# CS3072 - Final Project

IMDB Data Analysis

Hala Haneya

12/1/2021

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

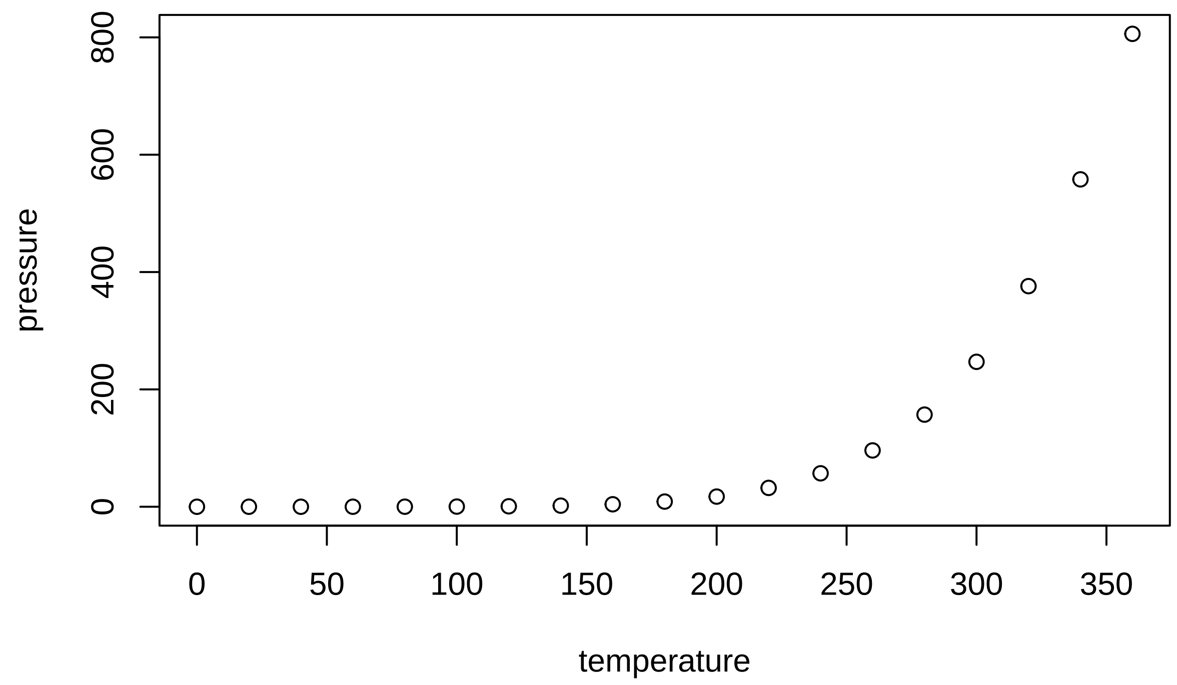
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

## Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

## Libraries

```
library(data.table)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

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

```
library(lubridate)
```

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

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

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

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.1.2
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.1.2
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(knitr)
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 4.1.2
```

```
library(RANN)
```

```
## Warning: package 'RANN' was built under R version 4.1.2
```

## Import Datasets

The data was imported using import dataset function in R Studio because we noticed it was much faster than using import as a code.

We will rename the datasets and remove the old ones.

```
movie_gross <- read.csv("zippedData/bom.movie_gross.csv.gz")
title_akas <- read.csv("zippedData/imdb.title.akas.csv.gz")
title_basics <- read.csv("zippedData/imdb.title.basics.csv.gz")
# title_crew <- read.csv("zippedData/imdb.title.crew.csv.gz")
# title_principals <- read.csv("zippedData/imdb.title.principals.csv.gz")
title_ratings <- read.csv("zippedData/imdb.title.ratings.csv.gz")
# movie_info <- read.csv("zippedData/rt.movie_info.tsv.gz")
movies_budget <- read.csv("zippedData/tn.movie_budgets.csv.gz")
movies <- read.csv("zippedData/tmdb.movies.csv.gz")
```

```
# rm(bom.movie_gross.csv)
# rm(imdb.title.akas.csv)
# rm(imdb.title.basics.csv)
# rm(imdb.title.crew.csv)
# rm(imdb.title.principals.csv)
# rm(imdb.title.ratings.csv)
# rm(rt.movie_info.tsv)
# rm(tn.movie_budgets.csv)
# rm(tmdb.movies.csv)
```

## Recommendations and Questions

### Question 1: Which studios has the highest profit margin?

We are looking to find the studios whose average profit margin is highest and we will first start by exploring the dataset.

```
dim(movie_gross)
```

```
## [1] 3387    5
```

```
summary(movie_gross)
```

```
##     title              studio          domestic_gross      foreign_gross     
##  Length:3387        Length:3387        Min.   :      100   Length:3387       
##  Class :character   Class :character   1st Qu.:   120000   Class :character  
##  Mode  :character   Mode  :character   Median :  1400000   Mode  :character  
##                                        Mean   : 28745845                     
##                                        3rd Qu.: 27900000                     
##                                        Max.   :936700000                     
##                                        NA's   :28                            
##       year     
##  Min.   :2010  
##  1st Qu.:2012  
##  Median :2014  
##  Mean   :2014  
##  3rd Qu.:2016  
##  Max.   :2018  
## 
```

```
group_by_studio <- movie_gross %>%
  group_by(studio) %>%
  summarise(number_titles = n_distinct(title))
group_by_studio
```

```
## # A tibble: 258 x 2
##    studio   number_titles
##    <chr>            <int>
##  1 ""                   5
##  2 "3D"                 1
##  3 "A23"                2
##  4 "A24"               49
##  5 "AaF"                1
##  6 "Abk."               1
##  7 "Abr."              10
##  8 "ADC"                2
##  9 "AF"                 6
## 10 "Affirm"             2
## # ... with 248 more rows
```

Here we select only those studios that have created more than 3 titles, since we have many of studios in our dataset.

```
studio_df <- group_by_studio %>%
  filter(number_titles > 3, !is.na(studio), studio != '')
studio_df
```

```
## # A tibble: 101 x 2
##    studio    number_titles
##    <chr>             <int>
##  1 A24                  49
##  2 Abr.                 10
##  3 AF                    6
##  4 Alc                   5
##  5 Amazon                7
##  6 Ampl.                 5
##  7 Anch.                18
##  8 Annapurna             6
##  9 ATO                   4
## 10 BG                   16
## # ... with 91 more rows
```

Removing redundant df to save memory.

```
rm(group_by_studio)
```

We only want the studios in grouped_by_studio, so we will save the studios in a vector and then filter the data in movie gross according to the required studios.

```
studios <- studio_df[['studio']]
```

```
movies_studios_budget <- movie_gross %>%
  filter(studio %in% studios)
```

Now, we will join the budgets table with the movies studios budget.

```
studios_plus_budget <-
  movies_budget %>%
  right_join(movies_studios_budget, by = c('movie' = 'title')) %>%
  filter(!is.na("production_budget"), !is.na("worldwide_gross")) %>%
  subset(select = -c(domestic_gross.x,domestic_gross.y,foreign_gross)) %>%
  drop_na("production_budget") %>%
  drop_na("worldwide_gross")
```

We remove commas and dollar signs from budget related columns and turning them numerical.

```
studios_plus_budget$production_budget =
  as.numeric(gsub("[\\$,]", "", studios_plus_budget$production_budget))
studios_plus_budget$worldwide_gross =
  as.numeric(gsub("[\\$,]", "", studios_plus_budget$worldwide_gross))
```

We calculate the profit and profit margin as follows.

```
studios_plus_budget <- studios_plus_budget %>%
  mutate(profit = worldwide_gross - production_budget) %>%
  mutate(profit_margin = profit / worldwide_gross)
```

