BIOSTATISTICS

Correlation and regression models. Grouping data with clustering methods.

Angela Jimeno Martin (ajimeno@usj.es)

June 2022

Contents

Introduction **1**
Linear regression model . . . 1
Goodness of fit . . . 6

Clustering methods **7**
K-Means algorithm . . . 8
Hierarchical clustering . . . 10

Exercises **13**

Bibliography **14**

Introduction

Linear regression models are based on how two quantitative variables, one **explanatory** and one **response**, are related to each other. They are an alternative study to those carried out in the previous units to try to create a model of how two variables are involved in a biological phenomenon. The idea of a model is to predict unknown values of the response variable as a function of the values of the explanatory variable, henceforth referred to as prediction.

However, before calculate the regression model, it is essential to prove that two variables are related, which can be assess though the value of **covariance**.

$$S_{xy} = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{n}$$

If the value of S_{xy} is distinct to 0, there is a correlation between x and y variables and then we may proceed adressing if the relation follows a linear model.

Linear regression model

The formula for a function to draw a straight line has the form:

$$y = bx + a$$

Where b is the *slope* and a is the independent term or *intercept.* The predictor variable is x and the response is y. In the real world finding a relationship between variables that results in a perfect straight line is almost impossible. But simple linear regression attempts to fit the data we have for several variables to a line that best represents the biological phenomenon being analysed.

The slope establishes the linear relationship that exists between the response variable and the predictor, and is interpreted as the estimator of change in the response variable associated with a one-unit increase in the predictor variable, which would be expected to increase on average as we move from unit to unit in the predictor variable.

In order to fit a linear model to a data set, it must be assumed that the data have certain characteristics:

1. For a given value of x, the true value of the mean value of y depends on x. This means that in a linear regression what is represented on the line are the mean values of y for a value of x, i.e. not exact ($\mu_{y|x}$).
2. Each observation in the y data set is independent, i.e. $y_i \neq y_j$.
3. Based on the above, the linear equation obtained in a linear regression is:

$$\mu_{y|x} = \beta_1 x + \beta_0$$

where β_1 is the slope and β_0 the ordinate at the origin.

4. The variance of y_i is constant, i.e. it has the property of homoscedasticity.
5. The distribution of response values, andi are normal.

The formula in point 3 talks about the average value that the response variable y_i can have for a given x. However, in our data set what we will have is an observed y_i value. Thus, the difference between the mean value provided by the linear regression ($\mu_{y|x}$) and the observed value in the data set (y_i) is the error or residual or *residual error*.

Thus, the value of y_i observed in the data set corresponds to:

$$y_i = \beta_1 x + \beta_0 + \epsilon_i$$

The residual error is thus the difference between the values estimated by the line and the observed values in the dataset.

$$\epsilon_i = y_i - \mu_{y|x}$$

In linear regression the sum of all these squared differences gives an idea of the variation that exists between the model and the actual data provided by the data set:

$$RSS = \sum_{i}^{n} \epsilon_i^2$$

This is called the **Sum of Squares of Residues (RSS)**.

If it is decided to make a linear fit by minimising the value of RSS as much as possible, the least-squares method is being used to make the fit, and the line obtained is called Least-squares regression line.

In this adjustment, the theoretical values ofβ_1 and β_0 , must be estimated, therefore their corresponding non-Greek letters are used. The calculation based on the least squares method of each estimator is performed as follows:

$$b_1 = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sum(x_i - \overline{x})^2}$$

$$b_0 = \overline{y} - b_1 \cdot \overline{x}$$

The R function that performs this calculation is `lm` and the parameters must be passed to it in the formula format: `y ~ x`.

`Example`: You have the following expression data for two genes:

```r
genes <- data.frame(gene1=c(-1.06,-0.81,-0.48,-0.42,-0.30,-0.35,-0.31,-0.18,-0.20,-0.11
                            ,-0.09,0.16,0.45,0.53,0.67,0.80,0.87,0.92),
                    gene2=c(-1.08,-1.02,-0.39,-0.48,-0.58,-0.24,-0.05,-0.33,0.51,-0.53,-0.47
                            ,0.10,0.39,0.11,0.52,0.34,1.08,1.21))
genes
```

```
##    gene1 gene2
## 1  -1.06 -1.08
## 2  -0.81 -1.02
## 3  -0.48 -0.39
## 4  -0.42 -0.48
## 5  -0.30 -0.58
## 6  -0.35 -0.24
## 7  -0.31 -0.05
## 8  -0.18 -0.33
## 9  -0.20  0.51
## 10 -0.11 -0.53
## 11 -0.09 -0.47
## 12  0.16  0.10
## 13  0.45  0.39
## 14  0.53  0.11
## 15  0.67  0.52
## 16  0.80  0.34
## 17  0.87  1.08
## 18  0.92  1.21
```

We plot the values of both variables against each other

```r
plot(x = genes$gene1,genes$gene2,main="Gene2 vs Gene1",pch=16,
     xlab="Gene1", ylab="Gene2")
```

