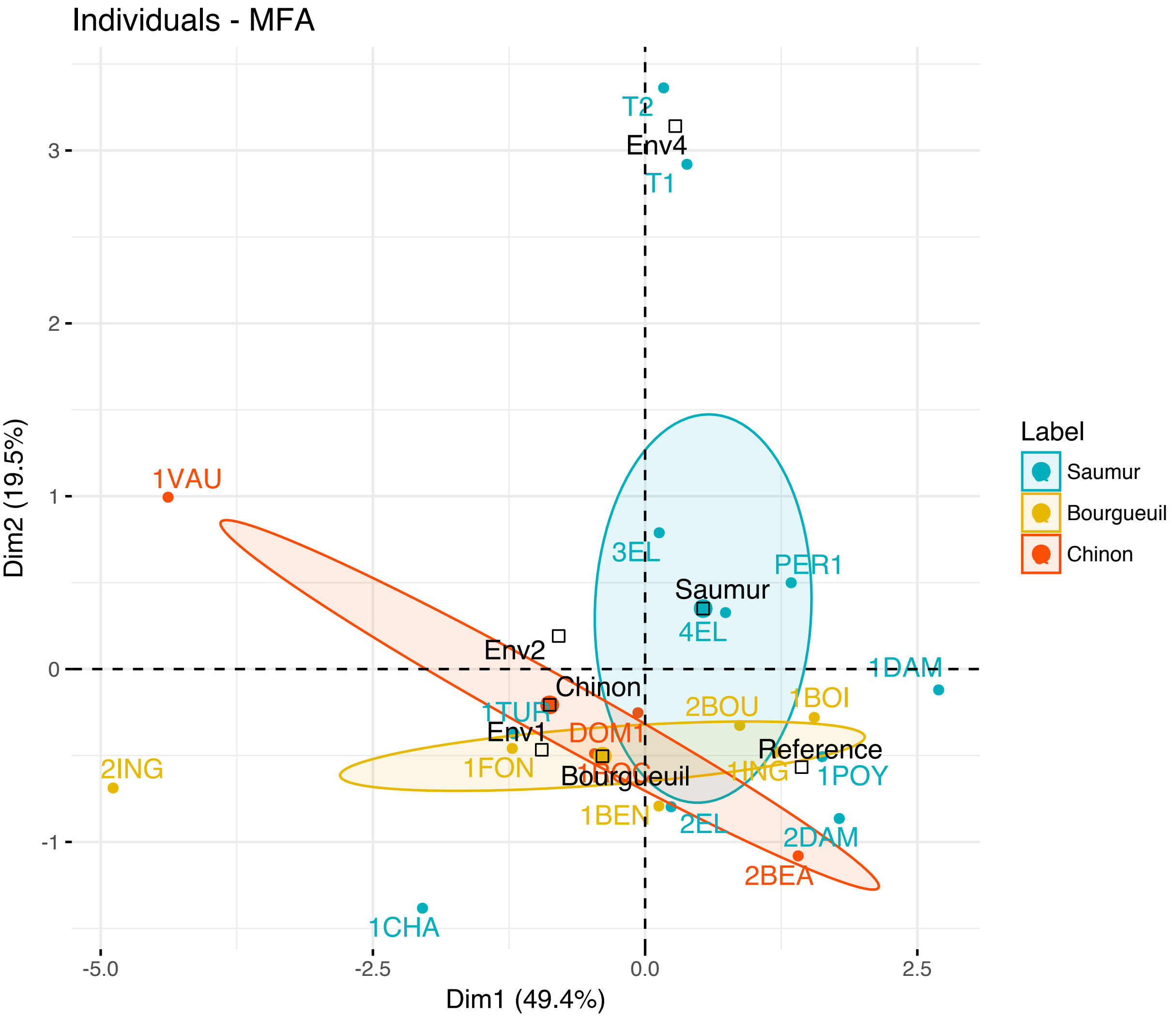
Most of the supplementary qualitative variable categories are close to the origin of the map. This result indicates that the concerned categories are not related to the first axis (wine "intensity" & "harmony") or the second axis (wine T1 and T2).

The category Env4 has high coordinates on the second axis related to T1 and T2.

The category "Reference" is known to be related to an excellent wine-producing soil. As expected, our analysis demonstrates that the category "Reference" has high coordinates on the first axis, which is positively correlated with wines "intensity" and "harmony".

