DATA EXPLORATION AND FEATURE REDUCTION

E. Fersini

WHY FEATURE REDUCTION?

Why even think about Feature Reduction?

- Naive theoretical view:
  More features
  => More information
  => More discrimination power.

- In practice:
  many reasons why this is not the case!
  - We need to identify features with a strong discriminative/predictive power

WHY FEATURE REDUCTION?

Chiwawa or Muffin?

WHY FEATURE REDUCTION?

- Many explored domains have hundreds to tens of thousands of variables/ features with many irrelevant and redundant ones!

- In domains with many features the underlying probability distribution can be very complex and very hard to estimate (e.g. dependencies between variables)!

- Irrelevant and redundant features can “confuse” the models!

- Limited training data!

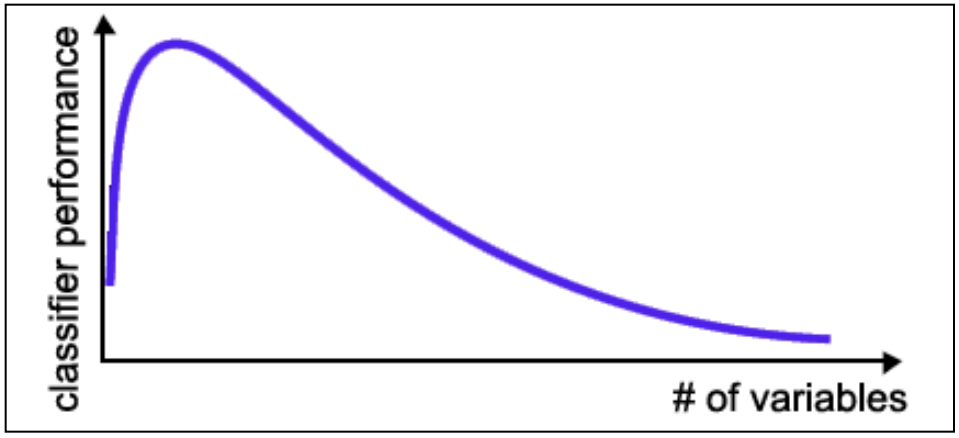
- Limited computational resources!

- **Curse of dimensionality**!

CURSE OF DIMENSIONALITY

- The required number of samples (to achieve the same accuracy) grows exponentionally with the number of variables!
- In practice: number of training examples is fixed!

  => the model's performance usually will degrade for a large number of features!



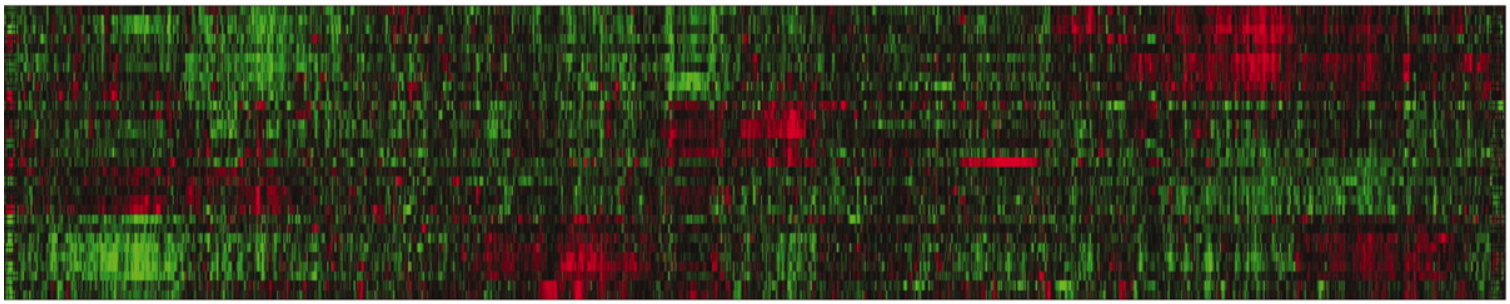
In many cases the information that is lost by discarding variables is made up for by a more accurate training in the lower-dimensional space!

