Box office Revenue Predictions using Random Forest

Emad Abdellatif

May 23, 2024

Contents

Introduction

1. Exploratory data analysis,
2. Feature engineering,
3. Treating missing values, and
4. Machine learning using random forest.

```
library(tidyverse) # Multiple packages
library(plotly) # Interactive visualizations
library(ggthemes) # Visualization themes
library(viridis) # Color scales
library(corrplot) # Correlation visualizations
library(gridExtra) # Grids for visualizations
library(VIM) # Visualizing missing values
library(lubridate) # Working with dates
library(randomForest) # Classification algorithm
library(tinytex)
library(webshot2)
```

Read in the train and test data sets and then bind the two sets using `bind_rows()` from the DPLYR package. We will do all feature engineering and data preparation on both data sets and then divide our data into train and test sets again later before creating our model.

```
#setwd("C:/Users/Emad Abdellatif/Desktop/Data Science Projects/Movies-Box-Office")
setwd("/Users/emadabdellatif/Documents/Projects/box-office-predictions")
train_data = read.csv('train.csv', na.strings=c("", '#N/A', '[]', '0'))
test_data = read.csv('test.csv', na.strings=c("", '#N/A', '[]', '0'))

full_data  <- bind_rows(train_data, test_data)
```

Lets explore our data and see how it looks like.

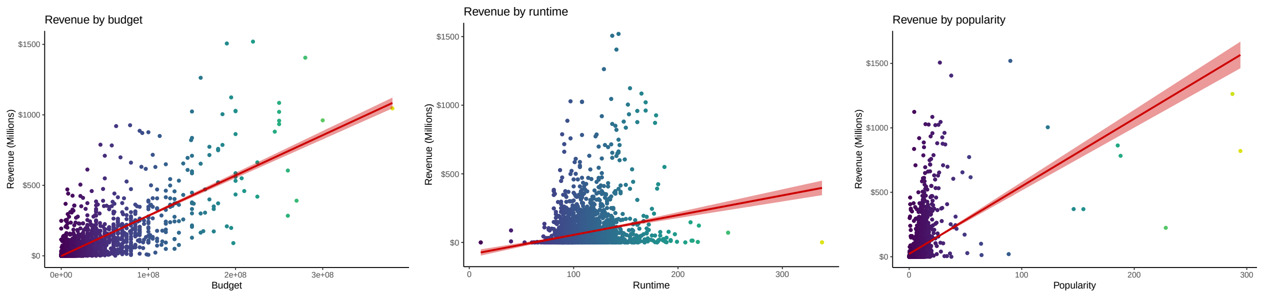
```
glimpse(full_data)
```

```
## Rows: 7,398
## Columns: 23
## $ id                    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1~
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Time Machine C~
## $ budget                <int> 14000000, 40000000, 3300000, 1200000, NA, 800000~
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{'id': 35, '~
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whiplash/", "ht~
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802", "tt182148~
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "en", "en", ~
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Princess Diaries ~
## $ overview              <chr> "When Lou, who has become the \"father of the In~
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.174936, 1.14807~
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "/w9Z7A0GHEh~
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id': 4}, {'nam~
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'United States of~
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9/12", "2/5/~
## $ runtime               <int> 93, 113, 105, 122, 118, 83, 92, 84, 100, 91, 119~
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'English'}]", "[{'~
## $ status                <chr> "Released", "Released", "Released", "Released", ~
## $ tagline               <chr> "The Laws of Space and Time are About to be Viol~
## $ title                 <chr> "Hot Tub Time Machine 2", "The Princess Diaries ~
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}, {'id': 96~
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', 'credit_id'~
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02c8c8', 'dep~
## $ revenue               <int> 12314651, 95149435, 13092000, 16000000, 3923970,~
```

With a quick observation, it looks like there are a few variables that could use some cleaning up. Because of this, we will use *regular expressions* and extract the appropriate information from them before using them in our model.

Exploratory Data Analysis

Lets begin by plotting our existing variables `budget`, `runtime`, and `popularity` in order to see their relation to the variable we are trying to predict, revenue.



We can see some clear trends that an increase in budget and popularity tend to lead to higher revenue. Runtime seems to show this trend as well, although not as strongly.

Feature Engineering

Collection

The strings in `belongs_to_collection` are messy and contain information we do not need. Lets use regular expressions to extract the collection names from these strings.

```r
full_data$collection_name <- str_extract(full_data$belongs_to_collection,
                pattern = "(?<=name\\'\\:\\s{1}\\').+(?=\\'\\,\\s{1}\\'poster)")
```

Now that we have extracted the collection names lets check the biggest collections.

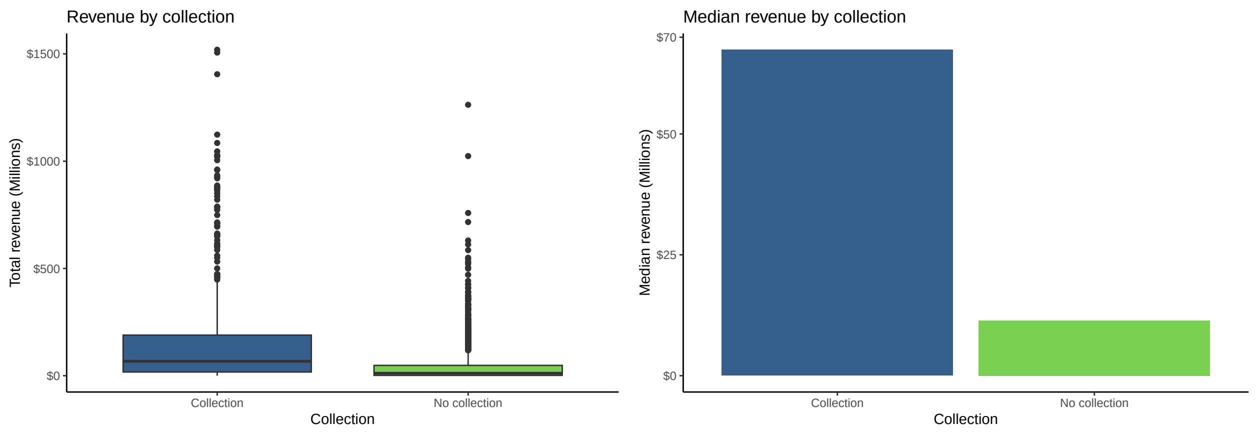
```r
full_data[1:3000,] %>%
   group_by(collection_name) %>%
   summarise(movie_count = n()) %>%
   arrange(desc(movie_count)) %>%
   filter(!is.na(collection_name)) %>%
   head(10)
```

```
## # A tibble: 10 x 2
##    collection_name                        movie_count
##    <chr>                                        <int>
##  1 James Bond Collection                           16
##  2 Friday the 13th Collection                       7
##  3 The Pink Panther (Original) Collection           6
##  4 Pokémon Collection                               5
##  5 Police Academy Collection                        5
##  6 Alien Collection                                 4
##  7 Ice Age Collection                               4
##  8 Paranormal Activity Collection                   4
##  9 Rambo Collection                                 4
## 10 Resident Evil Collection                         4
```

As each collection is fairly small we will engineer a new variable that consist of either being in a collection or not being in a collection.

```r
full_data$collection[!is.na(full_data$belongs_to_collection)] <- 'Collection'
full_data$collection[is.na(full_data$belongs_to_collection)] <- 'No collection'
```

Now lets plot our new variable to visualize how the two levels differ on revenue.