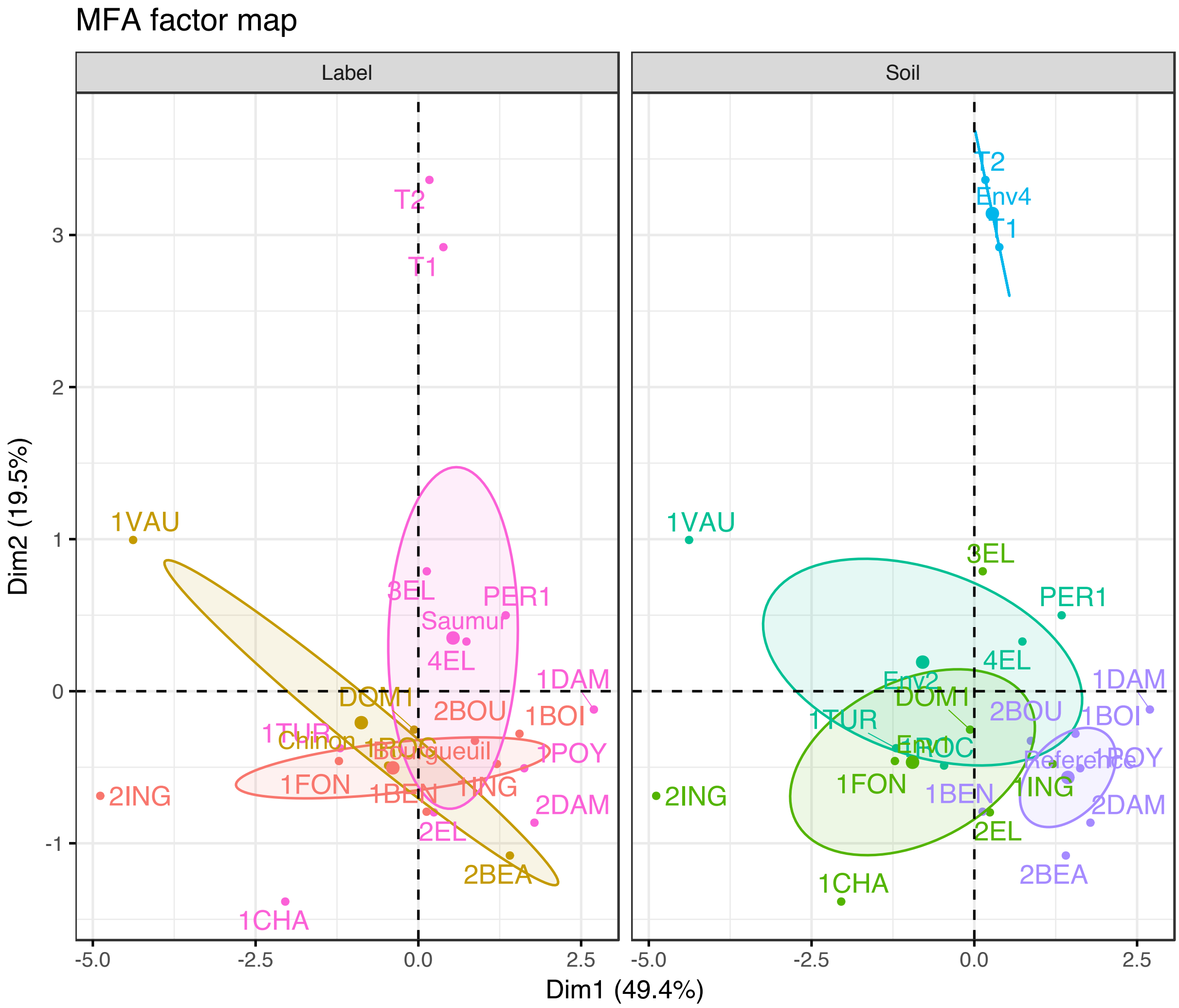
Note that, it's possible to color the individuals using any of the qualitative variables in the initial data table. To do this, the argument *habillage* is used in the *fviz_mfa_ind*() function. For example, if you want to color the wines according to the supplementary qualitative variable "Label", type this:

```
fviz_mfa_ind(res.mfa,
             habillage = "Label", # color by groups
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE, ellipse.type = "confidence",
             repel = TRUE # Avoid text overlapping
             )
```



If you want to color individuals using multiple categorical variables at the same time, use the function fviz_ellipses() [in *factoextra*] as follow:

```
fviz_ellipses(res.mfa, c("Label", "Soil"), repel = TRUE)
```



Alternatively, you can specify categorical variable indices:

```
fviz_ellipses(res.mca, 1:2, geom = "point")
```

7.3.4 Graph of partial individuals

The results for individuals obtained from the analysis performed with a single group are named *partial individuals*. In other words, an individual considered from the point of view of a single group is called partial individual.

In the default *fviz_mfa_ind*() plot, for a given individual, the point corresponds to the *mean individual* or the center of gravity of the partial points of the individual. That is, the individual viewed by all groups of variables.

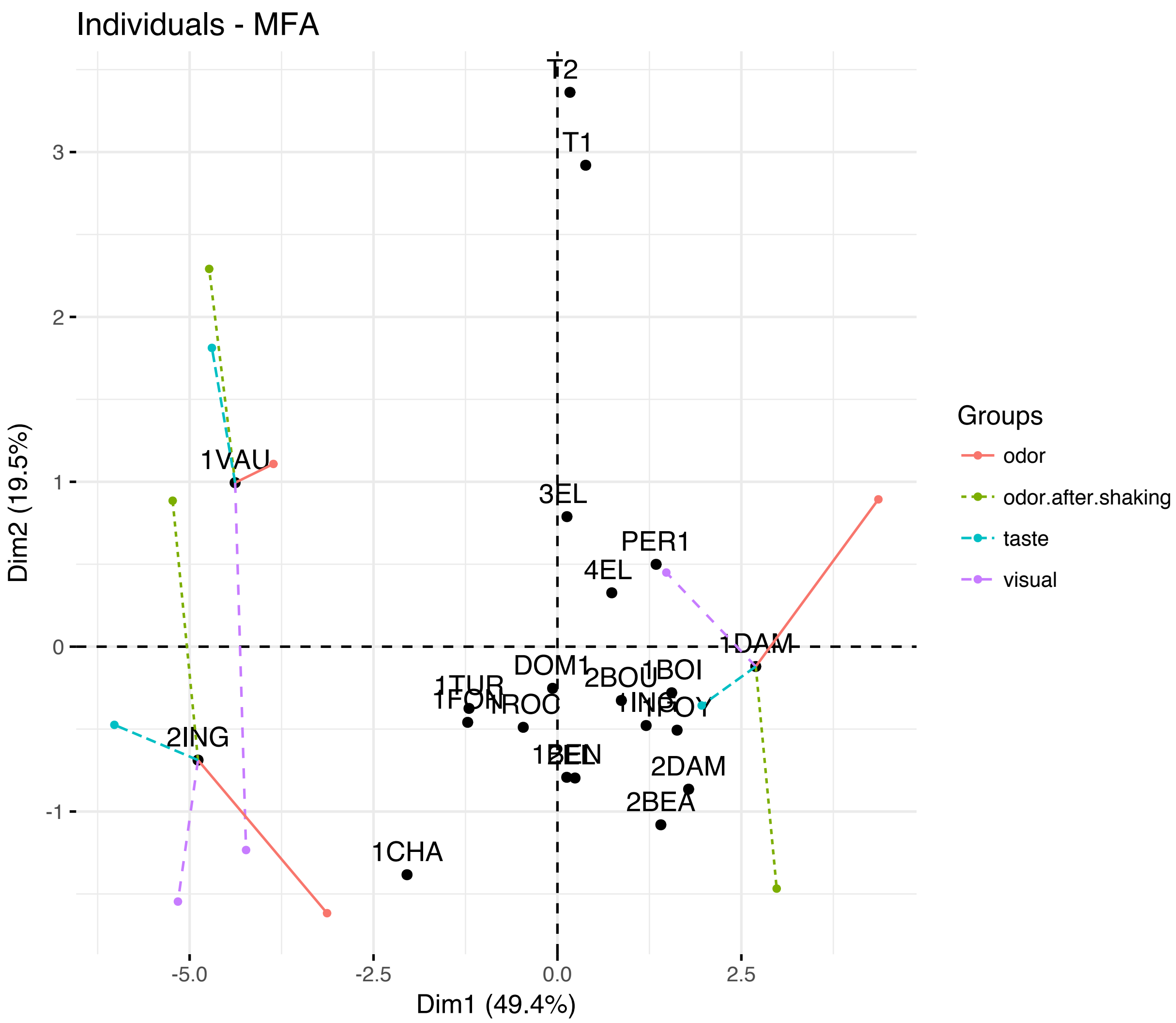
For a given individual, there are as many partial points as groups of variables.