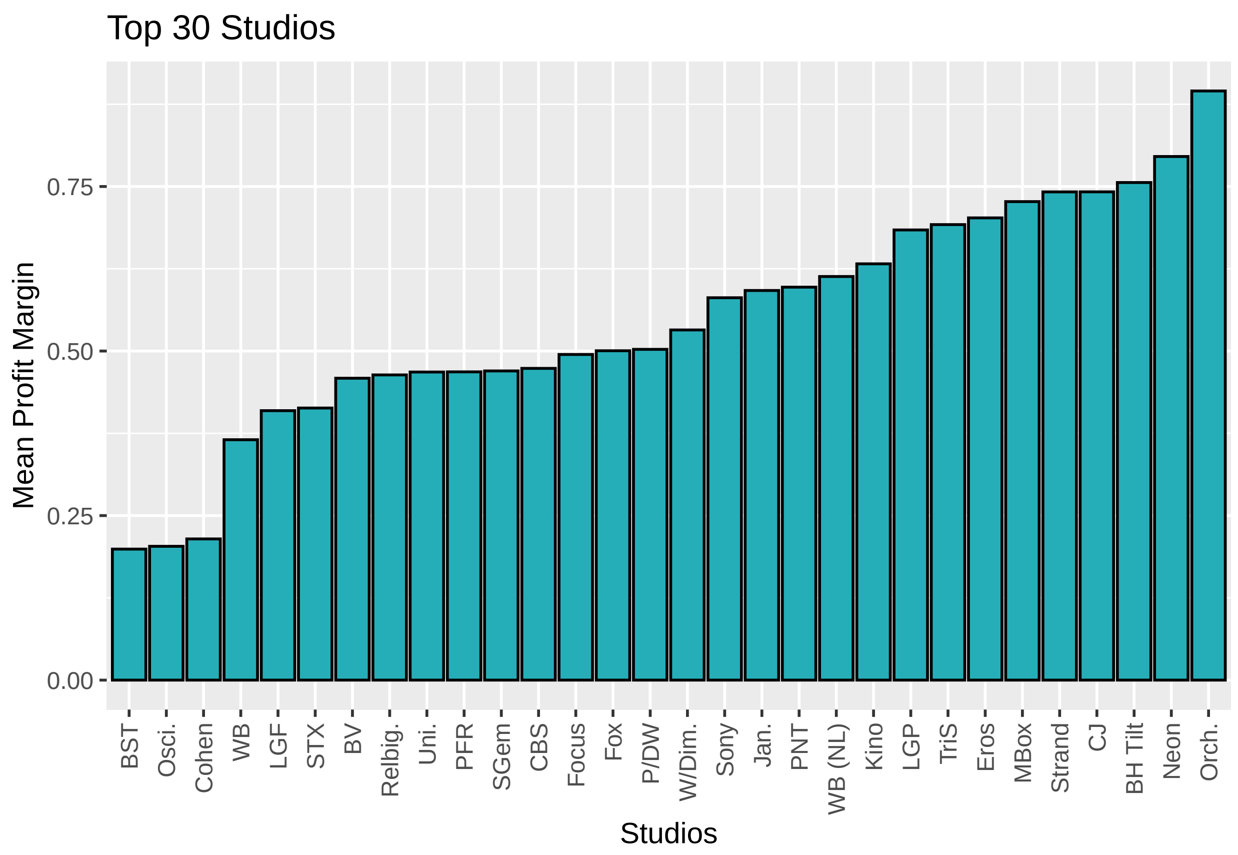
We find the mean profit margin for each studio.

```r
profit_by_studio <- studios_plus_budget %>%
  group_by(studio) %>%
  summarise(mean_profit_margin = mean(profit_margin))
```

These are the top 30 studios that are recommended to model best practices against.

```r
profit_by_studio %>%
  arrange(desc(mean_profit_margin)) %>%
  slice(1:30) %>%
  ggplot(mapping = aes(x = reorder(studio, mean_profit_margin), y = mean_profit_margin)) +
  geom_bar(stat = "identity", las = 2, fill = "#26aeb8", color = "black") +
  labs(title = "Top 30 Studios", x = "Studios", y = "Mean Profit Margin") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```

```
## Warning: Ignoring unknown parameters: las
```



Question 1 Conclusion: According to the mean profit margin, The Orchard (Orch.) has the highest profit margin.

## Question 2: What are the most profitable movies and how much should you spend?

To answer this question and provide a recommendation we'll make use of a budgets dataframe called imdb_budgets. Our analysis will require that we use the data to calculate profit and profit margin.

We remove commas and dollar signs from budget related columns and turning them numerical.

```
movies_budget$production_budget =
  as.numeric(gsub("[\\$,]", "", movies_budget$production_budget))
movies_budget$domestic_gross =
  as.numeric(gsub("[\\$,]", "", movies_budget$domestic_gross))
movies_budget$worldwide_gross =
  as.numeric(gsub("[\\$,]", "", movies_budget$worldwide_gross))
```

We remove the values less than zero and NAs.

```
movies_budget <-
  movies_budget %>%
  filter(production_budget > 0, worldwide_gross > 0,
         !is.na(production_budget), !is.na(worldwide_gross))
```

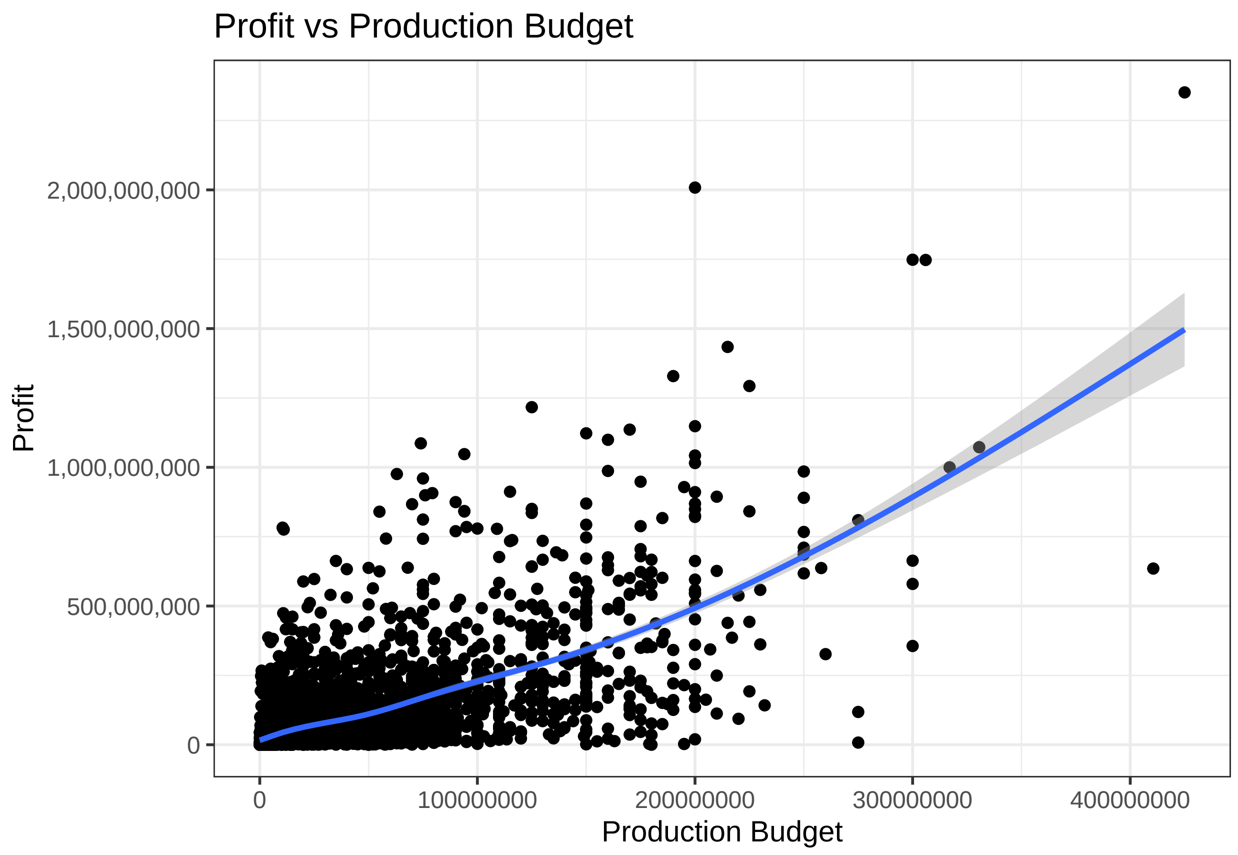
We calculate profit and profit margin as follows.

