Assignment 1

EMATM0061: Statistical Computing and Empirical Methods, Data Science MSc
Teaching Block 1, 2024

Before starting the assignment it is recommended that you first watch video lectures 2 and 3. You don't need to hand in this assignment.

1 Create a data frame

First open RStudio.

~~Step 1.~~ Within RStudio, find the Console window. In the Console, write a command to create a vector called `animals` which contains the names of a few animals. What type of vector is this? Your vector might look something like this:

```
## [1] "Snake" "Ostrich" "Cat" "Spider"
```

~~Step 2.~~ Create a vector of the same length called `num_legs` which gives the number of legs of the different animals. Your second vector will look something like this:

```
## [1] 0 2 4 8
```

~~Step 3.~~ Now combine these two vectors in a data frame called `animals_df` which has two columns - one with the names of animals, and another with their respective numbers of legs. The result should look something like this:

```
## animals num_legs
## 1 Snake 0
## 2 Ostrich 2
## 3 Cat 4
## 4 Spider 8
```

Step 4. Check the `Environment` window and find out what objects (variables) are available. You should be able to find objects called `animals`, `num_legs`, and `animals_df`. Can you find the data type of these objects there?

The functions required by the tasks above can be found in the example below.

```
city_name <- c( "Bristol", "Manchester", "Birmingham", "London") # vector of city names
population <- c(0.5,0.5,1,9) # vector of populations
cities_populations_df <-data.frame(city_name,population) # generating data frame
```

2 Check and delete objects

Apart from creating objects, you can also check and delete them. In this question, you will also learn how to find help documents of R functions.

Step 1. After you finish creating the data frame required by Question 1, get a list of objects in the working environment.

- You can use the function `ls()` which returns a vector of character strings giving the names of the objects in the environment. Type `?ls` to check its usage.
- Do you find the objects `animals`, `num_legs`, and `animals_df` again?

Sept 2. Use the function `rm` (type `?rm` to see examples of using `rm`) to remove the objects `num_legs` from the working environment. After that, check the list of objects in the current environment again, to see if `num_legs` has been removed.

Step 3. Remove all objects in the working environment.

3 Create a data frame in R Scripts

In Question 1, you created a data frame in the console in RStudio. Now you will create an R script and create a data frame using the script.

Step 0. Remove all objects in the working environment, similar to what you have done in Question 2.

Step 1. Go to File –> New File –> R Script to create your script. Save the file by clicking File –> Save and giving the file a name of your choice eg. "myFirstRScript".

Step 2. Now you can start writing codes in the script. Create a data frame `animals_df`, using the same code you did in Question 1

Step 3. You can run all the code within your script by clicking on the "Source" button on the top right (alternative click `Code -> Source`).

Step 4. Get a list of objects in the working environment. Is the list the same as what you got in Question 2 Step 1?

4 Create a data frame in R Markdown

Let's create an R markdown with html output as follows:

```
File -> New File -> R Markdown ...
```

You can then choose:

- A title for your document;
- An author name (for example your name);
- An output format. Let's choose HTML.

Then click "OK" to create an R markdown. This will create a template project. Explore the project and note its key features:

1. There is a block of **YAML** header at the start of the document which gives key information eg. title and output format.
2. You can create **headings of sections** with "#". Similarly, headings of subsections can be created with "##" and so on.
3. You can embed **blocks of R code** by using ``` ```{r} ``` before the R code and ``` ``` ``` afterwards.
4. By default both the code and its output are displayed. If you only want to include the output but not the code itself include the option `echo = FALSE` in the code prefix. If you don't want to include either then include the option `include = FALSE`.
5. You can include hyperlinks by writing <`url-name`>.

Save your R markdown file as "MyFirstRMarkdown". Remove everything in your R markdown except for the YAML section at the top with the title, author, date, and output. Then do the following steps.

Step 1. Insert a block of R code. In this block of code, create a data frame `animals_df`, using the same code you did in Question 1. In R Markdown, a block of code starts with ```` ```{r} ```` and ends with ```` ``` ````

Step 2. Insert another block of code to print `animals_df`.

Step 3. You can then generate an HTML file with the "Knit" button just above your scripting window. Check what you have in the HTML file.