The graph of partial individuals represents each wine viewed by each group and its barycenter. To plot the partial points of all individuals, type this:

```
fviz_mfa_ind(res.mfa, partial = "all")
```

If you want to visualize partial points for wines of interest, let say c("1DAM", "1VAU", "2ING"), use this:

```
fviz_mfa_ind(res.mfa, partial = c("1DAM", "1VAU", "2ING"))
```



Red color represents the wines seen by only the *odor* variables; violet color represents the wines seen by only the *visual* variables, and so on.

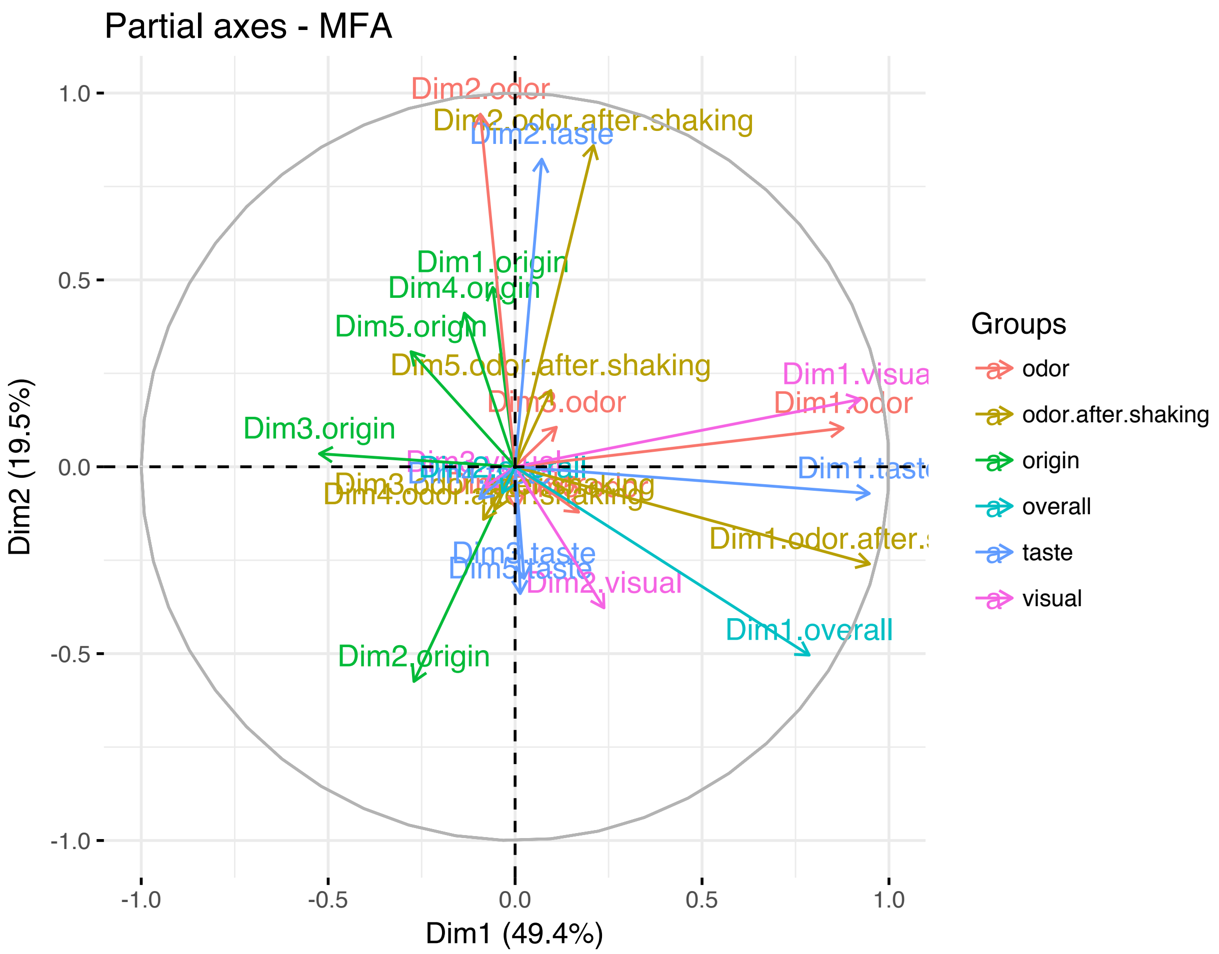
The wine *1DAM* has been described in the previous section as particularly "intense" and "harmonious", particularly by the *odor* group: It has a high coordinate on the first axis from the point of view of the *odor* variables group compared to the point of view of the other groups.

From the odor group's point of view, 2ING was more "intense" and "harmonious" than 1VAU but from the taste group's point of view, 1VAU was more "intense" and "harmonious" than 2ING.