EXAMPLE OF COD

Gene selection from microarray data

- Attributes/Features:
  gene expression coefficients corresponding to the amount of mRNA in a patien's sample (e.g. tissue biopsy)
- Task: Seperate healthy patients from cancer patients
- Usually there are only about 100 instances (patients) available for training and testing (!!!)
- Number of attributes/features (gene expression values) in the raw data: 6.000 – 60.000

patients

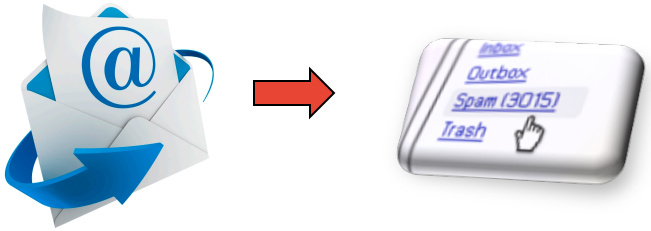


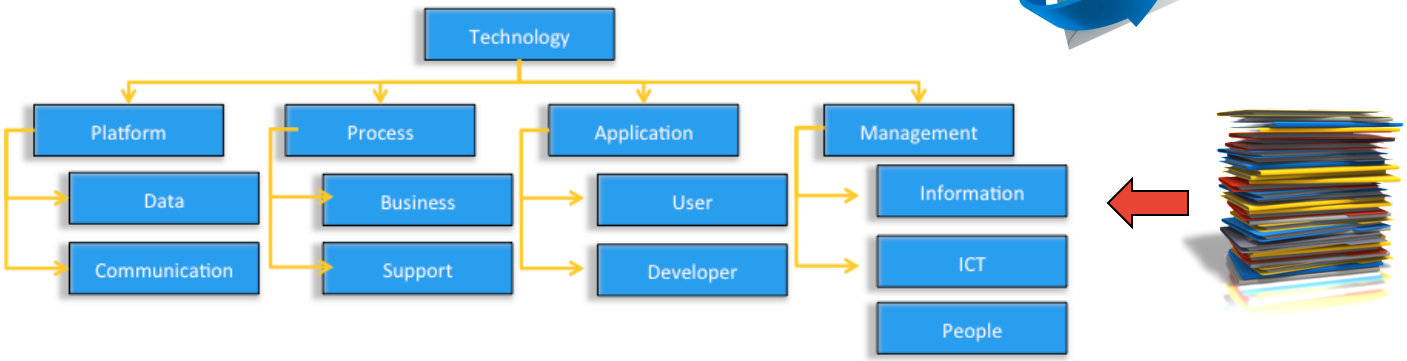
genes

EXAMPLE OF COD

Text-Categorization

- Documents are represented by a vector of dimension the size of the vocabulary containing word frequency counts
- Vocabulary (attributes) 15.000 words
  - each document is represented by a 15.000-dimensional vector
- Typical tasks:
  - Creation of taxonomies of documents
  - Detection of spam-email





COD AND HYPOTHESIS SPACES

Problem: make a decision whether to wait for a table at a restaurant, based on the following attributes:

1. **Alternate:** is there an alternative restaurant nearby?
2. **Bar:** is there a comfortable bar area to wait in?
3. **Fri/Sat:** is today Friday or Saturday?
4. **Hungry:** are we hungry?
5. **Patrons:** number of people in the restaurant (None, Some, Full)
6. **Price:** price range ($, $$, $$$)
7. **Raining:** is it raining outside?
8. **Reservation:** have we made a reservation?
9. **Type:** kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate:** estimated waiting time (0-10, 10-30, 30-60, >60)

