Module 4 - Supervised Learning: regression techniques.

4.1 Regression: what, why, and how?

Authors:

By Rafael Alcalá

Professor at the University of Granada

Andalusian Inter-University Institute in Data Science and Computational Intelligence (DaSCI)

And by Augusto Anguita-Ruiz

Postdoctoral Research Fellow at Barcelona Institute for Global Health- ISGlobal.

Reminder: Introduction to NoteBook.

In this *NoteBook* you will be guided, step-by-step, through loading a dataset to the descriptive analysis of its contents. The Jupyter NoteBook (*Python*) is an approach that combines text blocks (like this one) and code blocks or cells. The great advantage of this system is its interactivity because cells can be executed to directly check the results they contain.

Very important: the order of the instructions is fundamental and so each cell in this NoteBook must be executed sequentially. If any are omitted, the program may throw an error and so if there is any doubt, you will have to start from the beginning again.

First, it is very important to select "*Open in draft mode*" (draft mode) at the top left at the beginning. Otherwise, for security reasons, you will not be allowed to execute any code blocks. When the first of the blocks is executed, the following message will appear: "*Warning: This NoteBook was not created by Google*". Don't worry, you can trust the contents of the NoteBook and click on "*Run anyway*".

Let's start!

Click on the "*play*" button on the left side of each code cell. Remember that lines beginning with a hashtag (#) are comments and do not affect the execution of the script. You can also click on each cell and press "Ctrl+enter" ("Cmd+Enter" on Mac). Each time you execute a block, you will see the output just below it. The information is usually always the last statement, along with any *print()* commands present in the code.

INDEX

In this *NoteBook*:

1. We will learn the general concepts of regression methods.
2. We discuss their possible bioinformatics applications using biological datasets.

Contents:

1. INTRODUCTION TO REGRESSION TECHNIQUES AND THEIR RELEVANCE IN THE FIELD OF BIOINFORMATICS

High-throughput omics technologies such as **DNA microarrays**, **mass spectrophotometry** or protein **chips** have been widely used in recent years in **biomedical sciences** to unveil the molecular mechanisms underlying disease. Among other parameters, these techniques allow the expression level of thousands or millions of **genes, metabolites or proteins** to be simultaneously determined in a wide variety of biological tissues. The data derived from these techniques, collectively known as "omics data", are therefore **highly complex and dimensional** clusters. Their **main characteristics** include the fact that they present **continuous domains** (i.e., the expression of a gene/metabolite/protein will be measured as values from 0 upwards), the **phenomena of collinearity** between variables (i.e., two genes participating in the same cellular pathway will have very similar expression levels), and the presence of **epistatic interaction** phenomena (through which certain genes are able to modulate the effect that other genes have on the phenotype).

Therefore, it is vital that we identify the most appropriate analytical methods capable of dealing with these particularities. In most cases, a bioinformatics researcher working with omics data will be interested in (1) **identifying functional relationships** between each gene/metabolite/protein and the phenotype of interest (associations), (2) **reducing the number of starting genes/metabolites/proteins** by generating a less complex dataset, or (3) **generating a model with good predictive ability** for the phenotype of interest (output variable).