7.3.5 Graph of partial axes

The graph of partial axes shows the relationship between the principal axes of the MFA and the ones obtained from analyzing each group using either a PCA (for groups of continuous variables) or a MCA (for qualitative variables).

```r
fviz_mfa_axes(res.mfa)
```



It can be seen that, he first dimension of each group is highly correlated to the MFA's first one. The second dimension of the MFA is essentially correlated to the second dimension of the olfactory groups.

7.4 Summary

The multiple factor analysis (MFA) makes it possible to analyse individuals characterized by multiple sets of variables. In this article, we described how to perform and interpret MFA using FactoMineR and factoextra R packages.

7.5 Further reading

For the mathematical background behind MFA, refer to the following video courses, articles and books:

- Multiple Factor Analysis Course Using FactoMineR (Video courses). `https://goo.gl/WcmHHt`.

- Exploratory Multivariate Analysis by Example Using R (book) (Husson et al., 2017b).
- Principal component analysis (article) (Abdi and Williams, 2010). `https://goo.gl/1Vtwq1`.
- Simultaneous analysis of distinct Omics data sets with integration of biological knowledge: Multiple Factor Analysis approach (de Tayrac et al., 2009).

Part IV

Clustering

Chapter 8

HCPC: Hierarchical Clustering on Principal Components

8.1 Introduction

Clustering is one of the important data mining methods for discovering knowledge in multivariate data sets. The goal is to identify groups (i.e. clusters) of similar objects within a data set of interest. To learn more about clustering, you can read our book entitled “Practical Guide to Cluster Analysis in R” (`https://goo.gl/DmJ5y5`).

Briefly, the two most common clustering strategies are:

1. **Hierarchical clustering**, used for identifying groups of similar observations in a data set.
2. **Partitioning clustering** such as **k-means** algorithm, used for splitting a data set into several groups.

The HCPC (Hierarchical Clustering on Principal Components) approach allows us to combine the three standard methods used in multivariate data analyses (Husson et al., 2010):

1. Principal component methods (PCA, CA, MCA, FAMD, MFA),
2. Hierarchical clustering and
3. Partitioning clustering, particularly the k-means method.

> This chapter describes WHY and HOW to combine principal components and clustering methods. Finally, we demonstrate how to compute and visualize HCPC using R software.

8.2 Why HCPC?

Combining principal component methods and clustering methods are useful in at least three situations.

8.2.1 Case 1: Continuous variables

In the situation where you have a multidimensional data set containing multiple continuous variables, the *principal component analysis* (PCA) can be used to reduce the dimension of the data into few continuous variables containing the most important information in the data. Next, you can perform cluster analysis on the PCA results.

The PCA step can be considered as a denoising step which can lead to a more stable clustering. This might be very useful if you have a large data set with multiple variables, such as in gene expression data.

8.2.2 Case 2: Clustering on categorical data

In order to perform clustering analysis on categorical data, the *correspondence analysis* (CA, for analyzing contingency table) and the *multiple correspondence analysis* (MCA, for analyzing multidimensional categorical variables) can be used to transform categorical variables into a set of few continuous variables (the principal components). The cluster analysis can be then applied on the (M)CA results.

In this case, the *(M)CA* method can be considered as pre-processing steps which allow to compute clustering on categorical data.

8.2.3 Case 3: Clustering on mixed data

When you have a mixed data of continuous and categorical variables, you can first perform FAMD (*factor analysis of mixed data*) or MFA (*multiple factor analysis*). Next, you can apply cluster analysis on the FAMD/MFA outputs.

8.3 Algorithm of the HCPC method

The algorithm of the HCPC method, as implemented in the FactoMineR package, can be summarized as follow:

1. *Compute principal component methods*: PCA, (M)CA or MFA depending on the types of variables in the data set and the structure of the data set. At this step, you can choose the number of dimensions to be retained in the output by specifying the argument *ncp*. The default value is 5.
2. *Compute hierarchical clustering*: Hierarchical clustering is performed using the *Ward's criterion* on the selected principal components. Ward criterion is used in the hierarchical clustering because it is based on the multidimensional variance like principal component analysis.
3. *Choose the number of clusters based on the hierarchical tree*: An initial partitioning is performed by cutting the hierarchical tree.
4. *Perform K-means clustering* to improve the initial partition obtained from hierarchical clustering. The final partitioning solution, obtained after consolidation with

k-means, can be (slightly) different from the one obtained with the hierarchical clustering.

8.4 Computation

8.4.1 R packages

We'll use two R packages: i) *FactoMineR* for computing HCPC and ii) *factoextra* for visualizing the results.

To install the packages, type this:

```
install.packages(c("FactoMineR", "factoextra"))
```

After the installation, load the packages as follow:

```
library(factoextra)
library(FactoMineR)
```

8.4.2 R function

The function *HCPC*() [in *FactoMineR* package] can be used to compute hierarchical clustering on principal components.

A simplified format is:

```
HCPC(res, nb.clust = 0, min = 3, max = NULL, graph = TRUE)
```

- **res**: Either the result of a factor analysis or a data frame.
- **nb.clust**: an integer specifying the number of clusters. Possible values are:
  - *0*: the tree is cut at the level the user clicks on
  - *-1*: the tree is automatically cut at the suggested level
  - *Any positive integer*: the tree is cut with nb.clusters clusters
- **min, max**: the minimum and the maximum number of clusters to be generated, respectively
- **graph**: if TRUE, graphics are displayed

