HW1

Abdallah Al Rahbi

Question 2

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

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

#1)
```

```r
typeof(my_vec)
```

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

```r
# My_Vec contains "character" type data

#2)
my_vec_double = as.double(my_vec)
my_vec_int = as.integer(my_vec)

#3)

my_vec_bool = ifelse(my_vec_double <=0, T, F)

# Four elements of my_vec_bool are greater than 0

#4)
my_vec_double =
  my_vec_double %>%
  sort()
```

Question 3:

```r
#1)
matrix1 = matrix(c(1,2,3,4,5,6,7,8,9),
                 byrow = T,
                 nrow =3 )
matrix1
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

```r
matrix2= matrix(c(1:100,c((1:100)^2)),
                byrow = T,
                nrow =2)
matrix2
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,]    1    2    3    4    5    6    7    8    9    10    11    12    13    14
[2,]    1    4    9   16   25   36   49   64   81   100   121   144   169   196
     [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
[1,]    15    16    17    18    19    20    21    22    23    24    25    26
[2,]   225   256   289   324   361   400   441   484   529   576   625   676
     [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
[1,]    27    28    29    30    31    32    33    34    35    36    37    38
[2,]   729   784   841   900   961  1024  1089  1156  1225  1296  1369  1444
     [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
[1,]    39    40    41    42    43    44    45    46    47    48    49    50
[2,]  1521  1600  1681  1764  1849  1936  2025  2116  2209  2304  2401  2500
     [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
[1,]    51    52    53    54    55    56    57    58    59    60    61    62
[2,]  2601  2704  2809  2916  3025  3136  3249  3364  3481  3600  3721  3844
     [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
[1,]    63    64    65    66    67    68    69    70    71    72    73    74
[2,]  3969  4096  4225  4356  4489  4624  4761  4900  5041  5184  5329  5476
     [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
[1,]    75    76    77    78    79    80    81    82    83    84    85    86
[2,]  5625  5776  5929  6084  6241  6400  6561  6724  6889  7056  7225  7396
     [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
[1,]    87    88    89    90    91    92    93    94    95    96    97    98
[2,]  7569  7744  7921  8100  8281  8464  8649  8836  9025  9216  9409  9604
     [,99] [,100]
[1,]    99    100
[2,]  9801  10000
```

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

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

```
[1] -0.04084599
```

```r
#2)

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

    # Insert your code here
    count <- 0


    for(i in 1:n){
        for(j in 1: m){
            if(M[i,j] >= 0){
                count <- count + 1
            }
          }
      j = j +1
    }
    i = i + 1

    return(count)
}

#3)

col_wise_scan <- function(x){
    n <- nrow(x)
    m <- ncol(x)

    # Insert your code here
    count <- 0
    for(j in 1:m){
        for(i in 1:n){
            if(M[i,j] >= 0){
                count <- count + 1
            }
        }
      i = i +1
    }
  j = j + 1
```

```
    return(count)
}
```

4)

Since R uses column-major ordering, I expect the column wise scan to be faster

```
#5)

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

    total_time_taken <- final_time - initial_time
    return(total_time_taken)
}

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

```
$row_wise_time
Time difference of 0.00553894 secs

$col_wise_time
Time difference of 0.0001950264 secs
```

Row wise took longer to run than column wise as I expected in part 4

```
#6)


#100 by 100
M = M100 <- generate_matrix(100)

list(
    row_wise_time = time_scan(row_wise_scan, M100),
    col_wise_time = time_scan(row_wise_scan, M100)
)
```

```
$row_wise_time
Time difference of 0.0007321835 secs

$col_wise_time
Time difference of 0.0007898808 secs
```

```r
#1000 by 1000
M = M1000 <-generate_matrix(1000)

list(
    row_wise_time = time_scan(row_wise_scan, M1000),
    col_wise_time = time_scan(row_wise_scan, M1000)
)
```

```
$row_wise_time
Time difference of 0.0819509 secs

$col_wise_time
Time difference of 0.08212113 secs
```

```r
#5000 by 5000
M = M5000 <- generate_matrix(5000)

list(
    row_wise_time = time_scan(row_wise_scan, M5000),
    col_wise_time = time_scan(row_wise_scan, M5000)
)
```

```
$row_wise_time
Time difference of 2.161919 secs

$col_wise_time
Time difference of 2.12272 secs
```

For the 100 by 100 and the 5000 by 5000, column wise scanning is faster, however for the 1000 by 1000 matrix the row wise scanning is faster

Appendix

```
  sessionInfo()
```

```
R version 4.2.2 (2022-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22621)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

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

other attached packages:
[1] dplyr_1.0.10

loaded via a namespace (and not attached):
 [1] fansi_1.0.4      utf8_1.2.2       digest_0.6.31    R6_2.5.1
 [5] lifecycle_1.0.3  jsonlite_1.8.4   magrittr_2.0.3   evaluate_0.20
 [9] pillar_1.8.1     rlang_1.0.6      cli_3.6.0        renv_0.16.0-53
[13] vctrs_0.5.2      generics_0.1.3   rmarkdown_2.20   tools_4.2.2
[17] glue_1.6.2       xfun_0.36        yaml_2.3.7       fastmap_1.1.0
[21] compiler_4.2.2   pkgconfig_2.0.3  htmltools_0.5.4  tidyselect_1.2.0
[25] knitr_1.42       tibble_3.1.8
```

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

```
[1] TRUE
```