The aim is to find a line that is valid for predicting the values of gene 2 based on the observations of gene 1. Calculating the values of b_1 and b_0 manually:

$$b_1 = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sum(x_i - \overline{x})^2}$$

```r
gene1.mean <- mean(genes$gene1)
gene2.mean <- mean(genes$gene2)
b1 <- (sum((genes$gene1-gene1.mean)*(genes$gene2-gene2.mean)))/(sum((genes$gene1-gene1.mean)^2))
b1
```

```
## [1] 0.9707003
```

$$b_0 = \overline{y} - b_1 \cdot \overline{x}$$

```r
b0 <- gene2.mean - b1*gene1.mean
b0
```

```
## [1] -0.05540906
```

With this formula we would obtain the equation of the least squares regression line. By substituting observed values for x_i we would obtain the expected means for each y according to x.

$$\mu_{y|x} = 0.9707x - 0.0554$$

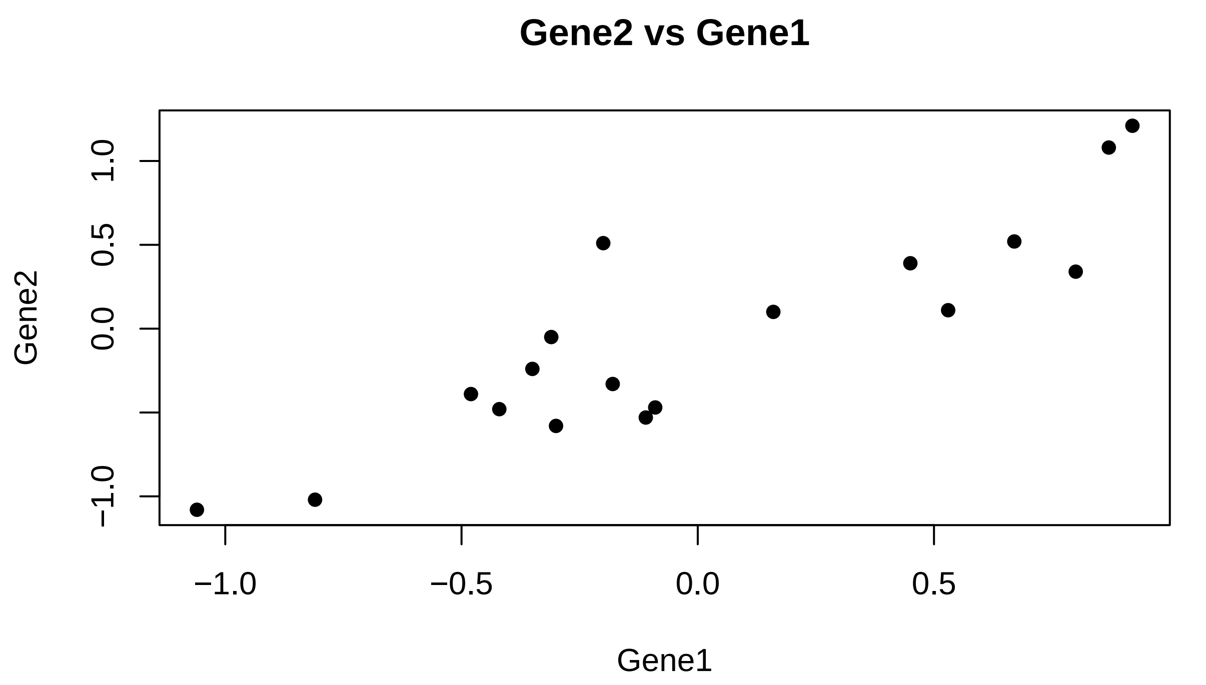


Figure 1: Gene2 vs Gene1

Then, expected values are calculated:

```
gene2.expected <- b1*genes$gene1 + b0
genes$gene2expected <- gene2.expected
genes[,2:3]
```

```
##    gene2 gene2expected
## 1  -1.08   -1.08435135
## 2  -1.02   -0.84167628
## 3  -0.39   -0.52134519
## 4  -0.48   -0.46310317
## 5  -0.58   -0.34661914
## 6  -0.24   -0.39515415
## 7  -0.05   -0.35632614
## 8  -0.33   -0.23013511
## 9   0.51   -0.24954911
## 10 -0.53   -0.16218609
## 11 -0.47   -0.14277208
## 12  0.10    0.09990299
## 13  0.39    0.38140607
## 14  0.11    0.45906209
## 15  0.52    0.59496013
## 16  0.34    0.72115116
## 17  1.08    0.78910018
## 18  1.21    0.83763519
```