8.4.3 Case of continuous variables

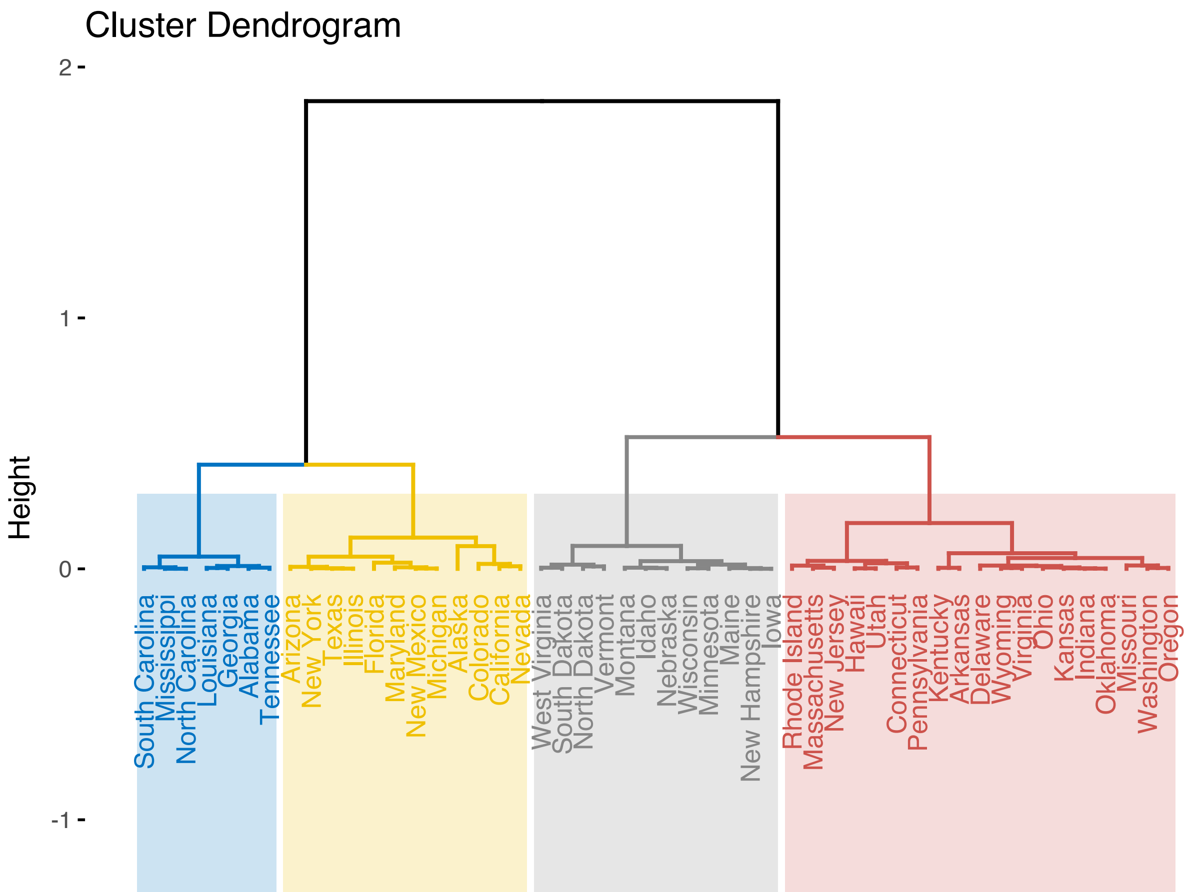
We start by computing again the principal component analysis (PCA). The argument *ncp = 3* is used in the function *PCA*() to keep only the first three principal components. Next, the HCPC is applied on the result of the PCA.

```
library(FactoMineR)
# Compute PCA with ncp = 3
res.pca <- PCA(USArrests, ncp = 3, graph = FALSE)
```

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

To visualize the dendrogram generated by the hierarchical clustering, we'll use the function *fviz_dend*() [in *factoextra* package]:

```
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
          )
```