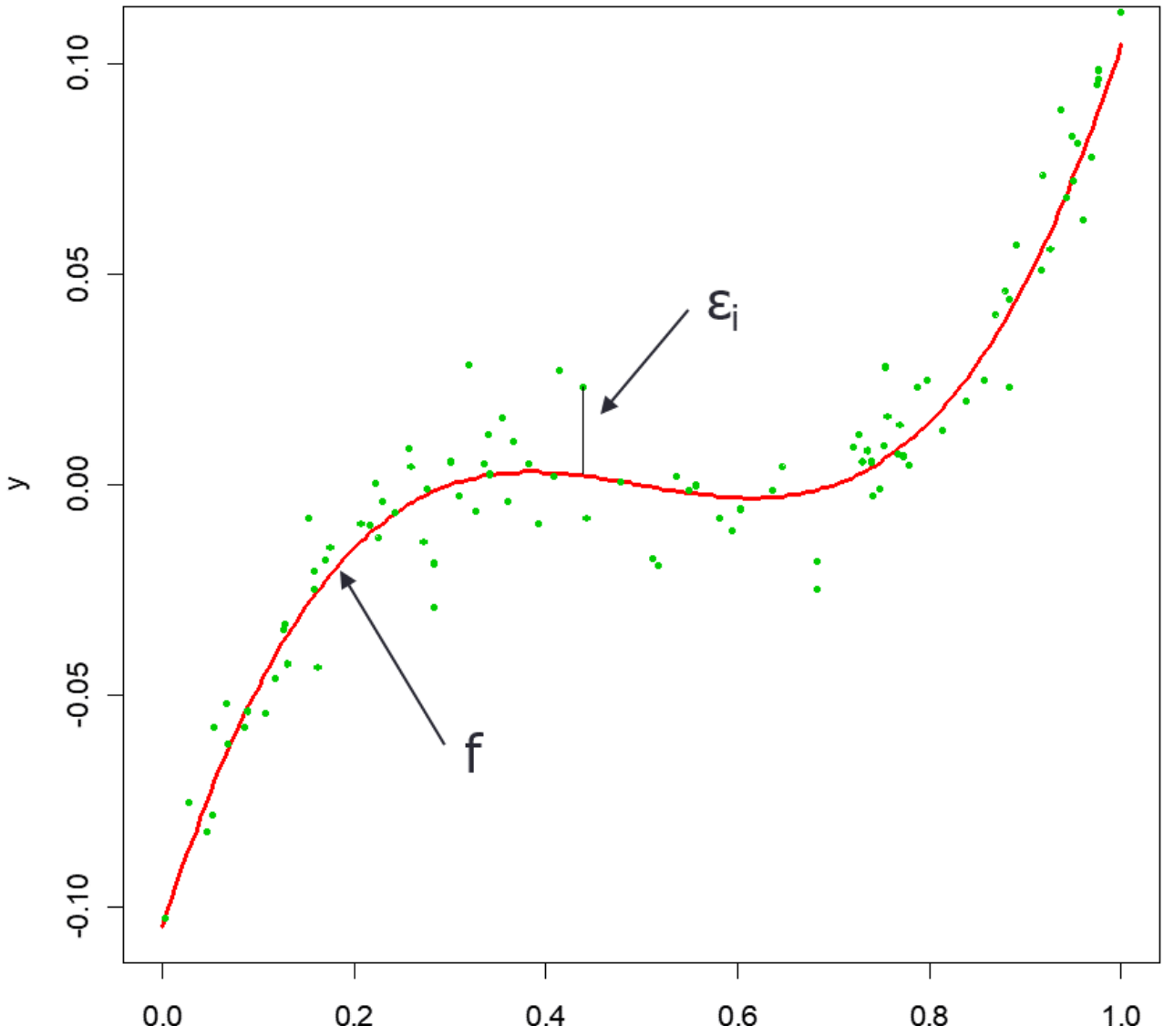
For all these purposes, **the regression techniques** introduced in this module are particularly appropriate, and have become established themselves as some of the *gold-standard* bioinformatics approaches. The main reasons for this are that the **design of these techniques is oriented to work with continuous variables**, their versatility in **identifying redundant variables**, and their capacity to **estimate** unknown values stand out. Beyond omics data, **regression techniques** have also proven to be **very useful** when working with **other biological data sources** commonly used in biomedicine, such as **microscopy-derived data** (in which images are transformed into numerical values), or **anthropometric, clinical, and biochemical measurements** that are usually obtained during a case-control study.

