FGV EMAp
ESCOLA DE
MATEMÁTICA
APLICADA

# Fundamentos de Data Science
# Assessed Coursework 1

This assignment must be submitted by July 22nd, 2021 at 8:00 am.
Late submissions will NOT be accepted.

This coursework is assessed and mandatory
and is worth 30% of your total final grade for this course.

## Learning outcomes assessed

This coursework will test some basic concepts of Matlab programming for data analysis, including writing short scripts, simple function and basic plotting.

## Instructions

### Identifier

Please choose a random number of 6 digits. Make sure that you keep a copy of that number as it will be used to provide the feedback (please avoid trivial numbers, such as 000000 or 123456. Also please avoid numbers starting with zero).

### Submission

Compress your files into a zip file and submit it through ECLASS. The zip file you submit cannot be overwritten by anyone else, and it cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submission after the deadline will NOT be accepted.

*If you have issues, at the very last minute, email your coursework as an attachment at alberto.paccanaro@fgv.br with the subject "URGENT – COURSEWORK 1 SUBMISSION". In the body of the message, explain the reason for not submitting through ECLASS.*

**IMPORTANT:** In this assignment, exercises 1, 5, 6, 7 require you to write scripts, while exercises 2, 3 and 4 require you to write a function. For this assignment you will have to submit a total of 5 files:

- A file containing all the scripts for exercises 1, 5, 6, 7. A template for this file, called Assignment1_scripts is provided on the course page on Eclass. You need to insert your code for each exercise where indicated.
- A file containing function *calcrectarea* required for exercise 2
- A file containing function *evenodd* required for exercise 4
- A file containing function *conversion* required for exercise 3
- The file *salesfigs.dat* that you will use in exercise 1

A template for each function is also provided on Eclass. Insert your code for each exercise in the corresponding file.

**Please do not change the above file names in your submission.**

**All the work you submit should be solely your own work.**
**Coursework submissions will be checked for this.**

**EXERCISE 1** (value: 2%)

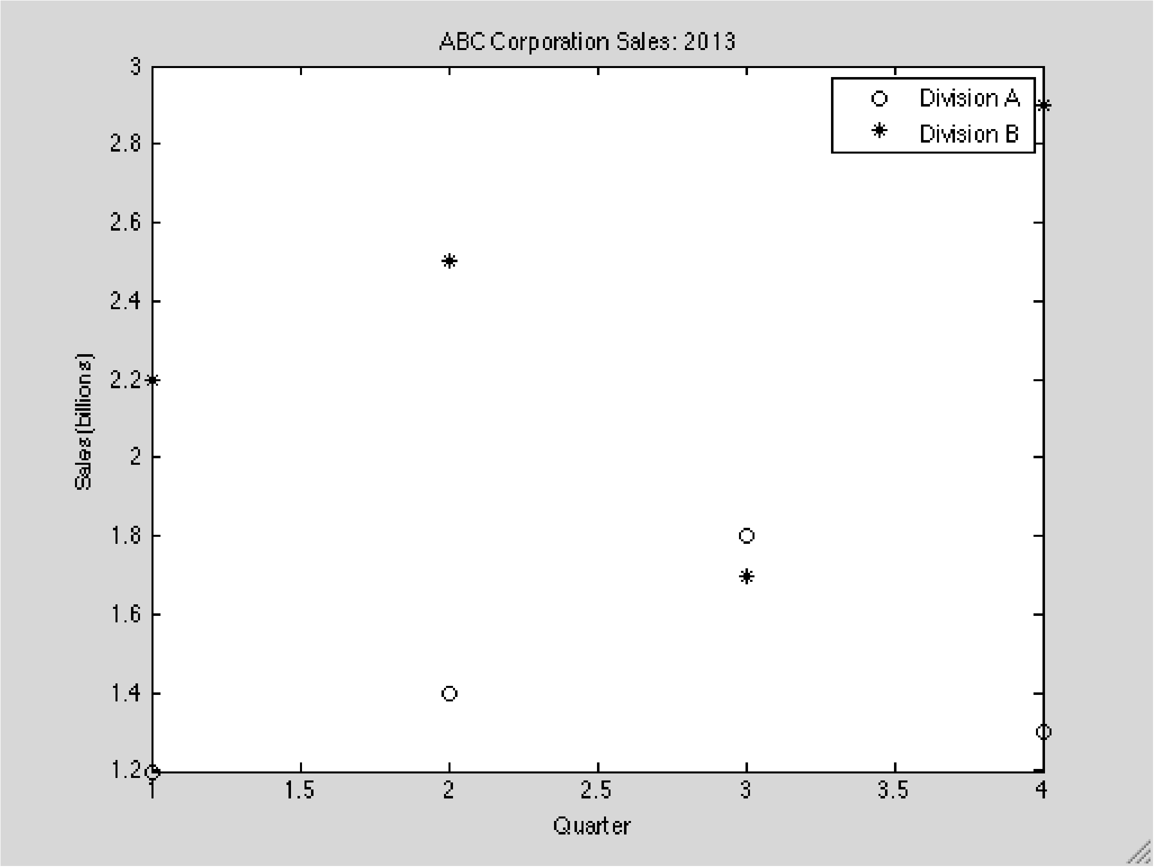
The sales (in billions) for two separate divisions of the ABC Corporation for each of the four quarters of 2013 are stored in a file called “salesfigs.dat”:

