STAT 413/613 Homework on Web Data: APIs and Scraping

Yunting Chiu

2020-10-13

Instructions

- Write your solutions **in this starter file**.
    - Modify the "author" field in the YAML header.
- Commit R Markdown and HTML files (no PDF files). **Push both .Rmd and HTML files to GitHub**.
    - Make sure you have knitted to HTML for your final submission.
- **Only include necessary code and data** to answer the questions.
- Most of the functions you use should be from the tidyverse. **Too much base R** will result in point deductions.
- Submit a response on Canvas that your assignment is complete on GitHub
- Feel free to use Pull requests and or email (attach your .Rmd) to ask me any questions.

Libraries

```
# This package is required for Accessing APIS (HTTP or HTTPS URLS from Web)
library(httr)
# This package exposes some additional functions to convert json/text to data frame
library(jsonlite)
# This library is used to manipulate data
library(tidyverse)
```

```
## ── Attaching packages ─────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.3     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ──────────── tidyverse_conflicts() ──
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
```

```
# This puts the key into your computer's key and credential manager for storage
library(keyring)
# Renviron file using the {usethis} package function
library(usethis)
# Add datetime formats
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# friendly for color blind people
library(ggthemes)
# extract elements from HTML files
library(rvest)
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
##
##     pluck
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

1 Using APIs

- Pick a website of your choice and use an API to download a data set. Convert elements of interest into a tibble and create a graph to answer a question of interest.
- State the question and interpret the plot

```
# edit key access
# usethis::edit_r_environ() # I called this one is "NASA_KEY_SECURE"
jsonMarsWeather <- GET("https://api.nasa.gov/insight_weather/?api_key=DEMO_KEY&feedtype=
json&ver=1.0", apikey = key_get("NASA_KEY_SECURE"))

# This method will tell us what is the type of response fetched from GET() call to the A
PI.
http_type(jsonMarsWeather)
```

```
## [1] "application/json"
```

```
# This method just verifies if the response is error free for processing
http_error(jsonMarsWeather)
```

```
## [1] FALSE
```

```
# check for our request
status_code(jsonMarsWeather)
```

```
## [1] 200
```

```
# glimpse(jsonMarsWeather)
```

```
# Shows raw data which is not structured and readable, if `as` is not specified, content does its best to guess which output is most appropriate.
jsonMarsWeatherText <- content(jsonMarsWeather, as = "text")
# print(jsonMarsWeatherText)

# Convert JSON reponse which is in text format to data frame using jsonlite package
MarsWeatherList <- fromJSON(jsonMarsWeatherText)