By definition, regression techniques study the relationships existing between **input variables** (independent variables), usually referred to as $X_1, X_2, \ldots, X_p$, and an **output variable** (dependent variable), termed Y. Thus, a regressor can estimate the expected value that Y would take, given a set of values for the variables X. Linking with the previous part of the course, regression methods are a **type of supervised machine learning** designed to predict a **continuous value**. To do this, models are built that specify the change in the response (output) of each input variable, assuming that all the other variables remain constant:

$$Y_i = f(X_i) + \epsilon_i$$

Where f is an **unknown function** and $epsilon$ is a **random error** whose mean is zero; the higher the variance of $epsilon$ the greater the difficulty in estimating f. Such variance is generally higher the smaller the number of instances available and/or the larger the number of input variables actually required, which is quite often the case in the field of bioinformatics.

A regression model can take many different **forms** including that of: a **function** (see Figure 1), **set of rules**, **neural network**, **support vector machine (SVM)**, among others. In this module, we will look very briefly at three of the best-known regression methods: **simple and multiple linear regression**, the k**-nearest-neighbors (KNN)** method, and the **M5 regression model**.


0.10
0.05
0.00
-0.05
-0.10
y
0.0
0.2
0.4
0.6
0.8
1.0
ε_i
f

2. REASONS FOR WANTING A GOOD ESTIMATE OF F: EXAMPLES IN BIOLOGICAL DATA SETS

In *Machine Learning*, and therefore also in the case of regression, data are used to "learn" f. As we have introduced throughout this capsule, there are two main reasons why we might want to get a good estimate of f as described below:

• **Prediction**: If we can produce a good estimate for f (and the variance of $epsilon$ is not too large) we can **make accurate predictions** for the output variable, Y, based on a new value of X. In this case, we try to learn models that are as accurate as possible. A very clear example of this type of application in bioinformatics is the use of **polygenetic risk scores** othwerwise known as *Genetic Risk Scores (GRS)*. GRSs are scoring systems that allow us to **assess an individual's genetic risk for a certain disease or phenotypic trait**. They are calculated as the **weighted sum of the number of risk alleles** an individual carries among a battery of genetic markers. Once the polygenetic risk score has been calculated for each individual, it is common to construct a **regression model** to predict the unknown Y_i values of a **continuous phenotype of interest** (e.g. survival time of a patient or high levels of a biomarker in blood) with these values as the X input variable. The great utility of polygenetic risk scores and their incorporation into predictive regression models lies in their ability to predict continuous phenotypes of interest with a large **anticipation in time** (since an individual's genetics is a parameter that can be assessed from birth). The application of predictive regression models based on *GRSs* to estimate future **body mass index (or obesity status)** in overweight children has been a widely used technique in recent years.

• **Interpretation/explanation/discovery**: Alternatively, we might also be interested in **discovering the type of relationships** between the output variable Y and each of the $X's$ input variables, as well as between the $X's$ variables themselves. To this end, we try to learn models that explain the variable Y from the $X's$ variables. Regression models are also widely used in bioinformatics for this purpose; especially in the field of omics. Notable examples include (1) the **discovery of genetic variants** responsible for the development of a disease (in this case, a continuous phenotype of interest such as high blood cholesterol levels), or (2) the **identification of omics interaction phenomena** such as those between genetic variants and metabolites (*mQTLs*), genetic variants and DNA methylation (*meQTLs*), or DNA methylation and metabolites (*mCpG*).

These two rationales are usually contradictory, with one almost always predominating over the other depending on the specific needs of the problem.

3. QUALITY METRICS FOR REGRESSION MODELS

A well-fit regression model results in estimated/predicted values close to the observed data values for an output or **dependent** variable Y, based on the observed values on other input or **independent** $X's$ variables. If there were no informative input variables, one would generally use the mean model, which directly uses the mean of the observed Y data as a prediction for any new unknown values. This would be the worst-case scenario when looking for linear relationships between the input variables and the output variable. Therefore, we must will always look for the fit of a proposed regression model to be better than that of the mean model.