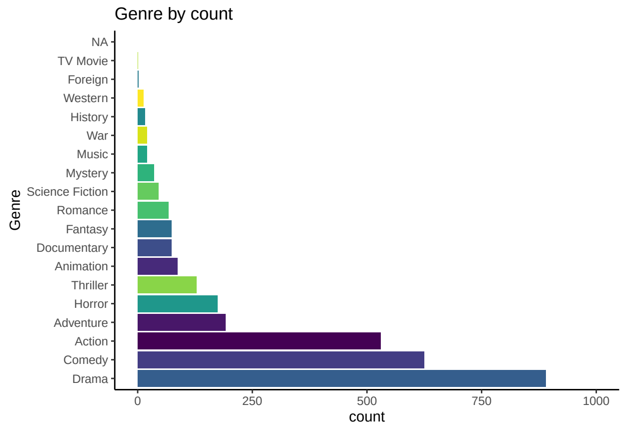
On average, movies that are in collections seem to be getting higher revenues as we can see by looking at the box plot and bar plot.

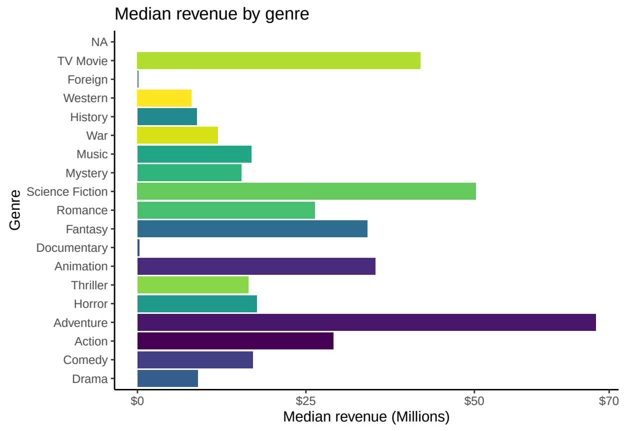
Main genre

We now want to extract the first genre from the `genres` strings to get the main genre for each movie. First, we will create a vector with the genres we want to extract. Next, we will extract the genres and add them to a new variable called `main_genre`.

```r
genres_matching_point <- "Comedy|Horror|Action|Drama|Documentary|Science Fiction|
              Crime|Fantasy|Thriller|Animation|Adventure|Mystery|War|Romance|Music|
              Family|Western|History|TV Movie|Foreign"

full_data$main_genre <- str_extract(full_data$genres, genres_matching_point)
```

Lets plot `main_genre` to find (1) how many movies there are per genre and (2) the median revenue by genre.





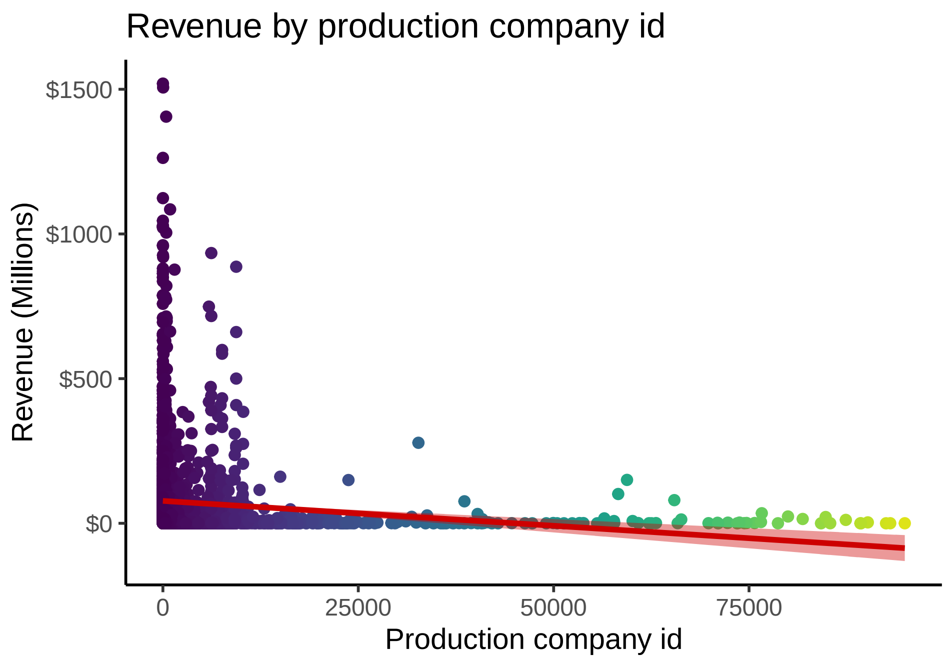
Here we can see that different genres seem to be making different revenues. Action movies seem to have the highest median revenue, followed by science fiction. One thing to note is that the median revenue for genres with few counts, such as TV Movie, might be over-/underestimations due to small sample sizes.

Production company id

We want to extract the first (and main) production company id from `production_companies` and create the new variable `prod_comp_id`.

```r
full_data$prod_comp_id <- str_extract(full_data$production_companies,
                                      pattern = "([0-9]+)")
```

Lets plot how this variable and see how it affects revenue.



Production companies with lower numbered id's seem to be making more revenue compared to the ones with higher id's. Lets check the correlation to confirm this.

```
cor(full_data$revenue, full_data$prod_comp_id, use = 'complete.obs')
```

```
## [1] -0.1282278
```

There seems to be a *small* negative correlation present.

Top production companies

Lets extract the main production company name from `production_companies`.

```
full_data$prod_comp_name <- gsub('(^\\[\\{\'name\'\\:\\s\'|\'\\,\\s\'id.*)', '',
                                  full_data$production_companies)
```

Check the top production companies ordered by number of movies produced.

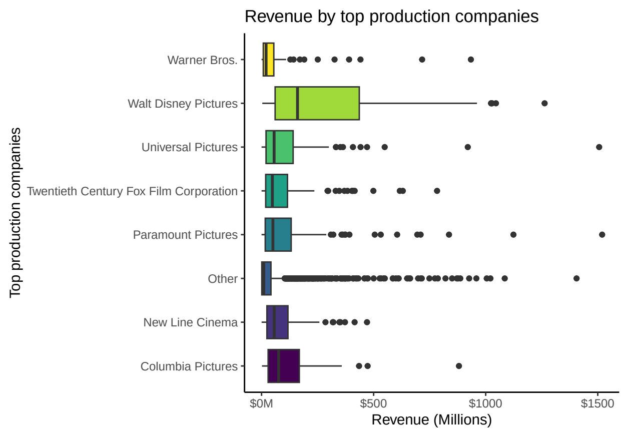
```
full_data[1:3000,] %>%
   group_by(prod_comp_name) %>%
   summarise(movie_count = n()) %>%
   arrange(desc(movie_count)) %>%
   filter(!is.na(prod_comp_name)) %>%
   head(10)
```

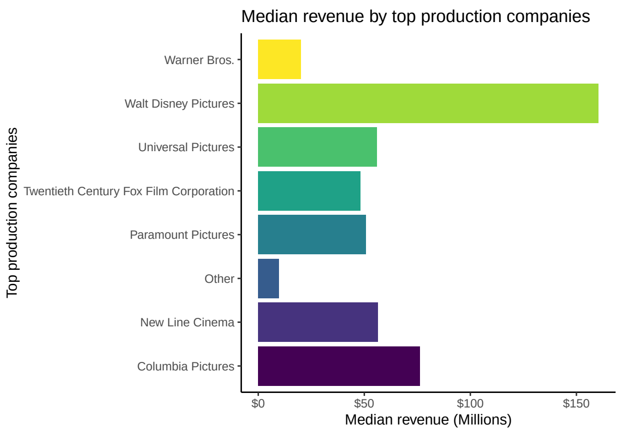
```
## # A tibble: 10 x 2
##    prod_comp_name                           movie_count
##    <chr>                                          <int>
##  1 Universal Pictures                               167
##   2 Paramount Pictures                           158
##   3 Twentieth Century Fox Film Corporation       122
##   4 Columbia Pictures                             90
##   5 Warner Bros.                                  70
##   6 New Line Cinema                               69
##   7 Walt Disney Pictures                          62
##   8 Columbia Pictures Corporation                 44
##   9 TriStar Pictures                              44
##  10 United Artists                                41
```

We will create a new variable called `top_prod_comp` (top production companies). We will create a separate category for each production company that has produced at least 60 movies that are present in our data set. All other production companies, including NAs, get put into an 'other' category.

```
full_data$top_prod_comp[full_data$prod_comp_name=='Universal Pictures'] <- 'Universal Pictures'
full_data$top_prod_comp[full_data$prod_comp_name=='Paramount Pictures'] <- 'Paramount Pictures'
full_data$top_prod_comp[full_data$prod_comp_name=='Twentieth Century Fox Film Corporation'] <- 'Twentiet
full_data$top_prod_comp[full_data$prod_comp_name=='Columbia Pictures'] <- 'Columbia Pictures'
full_data$top_prod_comp[full_data$prod_comp_name=='New Line Cinema'] <- 'New Line Cinema'
full_data$top_prod_comp[full_data$prod_comp_name=='Warner Bros.'] <- 'Warner Bros.'
full_data$top_prod_comp[full_data$prod_comp_name=='Walt Disney Pictures'] <- 'Walt Disney Pictures'

full_data$top_prod_comp[is.na(full_data$top_prod_comp)] <- 'Other'
```

Lets take a look at the effects of this variable on revenue.