# check the structure
#glimpse(MarsWeatherList)
# remove useless lists
MarsWeatherList02 <- MarsWeatherList[-c(8,9)]
```

- Tidy it !

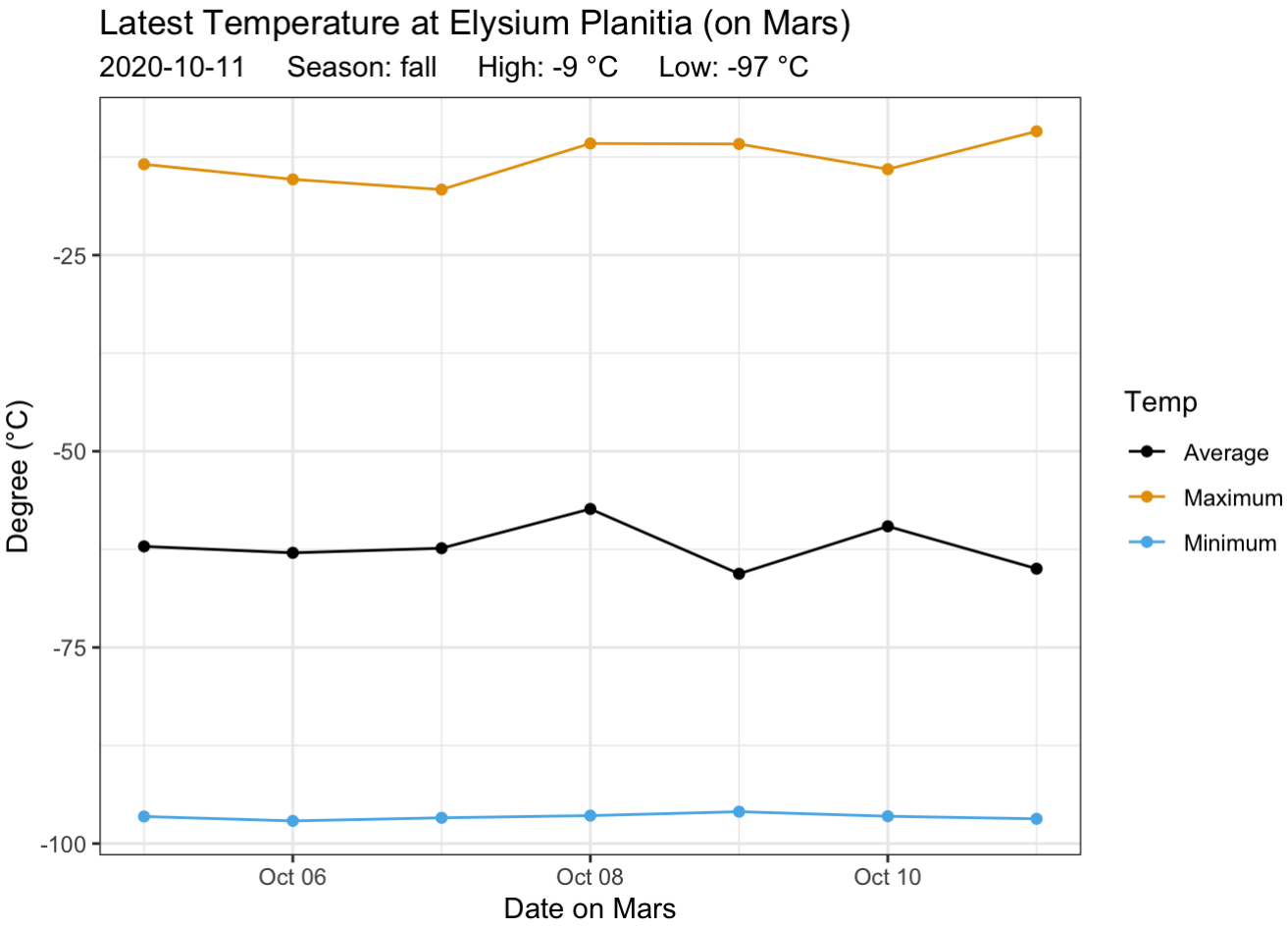
```
MarsWeather <- tibble(MarsWeather = MarsWeatherList02) # save as data frame
MarsWeather %>%
  unnest_wider(MarsWeather) %>%
  hoist(.col = AT, Average = "av", Minimum = "mn", Maximum = "mx") -> MarsWeather02
```

- Clean it !

```
MarsWeather02 %>%
  separate(First_UTC, c("MarsDate", "MarsTime"), sep = "T") %>%
  select(Average:Maximum, MarsDate, Season) %>%
  mutate(MarsDate = ymd(MarsDate)) %>%
  pivot_longer(c(Average:Maximum), names_to = "Temp", values_to = "values") ->
  MarsWeather03 # lubridate
```

- Hey! Show the latest temperature for one week on Mars.
- This graph is taking daily temperature measurements (avg, max, min) on the surface of Mars at Elysium Planitia, a flat, smooth plain near Mars' equator.