```
imdb_budgets <- movies_budget %>%
  mutate(profit = (worldwide_gross - production_budget)) %>%
  mutate(profit_margin = (worldwide_gross - production_budget)/worldwide_gross) %>%
  filter(profit_margin > 0, profit > 0)
```

We examine the overall trend of budget versus profit to see if there's any correlation.

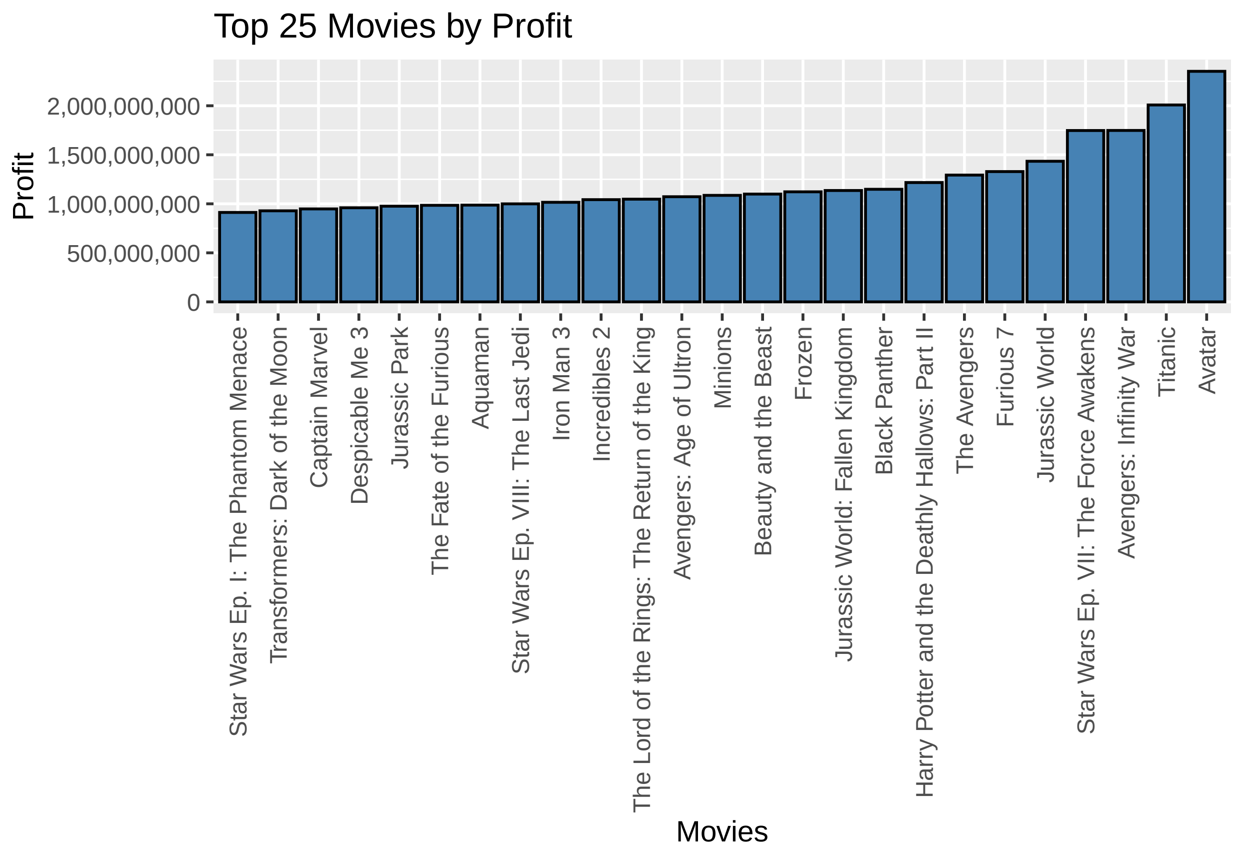
```
options(scipen=5)

imdb_budgets %>%
  ggplot(mapping = aes(x = production_budget, y = profit)) +
  geom_point() +
  geom_smooth() +
  labs(title = "Profit vs Production Budget", x = "Production Budget", y = "Profit") +
  scale_y_continuous(labels = comma) +
  theme_bw()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



We also take a look at the top 25 movies in terms of profit to understand their financial success and how closely we should attempt to emulate their budget.

```
options(scipen=3)
imdb_budgets %>%
  arrange(desc(profit)) %>%
  slice(1:25) %>%
  ggplot(mapping = aes(x = reorder(movie, profit), y = profit)) +
  geom_bar(position = "dodge", stat = "identity",
           fill = "steelblue", color = "black") +
  labs(title = "Top 25 Movies by Profit", x = "Movies", y = "Profit") +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```



Question 2 Conclusion: The most profitable movie is Avatar with a profit of about 3 billion us dollars. It is recommended to attempt to follow the budgets of the most profitable movies.

### Question 3: Which movie genres are most commonly produced and does quantity equate to higher net profits?

We join title_akas, title_basics and title_ratings as one large imdb.

```r
imdb <-
  title_akas %>%
  right_join(title_basics, by = c("title_id" = "tconst")) %>%
  left_join(title_ratings, by = c("title_id" = "tconst"))
```

We separate genres in their own dataframe. In the original dataset, they are comma separated under one variable.

```r
imdb_genres <- imdb %>%
  separate_rows(genres, sep = ",") %>%
  rename(genre = genres) %>%
  filter(!is.na(genre), genre != "\\N",
         !is.null(genre), genre != "Adult", genre != "")
```

We removed the original dataframes to save memory.

```
rm(title_akas)
rm(title_basics)
```

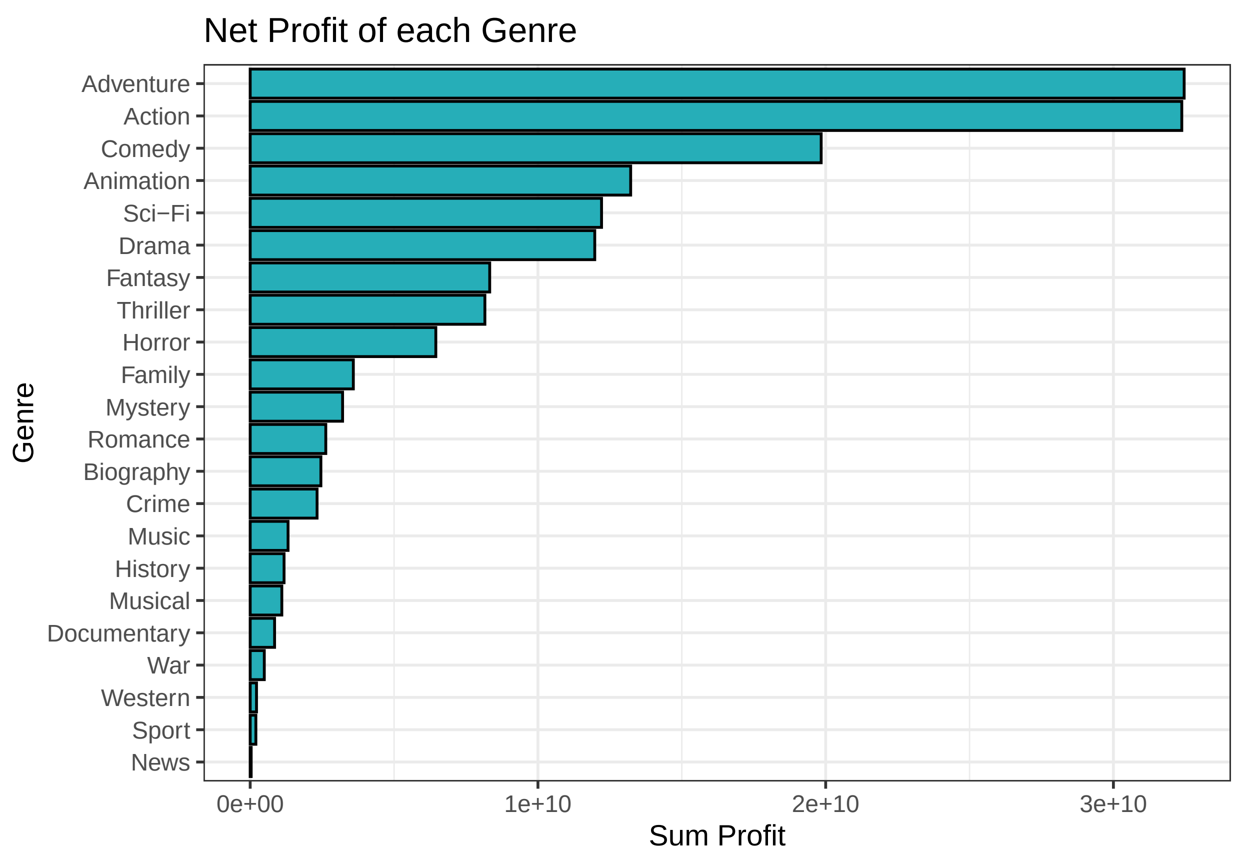
We join imdb budgets with genres to find net profit for each genre.