COD AND HYPOTHESIS SPACES

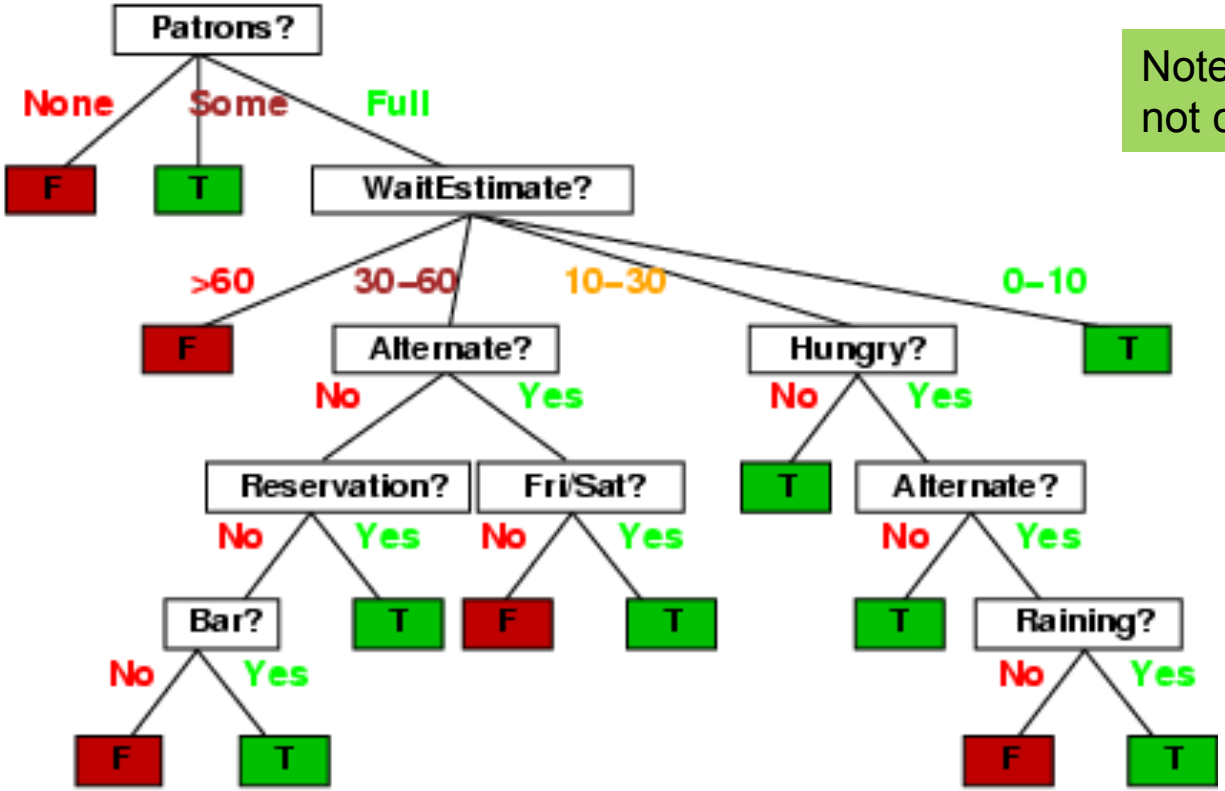
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

COD AND HYPOTHESIS SPACES

- One possible representation for hypotheses
- E.g., here is the "true" tree for deciding whether to wait:

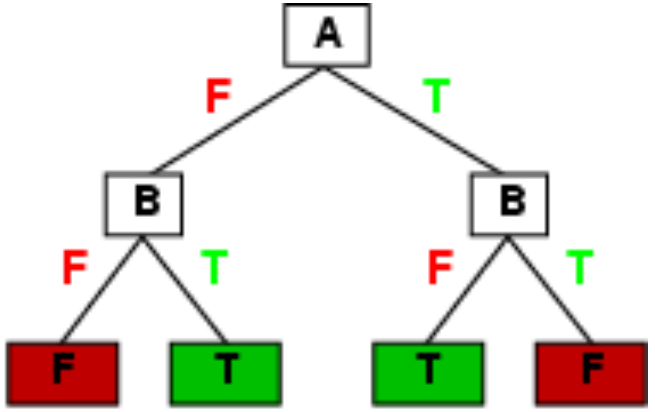


Note that ***type*** is not considered

COD AND HYPOTHESIS SPACES

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

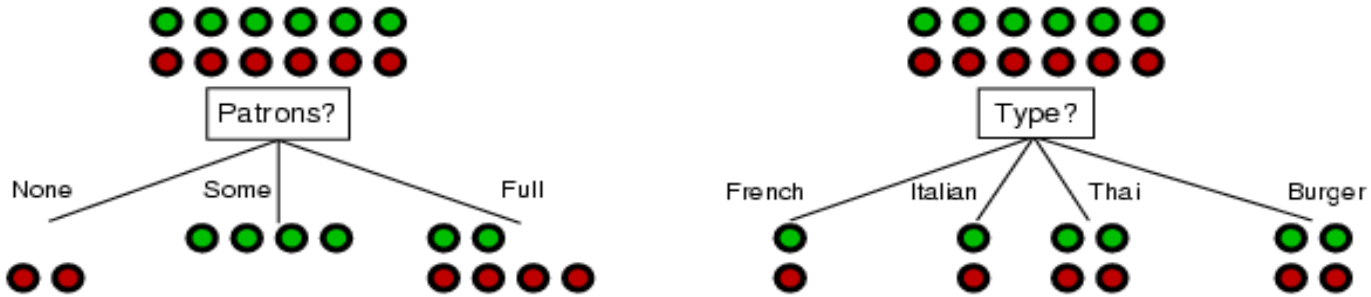


- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples

- Prefer to find more compact decision trees

COD AND HYPOTHESIS SPACES

- Limiting COD: find good features!
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



- *Patrons?* is a better choice

COD AND HYPOTHESIS SPACES

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions of n arguments

= number of distinct truth tables with 2^n rows

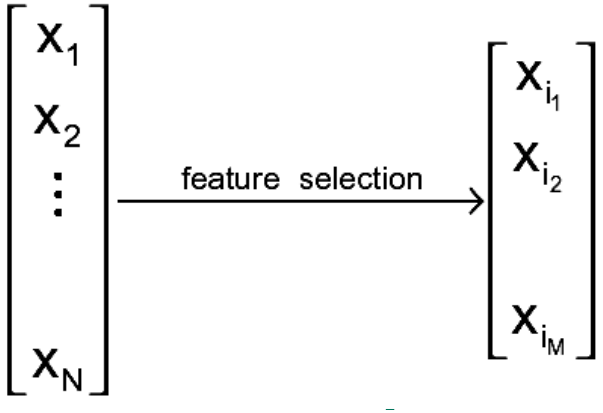
= 2^{2^n} truth tables (since each row has 2 possible function values)

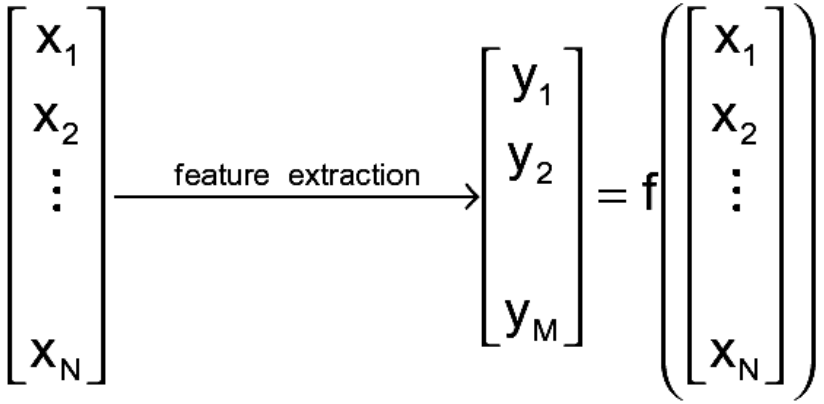
E.g., with 6 Boolean attributes, there are