Here we can see that the average revenue for a lot of the top production companies is higher than the 'other' production companies.

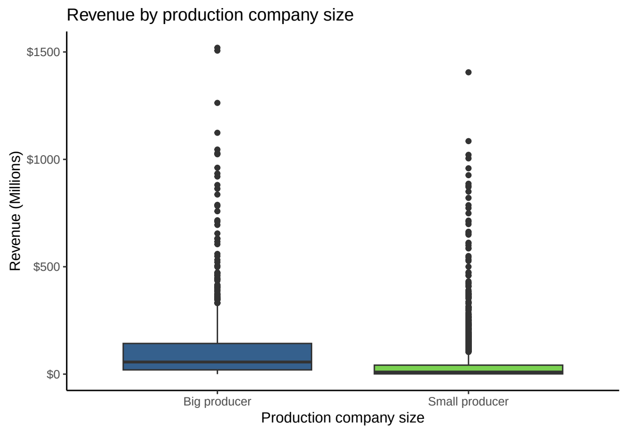
Production company size

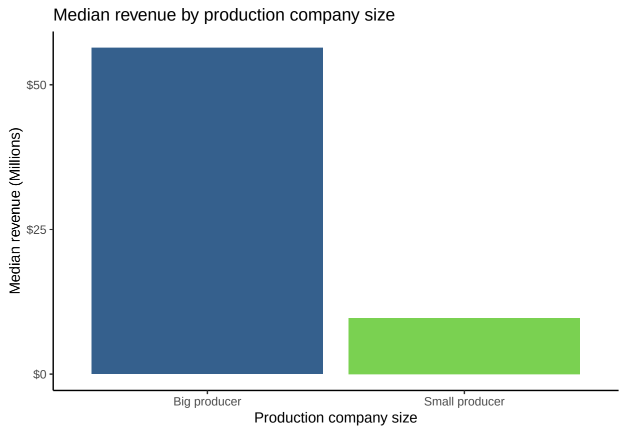
Now, lets move on to create `prod_comp_size` (production company size – *big producer* v. *small producer*). We will assign the production companies that have at least 60 movies each as big producers and all the rest as small producers. We will assume that all NAs are small producers.

```
full_data$prod_comp_size[full_data$prod_comp_name=='Universal Pictures'] <- 'Big producer'
full_data$prod_comp_size[full_data$prod_comp_name=='Paramount Pictures'] <- 'Big producer'
full_data$prod_comp_size[full_data$prod_comp_name=='Twentieth Century Fox Film Corporation'] <- 'Big pro
full_data$prod_comp_size[full_data$prod_comp_name=='Columbia Pictures'] <- 'Big producer'
```

```
full_data$prod_comp_size[full_data$prod_comp_name=='New Line Cinema'] <- 'Big producer'
full_data$prod_comp_size[full_data$prod_comp_name=='Warner Bros.'] <- 'Big producer'
full_data$prod_comp_size[full_data$prod_comp_name=='Walt Disney Pictures'] <- 'Big producer'

full_data$prod_comp_size[is.na(full_data$prod_comp_size)] <- 'Small producer'
```

Lets see how our new variable affects revenue.





Again, we can see that the big production companies are, on average, making more than the smaller production companies.

Top production countries

Lets extract the country abbreviations from the messy strings in `production_countries`.

```
full_data$prod_country <- str_extract(string = full_data$production_countries,
                                      pattern = "[:upper:]+")
```

Check the top countries by number of movies produced.

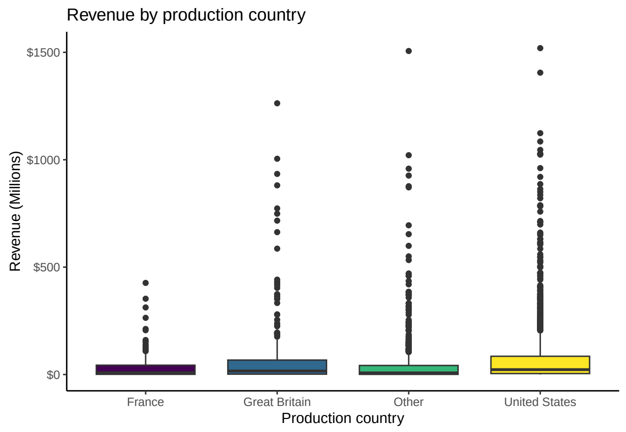
```
full_data[1:3000,] %>%
   group_by(prod_country) %>%
   summarise(movie_count = n()) %>%
   arrange(desc(movie_count)) %>%
   filter(!is.na(prod_country)) %>%
   head(10)
```

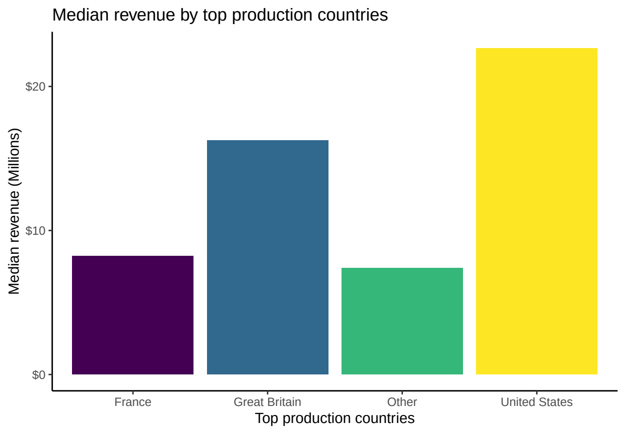
```
## # A tibble: 10 x 2
##    prod_country movie_count
##    <chr>              <int>
##  1 US                  1818
##  2 GB                   234
##  3 FR                   147
##  4 CA                    97
##  5 DE                    90
##  6 IN                    78
##  7 AU                    52
##  8 JP                    50
##  9 RU                    47
## 10 IT                    36
```

Separate into top production countries (criteria: 100+ movies) and 'other'.

```
full_data$top_prod_country[full_data$prod_country=='US'] <- 'United States'
full_data$top_prod_country[full_data$prod_country=='GB'] <- 'Great Britain'
full_data$top_prod_country[full_data$prod_country=='FR'] <- 'France'

full_data$top_prod_country[is.na(full_data$top_prod_country)] <- 'Other'
```

Plot our new `top_prod_country` variable.