5 Matrix operations

Use the `seq()` function to generate a sequence of numbers starting at 12 and decreasing to 2 in steps of -2. Call this vector `x_vect`. You may want to run `?seq` or `help(seq)` to help you do this. The vector `x_vect` should look like this:

```
## [1] 12 10 8 6 4 2
```

Now convert the vector `x_vect` into a matrix (with 2 rows and 3 coloumns) called X, using the `matrix()` function. The result should look like this:

```
##      [,1] [,2] [,3]
## [1,] 12   8    4
## [2,] 10   6    2
```

Next create a 2 by 2 matrix called Y consisting of a sequence of four numbers from 1-4. The matrix Y should look like this:

```
##      [,1] [,2]
## [1,] 1    3
## [2,] 2    4
```

In addition, create another 2 by 2 matrix called Z which looks as follows:

```
##      [,1] [,2]
## [1,] 4    8
## [2,] 6    10
```

Recall that for 2 by 2 matrices A, the transpose A^T is given by

$$A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix} \quad \text{where} \quad A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

Use the `t()` function to compute Y^T and Z^T.

Matrix sums. Now compute the matrix sums $Y + Z$ and $Y + Z$. The result in both cases should be the same. We call such operations commutative. This means that reversing the order does not change the result.

Matrix multiplication. Use R to compute the matrix products YZ and ZY. Are the results the same in both cases? Is matrix multiplication commutative?

Recall that the matrix multiplication between two 2 by 2 matrices A and B, is defined as

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}, \quad \text{where} \quad A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}.$$

In addition, use R to compute the matrix product YX. You should get something like this:

```
##      [,1] [,2] [,3]
## [1,] 42   26   10
## [2,] 64   40   16
```

What happens if you attempt to compute the matrix product XY? Explain the error message you get.

Matrix element-wise multiplication. Use R to compute the element-wise matrix products $Y \bigodot Z$ and $Y \bigodot Z$. Are the results the same in both cases? Is element-wise multiplication commutative?

Recall that the element-wise matrix multiplication between A and B is defined as

$$A \bigodot B = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix},$$

Matrix inverse. Compute the matrix inverse Y^{-1} via the `solve()` function:

```
solve(Y)
```

You should get something like this

```
##      [,1]  [,2]
## [1,] -2    1.5
## [2,] 1    -0.5
```

Next, check what you get from computing $Y^{-1}Y$ in R.

Now compute $Y^{-1}X$. Your result should look like this:

```
##      [,1]  [,2]  [,3]
## [1,] -9    -7    -5
## [2,] 7     5     3
```

Can you do this (obtain the results of $Y^{-1}X$) without first computing Y^{-1}? Try running the help command on the `solve()` function by typing `?solve` into the R console to find out how to do this.

6 Writing a function within R

First, create an R script (see also Question 3), and saving it using the name `mySecondRScript`.

To do the following steps, it is good to know how a function is defined in R as well as basic control flow statements in R (e.g., `if...else...`, and `for`). You may want to first look at the sample function called `is_prime` (provided at the end of this question) for function definition and control flow statements, before you continue.

Step 1. Within your script create a short function called `myFirstRFunc` which takes in a single numerical argument n and outputs the sum of all those numbers strictly below n which are divisible by either 2 or 7 or both.

For example, if $n = 14$ then it should be the sum of $2, 4, 6, 7, 8, 10, 12$, so `myFirstRFunc` applied to 14 gives the answer $2 + 4 + 6 + 7 + 8 + 10 + 12 = 49$.

Make sure your function includes useful comments. You may want to include a check so that your function produces an error if it is given an argument which isn't a non-negative integer.

Step 2. Run the script by clicking on the "Source" button on the top right. Then check what you get if you apply the function to 1000, i.e.,

```
myFirstRFunc(1000)
```

If you have been successful your function should produce the following output.

```
## [1] 284787
```

How to write a function in R? An example.

```
is_prime <- function(num){

  # This example function takes as input a positive integer and outputs Boolean

  stopifnot(is.numeric(num),num%%1==0,num>=0) # Stop if the input is not a non-negative integer

  t_val <- TRUE # Initialise truth value output with TRUE

  if(num<2){

    t_val<-FALSE # Output FALSE if input is either 0 or 1

  }else if(num>2){

    for(i in 2:sqrt(num)){ # Check possible divisors i no greater than sqrt(num)

      if(num%%i==0){
        t_val<-FALSE
        break     # if i divides num then num is not prime

      }
    }
  }

  return(t_val) # return the truth value which says whether or not num is prime
}
```

7 Futher R Markdown exercises

Next we will learn about how to include plots and mathematical formulae in R Markdown.

First, similar to what you did to set up `MyFirstRMarkdown` in Question 4, create a new R Markdown file and save it using the name `Assignment1RMarkdown`.

Step 1. Within your R markdown insert a section heading called "Wave plot".

Step 2. Insert a code block to do the following.