$2^{2^6} = 2^{64} =$ **18,446,744,073,709,551,616** possible functions/trees

FEATURE SELECTION VS FEATURE EXTRACTION

- **Two general approaches for dimensionality reduction**
  - Feature extraction: Transforming the existing features into a lower dimensional space
  - Feature selection: Selecting a subset of the existing features without a transformation





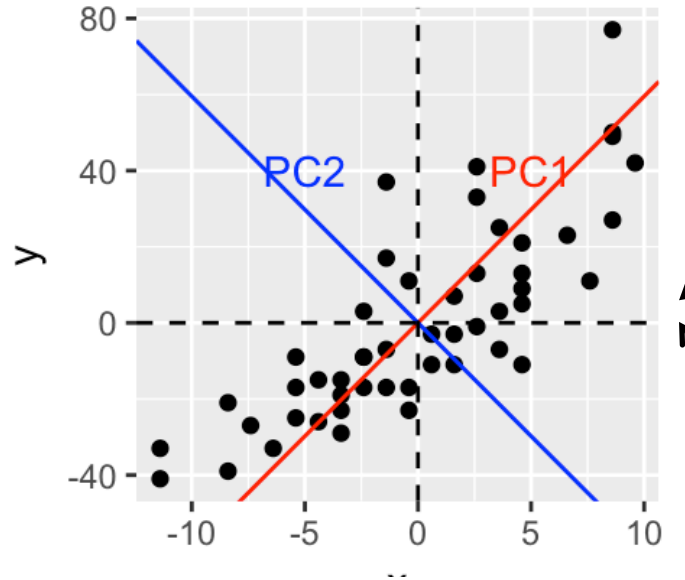
FEATURE EXTRACTION

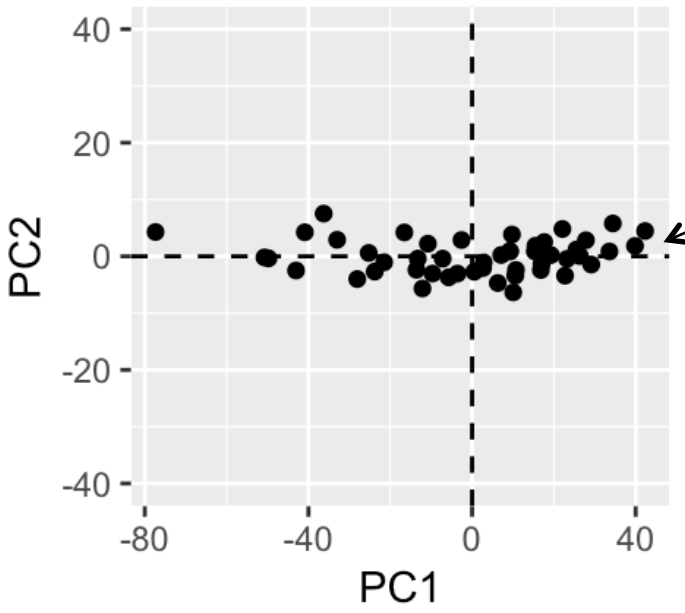
- Given a feature set $x=\{x_i \mid i=1...N\}$ find a mapping $y= f(x):R^N->R^M$ with M<N, such that the transformed feature vector y_i preserves (most of) the information or structure in R^N.
- An *optimal mapping* $y= f(x)$ will be one that results in no increase in the minimum probability of error
  - **Methods**:
    - Linear Transformations: **Principal Component Analysis**

PRINCIPAL COMPONENT ANALYSIS

- PCA is a type of **linear transformation** that fits the dataset to a new coordinate system:
  - most **significant variance** is found on the **first coordinate**
  - each **subsequent coordinate** is orthogonal to the last and has **less variance**.
- In practice:
  - we transform a set of x correlated variables to a set of p uncorrelated principal components.

UNDERSTANDING PCA





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".

