Assignment 2

Statistical Computing with R, 2023-24

Introduction

This assignment consists of 2 exercises that have been designed to test the skills you have learned in the first 8 lectures of Statistical Computing with `R`.

When solving the assignment, please bear in mind the following points:

1. you are warmly encouraged to use the packages and functions that you have encountered in the course so far, as well as the ones given as hints in this assignment;
2. the assignment is optional and not graded. You are allowed to discuss your solutions and collaborate with classmates, but please prepare your own solutions, submitting code and answers that you wrote by yourself - this way we can give you feedback about your thought process and code style, which will help you assess your level of preparation so far and will be useful to improve your skills before the exam;
3. try to structure your solutions clearly, and write tidy and commented code.

Preparing your solutions

Please use `R` Markdown to prepare and hand in your solutions:

1. use `R` Markdown to write your solutions to the assignment. Name your `.Rmd` file as follows: `surname_name_SCwR_A2.Rmd`, where `surname` is your surname, and `name` your name;
2. use section and subsection headers to structure the document, so that it is clear which exercise and question you are answering;
3. put your `R` code in code chunks. Keep the argument `echo = TRUE` (= don't hide your code in the compiled pdf!);
4. write your answers inline (= outside code chunks); please don't include your answers as comments inside code chunks!
5. compile your solutions as `.pdf`. If you have issues in compiling your document as pdf, you can reach out to the TAs for help!

Submission

The **deadline** for submissions is **Sunday 12th November 23:59**. To hand in your solutions, submit both your `.Rmd` and `.pdf` files through Brightspace.

Doubts / questions?

If you have any questions about the assignment, feel free to ask the TAs during the coding sessions, the question hour or via email to statcompr[at]gmail.com.

Good luck and have fun!

Exercise 1

Exercise 1 is about data visualization. Make sure that the charts you will create in this exercise are clear and readable, and make sure to include adequate titles and axis labels. Use the base `R` visualization functions covered so far in the course to create the charts in this exercise.

Data preparation

1. We will use a dataset that includes information on Airbnbs in the Ragusa province on Sicily (original source: "Airbnbs in Ragusa, Sicily" dataset from Kaggle). Download this file from Brightspace, and load it into R.
2. Create a subset of the dataset with the variables that we are going to use. These are:
   - neighbourhood_cleansed;
   - latitude;
   - longitude;
   - price.
3. We are now first going to clean the price-variable. To be able to use this variable as a numeric variable, take the following steps:
   - Remove the $ sign in front of the price;
   - Remove the ',' in prices over $1,000.00;
   - Convert the variable from character to numeric.

(Hint: take a look at the `gsub()` function.)

Prices per neighbourhood

4. Create a new dataframe that contains one column with the neighbourhood name, and one column with the average price per neighbourhood.
5. Create a vector with 12 colour names: one per neighbourhood. Name the vector `coloursvector`. Create a horizontal barplot showing the mean airbnb price per neighbourhood, and colour the bars according to `coloursvector`. Make sure that all neighbourhood names are visible (you can remove the y-axis label to increase readability).

Airbnb locations

6. Create a scatter plot with the latitude and longitude of the airbnbs. Colour the airbnbs based on their neighbourhoods. Use the same colour vector as the one you used in Exercise 2.5. To make sure you know which colours correspond to which level of the neighbourhood factor, use `col = coloursvector[as.factor(data$neighbourhood_cleansed)]`. The k-th colour in the colour vector now corresponds to the k-th neighbourhood in the factor levels.
7. Add a legend to the chart that matches the neighbourhood colours to the neighbourhood names.