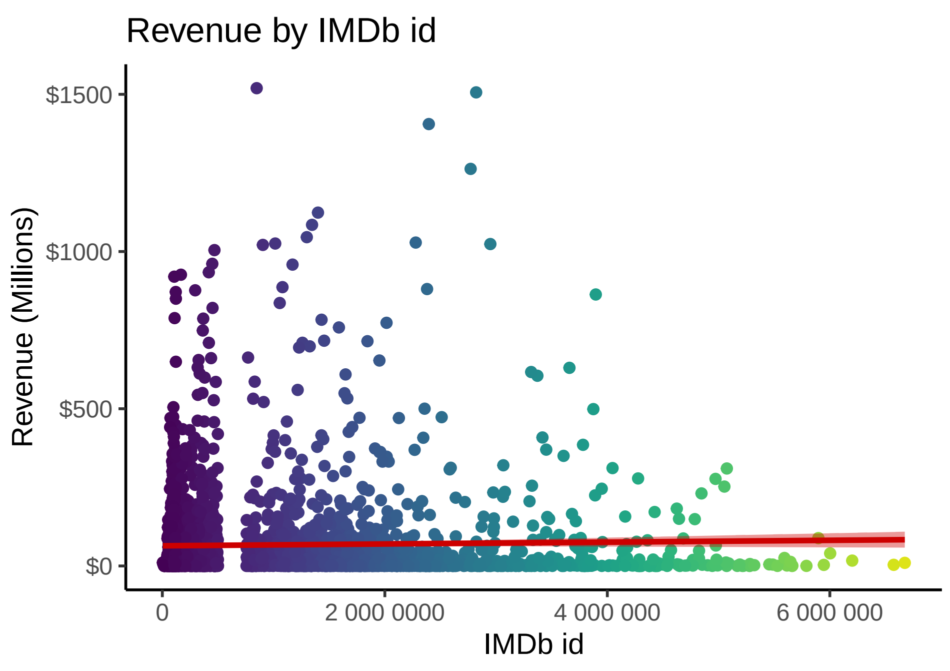
The U.S. and Great Britain seem to, on average, be getting more revenue than the countries that are not among the top production countries.

IMDB id

We will now extract the IMDb number from the `IMDb_id` string in order to see if this variable affects revenue. There will likely not be any correlation with this and revenue, but we will plot and explore this to make sure.

```
full_data$imdb_id_2 <- str_extract(full_data$imdb_id, '[0-9]+')
```

Plot this new variable.



There does not seem to be any trend between IMDb id and revenue. Lets check the correlation to make sure.

```
cor(full_data$revenue, full_data$imdb_id_2, use = 'complete.obs')
```

```
## [1] 0.02428141
```

This confirms that there is next to no correlation and that it is probably best to not include this variable in our prediction model.

Language

Lets take a look at the most common original languages for our movies.

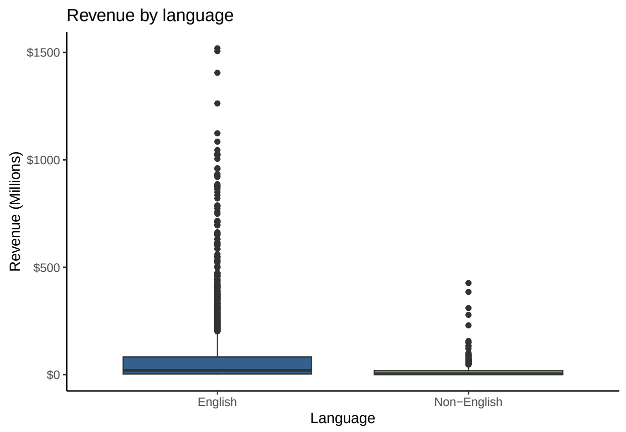
```
full_data[1:3000,] %>%
   group_by(original_language) %>%
   summarise(movie_count = n()) %>%
   arrange(desc(movie_count)) %>%
   filter(!is.na(original_language)) %>%
   head(10)
```

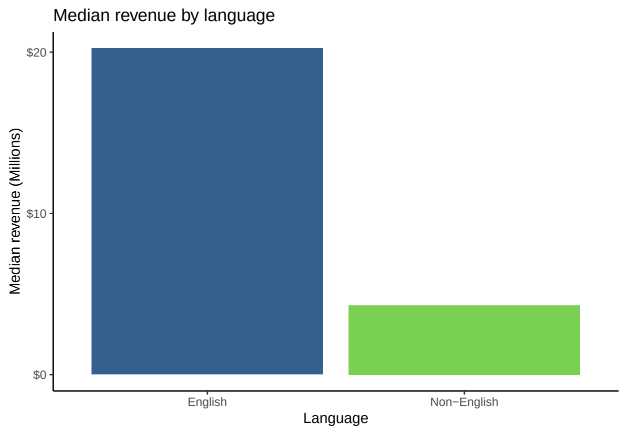
```
## # A tibble: 10 x 2
##    original_language movie_count
##    <chr>                   <int>
##  1 en                       2575
##  2 fr                         78
##  3 ru                         47
##  4 es                         43
##  5 hi                         42
##  6 ja                        37
##  7 it                        24
##  8 cn                        20
##  9 ko                        20
## 10 zh                        19
```

Since the absolute majority of the movies are English, with the second most popular language being French with 78 movies, we will create the variable `language` with levels English versus Non-English.

```
full_data$language[full_data$original_language=='en'] <- 'English'
full_data$language[is.na(full_data$language)] <- 'Non-English'
```

Now lets plot our new variable to see how it affects revenue.





Seems like English-language movies make on average about 5 times the revenue of non-English language movies.

Year, quarter, month, week, and weekday released

Now we will create 5 new variables: (1) `year_released`, (2) `quarter_released`, (3) `month_released`, (4) `week_released`, and (5) `weekday_released`.

Before creating our variables we will fix missing values for `release_date` so that we do not need to do so for each created variable later.

Lets see which rows have missing values for `release_date` and look up the titles and runtimes.

```
which(is.na(full_data$release_date))

full_data[3829, c('title', 'runtime')]
```

```
## [1] 3829
##                               title runtime
## 3829 Jails, Hospitals & Hip-Hop      90
```

I found online that the movie was released 3/20/2001 so I'll add this information.

```r
full_data$release_date[3829] <- '3/20/01'
```

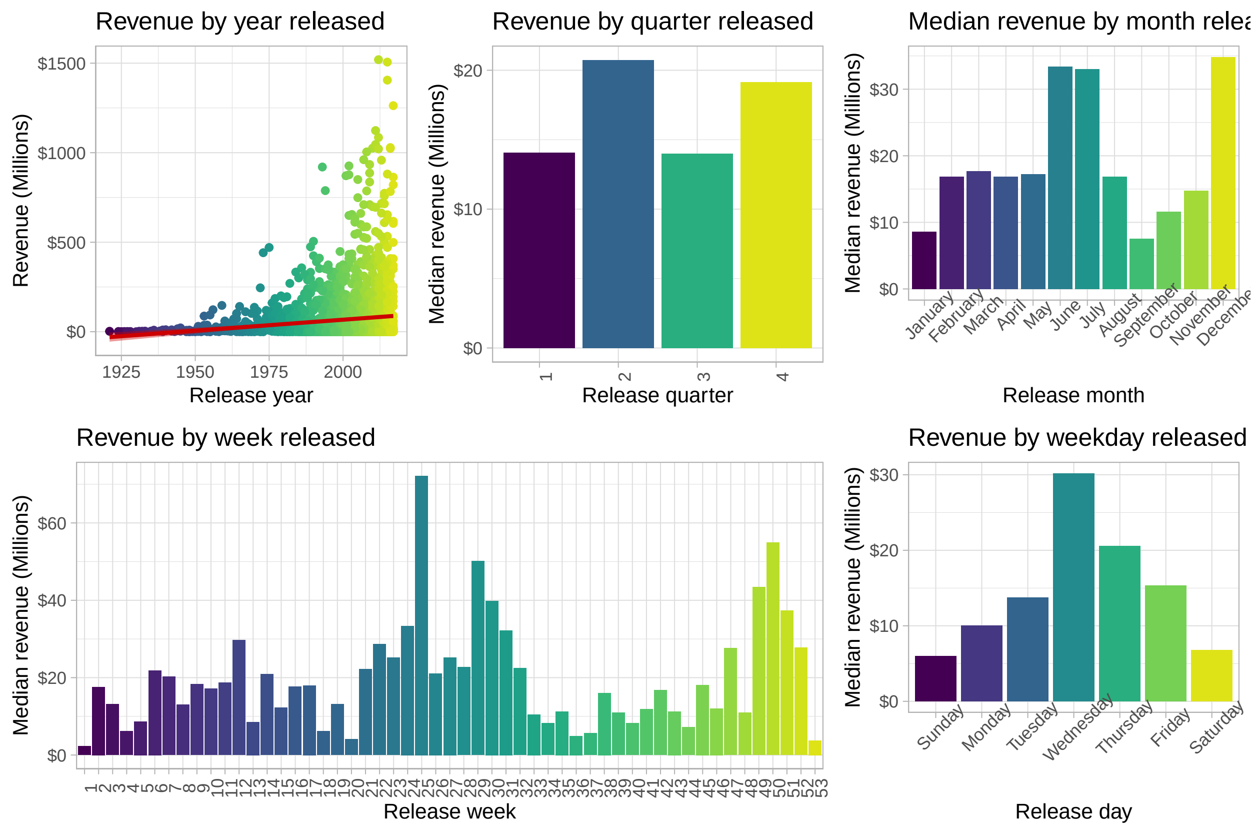
Our *release date* variable does not contain the century for the year the movies were released. Hence, we need to set the cut off value for century break as '20' (i.e., 1920). We will do this so that old movies (from 1921-1968) don't erroneously get classified as being from 20th century.

```r
full_data$release_date_mod <- parse_date_time2(full_data$release_date, "mdy",
                                               cutoff_2000 = 20)
```