In regression, as in other areas of *Machine Learning*, **many metrics** are available to assess the quality of the models obtained; including the **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, R^2 (when its calculation is possible and/or makes sense), and **mean absolute error (MAE)**, etc. In this section we will discuss the two most widely used statistics among the above: the R^2 and the **RMSE**. The calculation of both these parameters is based on two sums of squares: **total sum-of-squares (TSS)** and **residual sum-of-squares (RSS)**. On the one hand, the *TSS* measures how far the data are from the worst model that could be learned by the learning technique used. For example, how far the data are from a constant line distribution centered on the mean, which would be the worst model in the case of multivariate linear regression learning. *TSS* cannot be calculated (or it does not make sense to do so) for all regression methods and so *TSS-based measures* are not always available. On the other hand, the *RSS* measures how far the data are from the values predicted by the learned model (the difference in the absolute value).

Different combinations of these two values provide different information on how the regression model compares with models obtained by the same technique or by different regression techniques:

- R^2 **(or R-squared)**: The difference between *TSS* and *RSS* represents the improvement in prediction of the regression model compared to the worst model that could theoretically be obtained with that technique. Dividing that difference by *TSS* gives R^2. Thus, R^2 is the **proportional improvement in the prediction of the regression model, compared to the worst model**, and indicates the goodness of fit of the model for a particular technique. R^2 has the useful property that its scale is intuitive: it ranges from zero to one, with 0 indicating that the proposed model does not improve the prediction ability over the worst theoretical model for that technique, and 1 indicating perfect prediction. Improvement in the regression model results in proportional increases in R^2.

$$R^2 = (TSS - RSS)/TSS = 1 - RSS/TSS$$

- **RMSE (*Root Mean Squared Error*)**: The root mean squared error is the square root of the variance of the residuals. In this case, only the *RSS* (squared differences) is used and so it is applicable to models obtained by any regression technique. RMSE indicates the absolute fit of the model to the data; in other words, how close the observed data points are to the values predicted by the model. With n being the number of data and e_i each of the differences between the observed data and the model-predicted value ($e_i = y_i - y_i'$), RMSE can be calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$$

While R^2 is a relative measure of fit, **RMSE** is an absolute measure. Therefore, it has the useful property of being measured in the same units as the response variable. Lower values of **RMSE** indicate a better fit. **RMSE** is a good measure of the accuracy with which the model predicts the response, and is the criterion most commonly used to measure fit if the primary purpose of the model is prediction.

4. DESCRIPTION OF THE DATA SET USED IN THIS MODULE

Whenever we talk about **continuous, integer, or categorical variables** where the information can be represented numerically following a **logical order** (also known as **ordinary variables**), regression methods will be **applicable and useful**. This point is extremely important and is the main difference between regression and ranking methods. In classification, non-ordinal categorical variables will make sense. In regression, however, they have no place.

We could apply several different regression methods to the dataset we have been working with since the beginning of the course, ***The Cancer Genome Atlas (TCGA)*** according to different scenarios. For example, we could use a regression model to predict the **survival time** of melanoma patients based on the expression of genes previously shown to be directly involved in the pathophysiology of the disease. We could also use regression models to **discover relationships between variables (in this case, genes)** to reveal molecular interaction phenomena, whereby the products of one or more genes interact with each other to condition the appearance or worsening of the pathology.

Although, a priori, both these examples are valid, problems can arise when we want to make important inferences about parameters such as **patient survival** by using only **gene expression data**. Firstly, the expression of a gene represents a single, small piece within the complex puzzle of mechanisms underlying the appearance and malignization of a tumor. Secondly, although we may find several **alternative clinical variables** in the original dataset that could all be useful, most of them **do not present continuous domain**s.

Taking this into consideration, in this module, we propose the analysis of an **alternative dataset** consisting of clinical data on **childhood obesity**, obtained in patients who might present **insulin resistance**. This type of data, which includes both clinical and anthropometric data, has proven to be especially useful for predicting clinically relevant output variables such as the insulin resistance index (**HOMA-IR**). Among other applications, estimation of this index in overweight or obese children allows us to identify children at a considerable risk of developing type 2 diabetes in the future.