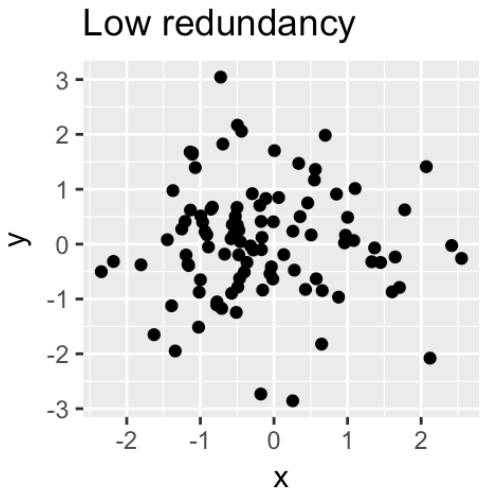
PC1 axis is the first **principal direction** along which the samples show the **largest variation**.

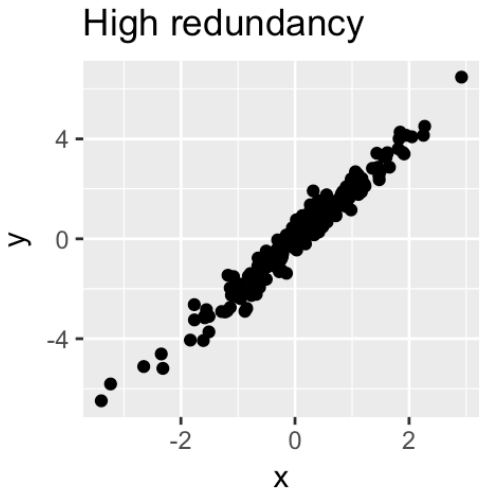
PC2 axis is the second most important direction and it is **orthogonal** to the PC1 axis.

The dimensionality of our data can be reduced to a single dimension by projecting each sample onto the first principal component.

UNDERSTANDING PCA

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





PCA IN PRACTICE

- 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],
  - *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.

PCA IN PRACTICE

- Install the two packages as follow and load them:

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

- We will use the demo data sets *decathlon2* from the factoextra package:

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

PCA IN PRACTICE

- The dataset describes athletes' performance during two sporting events (Desctar and OlympicG). It contains 27 individuals described by 13 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

PCA IN PRACTICE

Active individuals during PCA.

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

PCA IN PRACTICE

Supplementary individuals: The coordinates of these individuals will be predicted using the PCA information