```
imdb_genres_budget_rating <-
  imdb_budgets %>%
  inner_join(imdb_genres, by = c("movie" = "primary_title")) %>%
  filter(id == unique(id)) %>%
  rename(year = start_year)
```

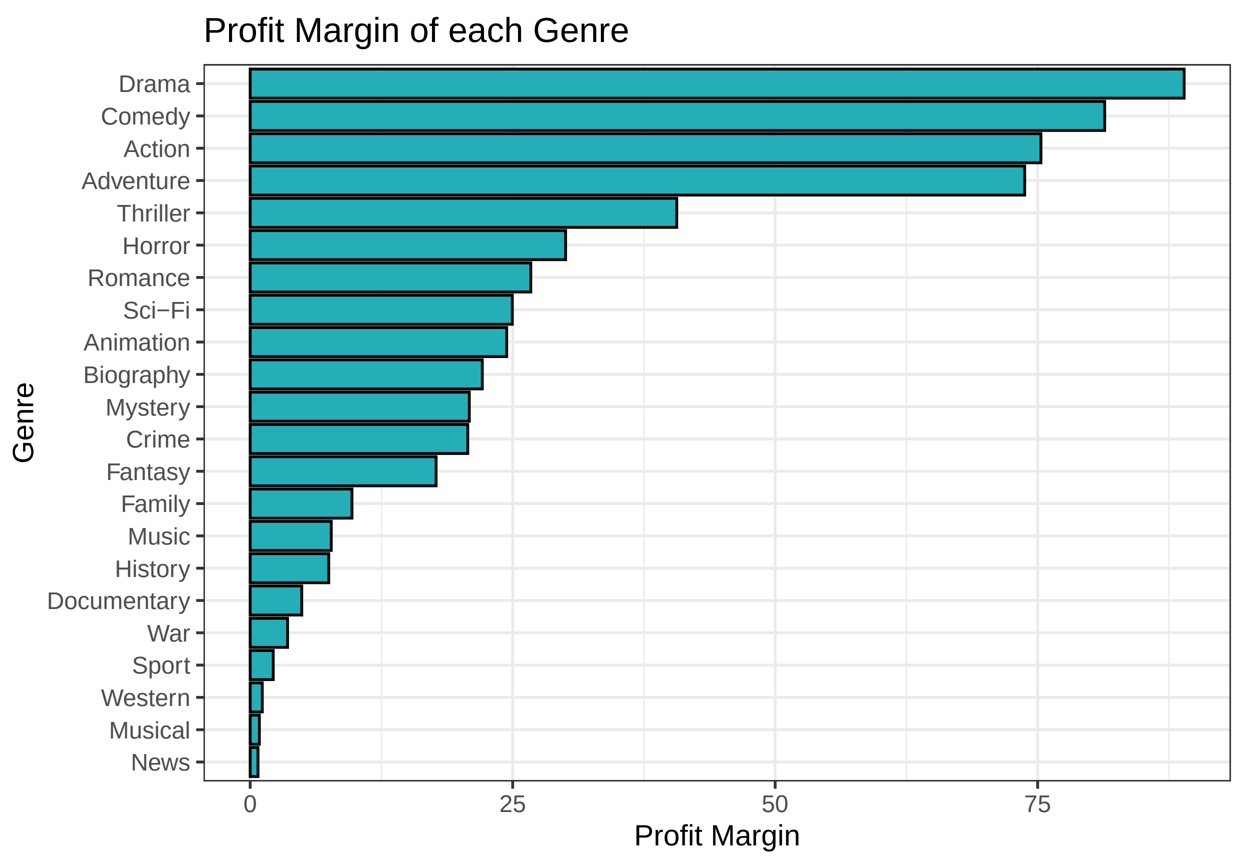
```
## Warning in id == unique(id): longer object length is not a multiple of shorter
## object length
```

```
imdb_genres_budget_rating <- imdb_genres_budget_rating %>%
  select(id,movie, release_date, year, profit, profit_margin,
         region, language, genre, averagerating)
```

```
options(scipen=5)

imdb_genres_budget_rating %>%
  group_by(genre) %>%
  summarise(sum_profit = sum(profit)) %>%
  ggplot(mapping = aes(x = reorder(genre, sum_profit), y = sum_profit)) +
  geom_bar(stat = "identity", fill = "#26aeb8", color = "black") +
  labs(title = "Net Profit of each Genre", x = "Genre", y = "Sum Profit") +
  scale_x_discrete(guide = guide_axis(check.overlap = TRUE)) +
  theme_bw() +
  coord_flip()
```



```
imdb_genres_budget_rating %>%
  group_by(genre) %>%
  summarise(sum_profit_margin = sum(profit_margin)) %>%
  ggplot(mapping = aes(x = reorder(genre, sum_profit_margin), y = sum_profit_margin)) +
  geom_bar(stat = "identity", fill = "#26aeb8", color = "black") +
  labs(title = "Profit Margin of each Genre", x = "Genre", y = "Profit Margin") +
  scale_x_discrete(guide = guide_axis(check.overlap = TRUE)) +
  theme_bw() +
  coord_flip()
```



Question 3 Conclusion: Adventure, Action and Comedy have the highest net profit of all genres. Analysis of profit margin shows that in addition to Adventure, Action and Comedy, Drama and Thriller also have financial success, with Drama the highest.

## Question 4: What is the best time of the year to release a movie?

```
str(imdb_genres_budget_rating)
```

```
## 'data.frame':    919 obs. of  10 variables:
##  $ id           : int  1 2 4 7 9 10 11 12 14 15 ...
##  $ movie        : chr  "Avatar" "Pirates of the Caribbean: On Stranger Tides" "Avengers: Age of Ultr
##  $ release_date : chr  "Dec 18, 2009" "May 20, 2011" "May 1, 2015" "Apr 27, 2018" ...
##  $ year         : int  2011 2011 2015 2018 2017 2015 2012 2018 2012 2010 ...
##  $ profit       : num  2351345279 635063875 1072413963 1748134200 355945209 ...
##  $ profit_margin: num  0.847 0.607 0.764 0.854 0.543 ...
##  $ region       : chr  "JP" "US" "ES" "DE" ...
##  $ language     : chr  "" "" "" "" ...
##  $ genre        : chr  "Horror" "Fantasy" "Adventure" "Action" ...
##  $ averagerating: num  6.1 6.6 7.3 8.5 6.5 6.8 8.4 7 6.6 7.8 ...
```

We will change the release date form chr to date type to get the month.

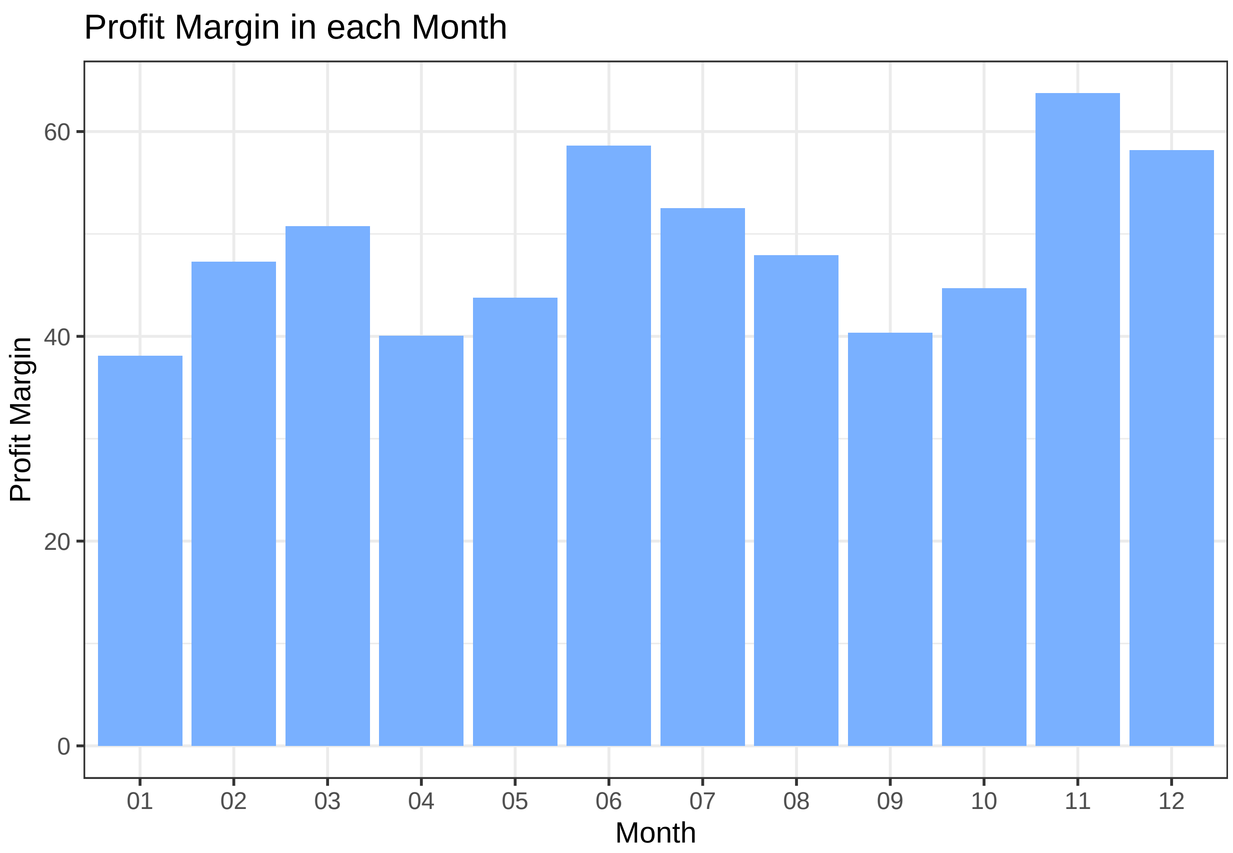
```r
imdb_genres_budget_rating$release_date <- mdy(imdb_genres_budget_rating$release_date)
class(imdb_genres_budget_rating$release_date)
```

```
## [1] "Date"
```

We extract the month number from the date and save it in a new variable.

