TMDB Box Office Prediction: Group 6

Abhinav Pateria, Ravi Gunjan, Saurabh Kumar Gupta, Vipul Kumar, Yash Panchal

07/11/2019

Introduction

In this porject we will be trying to predict how much revenue a movie makes at the box office. During this process we will be going through:

1 Exploratory data analysis

2 Feature engineering

3 Treating missing values

4 Machine learning using random forest.

First lets load all the packages that we will need.

```
# install.packages("Metrics")

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(Metrics)
```

Read the Data

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.

```
#read.csv("myRandomFile.csv", header=TRUE)



train_data = read.csv(file.choose(), header=TRUE, na.strings=c("", '#N/A', '[]', '0'))
test_data = read.csv(file.choose(), header=TRUE, na.strings=c("", '#N/A', '[]', '0'))

#
# train_data = read.csv("train.csv", header=TRUE, na.strings=c("", '#N/A', '[]', '0'))
# test_data = read.csv("test.csv", header=TRUE, na.strings=c("", '#N/A', '[]', '0'))

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

We will take a glimpse of our data (combined data) and see how it looks.

```
glimpse(full_data)
```

```
## Observations: 7,398
## Variables: 23
## $ ï..id                 <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1...
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Time M...
## $ budget                <int> 14000000, 40000000, 3300000, 1200000, NA...
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{'id...
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whiplas...
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802", "...
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "en"...
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Princess ...
## $ overview              <chr> "When Lou, who has become the \"father o...
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.174936,...
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "/w9...
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id': 4...
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'United S...
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9/12...
## $ runtime               <int> 93, 113, 105, 122, 118, 83, 92, 84, 100,...
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'English'}...
## $ status                <chr> "Released", "Released", "Released", "Rel...
## $ tagline               <chr> "The Laws of Space and Time are About to...
## $ title                 <chr> "Hot Tub Time Machine 2", "The Princess ...
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}, {...
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', 'cr...
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02c8c...
## $ revenue               <int> 12314651, 95149435, 13092000, 16000000, ...
```

With the help of glimpse of data we can categorize data into two types. a) One which is less messier, requires little or no cleaning - i..id, budget, homepage, imdb_id, original_language, original_title, overview, popularity, poster_path, release_date, runtime, status, tagline, title, revenue. b) Attribute which look quite messy, we will extract appropriate information from them before using in our model - belongs_to_collection, genres, production_companies, production_countries, spoken_languages, Keywords, cast,crew.

1. Exploratory Data Analysis

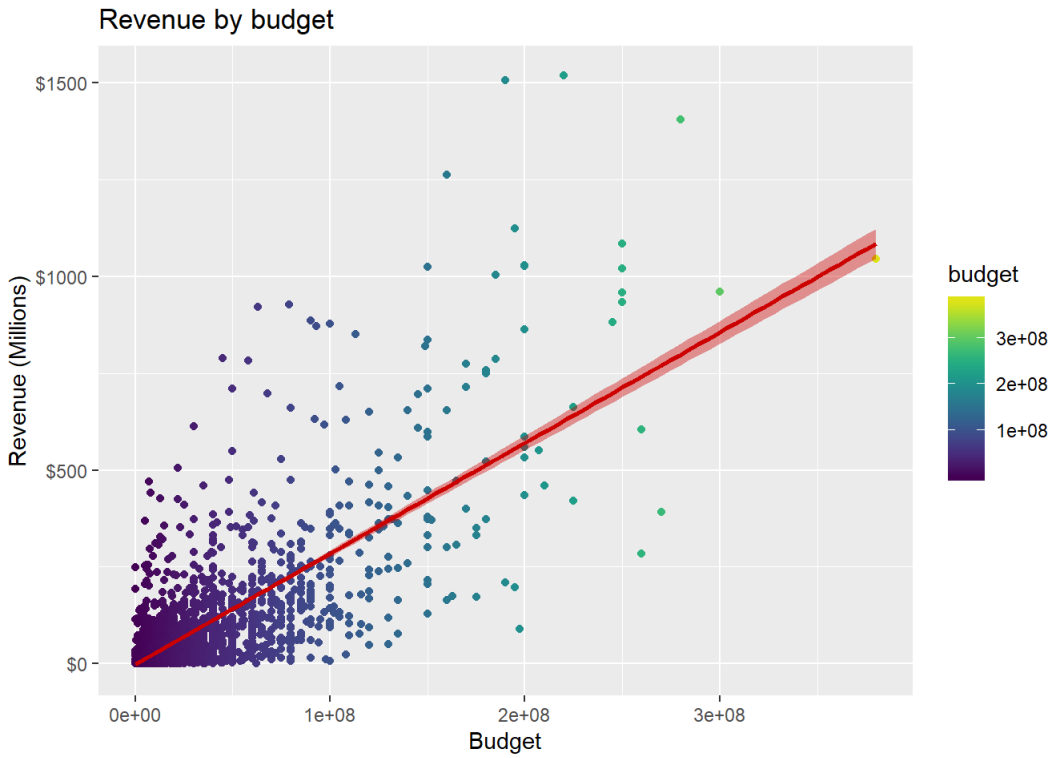
Lets begin by plotting our existing variables budget, runtime, and popularity in order to see their relation to the variable we are trying to predict, revenue.

Budget