1.2 1.4 1.8 1.3

2.2 2.5 1.7 2.9

First, create this file (you can use any editor, and then save it as “salesfigs.dat”).
Then, write a script that will

- load the data from the file into a matrix
- separate this matrix into 2 vectors.
- create the plot seen in the figure below (which uses black circles and stars as the plot symbols).



**EXERCISE 2** (value: 4%)

Write a function called *calcrectarea* that will receive lengths and widths of rectangles in centimetres as input arguments, and will return the areas of the rectangles. The function should work when:

- the inputs are just a single value, i.e. one scalar for the length and one scalar for the width. In this case the function will return one single output for the area.
- the inputs are vectors, i.e. one vector of lengths (***l***) and a corresponding vectors of widths (***w***), of the same length *n*. In this case the function will return *n* values for the areas where each is calculated as

$\boldsymbol{l}_i$ x $\boldsymbol{w}_i$ for each $i \leq n$. For this case, your code needs to handle the user error when the two vectors provided as inputs do not have the same length.

**EXERCISE 3** (value: 4%)

Create a function *conversion* that will take in input two arguments: (1) a single character that can be either ‘f’ for feet or ‘m’ for meters; (2) a single value or a vector. The function would then output the value(s) converted into meters (if ‘f’ was input) or into feet (if ‘m’ was input). The function should work when:

- The second input is just a single value for either feet or meters. In this case the function will produce one single output for the conversion into meters or feet, respectively.
- The second input is a vector containing *n* values in either feet or meters. In this case, the function will return a vector of *n* values for the conversions into meters or feet, respectively. Note that here, the user would provide only one value for measure type (either ‘f’ or ‘m’) as the *n* values in the vector are interpreted as being all of the same type (either feet or meters).
- Your code needs to handle the user error when the user provides a string different from ‘f’ or ‘m’.

**EXERCISE 4** (value: 4%)

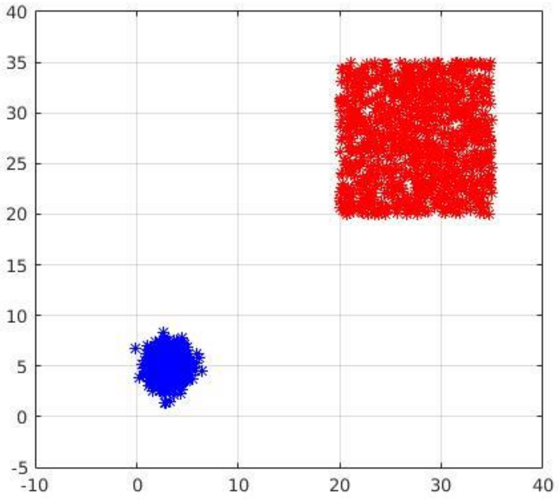
Write a function called *evenodd* that will take in input a single value *n* and then:

- Create a vector *v* of length *n* of random integers in the range [0 .. 30]
- Return only the elements of *v* which have an even value and are placed at odd positions in *v*, i.e. their indices in *v* is odd.