- Define a vector called x consisting of a sequence which starts at 0 and goes to 20 in increments of 0.01. You can do this using the `seq()` function.
- Next create a vector called y, which is of the same length as x, such that the i^{th} entry of y is equal to the sin function of the i^{th} entry of x. This can be done via the R function `sin()`.
- Then create a data frame called `sin_df` with two columns: x and y. You can inspect the first few rows of your data frame with the `head()` function like this:

```
head(sin_df,3)
##    x           y
## 1 0.00 0.000000000
## 2 0.01 0.009999833
## 3 0.02 0.019998667
```

If you have been successful you will observe a similar output after performing "knit".

- Write a statement to plot `sin_df`.You can do this using the `plot()` function. This aims to display a plot with x as the x-axis and y as the y-axis.

Step 3. Insert the following mathematical formula into your Markdown file:

$$\sin^2(x) + \cos^2(x) = 1.$$

About how to insert in the Markdown file mathematical formulae.

You can use R Markdown to render mathematical formulae. For example, you can write $y = \sin(x)$ inline to display "y = sin(x)".

You can also display more complex expressions. For example, the code below gives rise to the following output.

```
\[\sin(x)=\sum_{n=1}^{\infty}(-1)^{n+1}\cdot \frac{x^{2n-1}}{(2n-1)!}
\approx x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}\ldots. \]
```

The output will be:

$$\sin(x) = \sum_{n=1}^{\infty}(-1)^{n+1} \cdot \frac{x^{2n-1}}{(2n-1)!} \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \ldots.$$

```
\[ A=\left(\begin{matrix} a_{11} & a_{12}\\ a_{21} & a_{22} \end{matrix}\right). \]
```

The output will be:

$$A = \left(\begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix} \right).$$

For information about writing mathematical formulae in Rmarkdown documents visit: https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html#equations

8 Version control with RStudio and git

Next we will combine RStudio with git to create an R project with version control.

8.1 Connect RStudio to git

If you do not yet have a GitHub account sign up for one here https://github.com/.

Next, connect your RStudio to git via the `usethis` R package. Within RStudio perform the following steps within your R console:

```
install.packages("usethis") # Install the "usethis" R package
library(usethis) # Load the "usethis" R library
use_git_config(user.name = "Bob Smith", user.email = "bob@example.org") # Set profile info
```

The first step installs the `usethis` R package. The second step loads the `usethis` R library. You may be prompted to install RTools at this stage. If so follow the links provided by RStudio. The third step sets your git profile information. The user.name can be the same as the username you used to set up your git profile, but it is not necessary. The email needs to be the same email address as you used to set up your git profile.

8.2 Create an R project with version control

1. Create a project on GitHub. Go to https://github.com/ within your web browser and log into your git account. Create a new repository by clicking the green "New" button. Next:

- Give your repository a name eg. "`firstRProject`".
- Give your repository a description eg. "This is a repo for an R project".
- Check the box which says "Add a README file".
- Always be mindful of whether your repo is public or private.
- Click the green "Create repository" button.

You have now created a git repository. Next click the green "Code" button and copy the url provided. This is the repository url.

2. Create a project at RStudio. Go back to RStudio. Then go to:

```
File -> New Project... -> Version Control -> Git
```

- Below "Repository URL:" Paste the repository url which you copied from your git repo. (if you have chosen your project to be private, you may be asked to input your account details. Follow the instructions to gain access).
- Below "Project directory name:" Choose a directory name.
- Below "Create project as a subdirectory of:" Choose where to store your project on your local machine.
- Check the box marked "Open in new session".
- Click create project.

You have now created an R project with version control!

8.3 Committing and pushing

Within your new project click

```
File -> New File -> R Script
```

Type some R code in your new R script e.g.

```
a<-1
```

Now save your new file by selecting File $->$ Save before giving a name.

commit. Next go to the git tab at the top right of your screen. Then click the "commit" button.

On the left you will see a collection of check boxes. Here you can choose which files you want to add to your git repo. I suggest for now you check all of the files.

Write a commit message in the "Commit message" box eg. "This is my first commit". It's good practice to always include a commit message. Click the commit button. You should see a message which describes your changes. Commits act as a "snapshot" which records the current state of your repository.

push. You have now taken a local "snapshot" of your repository by performing a "commit". Next you want to record this snapshot on the central repository. This is done by performing a "Push". All you have to do is press the green "Push" button within the git panel in RStudio. You may be asked to enter your password or personal access tokens (you can create your token if needed https://github.com/settings/tokens). You typically only have to do this once.

Now check that your push has been successful. Go back to https://github.com/ and log into your account if you are not already logged in. Go to the repositories section and click on the name of the repository you created in the previous stage of this assignment. You should see a record of your most recent commit.

Additionally, you may want to move your previous R script entitled "myFirstRScript" (from Question 3) into this project. You can then "commit" and "push" to upload your script to your git account.