```
# create a graph
ggplot(data = MarsWeather03, aes(x = MarsDate, y = values, color = Temp)) +
  geom_line() +
  geom_point() +
  theme_bw() +
  scale_color_colorblind() +
  labs(title = "Latest Temperature at Elysium Planitia (on Mars) ",
       x = "Date on Mars", y = "Degree (°C)",
       subtitle = paste(tail(MarsWeather03$MarsDate, 1),"   ",
                        "Season:", tail(MarsWeather03$Season, 1),"   ",
                        "High:", round(tail(MarsWeather03$values, 1)),"°C", "   ",
                         "Low:", round(MarsWeather03$values[20]) ,"°C"))
```



- In this graph, we could see that is fall season now (Oct., 2020). Although the measuring place is approximately near to Mars' equator, the average temperature is lower than the negative 50 degrees. Also, Mars has a greater difference between night and day temperatures than Earth.

2 IMDB List of Oscar Winners

IMDB has a list of the Oscar Best Picture Winners (https://www.imdb.com/search/title/?count=100&groups=oscar_best_picture_winners&sort=year%2Cdesc&ref_=nv_ch_osc).

Scrape the following elements, convert the data into a tibble, tidy it, and clean it to answer the questions below: - Number - Title - Year - MPAA Rating - Length in minutes - Genre - Star Rating - Metascore Rating - Gross Receipts

Convert the data into a tibble, tidy it, and clean it to answer the following questions:

```
# save an HTML file
html_obj <- read_html("https://www.imdb.com/search/title/?count=100&groups=oscar_best_pi
cture_winners&sort=year%2Cdesc&ref_=nv_ch_osc")

# insert the found selectors as the value for the `css` argument
# using SelectorGadget with google extension
OscarPicsHtml <- html_nodes(html_obj,
                            css = ".ghost~ .text-muted+ span , .ratings-metascore , .
ratings-imdb-rating strong , .genre , .certificate , .runtime , .unbold , .lister-item-h
eader a")
OscarPicsText <- html_text(OscarPicsHtml)
head(OscarPicsText)
```

```
## [1] "1."
## [2] "Parasite"
## [3] "(2019)"
## [4] "R"
## [5] "132 min"
## [6] "\nComedy, Drama, Thriller           "
```

- Time to tidy the data

```
# create Number
tibble(text = OscarPicsText) %>%
  mutate(isPicsNumber = str_detect(text, "^\\d+\\.$")) ->
  OscarPics
# make sure there have 93 ranks corresponding to the website
# sum(rankingOscarPics$isPicsRank)

# create each segment
OscarPics %>%
  mutate(movieNum = cumsum(isPicsNumber)) %>%
  filter(movieNum > 0) ->
  OscarPics
```

- Make sure the number and titles first, so we recheck each pic again.

```
NumsTitles <- html_nodes(html_obj,
                              css = ".lister-item-header a , .text-primary")
NumsTitlesText <- html_text(NumsTitles)
# check the downloaded data
# head(NumsTitlesText)

tibble(text = NumsTitlesText) %>%
  mutate(rownum = row_number(),
         iseven = rownum %% 2 == 0,
         movie = rep(1:93, each = 2)) %>%
  select(-rownum) %>%
  pivot_wider(names_from = iseven, values_from = text) %>%
  select(-movie, "Rank" = "FALSE", pics = "TRUE") %>%
  mutate(Rank = parse_number(Rank)) ->
  picsNums
```

```
# create Titles (name of pics) that the pic names are corresponding to picsRank
OscarPics %>%
  mutate(isTitle = text %in% picsNums$pics) ->
  OscarPics

# create Years
OscarPics %>%
  mutate(isYear = str_detect(text, "\\(\\d+\\)")) ->
  OscarPics

# create Length in minutes
OscarPics %>%
  mutate(isMins = str_detect(text, "^\\d{1,3}+\\s+\\D{1,3}$")) ->
  OscarPics

# create genres
OscarPics %>%
  mutate(isGenre = str_detect(text, "^\\n+\\D")) ->
  OscarPics

# create Star Ratings
OscarPics %>%
  mutate(isStar = str_detect(text, "^\\d+\\.+\\d$")) ->
  OscarPics

# create Metascore Ratings
OscarPics %>%
  mutate(isMeta = str_detect(text, "^\\n+\\d")) ->
  OscarPics

# create Gross Receipts
OscarPics %>%
  mutate(isGross = str_detect(text, "^\\$")) ->
  OscarPics

# create MPAA Ratings- we keep it to the final step, be it's the most complicated
# OscarPics %>%
  # group_by(isPicsNumber, isTitle, isYear, isMins, isGenre, isStar, isMeta, isGross) %
>%
  # count()
OscarPics %>% # process of elimination
  mutate(isMPAA = !isPicsNumber & !isTitle & !isYear & !isMins & !isGenre & !isStar & !i
sMeta & !isGross) -> OscarPics
```

- case_when