It can be seen how some values are close. To determine what residual error we are making in predicting the values of each y_i with the line obtained it is necessary to calculate the residual error:

$$\epsilon_i = y_i - \mu_{y|x}$$

```
residues <- genes$gene2-genes$gene2expected
genes$residues <- residues
genes[2:4]
```

```
##    gene2 gene2expected      residues
## 1  -1.08   -1.08435135  4.351347e-03
## 2  -1.02   -0.84167628 -1.783237e-01
## 3  -0.39   -0.52134519  1.313452e-01
## 4  -0.48   -0.46310317 -1.689683e-02
## 5  -0.58   -0.34661914 -2.333809e-01
## 6  -0.24   -0.39515415  1.551542e-01
## 7  -0.05   -0.35632614  3.063261e-01
## 8  -0.33   -0.23013511 -9.986489e-02
## 9   0.51   -0.24954911  7.595491e-01
## 10 -0.53   -0.16218609 -3.678139e-01
## 11 -0.47   -0.14277208 -3.272279e-01
## 12  0.10    0.09990299  9.701311e-05
## 13  0.39    0.38140607  8.593934e-03
## 14  0.11    0.45906209 -3.490621e-01
## 15  0.52    0.59496013 -7.496013e-02
## 16  0.34    0.72115116 -3.811512e-01
## 17  1.08    0.78910018  2.908998e-01
## 18  1.21    0.83763519  3.723648e-01
```

Therefore to calculate y_1 would be obtained by means of:

$$y_1 = 0.9707x - 0.0554 + 4.35\mathring{u}10^{-3}$$

The sum of squared residuals, **RSS**, is:

```
RSS <- sum((genes$residues)^2)
RSS
```

```
## [1] 1.547075
```

With the R function, the data for the slope and the intercept are obtained with the least squares method, but not for the residuals.

```
lm(genes$gene2 ~ genes$gene1)
```

```
##
## Call:
## lm(formula = genes$gene2 ~ genes$gene1)
##
## Coefficients:
## (Intercept)  genes$gene1
##    -0.05541      0.97070
```

The representation of this line will be achieved by adding a `lines` function to the above graph.

```
plot(x = genes$gene1,genes$gene2,main="Gene2 vs Gene1",pch=16,
     xlab="Gene1", ylab="Gene2")
```

```
lines(genes$gene1,genes$gene2expected, col="blue") # Linear regression
segments(genes$gene1,genes$gene2expected,genes$gene1,genes$gene2,lty=2) # Vertical line
```

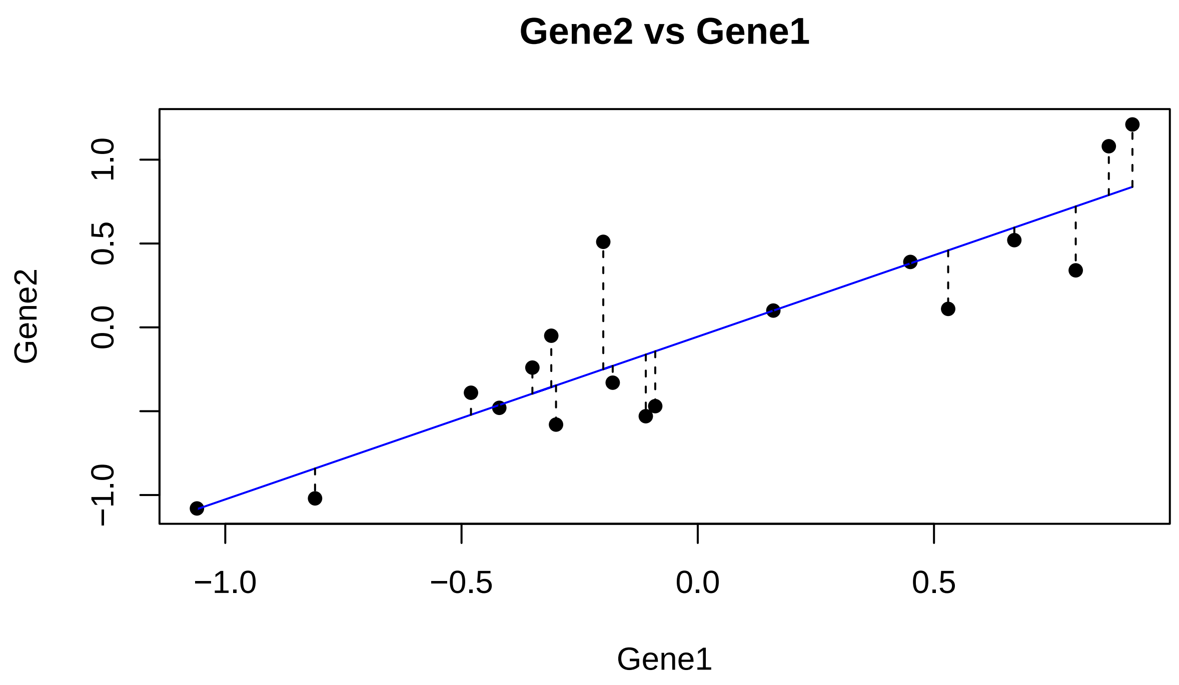


Figure 2: RSS adjuts

Goodness of fit

Goodness-of-fit refers to how well the regression model fits the observed data, and is represented by the R-squared value, R^2. The RSS (the sum of squared residuals) quantifies the discrepancy between the observed data and the regression line that represents them; the greater the RSS obtained, the greater the difference between what is expected by the model and what is observed. This value could be identified as the random variation that the response variable has to the model, which would be nothing more than the lack of fit to the model.

