MSCS 264: Homework #12

Due Tues, April 11 at 11pm

Al Ashir Intisar

As always, submit your knitted pdf file to Moodle, but be sure your RMarkdown file is saved and accessible in your Submit folder on the RStudio server.

A summary stats function

1. In this problem, we we will create a function called `summary_stats()` to produce a custom set of summary statistics (n, mean, median, sd, IQR, min, and max) for any variable that you input from a tibble. Follow the steps below to create your function.

(a) First, lets start with working code. Use `summarize` to calculate n, mean, median, sd, IQR, min, and max, for the hwy variable in the mpg dataset.

```
mpg|>
  summarise(n = n(), mean = mean(hwy), median = median(hwy), IQR = IQR(hwy), min = min(hwy), max = max(
```

```
## # A tibble: 1 x 6
##       n  mean median   IQR   min   max
##   <int> <dbl>  <dbl> <dbl> <int> <int>
## 1   234  23.4     24     9    12    44
```

(b) Copy your code from above and make it into a function with the name `summary_stats()`. It should have two inputs, `data` (the tibble of interest) and `x` (the variable of interest).

```
summary_stats <- function(tbl, variable){
  tbl|>
    summarise(n = n(), mean = mean({{variable}}), median = median({{variable}}), IQR = IQR({{variable}}

}
```

```
Hint: because of ["tidy evaluation"](https://dplyr.tidyverse.org/articles/programming.html), when you r
```

(c) Test your function on the hwy and cty variables from mpg.

```
summary_stats(mpg, hwy)
```

```
## # A tibble: 1 x 6
##       n  mean median   IQR   min   max
##   <int> <dbl>  <dbl> <dbl> <int> <int>
## 1   234  23.4     24     9    12    44
```

```
summary_stats(mpg, cty)
```

```
## # A tibble: 1 x 6
##       n  mean median   IQR   min   max
##   <int> <dbl>  <dbl> <dbl> <int> <int>
## 1   234  16.9     17     5     9    35
```

(d) Test your function on the age variable from gss_cat. Why do you get this output?

```
summary_stats(gss_cat, age)
```

```
## Error in `summarise()`:
## ! Problem while computing `IQR = IQR(age)`.
## Caused by error in `quantile.default()`:
## ! missing values and NaN's not allowed if 'na.rm' is FALSE
```

(e) Add an option to remove missing values, if any exist (and be sure the sample size `n` reflects the number of non-missing values). Thus, your function should have 3 inputs: `data` (the tibble of interest), `x` (the variable of interest), and `removeNA` (with a default value of FALSE).

```
summary_stats <- function(tbl, variable, removeNA = FALSE){

  if(removeNA == FALSE){
    tbl|>
    summarise(n = n(), mean = mean({{variable}}), median = median({{variable}}), IQR = IQR({{variable}}

  }
  else{
     tbl|>
      drop_na()|>
    summarise(n = n(), mean = mean({{variable}}), median = median({{variable}}), IQR = IQR({{variable}}

  }



}
```

(d) Test your new function on the hwy and cty variables from mpg, and the age variable from gss_cat.

```
summary_stats(mpg, hwy, TRUE)
```

```
## # A tibble: 1 x 6
##       n  mean median   IQR   min   max
##   <int> <dbl>  <dbl> <dbl> <int> <int>
## 1   234  23.4     24     9    12    44
```

```
summary_stats(mpg, cty, TRUE)
```

```
## # A tibble: 1 x 6
##       n  mean median   IQR   min   max
##   <int> <dbl>  <dbl> <dbl> <int> <int>
## 1   234  16.9     17     5     9    35
```

```
summary_stats(gss_cat, age, TRUE)
```

```
## # A tibble: 1 x 6
##       n  mean median   IQR   min   max
##   <int> <dbl>  <int> <dbl> <int> <int>
## 1 11299  47.2     46    27    18    89
```

Scraping movie data from IMDB

2. Go to the following website, and describe what the page shows: https://www.imdb.com/search/title?year=2017&title_type=feature&sort=boxoffice_gross_us,desc

Ans: The website page rates movies released in the year 2017 and also includes other data like title, genre, r rating status, number of revies, gross income from the movie etc. It is basically a movie rating web page.

3. The code below pulls the title and box office gross (money earned by tickets) for the top 50 grossing movies of a given year from the webpage! Look at "gross", "titles" and the resulting tibble! Turn this code into a function called "scrape_top50" with an input "movie_year". (Look closely! anywhere else that refers to 2017?)

```r
scrape_top50 <- function(movie_year){

url <- str_c("https://www.imdb.com/search/title?year=", movie_year, "&title_type=feature&sort=boxoffice_

# Don't worry about being able to write this code! We'll get there!
gross <- read_html(url) %>%
    html_nodes(".ghost~ .text-muted+ span") %>%
  html_text()

titles <- read_html(url) %>%
  html_nodes(".lister-item-header a") %>%
  html_text()

# This part you should understand!
tibble(movie = titles,
       gross = gross,
       year = movie_year)

}
```