```
OscarPics %>%
  mutate(key = case_when(isPicsNumber ~ "number",
                         isTitle ~ "title",
                         isYear ~ "year",
                         isMPAA ~"MPAA",
                         isMins ~ "minutes",
                         isGenre ~ "genre",
                         isStar ~ "starRating",
                         isMeta ~ "Metascore",
                         isGross ~ "gross")) %>%
  select(key, text, movieNum) %>%
  pivot_wider(names_from = key,
              values_from = text) -> OscarPics02
# head(OscarPics02)
```

- clean the data

```
# remove movieNum and set the suitable type for each variable
OscarPics02 %>%
  mutate(number = parse_number(number),
         year = parse_number(year),
         minutes = parse_number(minutes),
         genre = str_replace_all(genre, "\\n", ""),
         genre = str_squish(genre),
         starRating = parse_number(starRating),
         Metascore = str_replace_all(Metascore, "\\n", ""),
         Metascore = str_extract(Metascore, "\\d{0,3}"),
         Metascore = parse_number(Metascore),
         gross = parse_number(gross),  # Million
         movieNum = NULL) -> OscarPics03
```

1. Which two elements are missing the most from the movies?

- Metascore Rating (only have 76) and Gross Receipts (only have 83).

```
OscarPics03 %>%
  summarise_all(funs(sum(!is.na(.))))
```

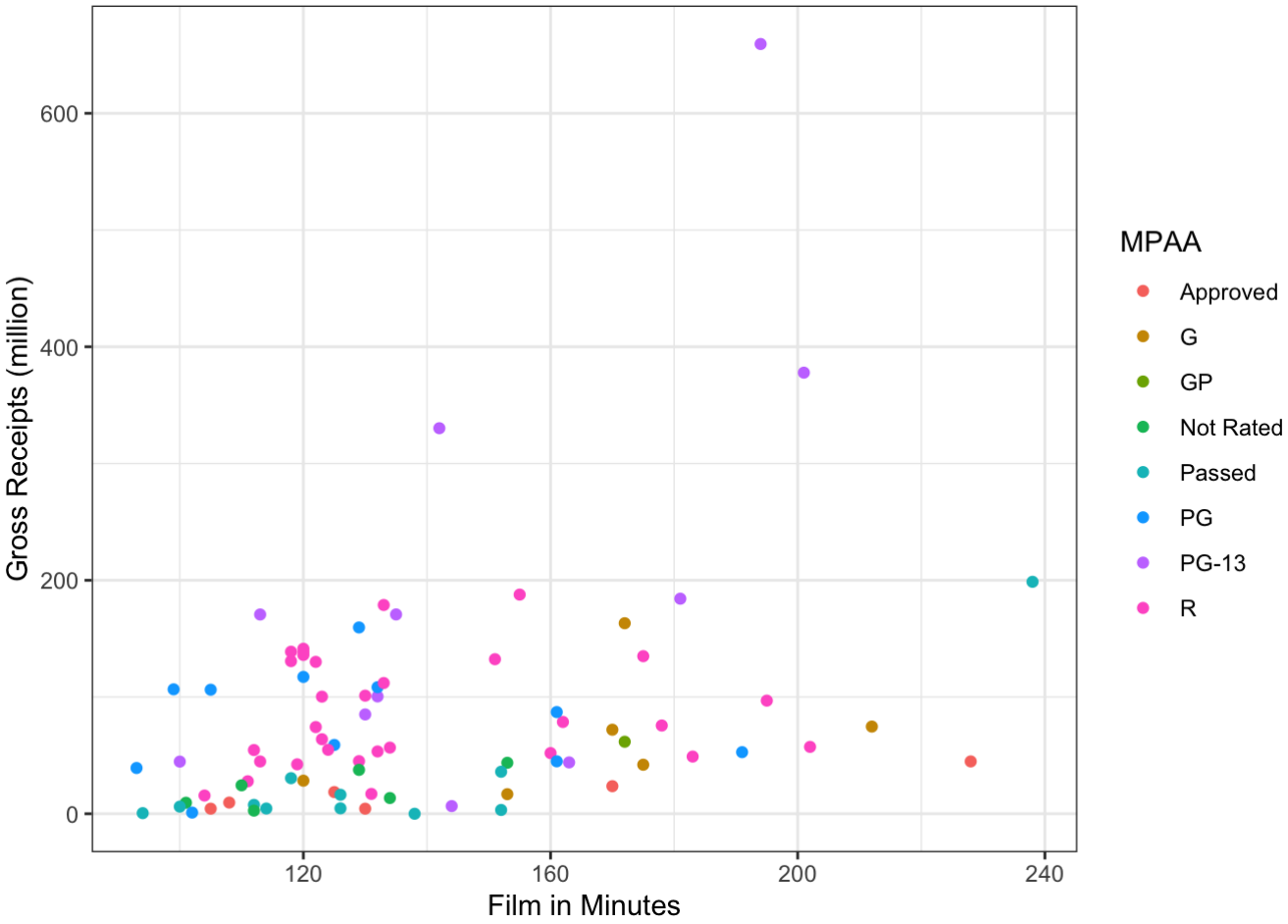
```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## # A tibble: 1 x 9
##   number title  year  MPAA minutes genre starRating Metascore gross
##    <int> <int> <int> <int>   <int> <int>      <int>     <int> <int>
## 1     93    93    93    93      93    93         93        76    83
```

2. Create a plot of the length of a film and its gross, color coded by rating.

- Does MPAA rating matter?