```
ggplot(full_data[1:3000,], aes(x = budget, y = revenue, color = budget)) +
  geom_point() +

  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(labels = c('$0', '$500', '$1000', '$1500')) +
  labs(title = 'Revenue by budget', x = 'Budget', y = 'Revenue (Millions)')
```

```
## Warning: Removed 812 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 812 rows containing missing values (geom_point).
```

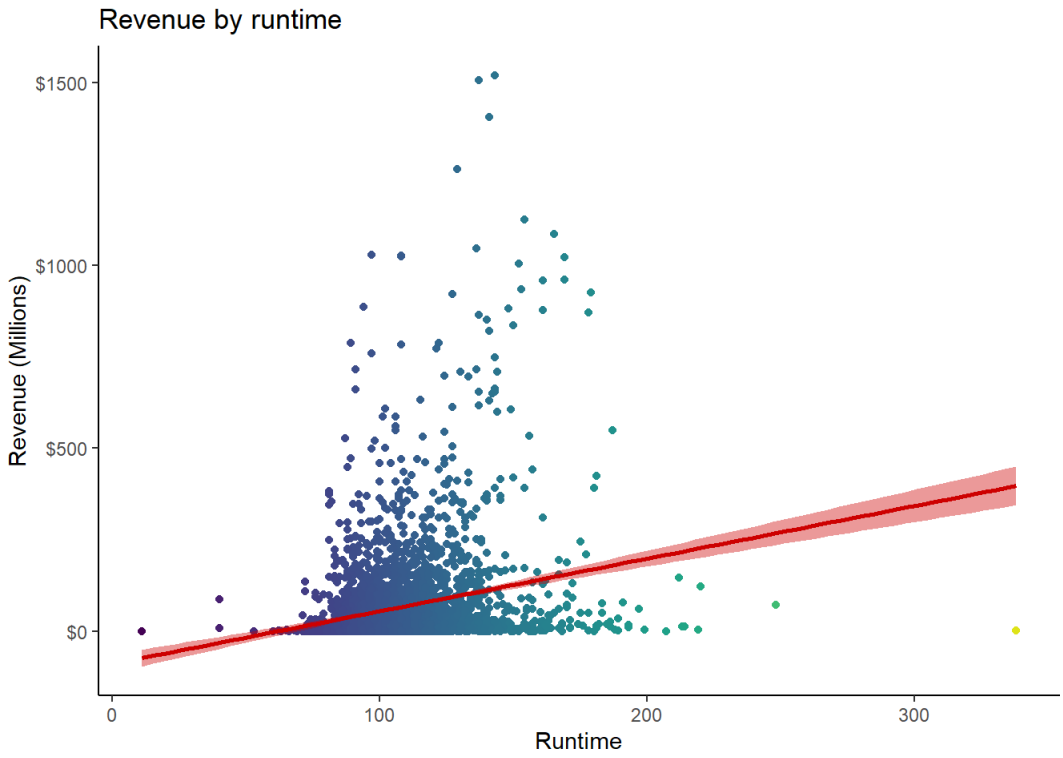


Runtime