Create year, quarter, month, week, and weekday released using the LUBRIDATE package.

```r
full_data$year_released <- ymd(full_data$release_date_mod) %>%
   lubridate::year()  # Grab year.

full_data$quarter_released <- ymd(full_data$release_date_mod) %>%
   lubridate::quarter()  # Grab quarter.

full_data$month_released <- ymd(full_data$release_date_mod) %>%
   lubridate::month(label = TRUE, abbr = FALSE)  # Grab month.

full_data$week_released <- ymd(full_data$release_date_mod) %>%
   lubridate::week()  # Grab week.

full_data$weekday_released <- ymd(full_data$release_date_mod) %>%
   lubridate::wday(label = TRUE, abbr = FALSE)  # Grab weekday.
```

Lets graph these variables to see how they affect revenue.



Here we can see that:

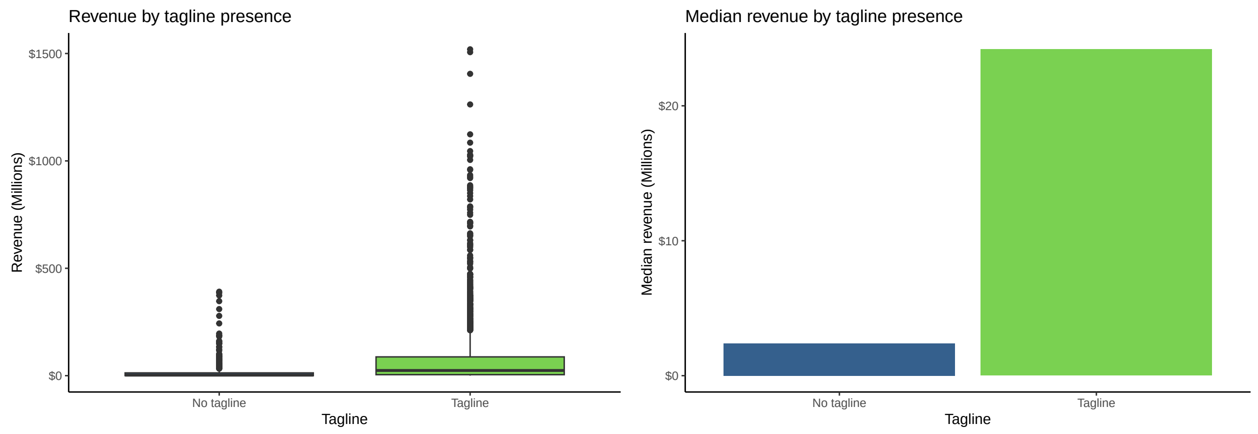
- The year plot seems to indicate revenue has been increasing over the years.
- Movies being released in June, July and December seem to be getting higher revenues. This is in line with what one would believe as a lot of blockbuster movies are released during the summer, while a lot of movies that are trying to compete for the Oscars are released in December.
- Movies that are released on Wednesdays seem to be getting somewhat higher revenues as well.

Tagline presence

Next we will feature engineer a `tagline_presence` variable by simply categorizing whether a movie has a tagline or not.

```r
full_data$tagline_presence[is.na(full_data$tagline)] <- 'No tagline'
full_data$tagline_presence[is.na(full_data$tagline_presence)] <- 'Tagline'
```

Next, lets create a bar plot of `tagline_presence` against `revenue`.



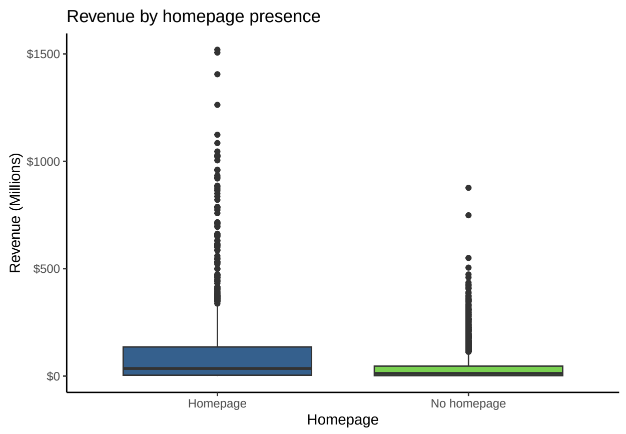
Seems like the median for movies with taglines is about 10 times that of movies without a tagline.

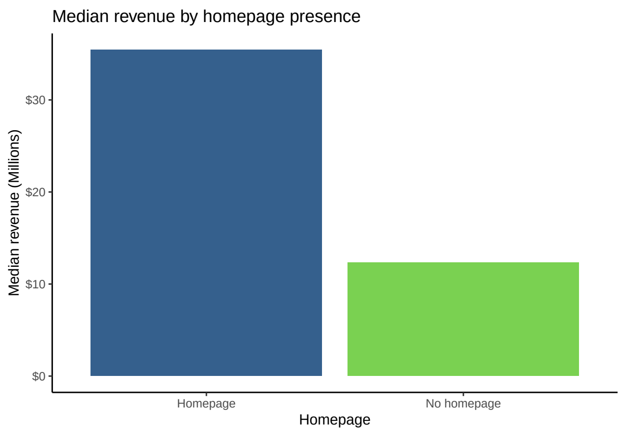
Homepage presence

Now we will create the `homepage_presence` variable by categorizing if the movies have a homepage or not.

```r
full_data$homepage_presence[is.na(full_data$homepage)] <- 'No homepage'
full_data$homepage_presence[is.na(full_data$homepage_presence)] <- 'Homepage'
```

Now lets plot this new feature.





