Lab 6

Jack Rellamas

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

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

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.

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

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

```
det_vector <- sapply(inv_list, det)
```

B. Skewness and Kurtosis

Skewness describes how asymmetrical the distribution of a numerical variable is around its mean. Given observations $x_1, \ldots, x_n$, we can calculate the sample skewness s of a variable using the following formula:

$$s = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right]^{3/2}}$$

Kurtosis is a measure of the "tailedness" of the distribution of a numerical variable is around its mean. Higher values of kurtosis indicate more extreme outliers. Given observations $x_1, \ldots, x_n$, we can calculate the sample kurtosis k of a variable using the following formula:

$$k = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^4}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right]^{2}} - 3$$

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.

```
flights <- nycflights13::flights

# function for skewness with given formula
skewness <- function(x) {
  # remove NA values from input vector
  x <- x[!is.na(x)]

  vector_mean = mean(x)

  sum_numerator = 0
  # calculate sum in numerator of skewness formula
  for (i in x) {
    term = (i - vector_mean)**3
    sum_numerator = sum_numerator + term
  }
  numerator = (1 / length(x)) * sum_numerator

  sum_denom = 0
  # calculate sum in  denom of skewness formula
  for (i in x) {
    term = (i - vector_mean)**2
    sum_denom = sum_denom + term
  }
  denom = ((1 / length(x)) * sum_denom)**(3/2)

  return(numerator / denom)
}

print(skewness(flights$arr_delay))
```

```
## [1] 3.7168
```

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.

```
# function for kurtosis with given formula
kurtosis <- function(x) {
  # remove NA values from input vector
  x <- x[!is.na(x)]

  vector_mean = mean(x)

  sum_numerator = 0
  # calculate sum in numerator of kurtosis formula
  for (i in x) {
    term = (i - vector_mean)**4
    sum_numerator = sum_numerator + term
  }
  numerator = (1 / length(x)) * sum_numerator

  sum_denom = 0
  # calculate sum in  denom of kurtosis formula
  for (i in x) {
    term = (i - vector_mean)**2
    sum_denom = sum_denom + term
  }
  denom = ((1 / length(x)) * sum_denom)**2

  return((numerator / denom) - 3)
}

print(kurtosis(flights$arr_delay))
```

```
## [1] 29.23258
```

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.

```
penguins <- palmerpenguins::penguins
get_column_skewness <- function(df) {
  # select only numeric cols
  numeric_only_df <- df %>%
    select(where(is.numeric))

  new_df <- NULL
  # puts skewness of each numeric col into new_df with same col names from original df, and skewness as
  for (col in colnames(numeric_only_df)) {
    s <- skewness(numeric_only_df[[col]])
```

```
    new_df[[get("col")]] <- c(s)
    new_df <- as.data.frame(new_df)
  }
  return(new_df)
}

penguin_skewness <- get_column_skewness(penguins)
print(penguin_skewness)
```

```
##   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g        year
## 1     0.05288481    -0.1428346         0.3441638    0.468264 -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)

  # 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))
}
```

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

The error was in the sorting the combined_times line. First, the program did not update the combined_times variable after sorting. Second, the program sorted in decreasing order, when it should have been sorted in increasing 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_corrected <- function(x, y) {
  # combine all runner times
  combined_times <- c(x, y)

  # sort all runner times from fastest to slowest
  # Change: sort(combined_times, decreasing = T) to combined_times <- sort(combined_times, decreasing =
  # Change: decreasing = T -> decreasing = F
  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))
}

team1 <- c(16.8, 21.2, 19.1)
team2 <- c(17.2, 18.1, 20.0)
correct_runner_ranks <- get_runner_ranks_corrected(team1, team2)
print(correct_runner_ranks)
```

```
## $x
## [1] 1 6 4
##
## $y
## [1] 2 3 5
```