```
ggplot(full_data[1:3000,], aes(x = runtime, y = revenue, color = runtime)) +
  geom_point() +
  # scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by runtime', x = 'Runtime', y = 'Revenue (Millions)')
```

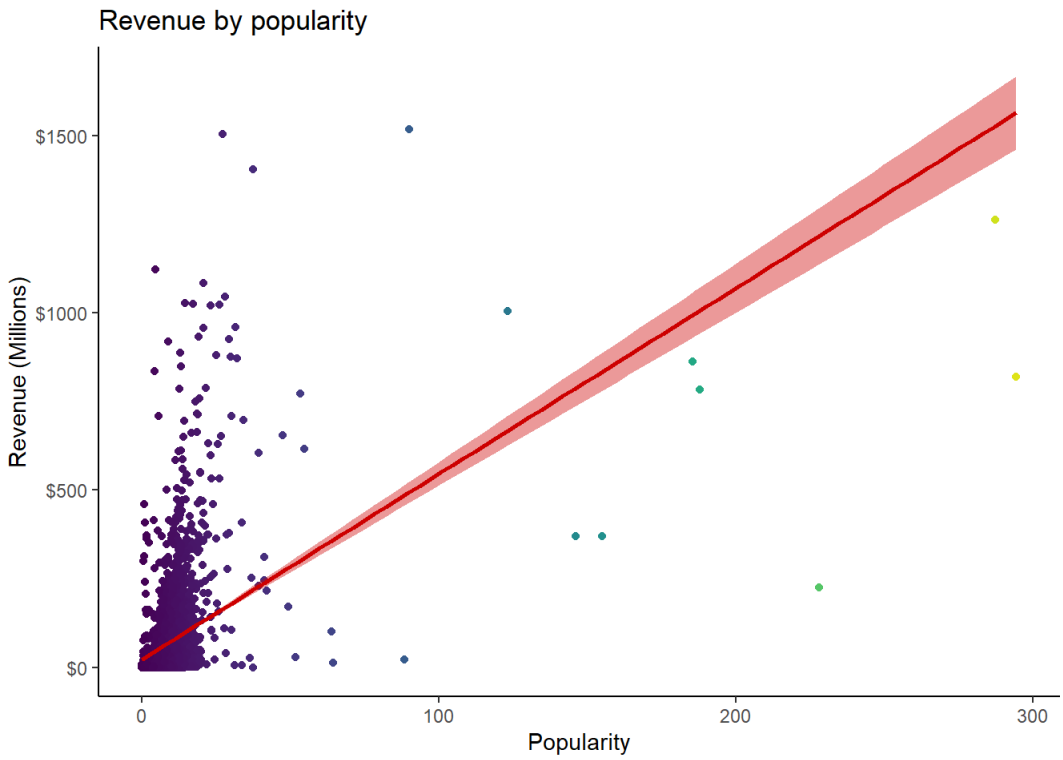
```
## Warning: Removed 14 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 14 rows containing missing values (geom_point).
```



Popularity

```
ggplot(full_data[1:3000,], aes(x = popularity, y = revenue, color = popularity)) +
  geom_point() +
  #  scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by popularity', x = 'Popularity', y = 'Revenue (Millions)')
```



From the above graphs we can see that an increase in the budget and the popularity tend to lead to higher revenue. Runtime also show an incresing trend with revenue but not as strong as budget and popularity.

Now, we will do feature engineering to create features for our machine learning algorithm.

2 Feature Engineering

Belongs to collection

Attribute belongs_to_collection are messy and unnecessary information is present which we do not need. To handle this, we will use regular expressions to extract the collection names from the strings in belongs_to_collection.

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

After the extraction of the collection names, we will check for the biggest collections.

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

```
## # A tibble: 409 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 PokÃ©mon 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
## # ... with 399 more rows
```

From the above table we can see that in each collection movie count is fairly small, so we will engineer a new variable that consist of either ‘being in a collection’ or ‘not being in a collection’.

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

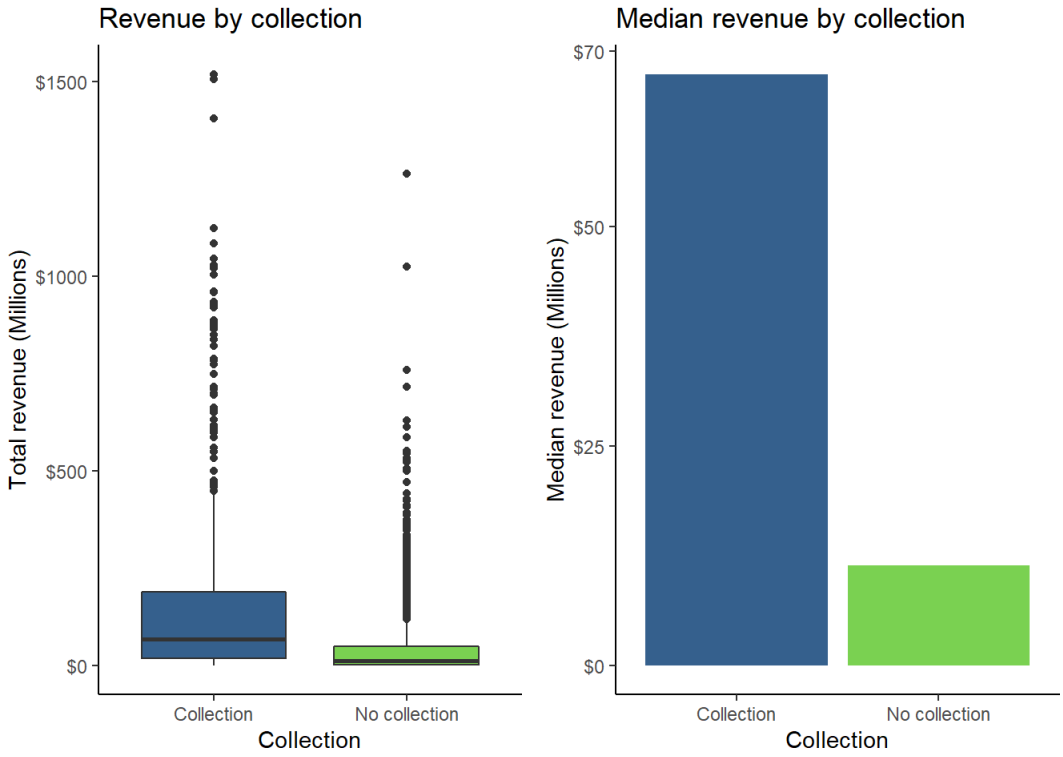
```
ggplot(data = full_data[1:3000,], aes(x = collection, y=revenue, fill=collection)) +
  # geom_boxplot(fill = c('grey50', 'red3')) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by collection', x = 'Collection', y = 'Total revenue (Millions)') -> p1