This obesity dataset contains information for **16 variables in 292 children** and was derived from a cohort of Spanish children aged 5–15 years, grouped into three experimental conditions (normal weight, overweight, and obese). A wide range of clinical, biochemical, and anthropometric (body composition) data, as well as lifestyle and physical activity data (obtained by accelerometers), are available for these children. By applying **regression techniques** on this dataset, it would be possible to study **the relationship between anthropometric, biochemical, physical activity, and insulin-resistance status variables**, as well as to **predict HOMA-IR based on the above**. In the analyses addressed in this module, **the HOMA-IR will be included as an output variable** from our regression models because it has been widely validated as a good indicator of insulin resistance and pre-diabetic status both in children and adults. We will use clinical and anthropometric data such as sex, age, pubertal stage, height, waist circumference and body mass index (BMI) as well as the main measures of physical activity such as the daily minutes of inactivity (sedentary behavior), or of mild, moderate, and vigorous physical activity, as input variables in this model. In addition, indicators of cardio-metabolic dysfunction other than HOMA-IR itself (such as HDLc and LDLc cholesterol, triglyceride, and blood pressure levels) will also be included. These factors are all related to the level of patient insulin resistance and present **continuous domains**. Therefore, they constitute a perfect set of input variables for the application of regression models.

5. WHY USING R? INSTALLING THE R ENVIRONMENT AND ITS LIBRARIES FOR REGRESSION

As we saw in Module 2 for data loading, and although Python is the language of choice for this course, the R libraries offer a **much more convenient** way to apply the regression algorithms we will see in this module. In R, the algorithms allow a direct formulation to be made using the variables of any data set. This allows a stepwise construction of the models, integrating/determining the really relevant variables and discovering the type of relationship/interaction between them. Likewise, in R more algorithms are available and, in our case, it is necessary to use the M5 algorithm (via *Cubist*) as it is almost a paradigm in the regression area (see the study carried out in 2019 among 164 algorithms by *Gacto et. al.* where a simple tree is shown to be competitive against the best ensembles of 500 trees - Bibliography section). Thus, in this module, we will use **functions of R to apply the regression algorithms** indicated.

The installation of R in our Google Colab environment was already explained in Module 2 of this MOOC, so here we directly include the relevant instructions. It should be remembered that all library installations we perform in the Google Colab environment will only remain active for a few hours, after which the installed libraries are removed. **Therefore, you will need to re-run the library installation codes in this section when you need to run notebooks containing R code**. In any case, the codes to perform this installation will be included at the beginning of each of the capsules where it is necessary.

Finally, remember that when a cell uses R code this is indicated by the **%%R** symbology in Python notebooks. By removing that line, which serves to indicate that what follows in the notebook is written in R, we could execute the same instructions in any R environment other than the notebook without any problems. For example, on our own Windows, Linux, MacOS machine using our local installation of R and RStudio.

The following code snippet, corresponding to the R installation, is exclusive to the use of Colab notebooks and would be equivalent to installing R andRStudio on our machines:

In [ ]:

```
# Estimated execution time: approx. 20 seconds.

### Installing R on Google Colab notebooks ###
!apt-get update
!apt-get install r-base
!pip install rpy2==3.5.1
%load_ext rpy2.ipython
print ("R installation on Google Colab completed")
```

```
Get:1 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InReleas
e [3,626 B]
Ign:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/
x86_64  InRelease
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repo
s/ubuntu1804/x86_64  InRelease
Get:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/
x86_64  Release [696 B]
Get:5 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InReleas
e [15.9 kB]
Hit:6 https://developer.download.nvidia.com/compute/machine-learning/repo
s/ubuntu1804/x86_64  Release
Get:7 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/
x86_64  Release.gpg [836 B]
Hit:8 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:9 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 k
B]
Get:10 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ Package
s [76.8 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:12 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Hit:14 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Get:15 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 k
B]
Get:16 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu180
4/x86_64  Packages [930 kB]
Get:17 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelea
se [21.3 kB]
Get:18 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main So
urces [1,827 kB]
Get:19 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Pa
ckages [1,474 kB]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Pa
ckages [840 kB]
Get:21 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64
Packages [806 kB]
Get:22 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packag
es [2,596 kB]
Get:23 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main am
d64 Packages [936 kB]
Get:24 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Pack
ages [2,252 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
[3,035 kB]
Get:26 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic/main am
d64 Packages [42.8 kB]
Fetched 15.1 MB in 4s (4,028 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
r-base is already the newest version (4.1.2-1.1804.0).
The following package was automatically installed and is no longer require
d:
  libnvidia-common-470
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 69 not upgraded.
Requirement already satisfied: rpy2 in /usr/local/lib/python3.7/dist-packa
ges (3.4.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-pac
kages (from rpy2) (2.11.3)
```

```
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packa
ges (from rpy2) (2018.9)
Requirement already satisfied: cffi>=1.10.0 in /usr/local/lib/python3.7/di
st-packages (from rpy2) (1.15.0)
Requirement already satisfied: tzlocal in /usr/local/lib/python3.7/dist-pa
ckages (from rpy2) (1.5.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-
packages (from cffi>=1.10.0->rpy2) (2.21)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.
7/dist-packages (from jinja2->rpy2) (2.0.1)
R installation on Google Colab completed
```

From here, removing the **%%R** command line, the resulting R code could be run, as indicated, in a local installation via RStudio. The next piece of code corresponds to the installation and import of the libraries necessary for the use of the regression algorithms indicated.

In [ ]:

```
# Estimated running time: 2:30 minutes approx.
# Libraries needed:
# ISLR for multivariate linear regression
# kknn for k-nearest neighbours regression
# Cubist for M5-based regression models

%%R
### Installation of the necessary libraries
install.packages(c("ISLR", "kknn", "Cubist"))
print ("Installation of the R libraries for this module done")

### Importing required libraries ###
require(ISLR)
require(kknn)
require(Cubist)
print ("Import of R libraries for this module completed")
```

```
R[write to console]: Installing packages into ‘/usr/local/lib/R/site-libra
ry’
(as ‘lib’ is unspecified)

R[write to console]: also installing the dependencies ‘plyr’, ‘igraph’, ‘r
eshape2’


R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/plyr
_1.8.6.tar.gz'

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 401191 bytes (391 KB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]:

R[write to console]: downloaded 391 KB


R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/igra
ph_1.2.11.tar.gz'

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 2398028 bytes (2.3 MB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: 

R[write to console]: downloaded 2.3 MB


R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/resh
ape2_1.4.4.tar.gz'

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 37307 bytes (36 KB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: 

R[write to console]: downloaded 36 KB


R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/ISLR
_1.4.tar.gz'

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 1756715 bytes (1.7 MB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]:

R[write to console]: downloaded 1.7 MB


R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/kknn
_1.3.1.tar.gz'

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 388410 bytes (379 KB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]:

R[write to console]: downloaded 379 KB


R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/Cubi
st_0.4.0.tar.gz'

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 1523565 bytes (1.5 MB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]:

R[write to console]: downloaded 1.5 MB


R[write to console]:

R[write to console]:
R[write to console]: The downloaded source packages are in
        ‘/tmp/RtmpEBN274/downloaded_packages’
R[write to console]:
R[write to console]:


[1] "Installation of the R libraries for this module done"

R[write to console]: Loading required package: ISLR

R[write to console]: Loading required package: kknn

R[write to console]: Loading required package: Cubist

R[write to console]: Loading required package: lattice


[1] "Import of R libraries for this module completed"
```

Important: if, at any time while monitoring of the capsules of this module, the following message appears after executing a code cell ***UsageError: Cell magic `%%R` not found*** , you will have to run the installation of R and the libraries previously mentioned in this section again.