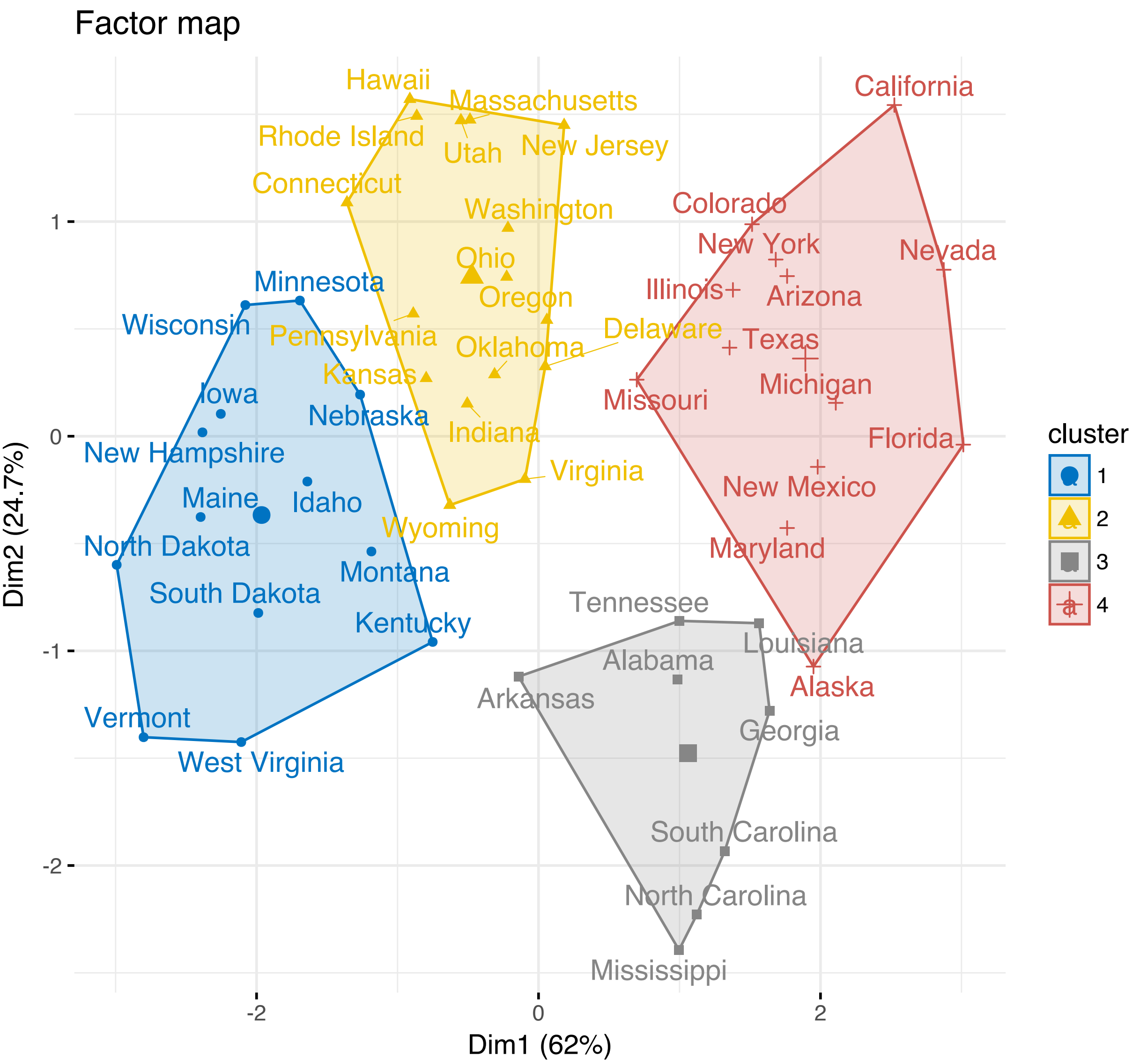


The dendrogram suggests 4 clusters solution.

It's possible to visualize individuals on the principal component map and to color individuals according to the cluster they belong to. The function *fviz_cluster*() [in *factoextra*] can be used to visualize individuals clusters.

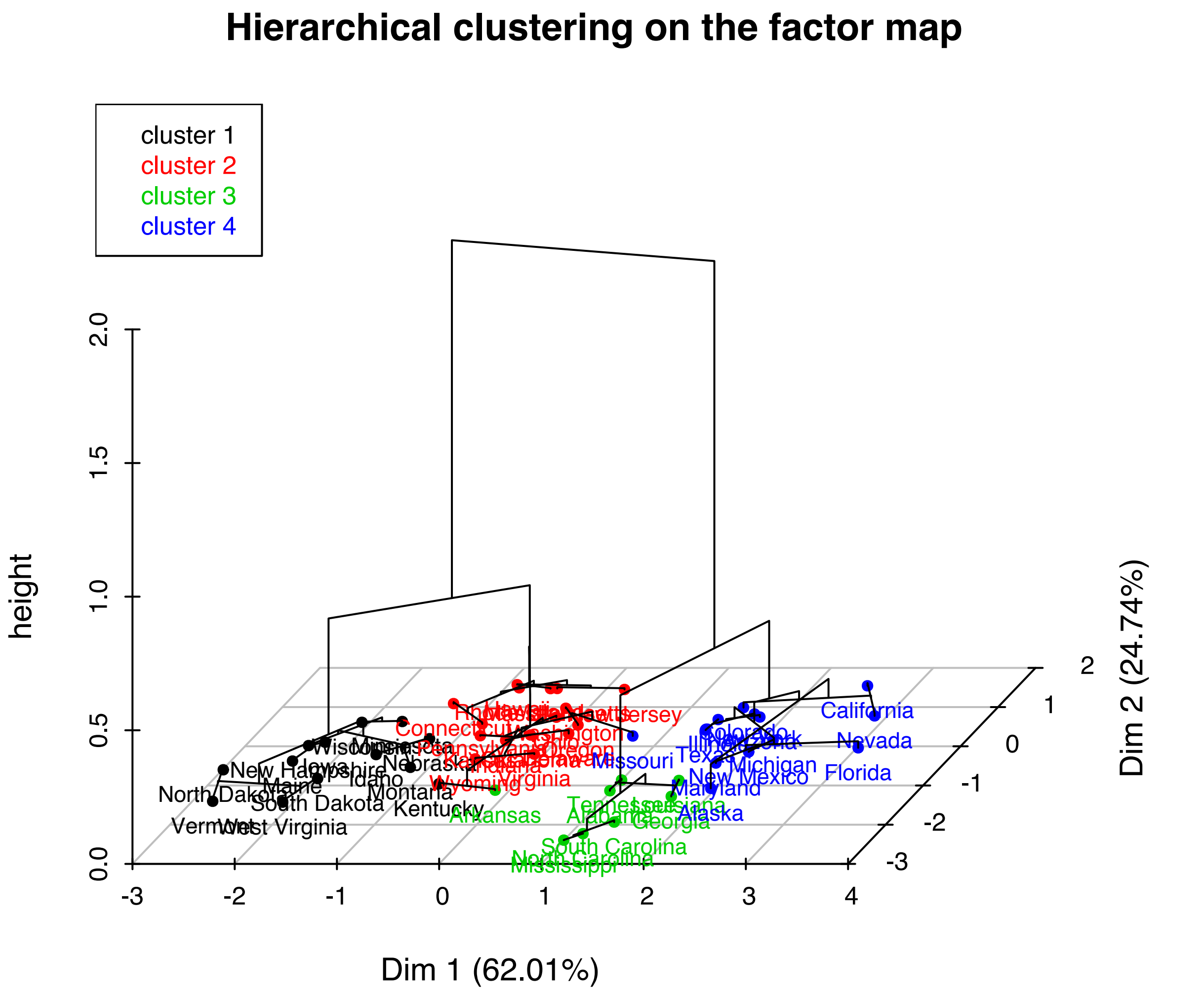
```
fviz_cluster(res.hcpc,
             repel = TRUE,            # Avoid label overlapping
             show.clust.cent = TRUE, # Show cluster centers
             palette = "jco",         # Color palette see ?ggpubr::ggpar
             ggtheme = theme_minimal(),
```

```
             main = "Factor map"
             )
```



You can also draw a three dimensional plot combining the hierarchical clustering and the factorial map using the R base function *plot*():

```
# 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

To display the original data with cluster assignments, type this:

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