ggplot(data = full_data[1:3000,], aes(x = collection, y = revenue, fill = collection)) +
  #  stat_summary_bin(fun.y = median, geom = "bar", fill = c('grey50', 'red3')) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 25000000, 50000000, 70000000),
                     labels = c('$0', '$25', '$50', '$70')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median revenue by collection', x = 'Collection', y = 'Median revenue (Millions)') -> p2
```

Box plot of Collection

Bar plot of Collection

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

```
grid.arrange(p1, p2, ncol = 2)
```



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.

```
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)
```

Movie counts by main genre

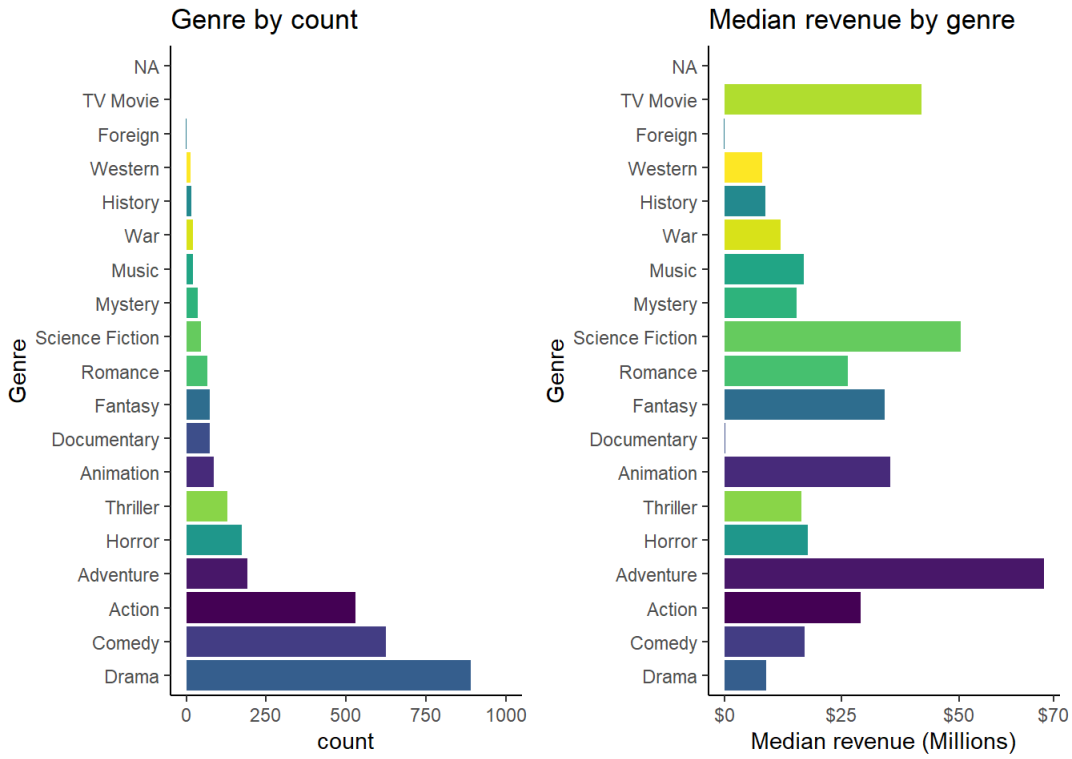
```
ggplot(full_data[1:3000,], aes(x = fct_infreq(main_genre), fill = main_genre)) +
  geom_bar() +
  #  scale_fill_grey() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  ylim(0, 1000) +
  coord_flip() +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Genre by count', x = 'Genre', y = 'count') -> p3
```

Median revenue by main genre