In contrast to this value would be the R^2: how well our regression model represents the observed data and explains the variation in the response variable. Its value is between 0 and 1, the larger the better the model fits the data. R^2 is calculated as 1 minus the ratio between the total variation of the response variable and the random variation of the response variable:

$$R^2 = 1 - \frac{\sum_i^n (y_i - \mu_{y|x})^2}{\sum_i^n (y_i - \overline{y})^2}$$

The denominator is also known as the **Total Sum of Squares**, **TSS**, and represents how much each observed value deviates from the mean obtained in the data set with those values. So R^2 could also be expressed as:

$$R^2 = 1 - \frac{RSS}{TSS}$$

The ratio

$$RSS/TSS$$

can be interpreted as the percentage of the variation that is not explained by the linear regression model. R^2 is also the result of the square of linear regression coefficient r:

$$r = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{(n-1)S_X S_Y}$$

Either this coefficient or R^2 determine the existance of a linear correlation between two variables when its value is significantly distinct to 0. This means that at least one coefficient of the linear model is not null:

$$\begin{cases} H_0: & \beta_i = 0\ \forall i \\ H_1: & \beta_i \neq 0 \text{ at least for one i} \end{cases}$$

Example: If we analyse the previous gene data:

```
num <- sum((genes$gene1 - gene1.mean)*(genes$gene2 - gene2.mean))
den <- (nrow(genes)-1)*sd(genes$gene1)*sd(genes$gene2)

correlation <- num/den
correlation <- correlation^2
correlation
```

```
## [1] 0.7790301
```

We may perform the contrast applyint the function `summary` to `lm` object:

```
summary(lm(genes$gene2 ~ genes$gene1))
```

```
##
## Call:
## lm(formula = genes$gene2 ~ genes$gene1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3811 -0.2196 -0.0084  0.1492  0.7595
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.05541    0.07330  -0.756    0.461
## genes$gene1  0.97070    0.12925   7.511 1.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.311 on 16 degrees of freedom
## Multiple R-squared:  0.779,  Adjusted R-squared:  0.7652
## F-statistic: 56.41 on 1 and 16 DF,  p-value: 1.246e-06
```

This is known as the **omnibus test** and performs an overall assessment of the model, to determine whether or not it is appropriate for modelling the data set provided. Its statistic is F , which gives rise to a probability value, p-value, which if it is less than 0.05 will indicate that the model is indeed appropriate for our data.

Clustering methods

In bioinformatics we can find a series of explanatory variables (which can be quantitative or categorical) that can predict the values of a response variable, which are unknown. This is a model. If the model uses the available data to predict new values, then it is a supervised learning method. Example of a supervised method: linear regression. Unsupervised learning methods attempt to identify the underlying data structure of a given set, without focusing on a specific variable.

Clustering is a very important class of unsupervised methods, used to identify subgroups within a population. For example, if we had a quantitative variable and we had to find out into how many categories it could be classified according to the variable's data, clustering would help us to do this classification. Classification is based on the notion of the degree of similarity or dissimilarity of the data, which is quantifiable with a measure of distance.

Distance squared

One of the most commonly used distances, for a random variable X for its observation i and its observation j,

$$d_{ij} = (x_i - x_j)^2$$

If we have in the dataset more than one random variable, for example p random variables $(X_1, \ldots X_p)$ the distance between each pair of observations for the different variables corresponds to:

$$d_{ij} = (x_{i1} - x_{j1})^2 + ... + (x_{ip} - x_{jp})^2$$

This measure of difference between observed values is the *squared Euclidean distance.*The R `dist` function allows to calculate Euclidean distances between data.

K-Means algorithm

It is the simplest algorithm that uses the Euclidean distance squared to quantify how different the data is. It is an iterative algorithm, in which we must first indicate the number of groups into which we believe the population is divided, which corresponds to the value of K. The objective will be to try to group the n observations into one of the K groups according to how different these observations are, i.e. the distance. With each iteration the observations are better distributed in the groups or clusters.

The steps to be followed to carry out a cluster classification by the K-means algorithm:

1. All observations are randomly divided into K groups or clusters.
2. Define a centre or centroid for each cluster, an imaginary observation that represents the mean of all observations in that cluster.
3. The Euclidean distance squared from each observation to all the centroids of the K groups is measured.
4. Each observation is assigned to the group whose Euclidean distance to the centre was the smallest.
5. Return to point 2, recalculate the centres and repeat this process until the values of the centres stop changing with each iteration.