```r
imdb_genres_budget_rating$release_month <- format(imdb_genres_budget_rating$release_date, "%m")
```

```r
imdb_genres_budget_rating %>%
  ggplot(mapping = aes(x = release_month, y = profit_margin)) +
  labs(title = "Profit Margin in each Month", x = "Month", y = "Profit Margin") +
  geom_bar(stat = "identity", fill = "#79b0ff") +
  theme_bw()
```



Question 4 Conclusion: November and December are months that bring the most profit. We believe this is due to the Christmas and New Year's Eve celebrations and holidays. At the third place is June. It could bring more profit due to it being during the Summer.

## Question 5: Which actors and directors tend to add the most value?

We are going to take a look at the average net profit across all movies. Then, we want to determine which actors and directors consistently appear in movies where the net profit substantially exceeds the average. We will represent this in a field called Value Above Replacement(VAR). To further simplify this concept; if across all movies the average net profit is 100 dollars and the average net profit of movies from 'Actor: X' is 200 dollars he/she would have a VAR of 2. This number represents X times over the average. To eliminate outliers we will look at actors who appear in 10 or more movies and directors who work in 5 or more.

We'll use the actors_df dataframe, adjust for inflation, and calculate profit as we did in Question 1. Then filter by actors who have starred in 10 or more movies.

```
actors <- read_csv('tables/Actors_Table.csv')
```

```
## Rows: 15320 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Movie, value, Release Date
## dbl (4): Year, Production Budget, Domestic Gross, Worldwide Gross

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
directors <- read_csv('tables/Directors_Table.csv')
```

```
## Rows: 4181 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Movie, value, Release Date
## dbl (4): Year, Production Budget, Domestic Gross, Worldwide Gross

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We rename the columns for consistency.

```
actors <- actors %>%
  rename(movie = 'Movie', year = 'Year',release_date = 'Release Date',
         production_budget = 'Production Budget',
         domestic_gross = 'Domestic Gross', worldwide_gross = 'Worldwide Gross')

directors <- directors %>%
  rename(movie = 'Movie', year = 'Year',release_date = 'Release Date',
         production_budget = 'Production Budget',
         domestic_gross = 'Domestic Gross', worldwide_gross = 'Worldwide Gross')
```

We calculate profit and profit margin like above.

```
actors <- actors %>%
  filter(production_budget > 0, worldwide_gross > 0) %>%
  mutate(profit = worldwide_gross - production_budget) %>%
  mutate(profit_margin = profit / worldwide_gross) %>%
  filter(profit > 0, profit_margin > 0)

directors <- directors %>%
  filter(production_budget > 0, worldwide_gross > 0) %>%
  mutate(profit = worldwide_gross - production_budget) %>%
  mutate(profit_margin = profit / worldwide_gross) %>%
  filter(profit > 0, profit_margin > 0)
```

We calculate VAR for actors.

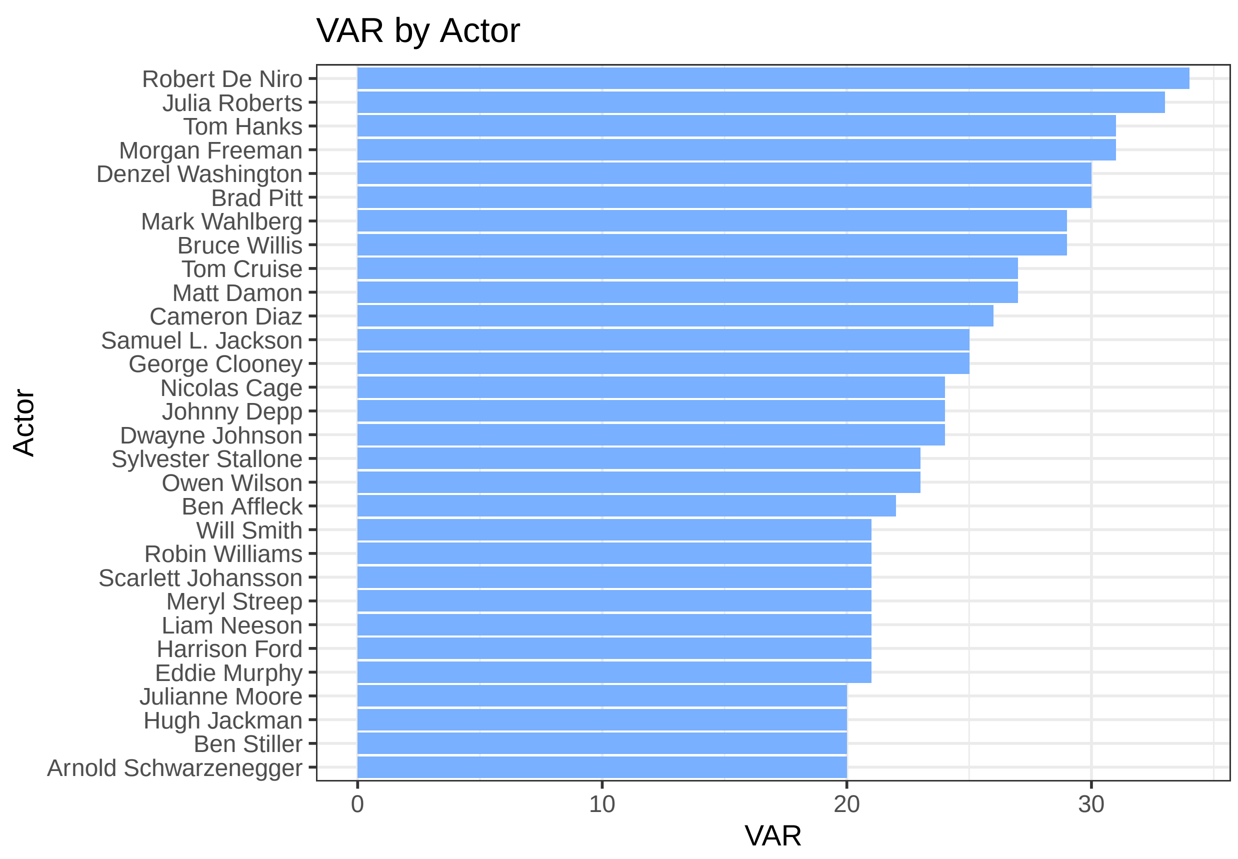
```
actors <- actors %>%
  group_by(value) %>%
  summarise(sum_profit = sum(profit), mean_profit = mean(profit),
            count_movies = n()) %>%
  filter(count_movies > 10) %>%
  mutate(VAR = sum_profit / mean_profit)
```

We calculate VAR for directors.

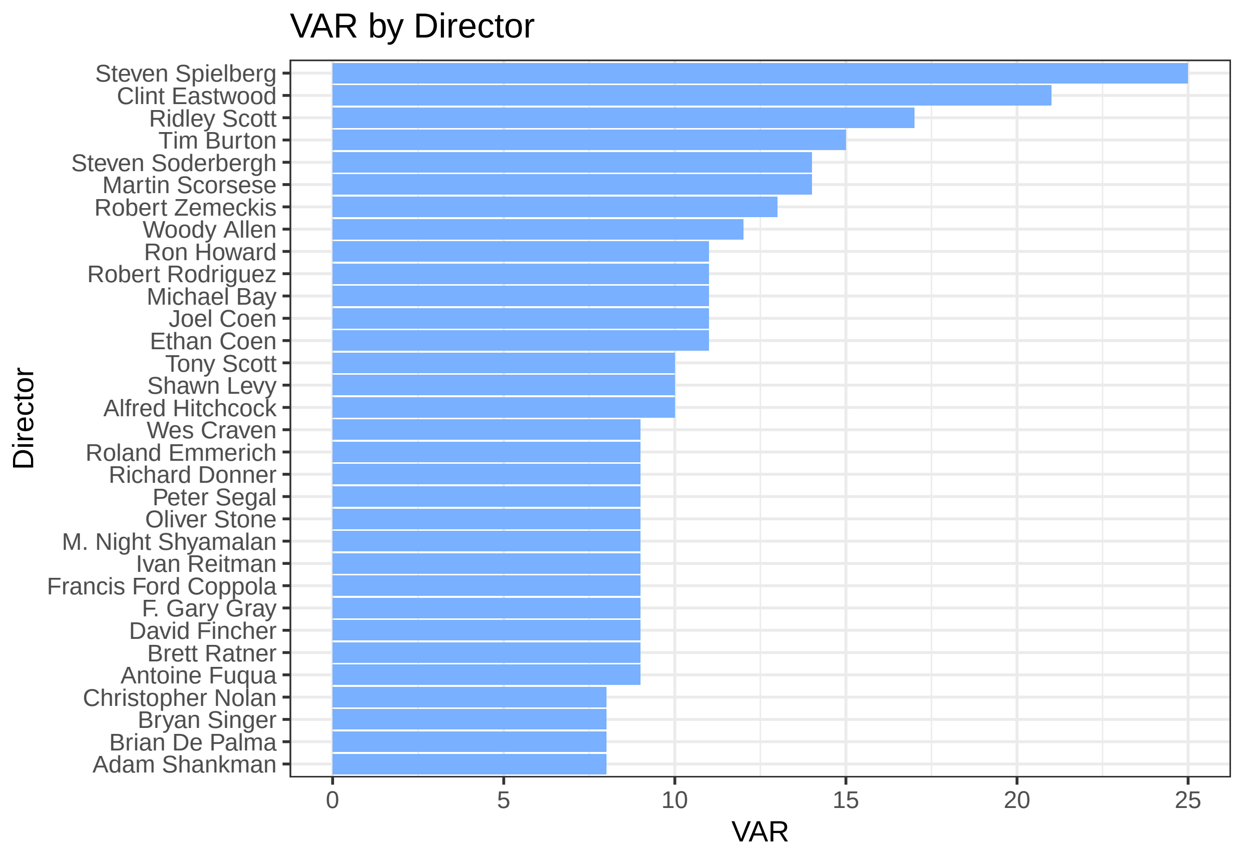
```
directors <- directors %>%
  group_by(value) %>%
  summarise(sum_profit = sum(profit), mean_profit = mean(profit),
            count_movies = n()) %>%
  filter(count_movies > 5) %>%
  mutate(VAR = sum_profit / mean_profit)
```

