Lab 6

Code

AUTHOR
Maxim Dokukin

Remember, **follow the instructions below and use R Markdown to create a pdf document with your code and answers to the following questions on Gradescope.** You may find a template file by clicking "Code" in the top right corner of this page.

A. Basic functions

Use the following code to create a list of four matrices:

```
set.seed(100)
matrix_list <- list(
  A = diag(5),
  B = matrix(rnorm(9), nrow = 3, ncol = 3),
  C = matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2),
  D = diag(c(1:5))
)
```

1. Use the `lapply` function to create a list of length four containing the inverse of these four matrices.

```
matrix_list_i <- lapply(matrix_list, solve)
```

2. Use the `sapply` function to create a vector of length four containing the determinants of these four matrices.

```
matrix_det <- sapply(matrix_list, det)
```

B. Skewness and Kurtosis

3. Write a function `skewness()` that takes as input a numeric vector `x` and returns the sample skewness. There are functions in R that compute skewness, but you cannot use any of them–write your own

implementation. You may remove all `NA` values by default. Use your function to compute the sample skewness of the `arr_delay` variable in the `flights` dataset contained in the `nycflights13` package.

```
skewness <- function(x) {
  x <- x[!is.na(x)]
  return( (sum((x - mean(x))^3) / length(x)) / ((sum((x - mean(x
}

library(nycflights13)
print(paste('Skewness : ', skewness(flights$arr_delay)))
```

```
[1] "Skewness :  3.71680044883328"
```

4. Write a function `kurtosis()` that takes as input a numeric vector `x` and returns the sample skewness. There are functions in R that compute kurtosis, but you cannot use any of them–write your own implementation. You may remove all `NA` values by default. Use your function to compute the sample kurtosis of the `arr_delay` variable in the `flights` dataset contained in the `nycflights13` package.

```
kurtosis <- function(x) {
  x <- x[!is.na(x)]
  return( (sum((x - mean(x))^4)/length(x)) / ((sum((x - mean(x)
}

print(paste('Kurtosis : ', kurtosis(flights$arr_delay)))
```

```
[1] "Kurtosis :  29.2325791555245"
```

5. Write a function `get_column_skewness()` that takes as input a data frame and calculates the skewness of each **numeric** variable. The output should be a data frame with two variables: `variable` containing the name of the variable and `skewness` containing the skewness. Your output data frame should only include the numeric variables. You may remove all `NA` values by default. Demonstrate your function on the `penguins` dataset.

```
get_column_skewness <- function(df) {
```

```
  df2 <- df[, sapply(df, is.numeric)]
  skewness_vals <- sapply(df2, function(x) skewness(x))

  return(data.frame(skewness = skewness_vals))
}

library(palmerpenguins)
data <- penguins

df_skw <- get_column_skewness(data)

print(df_skw)
```

```
                   skewness
bill_length_mm     0.05288481
bill_depth_mm     -0.14283463
flipper_length_mm  0.34416383
body_mass_g        0.46826396
year              -0.05349321
```

C. Finding an error

Suppose you have two teams of runners participating in a 5k. We wish to write a function that takes as input two vectors representing the times of the runners in each team and returns a list of two vectors representing the ranks of each team's runners.

For example, if the first team's times are `c(16.8, 21.2, 19.1)` and the second team's times are `c(17.2, 18.1, 20.0)`, the function should return `c(1, 6, 4)` for the first team and `c(2, 3, 5)` for the second team.

Below is a draft version of the function `get_runner_ranks()`. However, there is an error somewhere. Use any method we discussed in class to identify the error.

```
get_runner_ranks <- function(x, y) {
  # combine all runner times
  combined_times <- c(x, y)
```

```
  # sort all runner times from fastest to slowest
  sort(combined_times, decreasing = T)
  #not saving the sort result in the variable
  #also sort order should be increasing

  # create ranks vectors
  ranks_x <- numeric(length(x))
  ranks_y <- numeric(length(y))

  for (i in seq_along(ranks_x)) {
    # look up rank of time i in x in combined_times
    ranks_x[i] <- match(x[i], combined_times) #should be index
  }

  for (i in seq_along(ranks_y)) {
    # look up rank of time i in y in combined_times
    ranks_y[i] <- match(y[i], combined_times) #should be index
  }

  # return a list of first team and second team ranks
  return(list(x = ranks_x, y = ranks_y))
}
```

6. Explain in your own words what the error was.

Not saving sort results, and sorting in the incorrect order.

7. Below, write a corrected version of `get_runner_ranks()` and compute `get_runner_ranks(c(16.8, 21.2, 19.1), c(17.2, 18.1, 20.0))`.

```
get_runner_ranks <- function(x, y) {
  # combine all runner times
  combined_times <- c(x, y)

  # sort all runner times from fastest to slowest
  combined_times <- sort(combined_times, decreasing = F)

  # create ranks vectors
```

```
  ranks_x <- numeric(length(x))
  ranks_y <- numeric(length(y))

  for (i in seq_along(ranks_x)) {
    # look up rank of time i in x in combined_times
    ranks_x[i] <- match(x[i], combined_times)
  }

  for (i in seq_along(ranks_y)) {
    # look up rank of time i in y in combined_times
    ranks_y[i] <- match(y[i], combined_times)
  }

  # return a list of first team and second team ranks
  return(list(x = ranks_x, y = ranks_y))
}

get_runner_ranks(c(16.8, 21.2, 19.1), c(17.2, 18.1, 20.0))
```

```
$x
[1] 1 6 4

$y
[1] 2 3 5
```