As an example, let us use the dataset of the `Protein.txt` file containing data on protein consumption from different origins in different European countries, with the aim of classifying them according to the consumption of these proteins.

```r
proteins <- read.table(file="Protein.txt",sep = " ",header = TRUE)
```

We will start with 3 clusters based on countries that consume the most red meat and fish

```r
x <- proteins[,c("RedMeat","Fish")]
#x <- scale(x)
head(x,10)
```

```
##   RedMeat Fish
## 1    10.1  0.2
## 2     8.9  2.1
## 3     13.5  4.5
## 4      7.8  1.2
## 5      9.7  2.0
## 6     10.6  9.9
## 7      8.4  5.4
## 8      9.5  5.8
## 9     18.0  5.7
## 10    10.2  5.9
```

The R function that creates the cluster by the K-means algorithm is `kmeans`.

```
clus <- kmeans(x, centers = 3)
```

The object that this function creates is a kmeans. Then, using it is possible to find out the position of each country and which are the centroid values.

```
clus$cluster
```

```
##  [1] 1 1 2 1 1 3 3 3 2 3 1 2 1 1 3 1 3 1 3 3 2 2 1 2 1
```

```
clus$centers
```

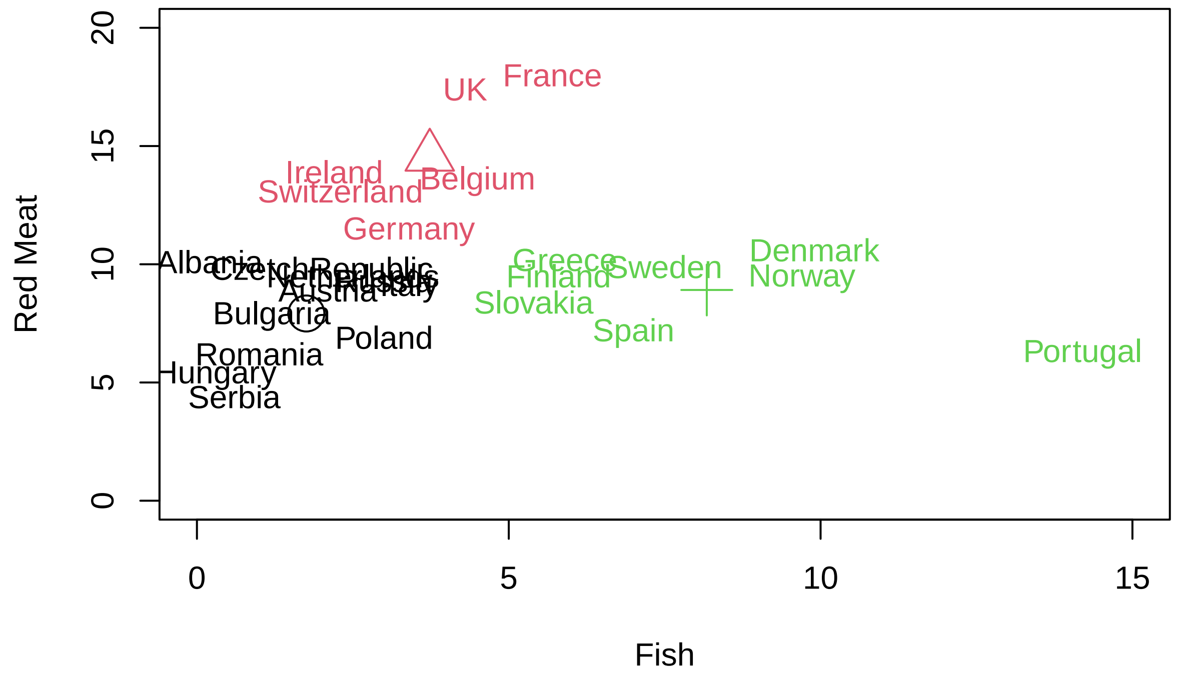
```
##     RedMeat     Fish
## 1  7.918182 1.754545
## 2 14.550000 3.733333
## 3  8.912500 8.175000
```

It is usefull to add a new column to the dataset with the information of the cluster that occupies.

```
proteins$clusterId <- clus$cluster
```

The representation of data around centroids can be visualized as follows:

```
plot(proteins$Fish, proteins$RedMeat, type = "n", xlab = "Fish",
     ylab = "Red Meat", xlim = c(0, 15), ylim = c(0, 20))

text(proteins$Fish, proteins$RedMeat, labels = proteins$Country,col=proteins$clusterId)
points(x=clus$centers[,2],y=clus$centers[,1], pch=c(1,2,3), col=c(1,2,3),cex=2.5)
```



For the first group of the cluster, in black, there are 11 countries, for the second group in red there are 6, and finally for the third group there are 8. You can see how they are distributed around the centroids, which are a circle, a triangle and a cross, respectively.

The first cluster contains countries that have relatively low fish and red meat consumption. The second cluster contains countries whose meat consumption is similar to that of the first group, but has higher fish consumption. And finally, the third cluster includes countries whose red meat consumption is higher than the other two while fish consumption remains lower than in the second cluster.