```
OscarPics03 %>%
  filter(!is.na(gross)) -> OscarPicsNoGrossNAs # remove the NAs in gross
  ggplot(data = OscarPicsNoGrossNAs, mapping = aes(x = minutes, y = gross, color = MPA
A)) +
  geom_point() +
  theme_bw() +
  labs(x = "Film in Minutes",
       y = "Gross Receipts (million)")
```

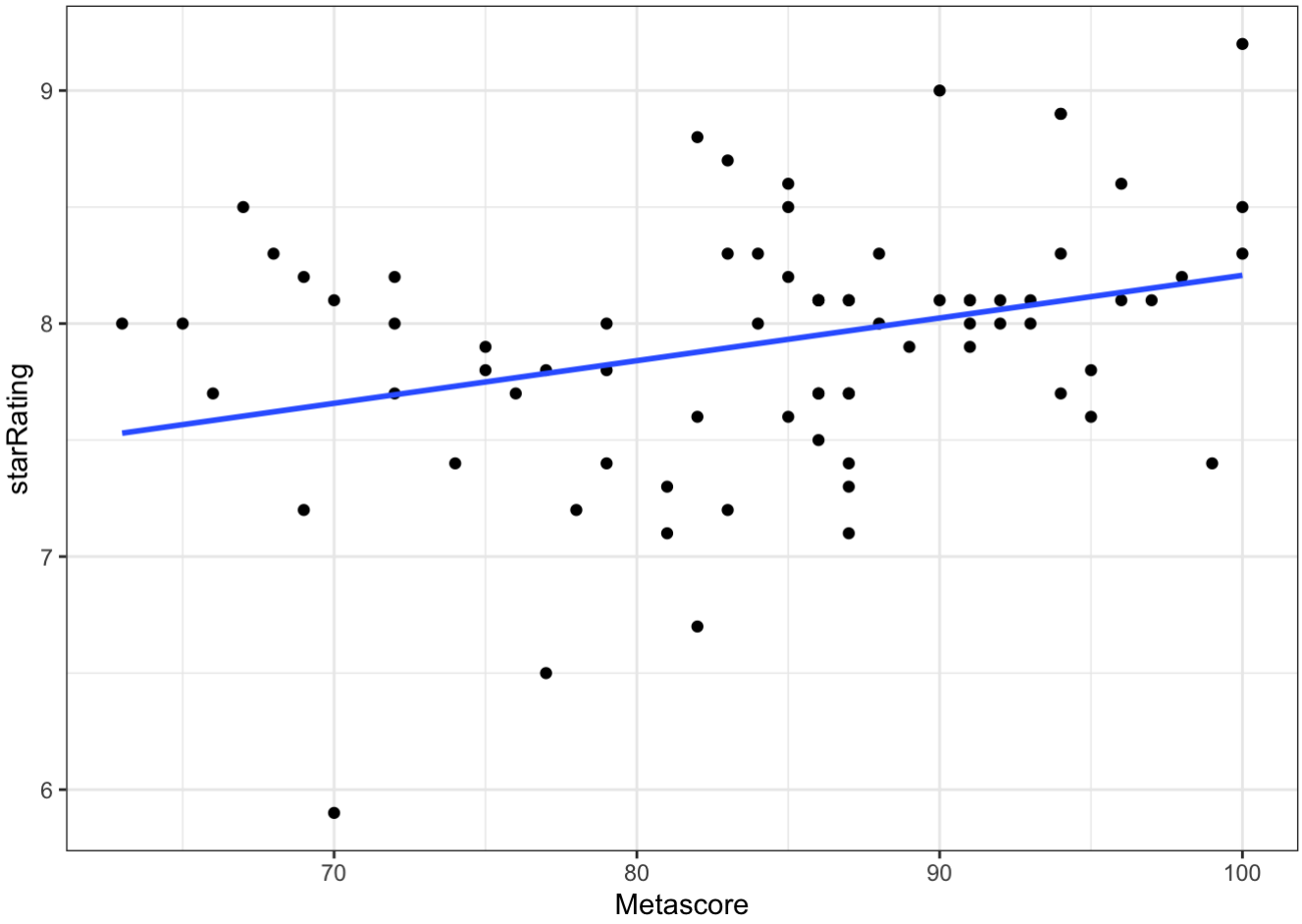


- The MPAA does not matter, because the rating of the movie does not determine the length of the firm.

3. Create a plot with a single Ordinary Least Squares smoothing line with no standard errors showing for predicting stars rating based on metacritic scores.

```
OscarPics03 %>%
  filter(!is.na(Metascore)) -> OscarPicsNoMetaNAs # remove the NAs in Metascore
  ggplot(data = OscarPicsNoMetaNAs, aes(x = Metascore, y = starRating)) +  # predict sta
rs rating as Y-axis
  geom_point()+
  geom_smooth(method = lm, se = FALSE) +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



- Is there a meaningful relationship in terms of the p-value and adjusted R-Squared?

```
# lm (y ~ x)
# x = predictive variable = independent variable = 自变量
# y = response variable = dependent variable = 因变量
scoreSLR <- lm(data = OscarPicsNoMetaNAs, starRating ~ Metascore)
summary(scoreSLR)
```