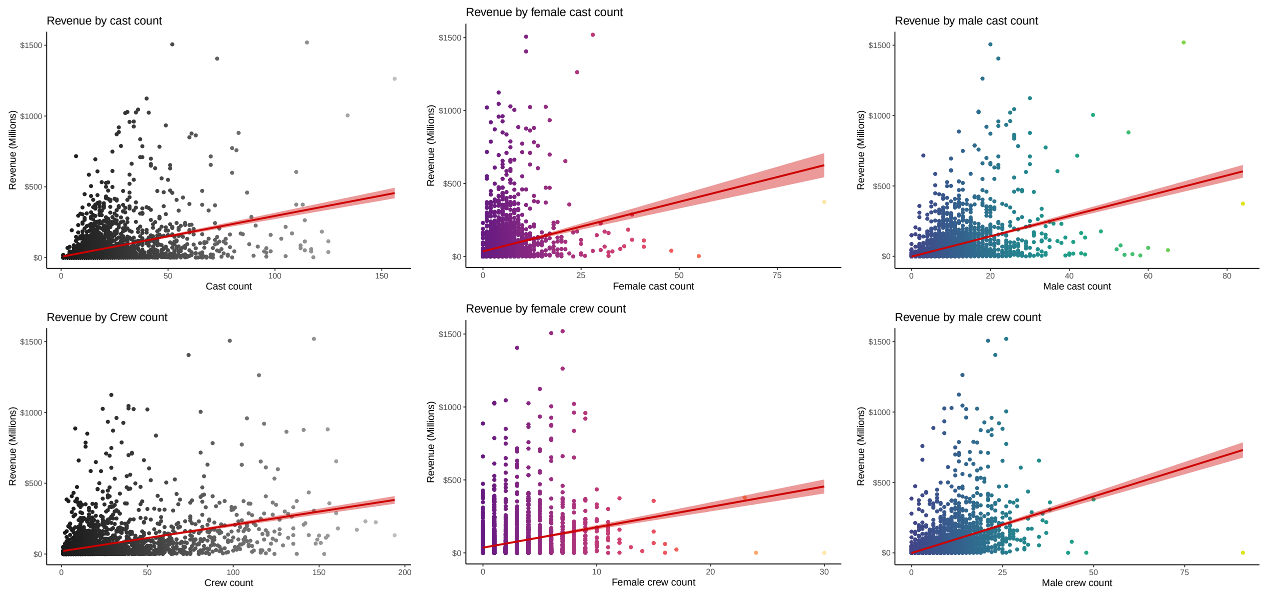
Movies with homepages seem to be making on average 3 times as much as movies without a homepage.

Gender of cast & crew

We will now create new variables to see how gender of cast and crew affect revenue.

```r
# Total cast count and by gender
full_data$number_of_cast <- str_count(full_data$cast, 'name')
full_data$female_cast <- str_count(full_data$cast, ('gender\'\\:\\s1'))
full_data$male_cast <- str_count(full_data$cast, ('gender\'\\:\\s2'))
full_data$unspecified_cast <- str_count(full_data$cast, ('gender\'\\:\\s0'))

# Total crew count and by gender
full_data$number_of_crew <- str_count(full_data$crew, 'name')
full_data$female_crew <- str_count(full_data$crew, ('gender\'\\:\\s1'))
full_data$male_crew <- str_count(full_data$crew, ('gender\'\\:\\s2'))
full_data$unspecified_crew <- str_count(full_data$crew, ('gender\'\\:\\s0'))
```

Lets plot this.



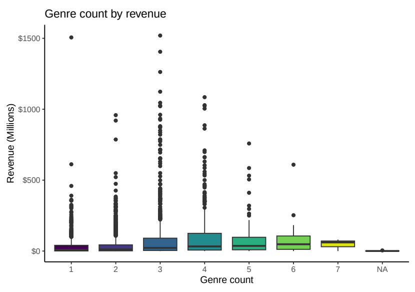
Here we can see the distribution in revenue by gender for cast and crew. There seems to be a quite clear trend that the more cast and crew the movie has, the higher the revenue.

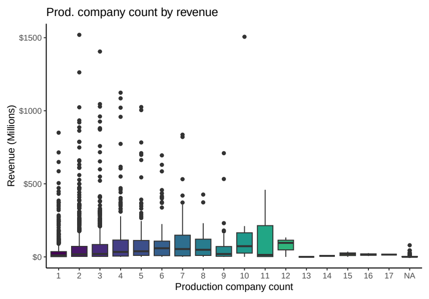
Number of...

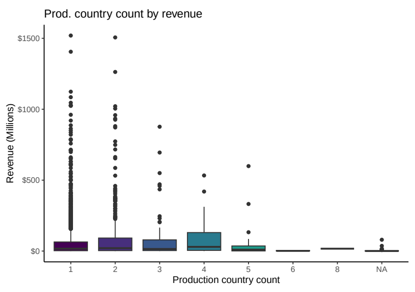
Now we will create multiple new features by extracting the number of occurrences of certain variables within a string. We will extract the number of: (1) **genres**, (2) **production companies**, (3) **production countries**, (4) **spoken languages**, and (5) **keywords**. We will do this by counting the occurrence of 'name' in each string.

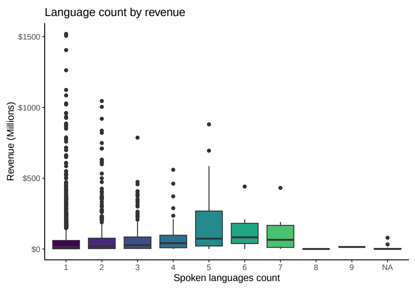
```r
full_data$number_of_genres <- str_count(full_data$genres, 'name')
full_data$number_of_prod_companies <- str_count(full_data$production_companies, 'name')
full_data$number_of_prod_countries <- str_count(full_data$production_countries, 'name')
full_data$number_of_spoken_languages <- str_count(full_data$spoken_languages, 'name')
full_data$number_of_keywords <- str_count(full_data$Keywords, 'name')
```

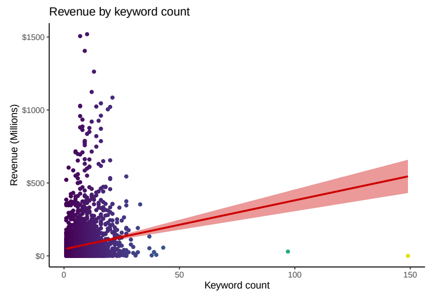
Now lets plot these variables.







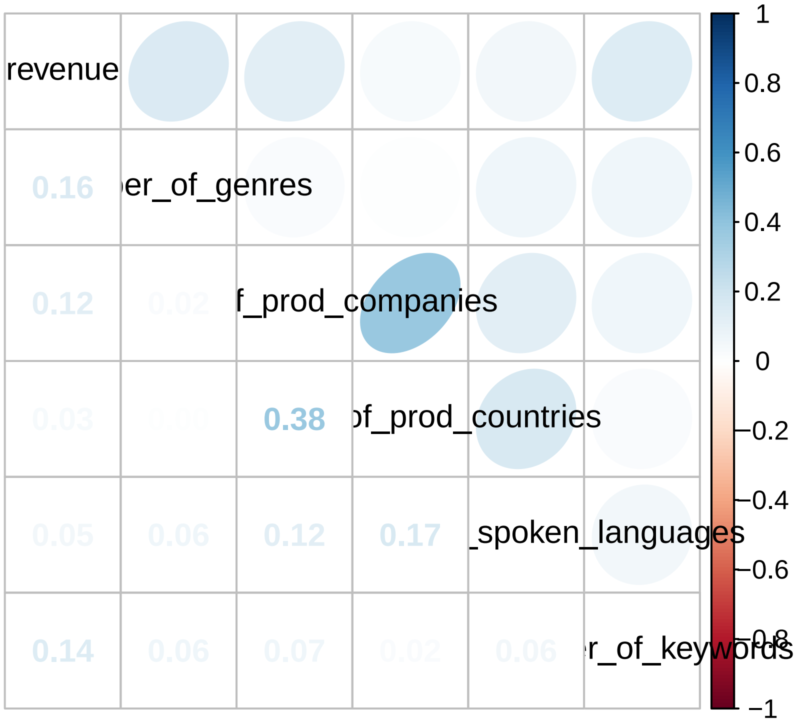




Here we can see that:

- The more genres a movie has, the higher the median revenue.
- The more production companies a movie has, the higher the revenue, up to 6 production companies. A higher number than that seems to have more volatile results. This might be explained by smaller sample sizes.
- There seems to be no clear trend between number of production countries and revenue.
- There seems like there is no clear trend for number of spoken languages either.
- There is a trend between more keywords and higher revenue.

Lets plot these variables on a correlation plot.