This algorithm has two limitations:

- all observations are assigned to groups and once they are in the group, they are not considered different from each other, but belong to that group and that's it.
- it is necessary to specify the number of clusters *a priori*. The choice should not be trivial, as it has a substantial impact on the results.

Hierarchical clustering

This is an alternative method to avoid the limitations that the K-means algorithm provides. The result is a tree or **dendogram**. The root of the tree contains all observations, and is the highest level. The leaves, on the other hand, are the lowest level and are unique to each observation.

There are two algorithms: agglomerative and divisive.

Agglomerative algorithm

The way to build the tree is from the bottom up.

- At the beginning, there are as many different clusters as there are observations.

- The next step is to converge two clusters with the smallest degree of difference (smallest distance).
- The following clusters converge with the next smallest degree of difference.
- This process is continued until only one cluster is present, i.e. the root of the tree is reached.

This algorithm is the most common and has three different methods of determining the distance between two clusters:

1. *Single linkage*: the cluster uses the minimum distance, di j, among all possible pairs as the distance between clusters.
2. *Complete linkage*: the cluster uses the maximum distance, di j, between all possible pairs in the cluster as the distance between clusters.
3. *Average linkage*: the cluster uses the average distance calculated between all points of the two clusters.

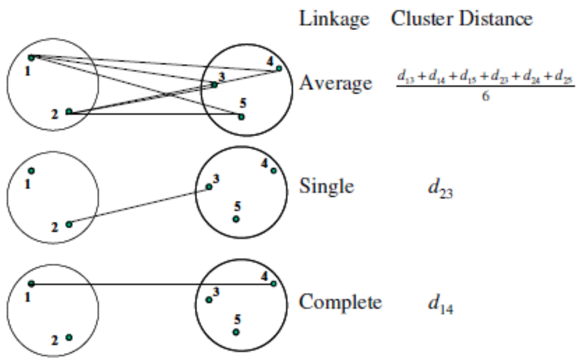
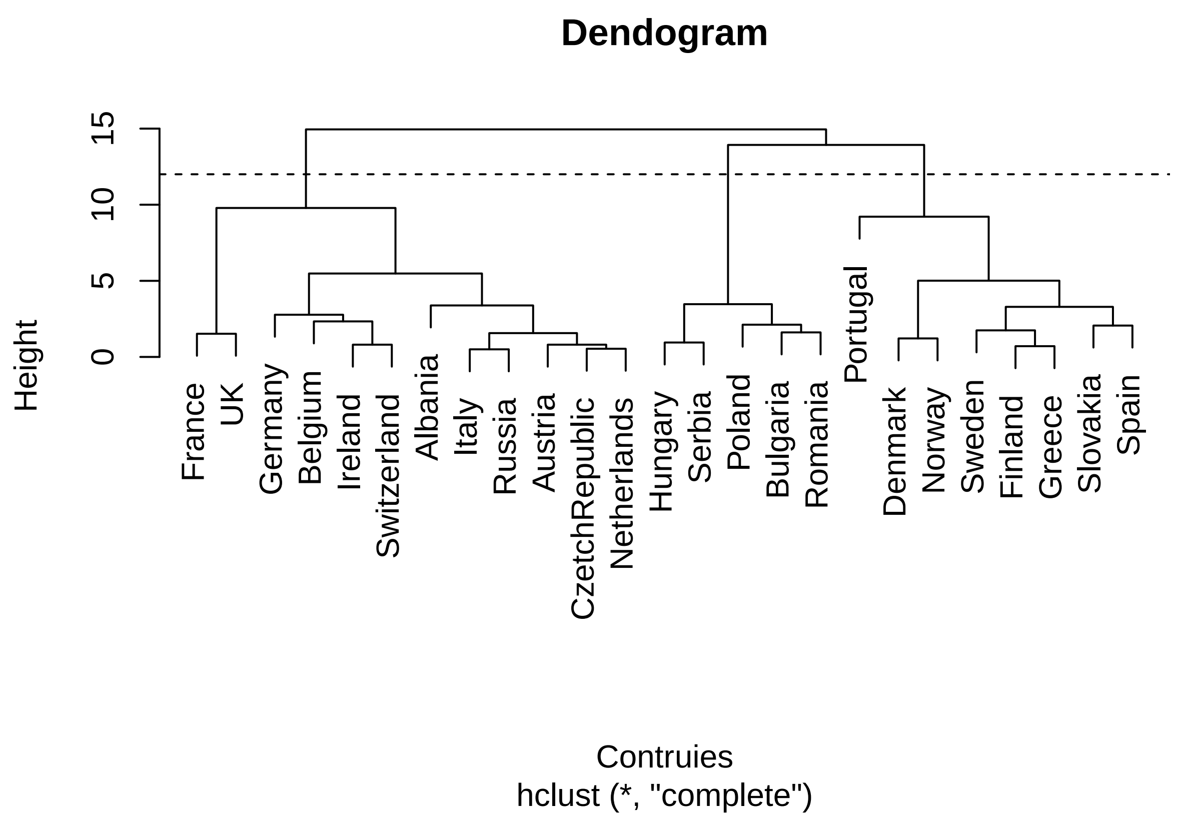