We visualize the results.

```
actors %>%
  arrange(desc(VAR)) %>%
  slice(1:30) %>%
  ggplot(mapping = aes(x = reorder(value, VAR), y = VAR)) +
  labs(title = "VAR by Actor", x = "Actor", y = "VAR") +
  geom_bar(stat = "identity", fill = "#79b0ff") +
  coord_flip() +
  theme_bw()
```



```r
directors %>%
  arrange(desc(VAR)) %>%
  slice(1:32) %>%
  ggplot(mapping = aes(x = reorder(value, VAR), y = VAR)) +
  labs(title = "VAR by Director", x = "Director", y = "VAR") +
  geom_bar(stat = "identity", fill = "#79b0ff") +
  coord_flip() +
  theme_bw()
```



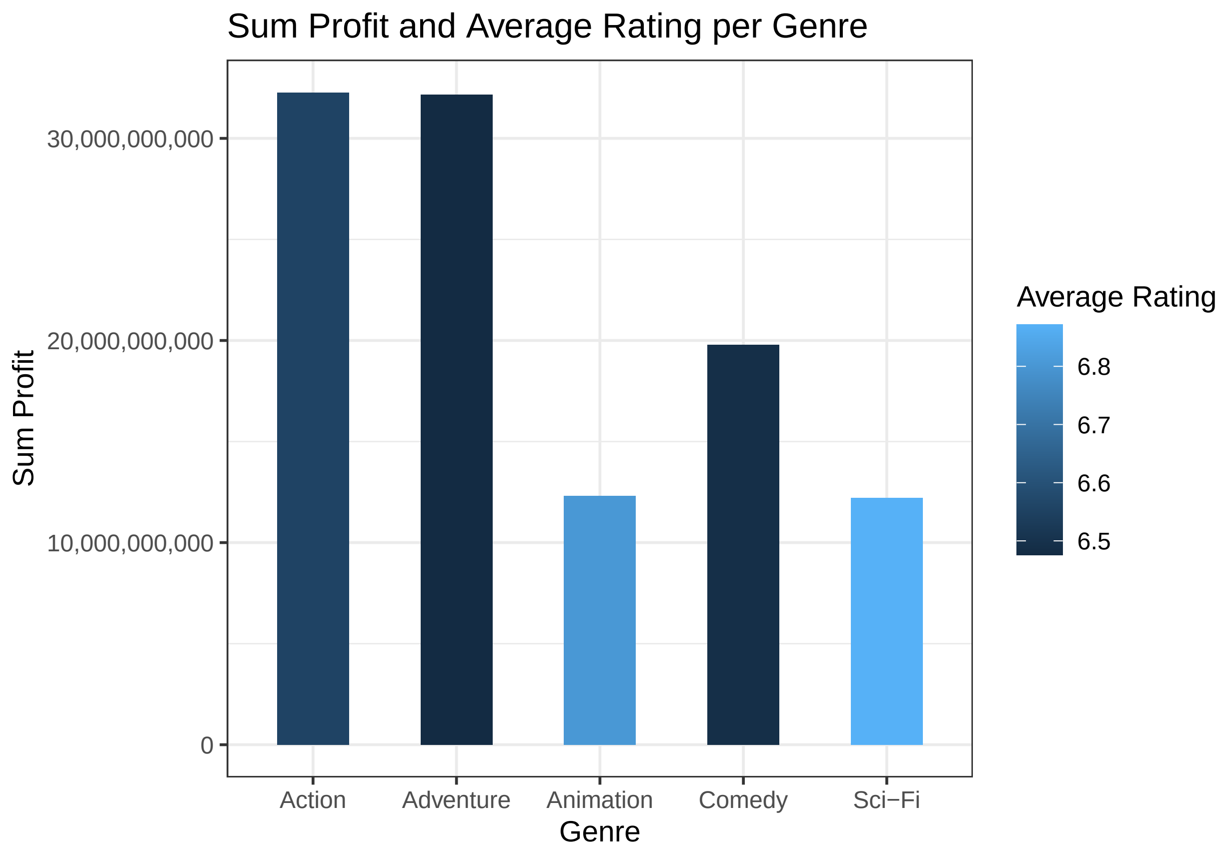
Question 5 Conclusion: According to the above calculation and graphs, actor Robert De Niro is the most valuable actor. Director Steven Spielberg is most valubale director.

### Question 6: What is the relationship between the genres, the net profit and

the imdb rating?

We start by grouping by the top 5 genres only. Then, we calculate the sum profit and mean rating in each genre.

```r
options(scipen=5)
imdb_genres_budget_rating %>%
  filter(genre %in% c('Action', 'Adventure',  'Animation', 'Comedy', 'Sci-Fi'),
         !is.na(averagerating)) %>%
  group_by(genre) %>%
  summarise(sum_profit = sum(profit), average_rating = mean(averagerating)) %>%
  ggplot(mapping = aes(x = genre, y = sum_profit, fill = average_rating)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.5) +
  labs(title="Sum Profit and Average Rating per Genre", fill = "Average Rating",
       x = "Genre", y = "Sum Profit") +
  scale_y_continuous(labels = comma) +
  theme_bw()
```



Question 6 Conclusion: The plot is interesting. The genres with low sum profit comparatively has the highest ratings. Sci-Fi has the lowest sum profit but has the highest average rating at 6.8.

## Machine Learning

We create a new dataframe for the modeling.

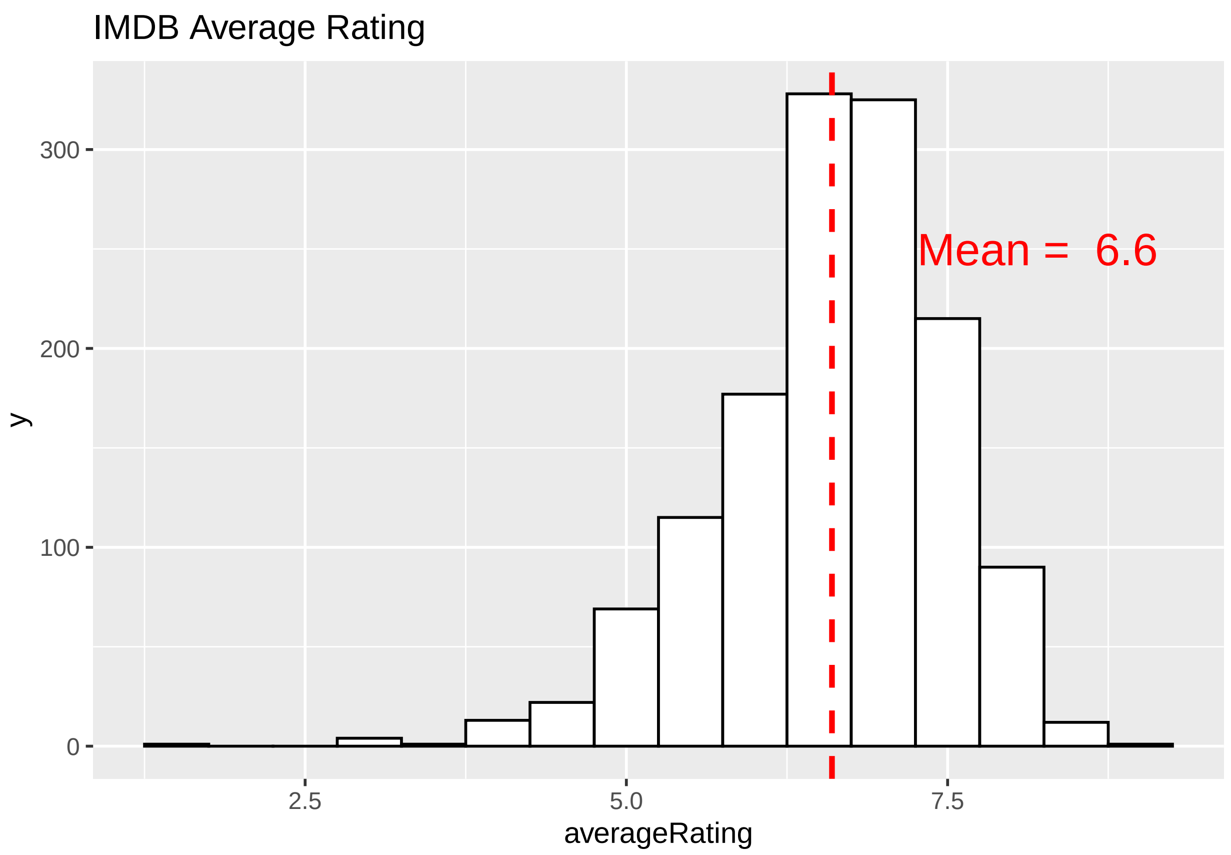
```
imdb <-
  imdb %>%
  distinct(title_id, .keep_all = TRUE) %>%
  select(primary_title, region, language, start_year, runtime_minutes,
         genres, averagerating, numvotes)
```

```
for_lm <-
  imdb_genres_budget_rating %>% inner_join(imdb, by = c("movie" = "primary_title")) %>%
  select(id, movie, year, release_month, genre, averagerating.x, numvotes,
         runtime_minutes, profit, profit_margin) %>%
  rename(averageRating = averagerating.x) %>%
  filter(!is.na(averageRating))
```