4. Now that you have written your function, you can scrape data for multiple years! Scrape the data for 2015 to 2019. Name each resulting tibble "movies_2015" etc. Then you can use this code to put them together into one dataset:

```r
movies_2015 <- scrape_top50(2015)
movies_2016 <- scrape_top50(2016)
movies_2017 <- scrape_top50(2017)
movies_2018 <- scrape_top50(2018)
movies_2019 <- scrape_top50(2019)

all_movies <- movies_2015 %>%
  bind_rows(movies_2016) %>%
  bind_rows(movies_2017) %>%
  bind_rows(movies_2018) %>%
  bind_rows(movies_2019)
```

5. If you weren't able to get all_movies, you can read in the datset from our class code folder! Use either version to:

a) create a numeric variable called "gross_millions", which is the gross in millions of US dollars.

```r
all_movies <- all_movies|>
  mutate(gross_millions = parse_number(gross))
```

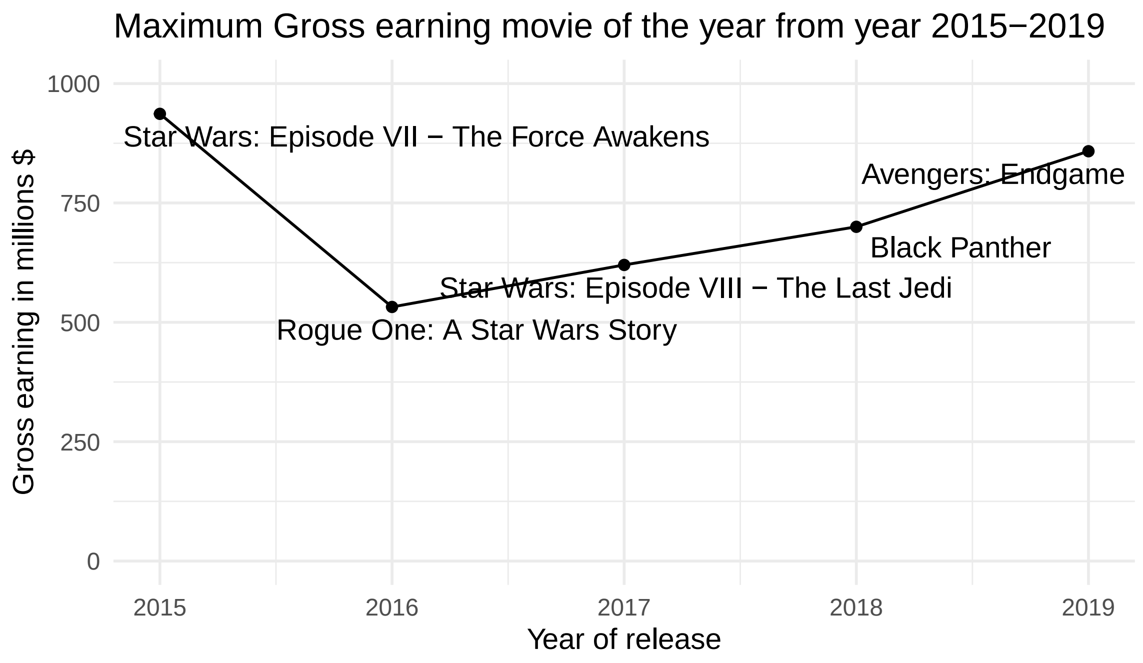
b) Use your summary_stats function to calculate a table of summary stats of "gross_millions" for each year. (Note: If you didn't get `summary_stats` working above, just calculate the mean gross_millions for each year).

```
all_movies|>
  group_by(year)|>
  summary_stats(gross_millions, TRUE)
```

```
## # A tibble: 5 x 7
##    year     n  mean median   IQR   min   max
##   <dbl> <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1  2015    50  167.   117.  106.  56.1  937.
## 2  2016    50  168.   123.  130.  65.1  532.
## 3  2017    50  169.   115.  120.  56.5  620.
## 4  2018    50  174.   124.  127.  58.0  700.
## 5  2019    50  177.   111.  100.  54.7  858.
```

c) From all_movies, get the top grossing movie for each year. Create a graph with year on the x axis, gross_millions on the y axis. Label each point with the title of the movie, and connect them with a line. Give your graph an appropriate title and axis labels. Use a new color and/or theme! Also use `coord_cartesian(ylim = c(0, 1000))` to make the y-axis go from 0 to 1 billion.

```
library(ggrepel)
all_movies|>
  group_by(year)|>
  slice_max(gross_millions, n = 1)|>
  ggplot(aes(x = year, y = gross_millions))+
  geom_point()+
  geom_line()+
  #geom_label(aes(label = movie))+
  geom_text_repel(aes(label = movie), nudge_y = -30, nudge_x = .3, segment.size = 0) +  # Add labels
  #geom_segment(aes(x = year, y = gross_millions, xend = year + 0.2, yend = gross_millions - 25),
               #arrow = arrow(length = unit(0.15, "cm")), show.legend = FALSE)+
  coord_cartesian(ylim = c(0, 1000))+
  labs(title = "Maximum Gross earning movie of the year from year 2015-2019", x = "Year of release", y
  theme_minimal()
```



d) Write the all_movies dataset to your submit folder!

```
write.csv(all_movies, "~/Mscs 264 S23/Submit Section A/Homeworks /all_movies.csv")
```