Figure 3: Types of linkage

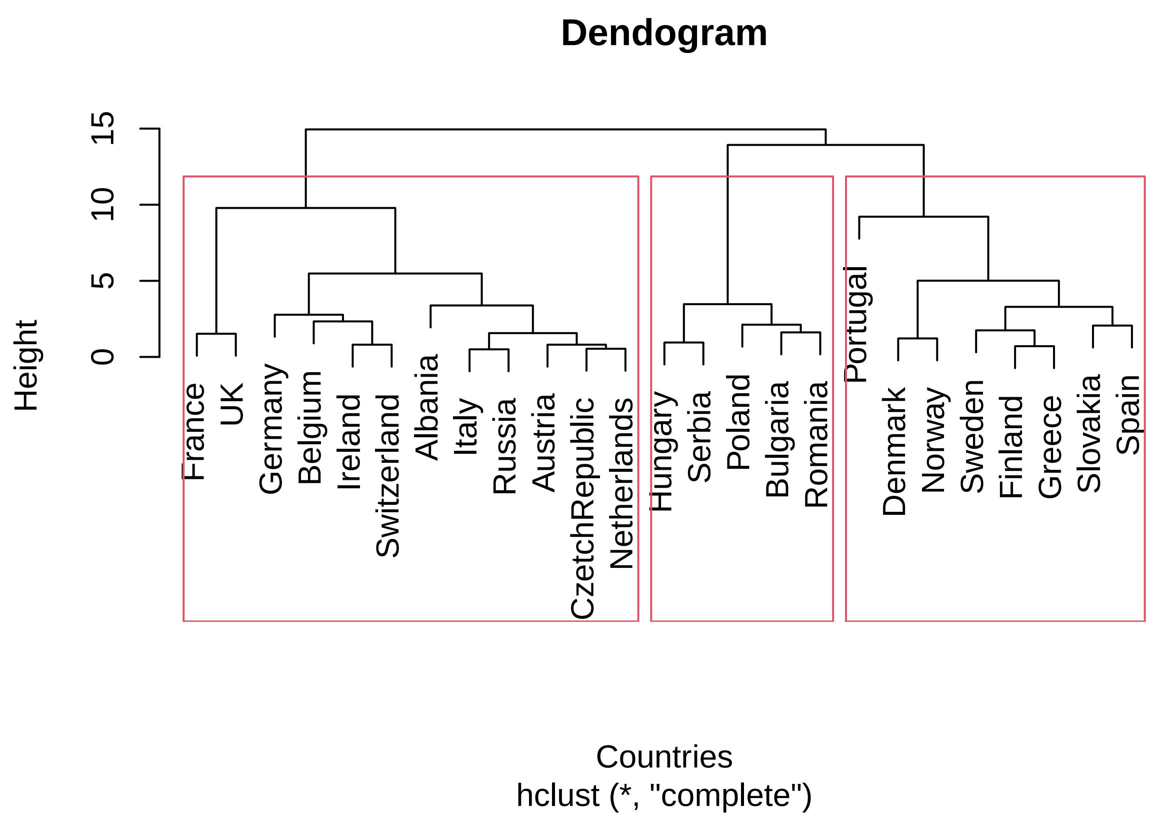
In R, the `hclust` function is the one used to obtain the values to be represented in the dendogram. It receives as an argument an object of the class `dist`, which is the result of calculating the distance between the observations for the chosen variables. The resulting object is of class `hclust`.

```r
# Using previous protein dataset, we determine Euclidean distance
d <- dist(x)

# This method allows different modes: single, complete, average, among others.
clus.h <- hclust(d = d,method = "complete")

# Dendrogram
plot(clus.h, main="Dendogram",labels = proteins$Country,xlab="Contruies")
abline(h=12,lty="dashed")
```



The horizontal line shows where the dendogram would be cut to create 3 clusters. This algorithm is very effective in giving an idea of how many different clusters I want to group my data into with the K-means algorithm.

```
plot(clus.h, main="Dendogram",labels = proteins$Country,xlab="Countries")
rect.hclust(clus.h, k =3)
```



With the `rect.hclust` function it is possible to obtain which data set belongs to which cluster. This information can be added to the dataset using the `cutree` function, and determine whether it matches that previously obtained by the K-means algorithm.

```
clus.h.id <- cutree(clus.h, k = 3)

proteins$hc <- clus.h.id
```

Divisive Algorithm

The way to build the tree is from the top down.

- At the beginning there is only one cluster, where all observations are grouped together.
- The next step is to separate into two clusters with the greatest degree of difference (greatest distance).
- The following clusters are then separated with the next highest degree of difference.
- This process is continued until there are only as many clusters as observations, i.e. the branches of the trees are reached.