We create a histogram for the IMDB average rating variable.

```
mean_rating <- round(mean(for_lm$averageRating, na.rm = TRUE), digits = 2)
```

```r
for_lm %>%
  ggplot(mapping = aes(x = averageRating)) +
  geom_histogram(binwidth=0.5, fill="white", color = "black" ) +
  geom_vline(aes(xintercept = mean_rating),
               color = 'red', size = 1, linetype = "dashed") +
   annotate("text",
          x = 8.2,
          y = 250,
          label = paste("Mean = ", mean_rating),
          col = "red",
          size = 6) +
  labs(title = "IMDB Average Rating")
```



The IMDB scores show a nice, mostly normal distribution centered around a mean of 6.6 with somewhat of a left-side skew. Given its distribution the IMDB rating (averageRating) was the chosen response variable.

Since the goal is to predict the popularity of a movie prior to its release, the prediction model uses only variables from the data set that could be known ahead of time. Thus, variables such as DVD release date, number of IMDB votes, profit, etc. were not chosen to be in the model. Variables with large domains, such as studio name, actor/director names, URLs, etc. were excluded as well.

The variables we chose (based on research) are :

1. genre
2. runtime_minutes
3. release_month

The release month was included assuming that movies released at certain times of the year may be more popular than others. Release year was discarded as being irrelevant (without time travel capabilities, no future movie will be released in a year that has already passed) and release day was thought to be too granular to be a worthwhile predictor.

### Linear Model

```r
model <- lm(formula = averageRating ~ genre + runtime_minutes + release_month,
     data = for_lm)
```

```r
summary(model)
```

```
##
## Call:
## lm(formula = averageRating ~ genre + runtime_minutes + release_month,
##     data = for_lm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5836 -0.4651  0.0689  0.5466  2.0907
##
## Coefficients:
##                      Estimate Std. Error t value    Pr(>|t|)
## (Intercept)         5.7120018  0.1538985  37.115     < 2e-16 ***
## genreAdventure     -0.0747208  0.0980612  -0.762    0.446215
## genreAnimation      0.4024223  0.1499619   2.683    0.007382 **
## genreBiography      0.6070454  0.1314885   4.617 0.000004301 ***
## genreComedy         0.0346178  0.0940622   0.368    0.712913
## genreCrime         -0.0228079  0.1394782  -0.164    0.870133
## genreDocumentary -1.0350864  0.3047789  -3.396    0.000705 ***
## genreDrama          0.4582707  0.0885904   5.173 0.000000268 ***
## genreFamily        -0.1374703  0.1837652  -0.748    0.454555
## genreFantasy       -0.3373034  0.1397382  -2.414    0.015929 *
## genreHistory        0.4858440  0.2343833   2.073    0.038390 *
## genreHorror        -0.5248496  0.1267933  -4.139 0.000037157 ***
## genreMusic          0.2932407  0.2506826   1.170    0.242317
## genreMusical        0.5652628  0.4887153   1.157    0.247645
## genreMystery       -0.1946419  0.1449086  -1.343    0.179450
## genreNews           0.6732386  0.8375020   0.804    0.421628
## genreRomance       -0.1910063  0.1390842  -1.373    0.169901
## genreSci-Fi         0.2004732  0.1493591   1.342    0.179768
## genreSport         -0.1715743  0.3804637  -0.451    0.652096
## genreThriller      -0.2205698  0.1157799  -1.905    0.057000 .
## genreWar            0.4829777  0.3784708   1.276    0.202147
## genreWestern       -0.0519751  0.5928849  -0.088    0.930157
## runtime_minutes     0.0046862  0.0008844   5.298 0.000000138 ***
## release_month02     0.0146294  0.1277243   0.115    0.908829
## release_month03     0.4525547  0.1192095   3.796    0.000154 ***
## release_month04     0.4678463  0.1213517   3.855    0.000121 ***
## release_month05     0.2780431  0.1300481   2.138    0.032710 *
## release_month06     0.2891195  0.1301962   2.221    0.026554 *
## release_month07     0.1032503  0.1331781   0.775    0.438321
## release_month08   0.3397706  0.1267837   2.680    0.007461 **
## release_month09   0.2867266  0.1289279   2.224    0.026332 *
## release_month10   0.3670335  0.1288796   2.848    0.004473 **
## release_month11   0.6453396  0.1209814   5.334 0.000000114 ***
## release_month12   0.4182717  0.1200349   3.485    0.000510 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8295 on 1248 degrees of freedom
##   (91 observations deleted due to missingness)
## Multiple R-squared:  0.1857, Adjusted R-squared:  0.1641
## F-statistic: 8.622 on 33 and 1248 DF,  p-value: < 2.2e-16
```

```
anova(model)
```

```
## Analysis of Variance Table
##
## Response: averageRating
##                   Df Sum Sq Mean Sq F value        Pr(>F)
## genre             21 139.31  6.6337  9.6411     < 2.2e-16 ***
## runtime_minutes    1  16.16 16.1558 23.4800 0.00000142062 ***
## release_month     11  40.32  3.6650  5.3266 0.00000002733 ***
## Residuals       1248 858.71  0.6881
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

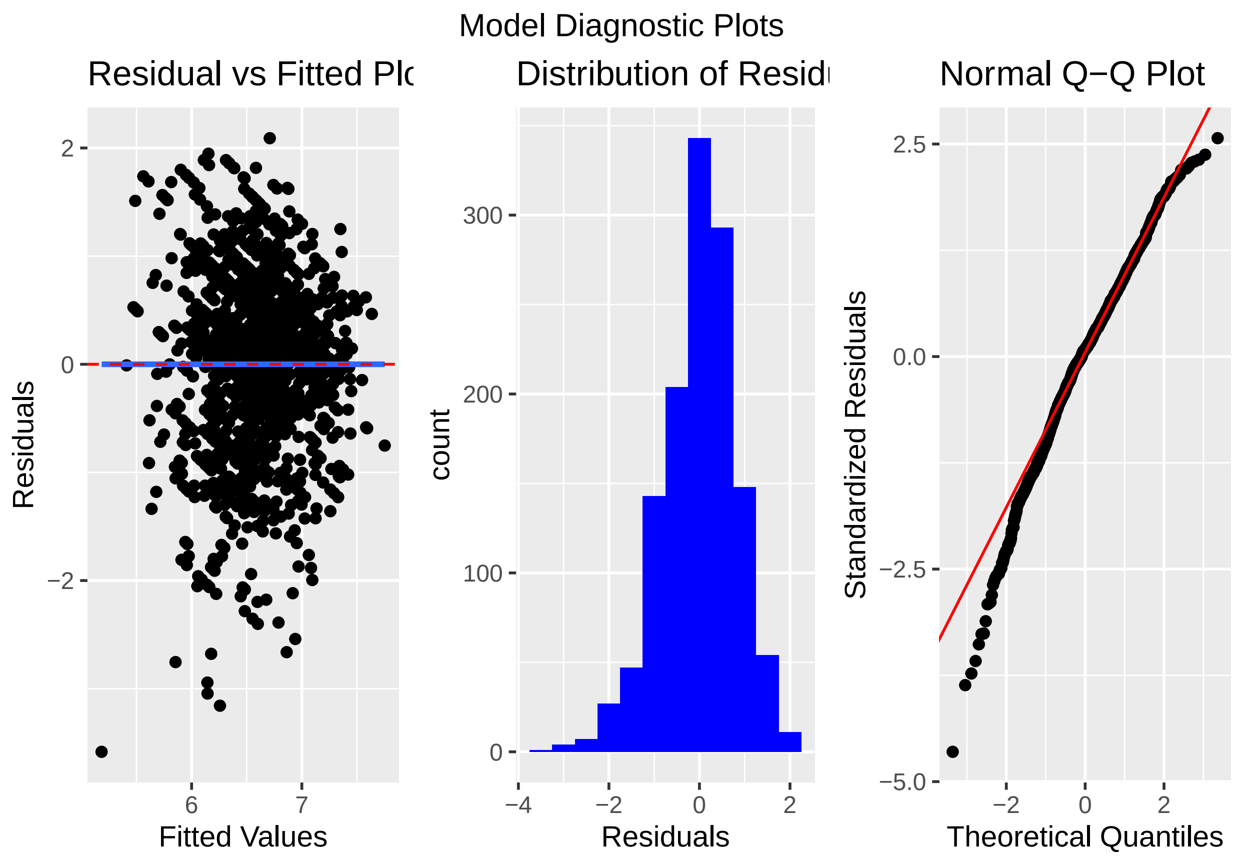
### Model Diagnostics