```
##             Murder Assault UrbanPop Rape clust
## Alabama       13.2     236       58 21.2     3
## Alaska        10.0     263       48 44.5     4
## Arizona        8.1     294       80 31.0     4
## Arkansas       8.8     190       50 19.5     3
## California     9.0     276       91 40.6     4
## Colorado       7.9     204       78 38.7     4
## Connecticut    3.3     110       77 11.1     2
## Delaware       5.9     238       72 15.8     2
## Florida       15.4     335       80 31.9     4
## Georgia       17.4     211       60 25.8     3
```

In the table above, the last column contains the cluster assignments.

To display quantitative variables that describe the most each cluster, type this:

```
res.hcpc$desc.var$quanti
```

Here, we show only some columns of interest: "Mean in category", "Overall Mean", "p.value"

```
## $`1`
##          Mean in category Overall mean  p.value
## UrbanPop             52.1        65.54 9.68e-05
## Murder                3.6         7.79 5.57e-05
## Rape                 12.2        21.23 5.08e-05
## Assault              78.5       170.76 3.52e-06
##
## $`2`
##          Mean in category Overall mean p.value
## UrbanPop            73.88        65.54 0.00522
## Murder               5.66         7.79 0.01759
##
## $`3`
##          Mean in category Overall mean  p.value
## Murder               13.9         7.79 1.32e-05
## Assault             243.6       170.76 6.97e-03
## UrbanPop             53.8        65.54 1.19e-02
##
## $`4`
##          Mean in category Overall mean  p.value
## Rape                 33.2        21.23 8.69e-08
## Assault             257.4       170.76 1.32e-05
## UrbanPop             76.0        65.54 2.45e-03
## Murder               10.8         7.79 3.58e-03
```

From the output above, it can be seen that:

- the variables UrbanPop, Murder, Rape and Assault are most significantly associated with the cluster 1. For example, the mean value of the Assault variable in cluster 1 is 78.53 which is less than it's overall mean (170.76) across all clusters. Therefore, It can be conclude that the cluster 1 is characterized by a low rate of Assault compared to all clusters.
- the variables UrbanPop and Murder are most significantly associated with the cluster 2.

...and so on ...

Similarly, to show principal dimensions that are the most associated with clusters, type this:

```
res.hcpc$desc.axes$quanti
```

```
## $`1`
##       Mean in category Overall mean  p.value
## Dim.1            -1.96    -5.64e-16 2.27e-07
##
## $`2`
##       Mean in category Overall mean  p.value
## Dim.2            0.743    -5.37e-16 0.000336
##
## $`3`
##       Mean in category Overall mean  p.value
## Dim.1            1.061    -5.64e-16 3.96e-02
## Dim.3            0.397     3.54e-17 4.25e-02
## Dim.2           -1.477    -5.37e-16 5.72e-06
##
## $`4`
##       Mean in category Overall mean  p.value
## Dim.1             1.89    -5.64e-16 6.15e-07
```

The results above indicate that, individuals in clusters 1 and 4 have high coordinates on axes 1. Individuals in cluster 2 have high coordinates on the second axis. Individuals who belong to the third cluster have high coordinates on axes 1, 2 and 3.

Finally, representative individuals of each cluster can be extracted as follow:

```
res.hcpc$desc.ind$para
```

```
## Cluster: 1
##         Idaho  South Dakota         Maine          Iowa New Hampshire
##         0.367         0.499         0.501         0.553         0.589
## --------------------------------------------------------
## Cluster: 2
##         Ohio     Oklahoma Pennsylvania       Kansas      Indiana
##        0.280        0.505        0.509        0.604        0.710
## --------------------------------------------------------
## Cluster: 3
##        Alabama South Carolina        Georgia      Tennessee      Louisiana
##          0.355          0.534          0.614          0.852          0.878
## --------------------------------------------------------
## Cluster: 4
##   Michigan    Arizona New Mexico   Maryland      Texas
##      0.325      0.453      0.518      0.901      0.924
```

For each cluster, the top 5 closest individuals to the cluster center is shown. The distance between each individual and the cluster center is provided. For example, representative individuals for cluster 1 include: Idaho, South Dakota, Maine, Iowa and New Hampshire.

8.4.4 Case of categorical variables

For categorical variables, compute CA or MCA and then apply the function *HCPC*() on the results as described above.

Here, we'll use the *tea* data [in *FactoMineR*] as demo data set: Rows represent the individuals and columns represent categorical variables.

We start, by performing an MCA on the individuals. We keep the first 20 axes of the MCA which retain 87% of the information.

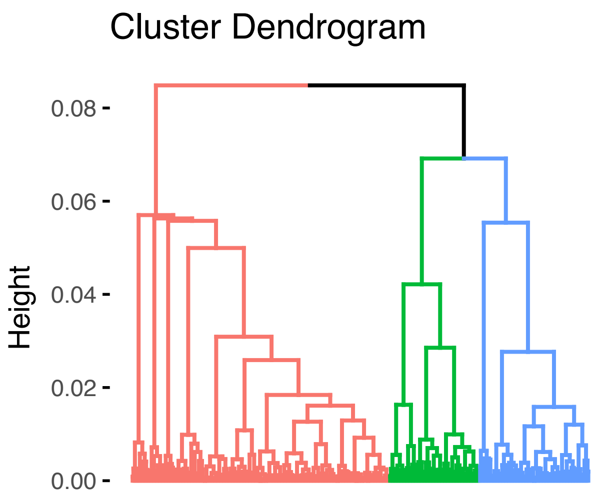
```
# Loading data
library(FactoMineR)
data(tea)

# Performing MCA
res.mca <- MCA(tea,
               ncp = 20,            # Number of components kept
               quanti.sup = 19,     # Quantitative supplementary variables
               quali.sup = c(20:36), # Qualitative supplementary variables
               graph=FALSE)
```

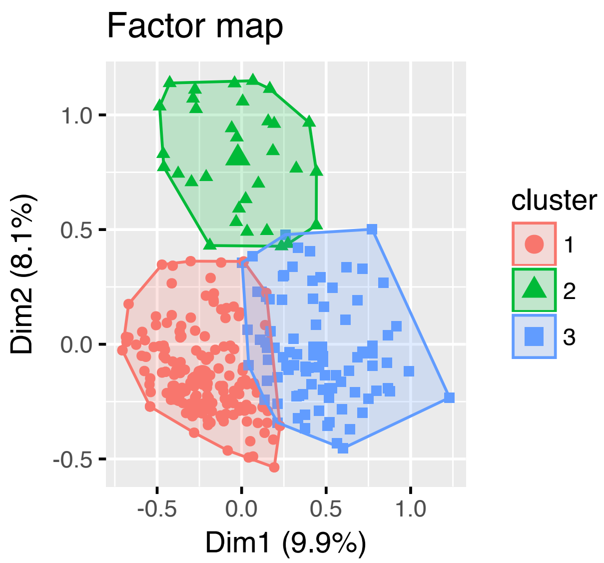
Next, we apply hierarchical clustering on the results of the MCA:

```
res.hcpc <- HCPC (res.mca, graph = FALSE, max = 3)
```

The results can be visualized as follow:

```
# Dendrogram
fviz_dend(res.hcpc, show_labels = FALSE)

# Individuals facor map
fviz_cluster(res.hcpc, geom = "point", main = "Factor map")
```





As mentioned above, clusters can be described by i) variables and/or categories, ii) principal axes and iii) individuals. In the example below, we display only a subset of the results.

- **Description by variables and categories**

```
# Description by variables
res.hcpc$desc.var$test.chi2
```

```
##          p.value df
## where   8.47e-79  4
## how     3.14e-47  4
## price   1.86e-28 10
## tearoom 9.62e-19  2
```

```
# Description by variable categories
res.hcpc$desc.var$category
```

```
## $`1`
##                       Cla/Mod Mod/Cla Global  p.value
## where=chain store        85.9    93.8   64.0 2.09e-40
## how=tea bag              84.1    81.2   56.7 1.48e-25
## tearoom=Not.tearoom      70.7    97.2   80.7 1.08e-18
## price=p_branded          83.2    44.9   31.7 1.63e-09
##
## $`2`
##                   Cla/Mod Mod/Cla Global  p.value
## where=tea shop       90.0    84.4   10.0 3.70e-30
## how=unpackaged       66.7    75.0   12.0 5.35e-20
## price=p_upscale      49.1    81.2   17.7 2.39e-17
## Tea=green            27.3    28.1   11.0 4.44e-03
##
## $`3`
##                             Cla/Mod Mod/Cla Global  p.value
## where=chain store+tea shop     85.9    72.8   26.0 5.73e-34
## how=tea bag+unpackaged         67.0    68.5   31.3 1.38e-19
## tearoom=tearoom                77.6    48.9   19.3 1.25e-16
## pub=pub                        63.5    43.5   21.0 1.13e-09
```

The variables that characterize the most the clusters are the variables "where" and "how". Each cluster is characterized by a category of the variables "where" and "how". For example, individuals who belong to the first cluster buy tea as tea bag in chain stores.

- **Description by principal components**

```
res.hcpc$desc.axes
```

- **Description by Individuals**

```
res.hcpc$desc.ind$para
```

8.5 Summary

We described how to compute hierarchical clustering on principal components (HCPC). This approach is useful in situations, including:

- When you have a large data set containing continuous variables, a principal component analysis can be used to reduce the dimension of the data before the hierarchical clustering analysis.

- When you have a data set containing categorical variables, a (Multiple)Correspondence analysis can be used to transform the categorical variables into few continuous principal components, which can be used as the input of the cluster analysis.

We used the FactoMineR package to compute the HCPC and the factoextra R package for ggplot2-based elegant data visualization.

8.6 Further reading

- Practical guide to cluster analysis in R (Book). `https://goo.gl/DmJ5y5`
- HCPC: Hierarchical Clustering on Principal Components (Videos). `https://goo.gl/jdYGoK`

Bibliography

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *John Wiley and Sons, Inc. WIREs Comp Stat*, 2:433–459.

Bendixen, M. (2003). A practical guide to the use of correspondence analysis in marketing research. *Marketing Bulletin*, 14.

Bendixen, M. T. (1995). Compositional perceptual mapping using chi-squared trees analysis and correspondence analysis. *Journal of Marketing Management*, 11(6):571–581.

de Tayrac, M., Lê, S., Aubry, M., Mosser, J., and Husson, F. (2009). Simultaneous analysis of distinct omics data sets with integration of biological knowledge: Multiple factor analysis approach. *BMC Genomics*, 10(1):32.

Gabriel, K. R. and Odoroff, C. L. (1990). Biplots in biomedical research. *Statistics in Medicine*, 9(5):469–485.

Greenacre, M. (2013). Contribution biplots. *Journal of Computational and Graphical Statistics*, 22(1):107–122.

Husson, F., Josse, J., and J., P. (2010). Principal component methods - hierarchical clustering - partitional clustering: why would we need to choose for visualizing data? *Unpublished data*.

Husson, F., Josse, J., Le, S., and Mazet, J. (2017a). *FactoMineR: Multivariate Exploratory Data Analysis and Data Mining*. R package version 1.36.

Husson, F., Le, S., and Pagès, J. (2017b). *Exploratory Multivariate Analysis by Example Using R*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1138196346.

Jollife, I. (2002). *Principal component analysis*. Springer-Verlag, New York, 2nd edition. ISBN 0-387-95442-2.

Kaiser, H. F. (1961). A note on guttman's lower bound for the number of common factors. *British Journal of Statistical Psychology*, 14:1–2.

Kassambara, A. and Mundt, F. (2017). *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.5.999.

Nenadic, O. and Greenacre, M. (2007). Correspondence analysis in r, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, 20(3):1–13.

Pagès, J. (2002). Analyse factorielle multiple appliquée aux variables qualitatives et aux données mixtes. *Revue Statistique Appliquee*, 4:5–37.

Pagès, J. (2004). Analyse factorielle de donnees mixtes. *Revue Statistique Appliquee*, 4:93–111.

Peres-Neto, P. R., Jackson, D. A., and Somers, K. M. (2005). How many principal components? stopping rules for determining the number of non-trivial axes revisited. *British Journal of Statistical Psychology*, 49:974–997.

Index