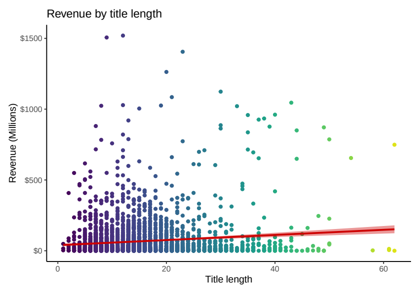
`number_of_spoken_languages` and `number_of_prod_countries` show no correlation with `revenue`. We have 2 options: either remove these variables or try to see if we can make the patterns stronger by bunching levels together. I tried both these options and got better results by removing the variables from the model.

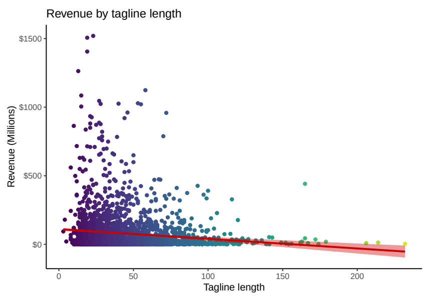
Length of...

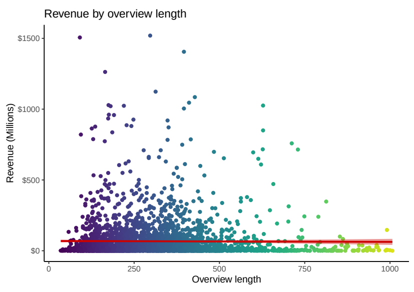
We will now create 3 additional variables, (1) `title_length`, (2) `overview_length`, and (3) `tagline_length` by extracting the lengths of the strings of the variables.

```
full_data$title_length <- str_length(full_data$title)
full_data$tagline_length <- str_length(full_data$tagline)
full_data$overview_length <- str_length(full_data$overview)
```

Lets plot these variables.







The correlation between these variables and revenue seem small. Lets take a look at what the actual correlations are.

```r
cor(full_data$revenue, full_data$title_length, use = 'complete.obs')
cor(full_data$revenue, full_data$tagline_length, use = 'complete.obs')
cor(full_data$revenue, full_data$overview_length, use = 'complete.obs')
```

```
## [1] 0.1096643
## [1] -0.1206023
## [1] -0.007450262
```

Here we can see that there is a weak correlation between title length and tagline length and revenue. There is no correlation between overview length and revenue so it is probably best to not include the variable in our model.

Last data preparations

Subsetting the data

Lets first create a subset containing all the variables we want to keep for our machine learning model.

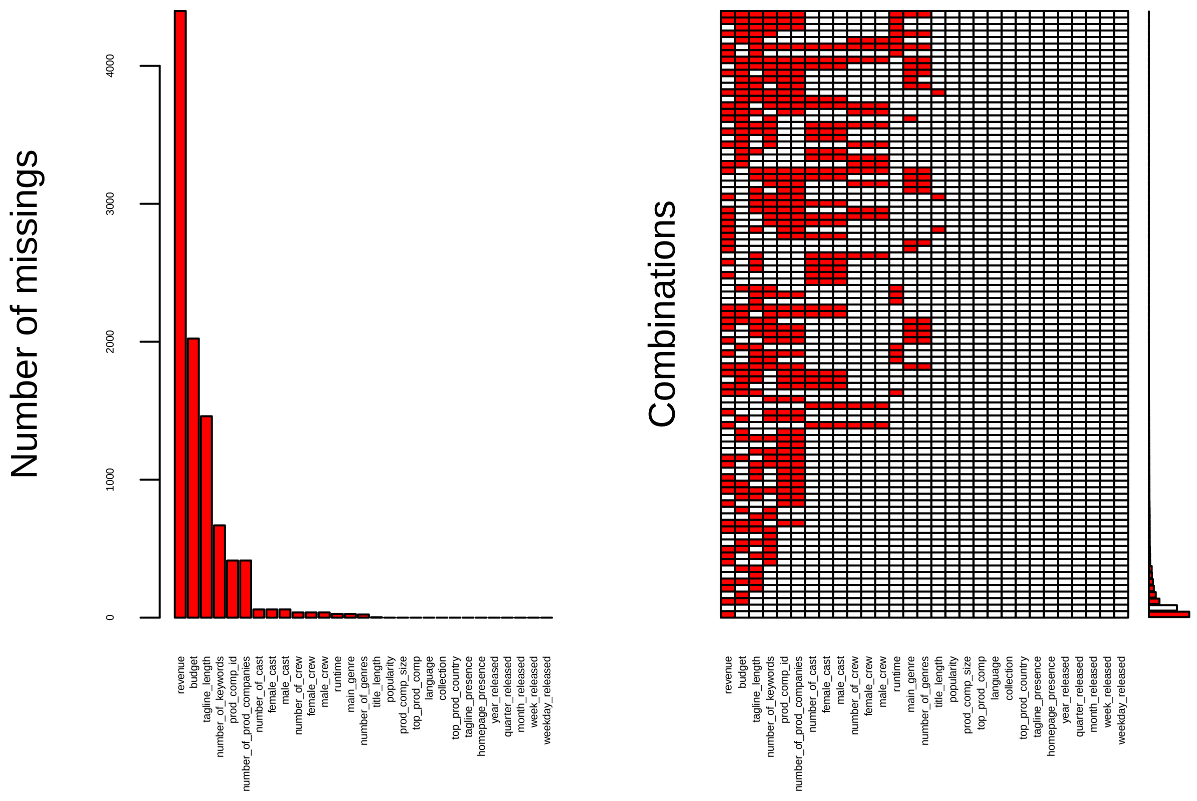
```r
full_data_subset <- subset(full_data,
                     select = c(popularity, runtime, budget, prod_comp_size,
                     top_prod_comp, prod_comp_id, main_genre, language, collection,
                     top_prod_country, tagline_presence, homepage_presence,
                     year_released, quarter_released, month_released, week_released,
                     weekday_released, number_of_keywords, number_of_prod_companies,
                     number_of_genres, title_length, tagline_length, number_of_cast,
                     number_of_crew, female_cast, male_cast, female_crew, male_crew,
                   # number_of_prod_countries, number_of_spoken_languages,
                   # imdb_id_2, overview_length, unspecified_cast, unspecified_crew,
                      revenue))
```

What missing values do we have?

We will now begin our process of **treating missing values** in our data set.

Lets first visualize the missing values in our data by using `aggr()` from the VIM package.

```r
aggr(full_data_subset, sortVars = TRUE, prop = FALSE, cex.axis = .35,
     numbers = TRUE, col = c('grey99','red'))
```



```
##
##  Variables sorted by number of missings:
##                  Variable Count
##                   revenue  4398
##                    budget  2023
##            tagline_length  1460
##        number_of_keywords   669
##              prod_comp_id   414
##  number_of_prod_companies   414
##            number_of_cast    60
##               female_cast    60
##                 male_cast    60
##            number_of_crew    38
##               female_crew    38
##                 male_crew    38
##                   runtime    27
##                main_genre    26
##          number_of_genres    23
##              title_length     3
##                popularity     0
##            prod_comp_size     0
##             top_prod_comp     0
##                  language     0
##                collection     0
##          top_prod_country     0
##          tagline_presence     0
##         homepage_presence     0
##             year_released     0
##          quarter_released     0
##            month_released     0
##             week_released     0
##          weekday_released     0
```