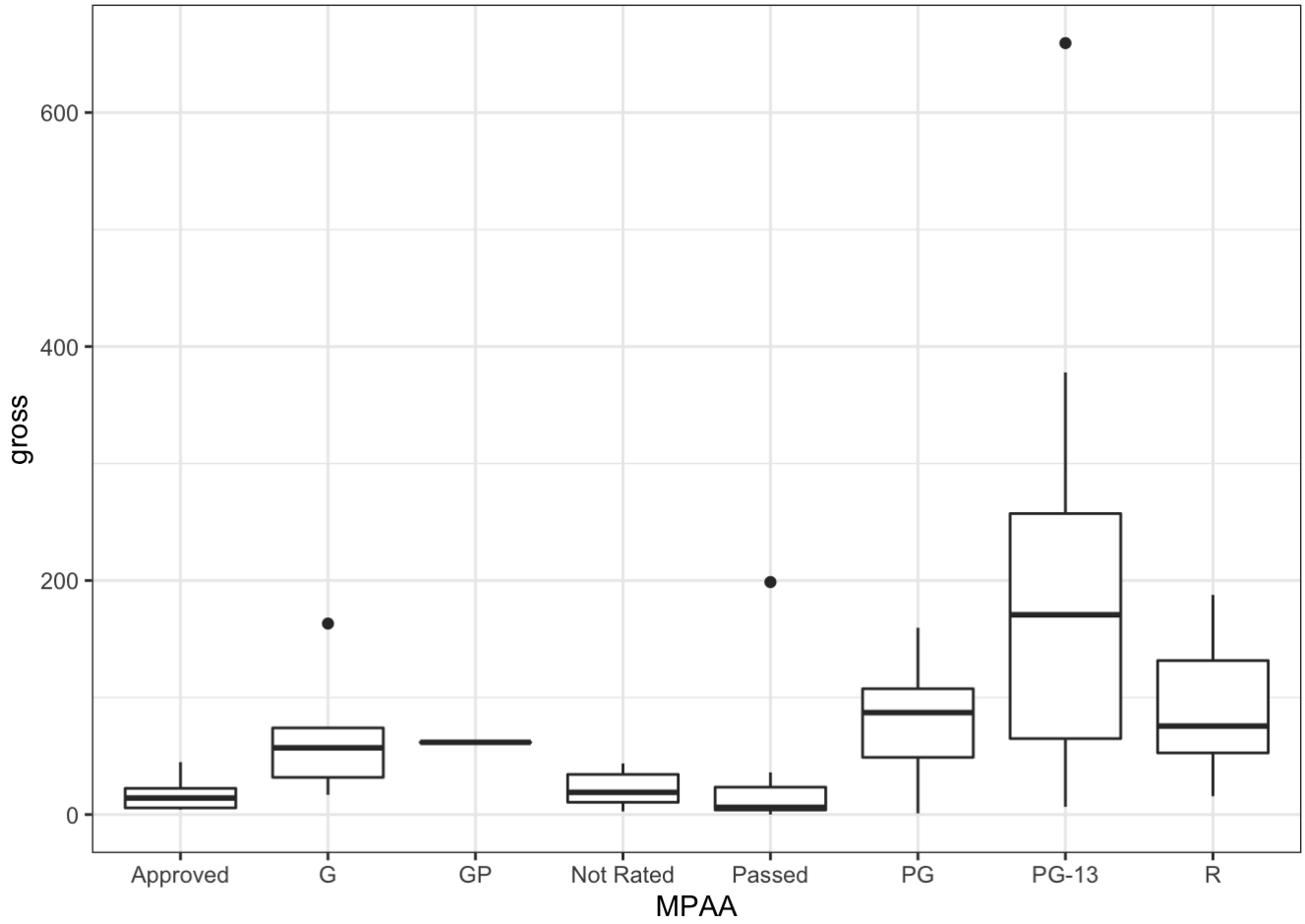
```
##
## Call:
## lm(formula = starRating ~ Metascore, data = OscarPicsNoMetaNAs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75784 -0.28695  0.03461  0.30735  0.99310
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.376689   0.558256  11.423  < 2e-16 ***
## Metascore   0.018302   0.006576   2.783  0.00683 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.53 on 74 degrees of freedom
## Multiple R-squared:  0.09474,    Adjusted R-squared:  0.08251
## F-statistic: 7.745 on 1 and 74 DF,  p-value: 0.006832
```

- H0: $\beta 1 = 0$, H1: $\beta 1 \neq 0$
- Based on the p-value: 0.006832(< 0.05), we have evidence to reject the H0 hypothesis. In this case, there is a possible linear relationship between star rating and metacritic scores. However, the low adjusted R-squared 0.08251 shows that this model does not explain much of variation of the data meaning that the linear model may not be the best model for this dataset. Because only 8.25% of variance is explained by this regression model.

