Exercises for Lecture 7

Statistical Computing with R, 2023-24

Exercise 1

1. Create an RStudio project and link it to the folder where you have saved all your files from Lecture 7
2. Create an R Markdown document where you will write your solutions to this assignment. Save the document in the project folder. Make sure to use section (#) and subsection (##) headers to clearly structure your solutions
3. Compile the RMarkdown document by "knitting it" to a `pdf` output. If you have issues compiling to `pdf`, ask for help from the TAs. They might be able to help you solving the problem. If the problem persists, compile your solutions as `html` file

Exercise 2

Write a function that takes a dataframe as input and does the following:

- Check if the dataframe contains any numeric, character or factor variables. If not, return the message "This dataframe contains no numeric, character or factor values".
- If the dataframe contains at least one numeric, character or factor variable, then:
  - Create a boxplot for each numeric variable
  - Create a barplot for each character/factor variable

Check your function using a the `gala` and `amlxray` datasets, which are available in the package `faraway`.

Exercise 3

The `state.x77` matrix from the `MASS` package contains (outdated) information on the 50 US states. To get it, load the `MASS` library:

```
library(MASS)
?state # check the help page out!
head(state.x77)
```

```
##            Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
```

```
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
```

1. Compute the mean, median, standard deviation and skewness of each of the numeric variables in the data frame. Store the results in a data frame where each row corresponds to a variable, and each column to a summary statistics
2. Draw a histogram or a density plot for each variable. Make sure the name of each variable is included in the title of the chart
3. Compare the plots to the summary variables: which variables appear to be symmetric? Which ones are positively skewed, and which ones negatively skewed?
4. Make a scatter plot of `Illiteracy` versus `Life Exp`. Do you notice any trend?
5. Use the function `cor.test( )` to evaluate the correlation between `Illiteracy` and `Life Exp`. Can you think of a plausible explanation for this correlation?

Exercise 4

Execute the following code:

```
c1 = c(0, 1)
c2 = expand.grid(c1, c1)
c2s = rowSums(c2)
```

1. Create a chart to visualize the frequency distribution of `c2s`. Inspect `c2`, `c2s`, and the chart. What are the dimensions of `c2`? What do you think `expand.grid()` and `rowSums()` do? (Check out the documentation of the two functions to double-check your answer)
2. Create the equivalent of `c2` and `c2s`, but repeating `c1` 5 times. Call the outcomes `c5` en `c5s` respectively. Create a chart to visualize the frequency distribution of `c5s` Can you notice something happening to the distribution of `c5s`? What are the dimensions of `c5`?
3. Now use `expand.grid(replicate(15, c, simplify = F))` to create `c15`, and use `c15` to create `c15s`. Again, visualize the frequency distribution of `c15s`. Can you predict and/or calculate the dimensions of `c15`?
4. Calculate and save the range, mean, and standard deviation of `c15s`. Are these numbers surprising? If so, how could you verify if these numbers are correct? (Hint: think about what you've learnt in the Probability course!)
5. Use the `rnorm( )` function to draw $n = 10^3$ random numbers from a normal distribution with mean `mean(c15s)` and standard deviation `sd(c15s)`. Create a histogram of your draws (set the parameter `breaks = 16`). What are your conclusions?
6. Compare the histogram of `c15s` to the density of a normal distribution with mean $\mu =$ `mean(c15s) - 0.5` and standard deviation $\sigma =$ `sd(c15s)`; combine the histogram and the density curve in the same chart. What are your conclusions? (Hints: 1. for the histogram, use `prob = T`; 2. to add the normal curve, use `dnorm(x, mean, sd)` and

```
    add = T)
```

Exercise 5

How many elements in the Fibonacci sequence do you need to sum, before the sum of such elements reaches 22000?

Exercise 6 (extra exercise)

Note: this exercise is not material for the exam; it goes beyond the scope of the material you covered so far. It is a fun one, so if you want to, give it a go!

Part 1: manipulating images in `R`

Download the file `academiegebouw.jpg` from https://github.com/mirkosignorelli/Teaching/tree/main/SCwR_course:

This image contains 666 x 1000 pixels. In this exercise, we will manipulate this image in `R`. An image has 3 channels Red, Green and Blue (RGB). Those values range from 0 to 255. Install the `jpeg` package and see `?readJPEG`.

```
install.packages("jpeg")
```

1. Import the image in `R`, and assign it to an object called `my_image`. What type of object is it? (Tip: use `is(my_image)` to verify that). Do the dimensions of `my_image` make sense?

2. Black and white images are 2 dimensional arrays, typically obtained by computing the average of the 3 color channels. Convert the image to black and white, and export it as `.jpg` file using the `writeJPEG( )` function.

3. Create a function to convert the 3D array to a `data.frame` with 5 variables: column, row, red, blue and green. The resulting data frame should have 666 x 1000 = 666000 rows (to wit, one row for each pixel). You may find useful the function `expand.grid()`. See `?expand.grid`. Notice that a the 3D array is a 2D array in 3 different channels. In `R` you can get all the values from one of the channels with `my_image[ , , channel_number]`. The latter is a 2D array.

4. Check how many different colors exist in the your `data.frame`. Remember that a color is the combination of red, green and blue.

Part 2: compressing an image using the k-means method

NB: the remainder of this exercise is a bit more advanced than usual, so don't worry if you have troubles understanding it or implementing it!

K-means is a simple clustering algorithm that can be shortly summarized as follows:

1. Choose randomly k starting points as centroids.
2. While not converged (no movement in the centroids):
    1. Assign each point to the nearest centroid.
    2. Update the centroid with the mean of all the points under that cluster.

In the context of image compression, the idea is that the k-means algorithm can be used to summarize the original image using less memory (the stronger the compression and memory saving, the smaller the number of unique colors). To apply the k-means algorithm, you can use the function `kmeans()` (see `?kmeans()`).

5. Apply `kmeans()` function to the red, green and blue colors of our `data.frame`. Use 40 centroids. Store the clustered values, and check the value of the centroids. Those are your new colors!

6. Use the new centroids as the value for your colors in each pixel. The pertenence of each row in the `data.frame` to each cluster is given by the function. Substitute the values of each row for the values of its cluster. Notice that `kmeans()` returns the value of each center and a vector with cluster label of each row.

7. Convert the new `data.frame` to a 3D array, save it to a `.jpg` file and visualize it. What happens if you reduce the number of centroids?