Treating missing values

For simplicity sake and because most our variables have few missing values, we will use the mean and mode to treat most of our NAs. We will impute and predict the missing values for `budget` later as it has a lot of missing values.

```r
full_data_subset$runtime[is.na(full_data_subset$runtime)] <- mean(full_data_subset$runtime, na.rm = TRU
full_data_subset$number_of_cast[is.na(full_data_subset$number_of_cast)] <- mean(full_data_subset$number_
full_data_subset$number_of_crew[is.na(full_data_subset$number_of_crew)] <- mean(full_data_subset$number_
full_data_subset$tagline_length[is.na(full_data_subset$tagline_length)] <- mean(full_data_subset$taglin
full_data_subset$title_length[is.na(full_data_subset$title_length)] <- mean(full_data_subset$title_leng
full_data_subset$female_cast[is.na(full_data_subset$female_cast)] <- mean(full_data_subset$female_cast,
full_data_subset$male_cast[is.na(full_data_subset$male_cast)] <- mean(full_data_subset$male_cast, na.rm
full_data_subset$female_crew[is.na(full_data_subset$female_crew)] <- mean(full_data_subset$female_crew,
full_data_subset$male_crew[is.na(full_data_subset$male_crew)] <- mean(full_data_subset$male_crew, na.rm
full_data_subset$main_genre[is.na(full_data_subset$main_genre)] <- "Drama"
full_data_subset$number_of_genres[is.na(full_data_subset$number_of_genres)] <- 1
full_data_subset$number_of_prod_companies[is.na(full_data_subset$number_of_prod_companies)] <- 1
full_data_subset$number_of_keywords[is.na(full_data_subset$number_of_keywords)] <- 0
full_data_subset$prod_comp_id[is.na(full_data_subset$prod_comp_id)] <- 10000
```

Lets log transform our variables with skewed distributions.

```r
full_data_subset <- mutate(full_data_subset,
                           budget = log10(budget + 1),
                           year_released = log10(year_released),
                           popularity = log10(popularity + 1),
                           revenue = log10(revenue + 1))
```

For budget we have 2023 NAs. We will use a linear model to predict these values.

```r
# Create linear model to predict budget.
lm_budget <- lm(budget ~ number_of_cast + number_of_crew + year_released +
                   popularity + runtime + number_of_genres + prod_comp_id +
                   main_genre,
                data = full_data_subset, na.action = na.omit)

# Predict all NAs in budget with lm_budget.
full_data_subset$budget[is.na(full_data_subset$budget)] <- predict(lm_budget)
```

Final preparations

Lets create a last variable using budget divided by year of release. It might hold important information regarding the effect of year on budget.

```
full_data_subset$budget_year_ratio <- full_data_subset$budget/full_data_subset$year_released
```

Correct variable types.

```
full_data_subset <- full_data_subset %>% mutate_if(is.character, as.factor)
full_data_subset$weekday_released <- factor(full_data_subset$weekday_release, ordered = FALSE)
full_data_subset$month_released <- factor(full_data_subset$month_released, ordered = FALSE)
full_data_subset$quarter_released <- factor(full_data_subset$quarter_released)
```

Split `full_data_subset` back into train and test data sets.

```
train <- full_data_subset[1:3000,]
test <- full_data_subset[3001:7398,]
```

Machine Learning

We will now begin our machine learning using **Random Forest**.

Set a random seed for reproducability of results and create random forest model.

```
set.seed(222)

rf_model <- randomForest(revenue ~ .,
                         data = train,
                         ntree = 501,
                         replace = TRUE,
                         nodesize = 9,
                         importance = TRUE); print(rf_model)
```

```
##
## Call:
##  randomForest(formula = revenue ~ ., data = train, ntree = 501,      replace = TRUE, nodesize = 9, i
##                Type of random forest: regression
##                      Number of trees: 501
## No. of variables tried at each split: 9
##
##           Mean of squared residuals: 0.742666
##                     % Var explained: 57.26
```

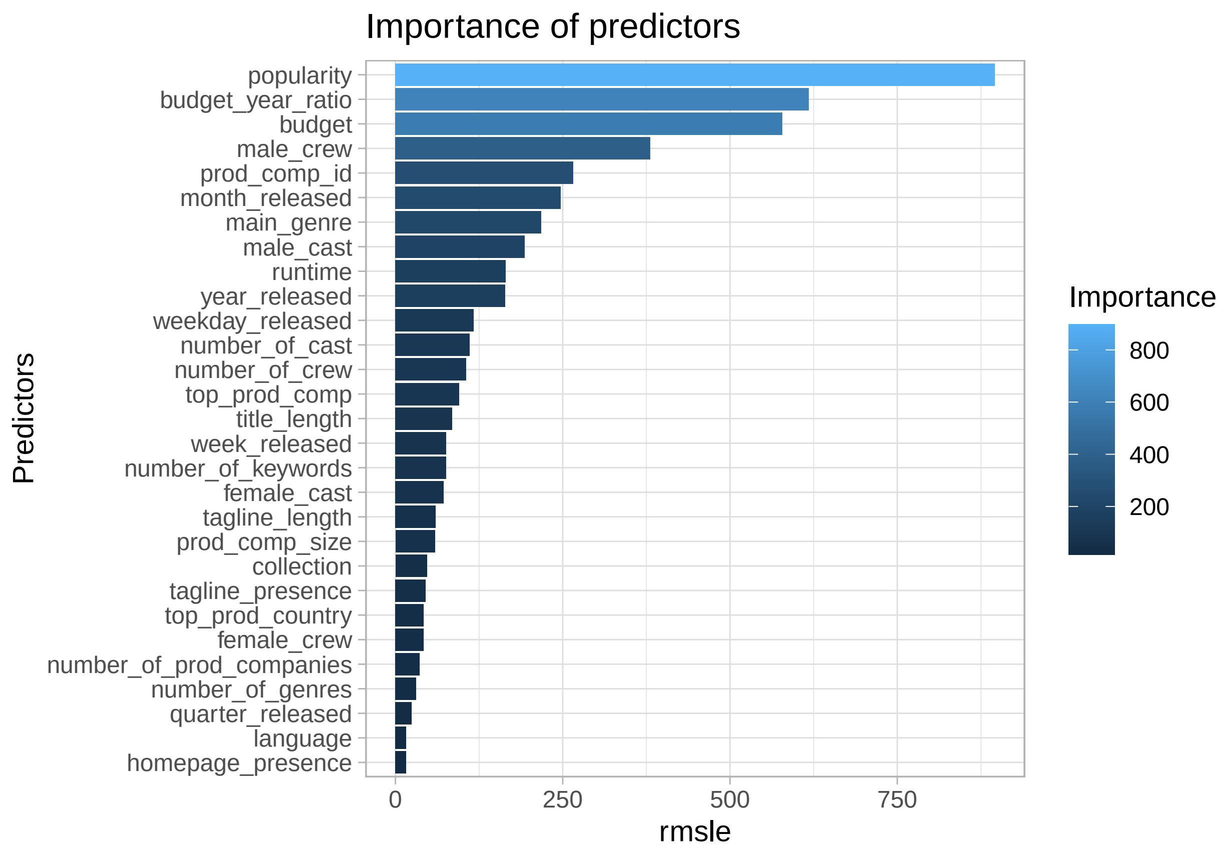
Here we can see our mean of squared residuals and the percentage variance explained by our model.

Importance of variables

Lets create a plot to visualize our variables by importance.

```
knitr::opts_chunk$set(echo = TRUE)
# Create an object for importance of variables
importance <- importance(rf_model)
```

```r
# Create data frame using importance.
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[,'IncNodePurity'], 0))

# Create interactive plot.
#ggplotly(
  ggplot(varImportance, aes(x = reorder(Variables, Importance),
                           y = Importance, fill = Importance)) +
       geom_bar(stat='identity') +
       labs(title = 'Importance of predictors', x = 'Predictors', y = 'rmsle') +
       coord_flip() +
       theme_light()#)
```



Here we can see our variables by importance. It looks like *popularity* and *budget* are the most important variables in our model followed by a slew of the variables that we created. This plot is interactive so run your mouse over the bars for more information about them.

Prediction

Now that we have created our model, we will use it to predict revenue on a test data set.

Run prediction with the test data.

```r
prediction <- predict(rf_model, test)
```

Save the solution to a data frame with two columns: id and revenue.

```r
solution <- data.frame(id = full_data[3001:7398,]$id, revenue = prediction)
```

Reverse the log transformation of revenue before exporting the solution.

```r
solution <- as_tibble(solution) %>%
   mutate(revenue = 10^revenue)
```

Lastly, lets write the solution to file.

```r
write.csv(solution, file = 'Box_office_prediction.csv', row.names = F)
```