```
# Supplement the model data to make it easier to produce the diagnostic plots.
pMod <- fortify(model)

# Create residuals scatter plot.
p1 <- ggplot(pMod, aes(x=.fitted, y=.resid))+geom_point() +
      geom_smooth(se=FALSE)+geom_hline(yintercept=0, col="red", linetype="dashed") +
      xlab("Fitted Values")+ylab("Residuals") +
      ggtitle("Residual vs Fitted Plot")

# The following is a bunch of extra code to get around ggplot not being able
# to automatically draw a normal distribution line on a QQ plot.
# This code comes from a blog post at http://mgimond.github.io/ES218/Week06a.html
pMod$.qqnorm <- qqnorm(pMod$.stdresid, plot.it=FALSE)$x
y <- quantile(pMod$.stdresid, c(0.25, 0.75), na.rm = TRUE) # Find the 1st and 3rd quartiles
x <- quantile(pMod$.qqnorm, c(0.25, 0.75), na.rm = TRUE)   # Find the 1st and 3rd quartiles
slope <- diff(y) / diff(x)             # Compute the line slope
int <- y[1] - slope * x[1]             # Compute the line intercept

# Create residuals QQ plot.
p2 <- ggplot(pMod, aes(.qqnorm, .stdresid)) +
      geom_point(na.rm = TRUE) +
      geom_abline(intercept=int, slope=slope, color="red") +
      xlab("Theoretical Quantiles")+ylab("Standardized Residuals") +
      ggtitle("Normal Q-Q Plot")
```

```r
# Create residuals histogram plot.
p3 <- ggplot(data=pMod, aes(x=.resid)) +
      geom_histogram(binwidth=0.5, fill="blue") +
      xlab("Residuals") +
      ggtitle("Distribution of Residuals")
```

```r
grid.arrange(p1, p3, p2, nrow=1, top="Model Diagnostic Plots")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The model diagnostic plots above show that the model is passable. There is good scatter of the residuals around zero for the range of fitted values (the mean value of the residuals is, in fact, zero). The residuals Q-Q plot and distribution histogram show a pretty normal distribution, and one that mimics the left-hand skew of the original rating scores.

Overall, the evidence points toward the final model being valid.

### Prediction

```r
# Use the final model to generate rating predictions for Dirty Grandpa released
# in January 2016 and for Deadpool released in February 2016.
dataDG <- data.frame(genre="Comedy", runtime_minutes=102, release_month = "01")
predDG <- predict(model, dataDG, interval="predict")
```

```
dataDead <- data.frame(genre="Action", runtime_minutes=108, release_month = "02")
predDead <- predict(model, dataDead, interval="predict")

# Show prediction results.
df <- data.frame(t=c("Dirty Grandpa", "Deadpool"),
                 p=c(sprintf("%2.1f", predDG[1]),
                     sprintf("%2.1f", predDead[1])),
                 i=c(sprintf("%2.1f - %2.1f", predDG[2], predDG[3]),
                     sprintf("%2.1f - %2.1f", predDead[2], predDead[3])),
                 r=c("6.0", "8.1"))
kable(df, col.names=c("Movie Title", "Predicted Rating", "95% Prediction Interval", "Actual Rating"))
```

| Movie Title | Predicted Rating | 95% Prediction Interval | Actual Rating |
|---|---|---|---|
| Dirty Grandpa | 6.2 | 4.6 - 7.9 | 6.0 |
| Deadpool | 6.2 | 4.6 - 7.9 | 8.1 |

As can be seen, the model was very close in predicting the rating for Dirty Grandpa, but significantly off in its prediction for Deadpool; the real rating for which is even outside of the 95% confidence prediction interval (the interval around the predicted rating score within which we are 95% confident the real movie score would fall).

Note that the 95% confidence prediction intervals are very wide. This is a reflection of the poor predictive capability of the model (further evidenced by its F-statistic and adjusted R-square values).

## More Prediction ? (I can't do anymore - Hala)

Split into training and validation

```
# Create the training and test datasets
set.seed(100)

# Step 1: Get row numbers for the training data
trainRowNumbers <- createDataPartition(for_lm$averageRating, p=0.8, list=FALSE)

# Step 2: Create the training  dataset
trainData <- for_lm[trainRowNumbers,]

# Step 3: Create the test dataset
testData <- for_lm[-trainRowNumbers,]
```

### Descriptive Statistics

```
skimmed <- skim_to_wide(trainData)
```

```
## Warning: 'skim_to_wide' is deprecated.
## Use 'skim()' instead.
## See help("Deprecated")
```

```r
skimmed[, c(1:5, 9:11, 13, 15:16)]
```

Table 2: Data summary

| | |
|---|---|
| Name | .data |
| Number of rows | 1100 |
| Number of columns | 10 |
| Column type frequency: | |
| character | 3 |
| numeric | 7 |
| Group variables | None |

**Variable type: character**

| skim_variable | n_missing | complete_rate | min | whitespace |
|---|---|---|---|---|
| movie | 0 | 1 | 2 | 0 |
| release_month | 0 | 1 | 2 | 0 |
| genre | 0 | 1 | 3 | 0 |

**Variable type: numeric**

| skim_variable | n_missing | complete_rate | mean | sd | p25 | p75 | p1 |
|---|---|---|---|---|---|---|---|
| id | 0 | 1.00 | 52.49 | 28.46 | 29.00 | 77.00 | 10 |
| year | 0 | 1.00 | 2014.01 | 2.48 | 2012.00 | 2016.00 | 201 |
| averageRating | 0 | 1.00 | 6.60 | 0.90 | 6.10 | 7.20 | |
| numvotes | 167 | 0.85 | 138952.08 | 179294.52 | 16465.00 | 186788.00 | 138776 |
| runtime_minutes | 73 | 0.93 | 102.04 | 28.04 | 92.00 | 118.00 | 28 |
| profit | 0 | 1.00 | 162219543.94 | 235283389.95 | 25371783.25 | 209068670.25 | 235134527 |
| profit_margin | 0 | 1.00 | 0.65 | 0.23 | 0.51 | 0.82 | |

**Handling missing values**

```r
# Create the knn imputation model on the training data
preProcess_missingdata_model <- preProcess(trainData, method='knnImpute')
preProcess_missingdata_model
```

```
## Created from 915 samples and 10 variables
##
## Pre-processing:
##   - centered (7)
##   - ignored (3)
##   - 5 nearest neighbor imputation (7)
##   - scaled (7)
```

```r
# Use the imputation model tro predict the values of missing data points
trainData <- predict(preProcess_missingdata_model, newdata = trainData)
anyNA(trainData)
```

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

## Conclusion

In answer to the research question stated in Part 2, yes, it is possible to predict the popularity of a movie based upon basic movie characteristic data. In this analysis, a valid, parsimonious, multi-variable, linear regression model was created that proved to have some capability for predicting movie popularity as indicated by IMDB movie rating score.

But, there is much room for improvement. As shown in Part 5, the predictive power of the model is limited. Further analysis could pursue the following suggestions for improving the model.

Start with a larger analysis sample to capture more variability in the population data. Use a stratified sample reflecting the true proportion of movie genres in the population rather than a simple random sample. Create separate models for each movie genre. Identify other movie characteristic data to add to the model; identificaton of sequels and their ratings or searching for keywords in the movie title or description, for example.