4. Use an appropriate plot to compare the gross receipts by MPAA rating.

- Which MPAA rating has the highest median gross receipts?
- The below boxplot shows the PG-13 has the highest median gross receipts.

```
ggplot(data = OscarPicsNoGrossNAs, aes(x = MPAA, y = gross)) +
geom_boxplot() +
theme_bw()
```



- Which are the R-rated movies in the overall top 10 of gross receipts?
- Gladiator and Rain Man.

```
OscarPicsNoGrossNAs %>%
 arrange(desc(gross)) %>%
 select(title, year, MPAA, gross) %>%
 slice(1:10)
```

```
## # A tibble: 10 x 4
##    title                                          year MPAA   gross
##    <chr>                                         <dbl> <chr>  <dbl>
##  1 Titanic                                        1997 PG-13   659.
##  2 The Lord of the Rings: The Return of the King  2003 PG-13   378.
##  3 Forrest Gump                                   1994 PG-13   330.
##  4 Gone with the Wind                             1939 Passed  199.
##  5 Gladiator                                      2000 R       188.
##  6 Dances with Wolves                             1990 PG-13   184.
##  7 Rain Man                                       1988 R       179.
##  8 A Beautiful Mind                               2001 PG-13   171.
##  9 Chicago                                        2002 PG-13   171.
## 10 The Sound of Music                             1965 G       163.
```

- Extra Credit (1 pts): Use one-way analysis of variance to assess the level of evidence for whether all ratings have the same mean gross receipts. Provide your interpretation of the results.

```
allDiff <- aov(gross ~ MPAA, data = OscarPicsNoGrossNAs) # value ~ group
summary(allDiff)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## MPAA         7 228706   32672   4.835 0.000156 ***
## Residuals   75 506820    6758
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Multiple pairwise-comparison between the means of groups
    - In one-way ANOVA test, a significant p-value (less than 0.05) indicates that some of the group means are different, but we don't know which pairs of groups are different.
    - It's possible to perform multiple pairwise-comparison, to determine if the mean difference between specific pairs of group are statistically significant.
- H0 : all group means are equal.
- H1: at least one group has a different mean from another group.
- We have a evidence to reject H0 since p-value = 0.000156 < 0.05 which means that at least one rating has a different mean from another rating.

3 Extra Credit 1 Pts

- Listen to the AI Today podcast on Machine Learning Ops (https://podcasts.apple.com/us/podcast/ai-today-podcast-artificial-intelligence-insights-experts/id1279927057?i=1000468771571) and provide your thoughts on the following questions:

1. Does knowing about Git and GitHub help you in understanding the podcast?

- Git and GitHub are good tool for software and data engineers. They could easier manage and track the ongoing project status with their co-workers, especially Git and GitHub are convenient for remote collaboration (such as work from home since COVID-19). Moreover, we learned devtools package on week 5. The aim of devtools is to make package development easier by providing R functions. Marsden has mentioned devtools in the podcast, data engineers may not have programming backgrounds, thus devtools is good for data engineers to develop new packages.

2. How do you think the ideas of ML OPs will affect your future data science projects? You may also want to check out this article on Towards Data Science (https://towardsdatascience.com/ml-ops-machine-learning-as-an-engineering-discipline-b86ca4874a3f).

- As the code and data evolve independently, and typical Data scientists are not often as trained engineers, so they do not always follow good DevOps practices. "MLOps is a new discipline for collaboration and communication between data scientists and information technology (IT) professionals while automating and productizing machine learning algorithms" (Talagala, 2018). ML Ops is like a bridge that connects ML engineers and Data engineers having a smooth workflow, avoiding information asymmetry, and maintaining ML systems in production reliably and efficiently.