Homework 1

Ziyao Yang

Table of contents

❗ Due: Sun, Jan 29, 2023 @ 11:59pm

Please read the instructions carefully before submitting your assignment.

1. This assignment requires you to:
   - Upload your Quarto markdown files to a `git` repository
   - Upload a `PDF` file on Canvas
2. Don't collapse any code cells before submitting.
3. Remember to make sure all your code output is rendered properly before uploading your submission.

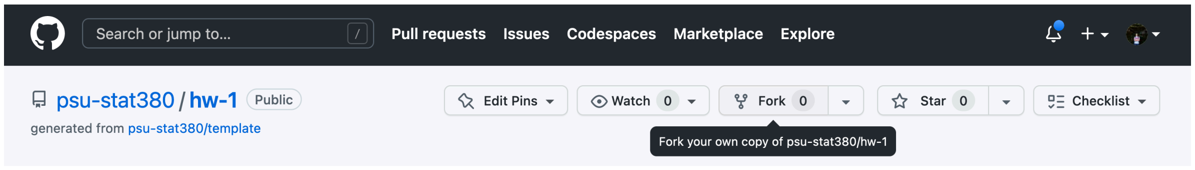
Please add your name to the the author information in the frontmatter before submitting your assignment.

Question 1

20 points

In this question, we will walk through the process of *forking* a `git` repository and submitting a *pull request*.

1. Navigate to the Github repository here and fork it by clicking on the icon in the top right



   Provide a sensible name for your forked repository when prompted.

2. Clone your Github repository on your local machine

```
$ git clone <<insert your repository url here>>
$ cd hw-1
```

   Alternatively, you can use Github codespaces to get started from your repository directly.

3. In order to activate the `R` environment for the homework, make sure you have `renv` installed beforehand. To activate the `renv` environment for this assignment, open an instance of the `R` console from within the directory and type

```
renv::activate()
```

   Follow the instrutions in order to make sure that `renv` is configured correctly.

4. Work on the *reminaing part* of this assignment as a `.qmd` file.

   - Create a `PDF` and `HTML` file for your output by modifying the YAML frontmatter for the Quarto `.qmd` document

5. When you're done working on your assignment, push the changes to your github repository.

6. Navigate to the original Github repository here and submit a pull request linking to your repository.

   Remember to **include your name** in the pull request information!

If you're stuck at any step along the way, you can refer to the official Github docs here [1]

—

Question 2

30 points

Consider the following vector

```r
my_vec <- c(
    "+0.07",
    "-0.07",
    "+0.25",
    "-0.84",
    "+0.32",
    "-0.24",
    "-0.97",
    "-0.36",
    "+1.76",
    "-0.36"
)
```

For the following questions, provide your answers in a code cell.

1. What data type does the vector contain?

```r
    typeof(my_vec)
```

```
[1] "character"
```

2. Create two new vectors called `my_vec_double` and `my_vec_int` which converts `my_vec` to Double & Integer types, respectively,

[1] I was unable to render the pdf file. I upgrade R version and all the packages, followed the instructions returned by "renv::status()" and synchronized the project with the lockfile. I'm not sure if these are the right process to solve the problem but I did successfully rendered pdf after these steps.

```r
my_vec_double <- as.double(my_vec)
my_vec_double
```

```
 [1]  0.07 -0.07  0.25 -0.84  0.32 -0.24 -0.97 -0.36  1.76 -0.36
```

```r
my_vec_int <- as.integer(my_vec)
my_vec_int
```

```
 [1] 0 0 0 0 0 0 0 0 1 0
```

3. Create a new vector `my_vec_bool` which comprises of:

   - TRUEif an element in `my_vec_double` is ≤ 0
   - FALSE if an element in `my_vec_double` is ≥ 0

```r
my_vec_bool <- ifelse(my_vec_double<=0, yes = TRUE, no = FALSE)
# ifelse(my_vec_double<=0, TRUE, FALSE)
# my_vec_bool <- my_vec_double<=0
my_vec_bool
```

```
 [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
```

How many elements of `my_vec_double` are greater than zero?

```r
sum(my_vec_double > 0)
```

```
[1] 4
```

4. Sort the values of `my_vec_double` in ascending order.

```r
sort(my_vec_double)
```

```
 [1] -0.97 -0.84 -0.36 -0.36 -0.24 -0.07  0.07  0.25  0.32  1.76
```

—

Question 3

50 points

In this question we will get a better understanding of how R handles large data structures in memory.