**EXERCISE 5** (value: 4%)

Create a script that will:

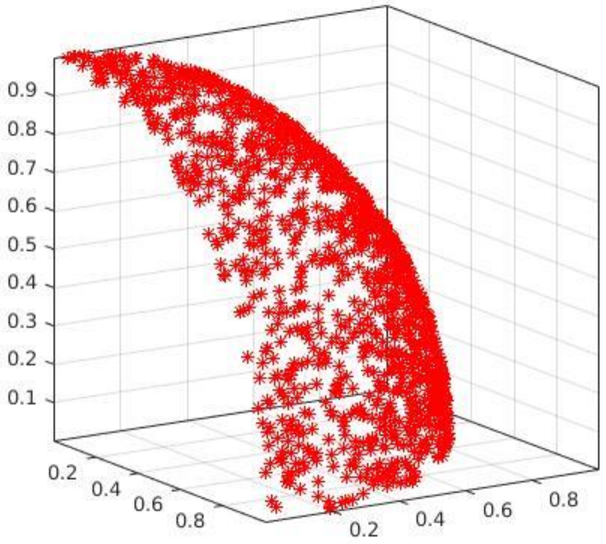
- Generate 1000 points, in 2 dimensions, which are uniformly distributed in the range: xmin=20, xmax=35, ymin=20, ymin=35.
- Plot the points, as red stars, in a figure whose axis are set in the range xmin=-10, xmax=40, ymin=-5, ymin=40
- Add to the same figure 1000 points, in 2 dimensions, which are normally distributed with a mean value of (3,5) and unit variance. These points should be denoted by a blue star.



**EXERCISE 6** (Value: 5%)

Create a script that will:

- Generate 10000 points, in 3 dimensions, normally distributed with zero mean and unit variance.
- Extract those points for which all the 3 components are positive and then:
  - Normalize them to have unit length (i.e. the norm of the vector is equal to 1)
  - Plot them in 3 dimensions. Your figure should look something like the one given below.



**EXERCISE 7** (Value: 7%)

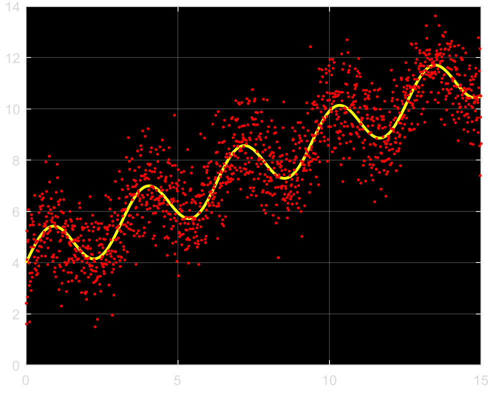
Assume that we have a process following the equation:

y = sin(2x) + 1/2 x + 4  with x in [0, 15]

We can measure y at intervals of 0.01, but these values are corrupted by a Gaussian noise with zero mean and unit variance. Write a script that implements the following points 1-8:

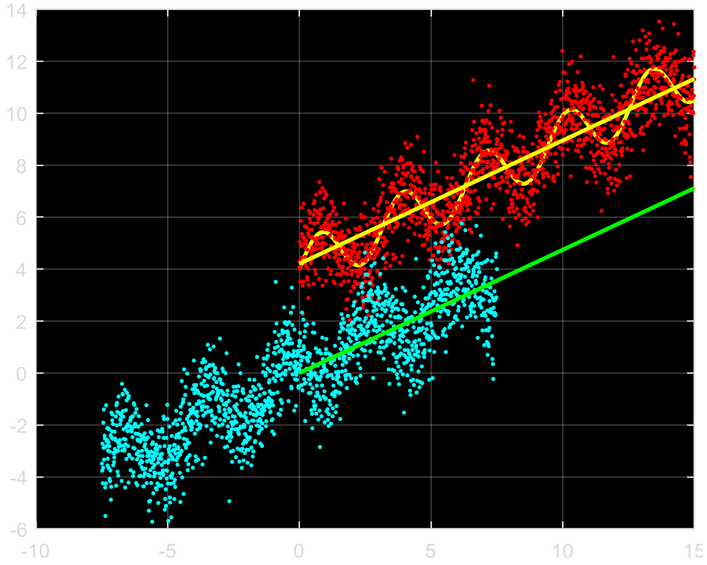
1) Create the dataset
2) Plot the dataset as red dots
3) Plot the above equation in yellow

(after point 3, your plot should look something like the figure below: )



4) mean centre the data and plot it in cyan (add it to your existing drawing)
5) assuming that *y = x * w + noise* learn w by least squares optimization using the mean centred data.
6) plot the linear model in green
7) re-plot the model centred on the original data (in yellow)

Your final plot should look something like this:



## Marking Criteria

In order to obtain full marks for each question, you must answer it correctly and  completely.
**Marks will be given for writing compact, vectorised code and avoiding the use of “loops” (*for* or *while* loops) for carrying out operation on matrix and vector elements.**