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

PCA IN PRACTICE

Active variables
used for PCA

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

PCA IN PRACTICE

Supplementary quantitative variables
predicted by PCA

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

PCA IN PRACTICE

Supplementary qualitative variables
used to group individuals

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

PCA IN PRACTICE

- 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
```

PCA - STANDARDIZATION

- In PCA, **variables** are often **scaled** (i.e. standardized). This is particularly recommended when variables are measured in different scales (e.g: 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 standard deviation equal to one and mean equal to zero.
- 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).

PCA - STANDARDIZATION

- 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**.

- REMARK: by default, the function PCA() [in FactoMineR], standardizes the data automatically during the PCA

PCA IN PRACTICE

- 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.

PCA IN PRACTICE

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

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

- Results of PCA can be easily visualized and interpreted

PCA IN PRACTICE

- We'll use the factoextra R package to help in the interpretation of PCA.
- 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.

PCA - EIGENVALUES

- 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.
  - 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.
  - We analyse the eigenvalues and the proportion of variances (i.e., information) retained by the principal components (PCs).

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

PCA - EIGENVALUES

Proportion of variation explained by each eigenvalue.

Variance percentage: 4.124 over 10 equals 0.4124. This means that 41.24% of the variation is explained by this first eigenvalue.

	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.

Cumulative percentage is obtained by adding the successive proportions of variation explained to obtain the running total. This means that about 59.627% of the variation is explained by the first two eigenvalues together.

PCA - EIGENVALUES

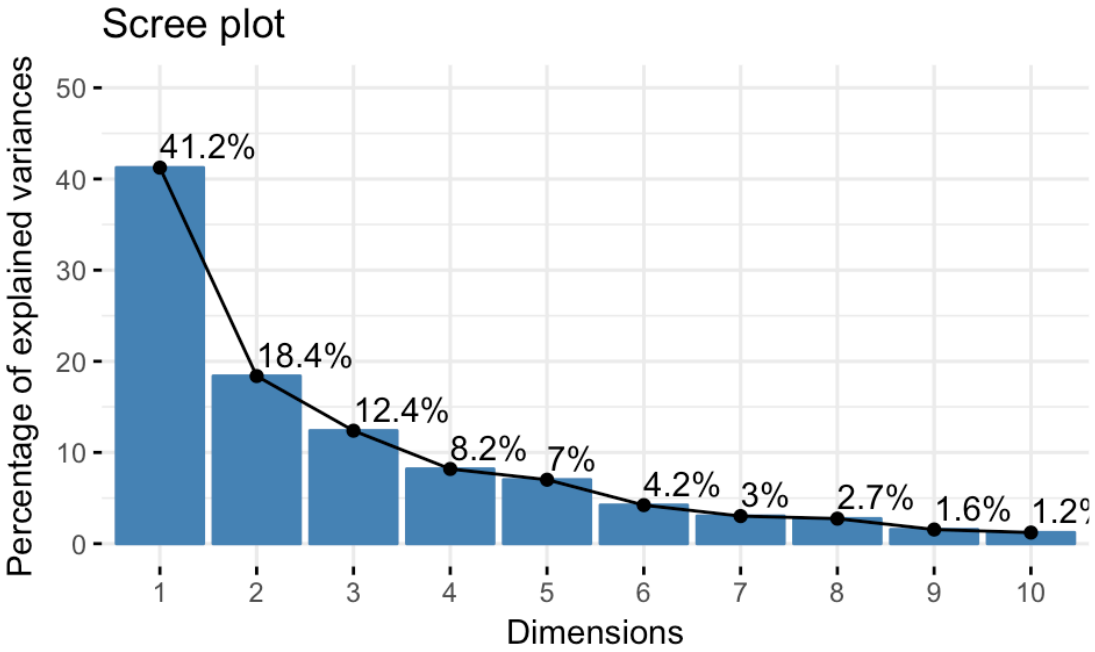
- Eigenvalues can be used to determine the number of principal components to retain after PCA (Kaiser 1961):
  - An **eigenvalue > 1** indicates that PCs account for more variance than accounted by one of the original variables in standardized data. This is commonly used as a cutoff point for which PCs 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.

PCA - EIGENVALUES

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

PCA - EIGENVALUES

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



PCA - VARIABLES

- A simple method to extract the results, for variables, from a PCA output is to use the function *get_pca_var()* [factoextra package].

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

- 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).

PCA - VARIABLES

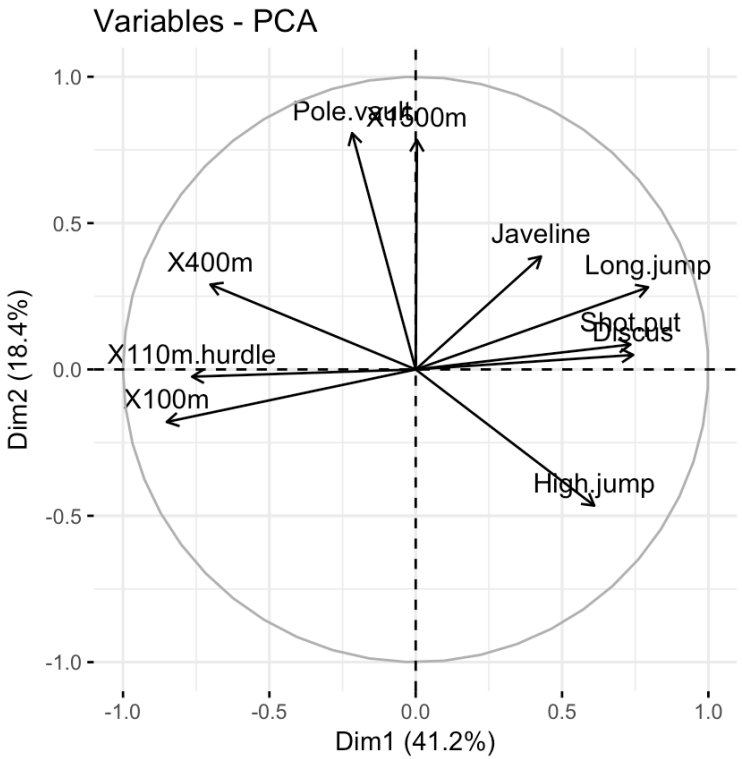
- The correlation between a variable and a principal component (PC) is used as the **coordinates of the variable** on the PC.
- The representation of variables differs from the plot of the observations:
  - The variables are represented by their correlations.