6. LOADING THE DATA

Once this introductory capsule is completed, in the next capsules we will delve into details about three of the best-known regression methods, applying them to our obesity dataset. These are:

- **Simple and/or multiple linear regression**.
- The **k-nearest neighbors algorithm for regression (KNN)**.
- The **M5 regression tree** (via *Cubist* in our case).

The **lines of code** required to **load** the obesity dataset into the R environment are shown below. These will be used in the next capsules of this module. Keep in mind that the file we are going to read is in a **table format with a header**. This is the standard format and so in our case, each column represents a variable of the problem. The last column of the data set corresponds to the **output variable** we want to infer; in other words, the output of our regression model. The first line of the data file (table header) includes the names of all the variables separated by commas, with each name corresponding to each of the aforementioned columns. Likewise, the remaining lines of the file (rows of the table) include numerical values separated by commas. It is vital that this numerical information is presented in an English to match the format of the algorithms we will use. Otherwise, the separating comma of the data could be confused with what would be the decimal point in a normal Spanish format. Starting from the second line of the file, each line (table row) contains the information related to one child. The ideal file type for this purpose is **.CSV** (comma-separated values) because this format can be easily generated or edited in Microsoft Excel or OpenOffice Calc.

The data is available via Google Drive in a file called homa.csv. However, if we were working locally with RStudio and R or with another data file, we would only have to change the https://drive.google (https://drive.google) (https://drive.google (https://drive.google))... URL by adding "\:\\\" (e.g., "C:\Data\homa.csv\" in case of the Windows operating system). The "head()" method is not necessary but is used to verify that the data was read correctly; it should show the column names and row numbering along with a small portion of the data.

In [ ]:

```
# Estimated execution time: 2 seconds approx.

%%R
### Reading
data <- read.csv(url("https://drive.google.com/uc?id=1GO2NBxYw54K6HkN-YgXbNadrLo5O6-0
u"))

### Visualisation of a small part of the data
head(data)
```

```
  Sex  Age Tanner Height   BMI   WC TAGmgDL HDLCmgDL LDLCmgDL SBP DBP Sede
ntary
1   1  9.5      0   1.55 11.34 60.0      55       51       93  97  60  41
1.0893
2   1  8.0      0   1.15 12.40 46.3      51       70       59  90  55  43
5.6071
3   0 10.5      0   1.42 12.99 67.5      65       60       96  96  54  48
3.9048
4   0  8.1      0   1.27 13.43 53.1      41       78      100 108  46  42
9.2976
5   1 10.4      0   1.32 13.72 51.9      39      100      120 107  69  51
2.0714
6   0 10.4      0   1.29 14.02 54.9      57       76       73  87  59  45
1.2321
     Light Moderate  Vigorous HOMA
1 321.5804 22.13393  3.982143 1.98
2 316.9762 48.05952 14.273810 0.87
3 337.7857 33.30952  7.988095 1.46
4 241.9762 39.67857 11.821429 1.07
5 216.0357  9.75000  2.410714 0.80
6 257.6429 36.40179  9.767857 1.35
```

REFERENCES

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. An Introduction to Statistical Learning with Applications in R Springer, 2013 (**Chapter 03**)
- McDonald, J.H. Handbook of Biological Statistics (3rd ed.). Sparky House Publishing, Baltimore, Maryland, 2014. Pages 190-208 in the printed version
- Usando rpy2 en notebooks: https://rpy2.github.io/doc/latest/html/notebooks.html (https://rpy2.github.io/doc/latest/html/notebooks.html)
- Usando read.csv de R: https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/read.table (https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/read.table)

ADDITIONAL REFERENCES

- M.J. Gacto, J.M. Soto-Hidalgo, J. Alcalá-Fdez, and R. Alcalá (2019). Experimental Study on 164 Algorithms Available in Software Tools for Solving Standard Non-Linear Regression Problems. IEEE Access 7, 2019, pp. 108916-108939; https://doi.org/10.1109/ACCESS.2019.2933261 (https://doi.org/10.1109/ACCESS.2019.2933261)