1. Provide R code to construct the following matrices:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \dots & 100 \\ 1 & 4 & 9 & 16 & 25 & \dots & 10000 \end{bmatrix}$$

Tip

Recall the discussion in class on how R fills in matrices

In the next part, we will discover how knowledge of the way in which a matrix is stored in memory can inform better code choices. To this end, the following function takes an input n and creates an $n \times n$ matrix with random entries.

```r
generate_matrix <- function(n){
    return(
        matrix(
            rnorm(n^2),
            nrow=n
        )
    )
}
```

For example:

```r
generate_matrix(4)
```

```
           [,1]       [,2]        [,3]       [,4]
[1,]  1.5318180 -0.9983621  0.47027687 -0.3500899
[2,] -0.2922491  0.4676499  1.54509368  0.5047034
[3,]  0.3214969  1.7944968 -0.25521193 -0.5990463
[4,]  0.6873778  1.6973251 -0.01395378 -0.9796172
```

Let M be a fixed 50×50 matrix

```r
M <- generate_matrix(50)
mean(M)
```

```
[1] 0.01430765
```

2. Write a function `row_wise_scan` which scans the entries of `M` one row after another and outputs the number of elements whose value is ≥ 0. You can use the following **starter code**

```
row_wise_scan <- function(x){
    n <- nrow(x)
    m <- ncol(x)

    # Insert your code here
    count <- 0
    for(...){
        for(...){
            if(...){
                count <- count + 1
            }
        }
    }

    return(count)
}
```

3. Similarly, write a function `col_wise_scan` which does exactly the same thing but scans the entries of `M` one column after another

```
col_wise_scan <- function(x){
    count <- 0

    ... # Insert your code here

    return(count)
}
```

You can check if your code is doing what it's supposed to using the function here[2]

4. Between `col_wise_scan` and `row_wise_scan`, which function do you expect to take shorter to run? Why?

[2] If your code is right, the following code should evaluate to be `TRUE`

```
sapply(1:100, function(i) {
    x <- generate_matrix(100)
    row_wise_scan(x) == col_wise_scan(x)
}) %>% sum == 100
```

5. Write a function `time_scan` which takes in a method `f` and a matrix `M` and outputs the amount of time taken to run `f(M)`

```
time_scan <- function(f, M){
    initial_time <- ... # Write your code here
    f(M)
    final_time <- ...  # Write your code here

    total_time_taken <- final_time - initial_time
    return(total_time_taken)
}
```

Provide your output to

```
list(
    row_wise_time = time_scan(row_wise_scan, M),
    col_wise_time = time_scan(row_wise_scan, M)
)
```

Which took longer to run?

6. Repeat this experiment now when:

    - `M` is a 100×100 matrix
    - `M` is a 1000×1000 matrix
    - `M` is a 5000×5000 matrix

What can you conclude?

—

Appendix

Print your `R` session information using the following command

```
sessionInfo()
```

```
R version 4.2.2 (2022-10-31)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Monterey 12.6

Matrix products: default
```

```
BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.2.2  fastmap_1.1.0   cli_3.6.0       htmltools_0.5.4
 [5] tools_4.2.2     rstudioapi_0.14 yaml_2.3.7      rmarkdown_2.20
 [9] knitr_1.42      xfun_0.36       digest_0.6.31   jsonlite_1.8.4
[13] rlang_1.0.6     renv_0.16.0-53  evaluate_0.20
```