Exercises

1. The `Indometh` dataset, which is included in the R dataset package, collects data from 6 individuals who were intravenously injected with the drug indomethacin. The dataset includes the time (hours) after injection when the blood samples were collected and the plasma concentration obtained. Answer the following questions about the experiment:

   a) Plot the time vs. concentration data. Is it a linear relationship?

b) Apply a logarithmic transformation to the data and re-plot. Is there now a linear relationship?

c) Find the linear regression line for the data transformed with the `lm` function. Interpret the result.

d) Add the linear regression line found to the previous graph.

e) How good is the fit of the data? Manually find the value of R^2.

f) Indicate if the linear regression model found is adequate to determine the time needed to wait to draw blood as a function of the concentration.

2. For this exercise is necessary to install and load the package **`ISLR`**:

```
install.packages("ISLR")
```

```
library(ISLR)
```

The `NCI60` object contains data on the expression levels of 6830 genes from 64 lines of different cancers.

```
str(NCI60)
```

```
## List of 2
##  $ data: num [1:64, 1:6830] 0.3 0.68 0.94 0.28 0.485 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:64] "V1" "V2" "V3" "V4" ...
##   .. ..$ : chr [1:6830] "1" "2" "3" "4" ...
##  $ labs: chr [1:64] "CNS" "CNS" "CNS" "RENAL" ...
```

On one hand, within `data`, there are expression data collected on the 64-sample microarray (rows) for 6830 genes (columns). On the other hand, in `labs` are the names of the cancer types.

```
data.nci <- NCI60$data
labels.nci <- NCI60$labs
```

The goal of the work is to see how the microarray expression data collected in ‘`data.nci` is clustered. In this way, it could be observed if there are genes that show alteration in their expression in a common way to the cancers that are collected in `tags.nci`. To do so, generate a markdown report showing how you follow the clustering analysis by doing the following steps:

1. Transform all the expression data from the microarrays so that they are in the same order of magnitude.
2. Visualize how many of the samples there are for each cancer type using a function that displays in a cross table the count within each cancer category in the `labels.nci` variable.
3. Calculate the Euclidean distance between all expression values.
4. Perform clustering using hierarchical clustering with the agglomerative algorithm and complete linkage as the method.
5. Represent the obtained dendogram using the cancer type labels.
   Note: if the letters are very large, use the parameter cex=0.5.
6. Group the results of the above dendogram into 4 groups.
7. Obtain to which of the clusters obtained above each of the samples belongs. Create a dataset with a column containing the cluster information and another column containing the type of cancer.
8. Explain the results you observe.

Bibliography

Dib, Linda, and Alessandra Carbone. 2012. “CLAG: An Unsupervised Non Hierarchical Clustering Algorithm Handling Biological Data.” *BMC Bioinformatics* 13 (1): 1–14.

Gönen, Mehmet, and Adam A Margolin. 2014. "Localized Data Fusion for Kernel k-Means Clustering with Application to Cancer Biology." *Advances in Neural Information Processing Systems* 27.

Hussain, Hanaa M, Khaled Benkrid, Huseyin Seker, and Ahmet T Erdogan. 2011. "Fpga Implementation of k-Means Algorithm for Bioinformatics Application: An Accelerated Approach to Clustering Microarray Data." In *2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 248–55. IEEE.

Kodinariya, Trupti M, and Prashant R Makwana. 2013. "Review on Determining Number of Cluster in k-Means Clustering." *International Journal* 1 (6): 90–95.

Loewenstein, Yaniv, Elon Portugaly, Menachem Fromer, and Michal Linial. 2008. "Efficient Algorithms for Accurate Hierarchical Clustering of Huge Datasets: Tackling the Entire Protein Space." *Bioinformatics* 24 (13): i41–49.

Lu, Yi, Shiyong Lu, Farshad Fotouhi, Youping Deng, and Susan J Brown. 2004. "Incremental Genetic k-Means Algorithm and Its Application in Gene Expression Data Analysis." *BMC Bioinformatics* 5 (1): 1–10.

Mac Nally, Ralph. 2000. "Regression and Model-Building in Conservation Biology, Biogeography and Ecology: The Distinction Between–and Reconciliation of–'Predictive'and 'Explanatory'models." *Biodiversity & Conservation* 9 (5): 655–71.

Mead, Roger, Robert N Curnow, and Anne M Hasted. 2017. *Statistical Methods in Agriculture and Experimental Biology*. Chapman; Hall/CRC.

Wei, Dan, Qingshan Jiang, Yanjie Wei, and Shengrui Wang. 2012. "A Novel Hierarchical Clustering Algorithm for Gene Sequences." *BMC Bioinformatics* 13 (1): 1–15.

Welham, Suzanne Jane, Salvador Alejandro Gezan, Suzanne J Clark, and Andrew Mead. 2014. *Statistical Methods in Biology: Design and Analysis of Experiments and Regression*. CRC Press.

Wilkin, Gregory A, and Xiuzhen Huang. 2007. "K-Means Clustering Algorithms: Implementation and Comparison." In *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, 133–36. IEEE.