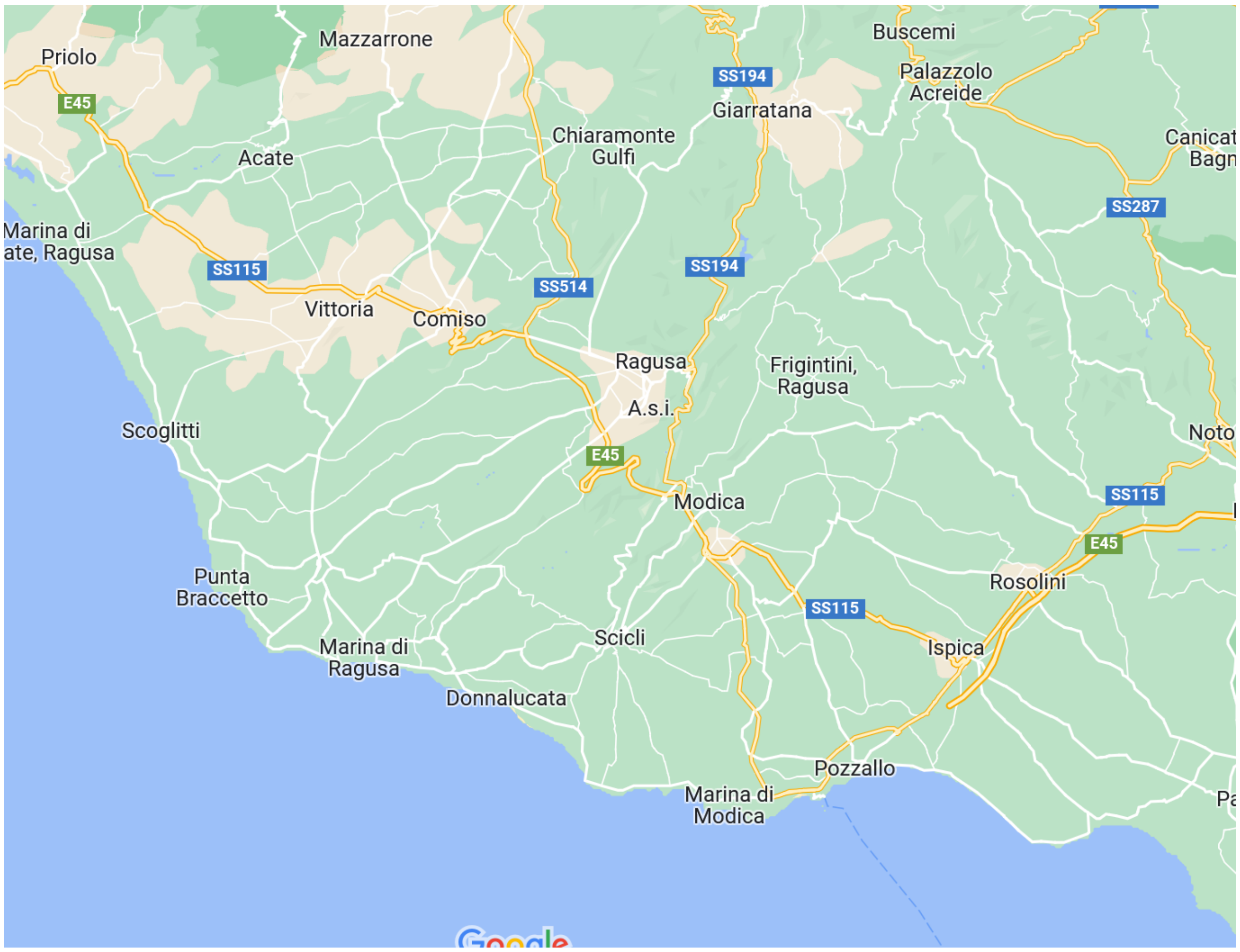

Figure 1: Ragusa province - Italy

9. Does the map you created correspond to the map of the Ragusa province in Figure 1? (You may use google maps to zoom in more on the different areas of the map)
10. Search for the "Ragusa cathedral" on google maps, and find its longitude and latitude. Clearly mark the location of this cathedral on your plot with a symbol of your choice.
11. Add a second legend that shows how the Ragusa cathedral is marked on the map.
12. Save your plot as a `.jpg` to your working directory.

Exercise 2

In this exercise you are going to write a function that implements a stock investing simulator where we invest a certain amount of money to buy some shares at one date, wait, and sell all our shares at a later date. Specifically, we invest `investment_USD` dollars to buy an integer number of APPL shares with `Open` price at `start_date`. We then simulate how much money we would have if we sold all of the shares at any day between the `start_date` and a given `end_date`.

To implement the simulator, we need to define a few quantities of interest. Let:

- t be a time variable that measures the number of days from the `start_date` (such that $t = 0$ corresponds the `start_date` itself);
- $v(t)$ be the value of one share at time t;
- I be the initial investment in USD.

The number of shares s bought at `start_date` can be obtained as

$$s = \left\lfloor \frac{I}{v(0)} \right\rfloor ,$$

where $\lfloor\ \rfloor$ denotes the modulo operator. Keep in mind that because $s \in \mathbb{N}$, you might have a non-zero amount of money left over, namely $I - s \times v(0)$ dollars.

The value of the investment at time t is given by

$$V(t) = s \times v(t) + (I - s \times v(0)),$$

and the profit / loss at time t is

$$\pi(t) = V(t) - V(0).$$

Task 1

Download the AAPL stock data from this URL, and import it in `R`. Make sure the `Open` column is numeric, and the `Date` column is of Date type. To convert the date, you may use `as.Date(AAPL$Date, format = "%Y-%m-%d")`.

Task 2

Write a function called `simulate_investment()` that takes inputs `start_date` (character), `end_date` (character) and `investment_USD` (numeric), and returns a summary data frame as output. The output dataframe should contain the value of our investment on each day between `start_date` and `end_date`, and should have three columns: `Date`, investment `Value` $v(t)$ and `Profit` $\pi(t)$.

When implementing this function, try to keep the following requirements in mind:

- your function takes in `start_date` and `end_date` as character strings, but the code inside your function should interpret these as correctly formatted dates (`as.Date()` may once again be useful here);
- your function should comprehensively check for bad inputs / impossible conditions (there are many!), and **return informative errors or warnings**. Think about the many ways in which the simulator function may fail, for example: the dates don't exist, are outside the temporal range of the data, are reversed, are not correctly formatted dates, and so on. Try to be as comprehensive as possible with your checks.

For example,

```r
df_summary <- simulate_investment(start_date = "2015-10-01",
                                  end_date = "2015-10-11",
                                  investment_USD = 10000)
```

should create a dataframe `df_summary` with 11 rows (one each for every day until 2015-10-11) that looks like this:

Table 1: Simulation Results

Date	Value	Profit
2015-10-01	10000.00	0.00
2015-10-02	9903.54	-96.46
2015-10-03	9903.54	-96.46
2015-10-04	9903.54	-96.46
2015-10-05	10073.71	73.71
2015-10-06	10141.96	141.96
2015-10-07	10242.97	242.97
2015-10-08	10101.92	101.92
2015-10-09	10084.63	84.63
2015-10-10	10084.63	84.63
2015-10-11	10084.63	84.63

Task 3

Simulate investing 10000 `$` between `01/10/2011` and `11/10/2011`. Display the resulting data frame.

Task 4

Simulate investing 5000 `$` between `01/10/2011` and `01/10/2016`. Create a line plot of the resulting summary dataframe with `Date` on the x axis and `Value` on the y axis, paying attention to all visual elements in the chart (make sure it looks good!).

By looking at the chart, when would it have been best to sell?

Task 5

Use the output dataframe from the previous question to find the exact date where the `Value` of the investment was the highest. Return both the date and the highest value.