```
# 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
```

PCA - VARIABLES

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



- 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. Variables that are away from the origin are well represented.

PCA - INDIVIDUALS

- The information 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 individuals 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"
```

PCA - INDIVIDUALS

- The information 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 individuals 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"
```

PCA - INDIVIDUALS

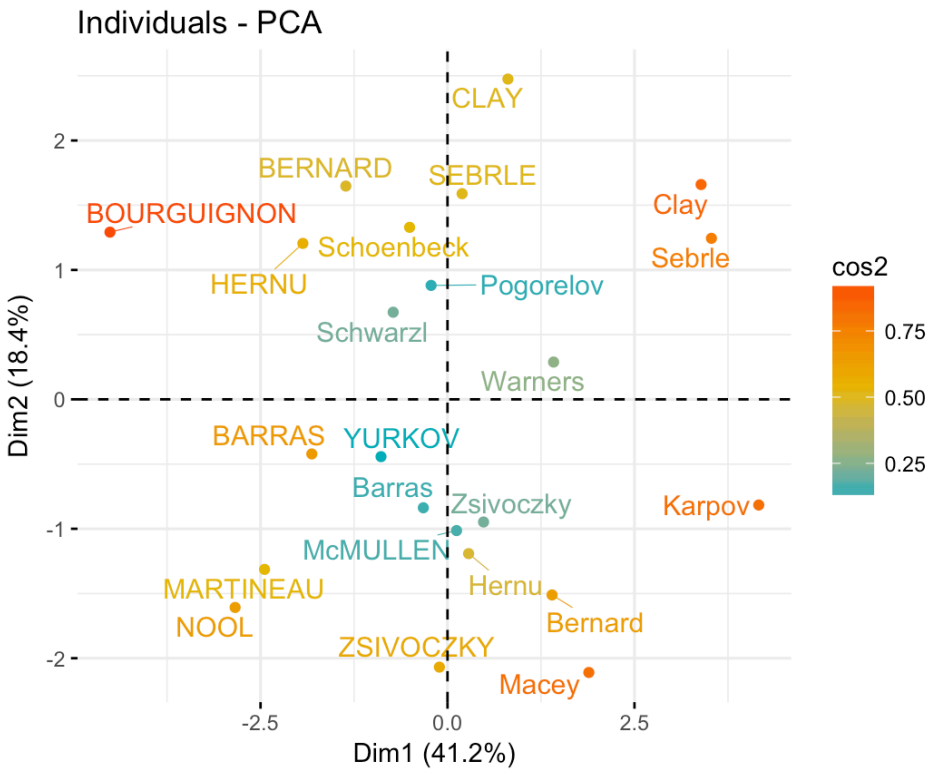
- Plots for individuals: quality and contribution
  - The *fviz_pca_ind()* is used to produce the graph of individuals:

```
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)
)
```

PCA - INDIVIDUALS



- A high cos2 indicates a good representation of the individual on the principal component.
- A low cos2 indicates that the individual is not perfectly represented by the PCs.

ASSIGNMENT

- Try to analyse the iris dataset to find:
  - How many components are necessary to explain the variance of the data
  - Determine if the selected components are "good" to explain different species

- TIP:

```
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"
             )
```