```
ggplot(full_data[1:3000,],  aes(x = fct_infreq(main_genre),  y=revenue, fill = main_genre)) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  #  scale_fill_grey() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  scale_y_continuous(breaks = c(0, 25000000, 50000000, 70000000),
                     labels = c('$0', '$25', '$50', '$70')) +
  coord_flip() +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median revenue by genre', x = 'Genre', y = 'Median revenue (Millions)') -> p4
```

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

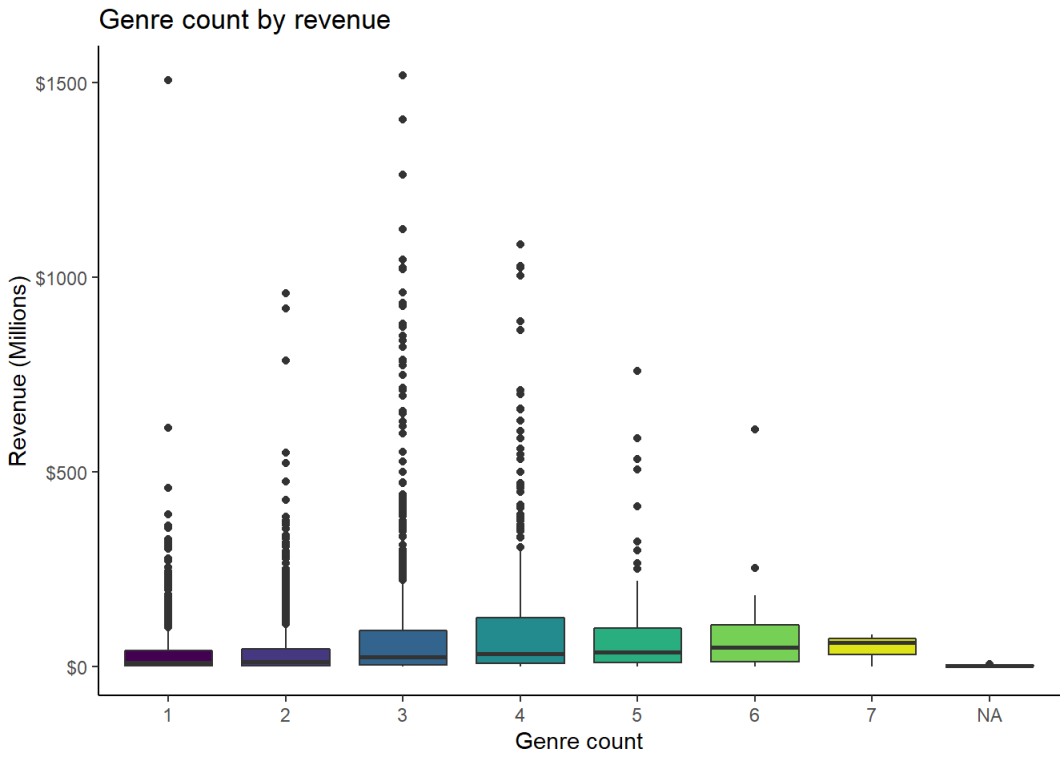
```
grid.arrange(p3, p4, ncol = 2)
```



Here we can see that different genres seem to be making different revenues. Adventure 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.

Genre Count

```
full_data$number_of_genres <- str_count(full_data$genres, 'name')

ggplot(full_data[1:3000,], aes(x = as.factor(number_of_genres), y = revenue,
                               fill=number_of_genres)) +
  geom_boxplot() +
  #  scale_fill_gradient(low = "grey40", high = "red3") +
  scale_fill_viridis(begin = 0, end = .95, option = 'D') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Genre count by revenue', x = 'Genre count', y = 'Revenue (Millions)')
```




```
full_data$number_of_genres[which(is.na(full_data$number_of_genres))]<-0
a<-full_data$revenue[1:3000]
b<-full_data$number_of_genres[1:3000]
cor(a,b)
```

```
## [1] 0.1636539
```

The trend in genre count increases from 1 to 4, reaches its peack and then shows a dreasing tendency. We can say that movies with 3-4-5-6 genre seems to generate a good revenue.

The correlation between Revenue and number of Genre a movie contains is 0.1636539.

Homepage

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

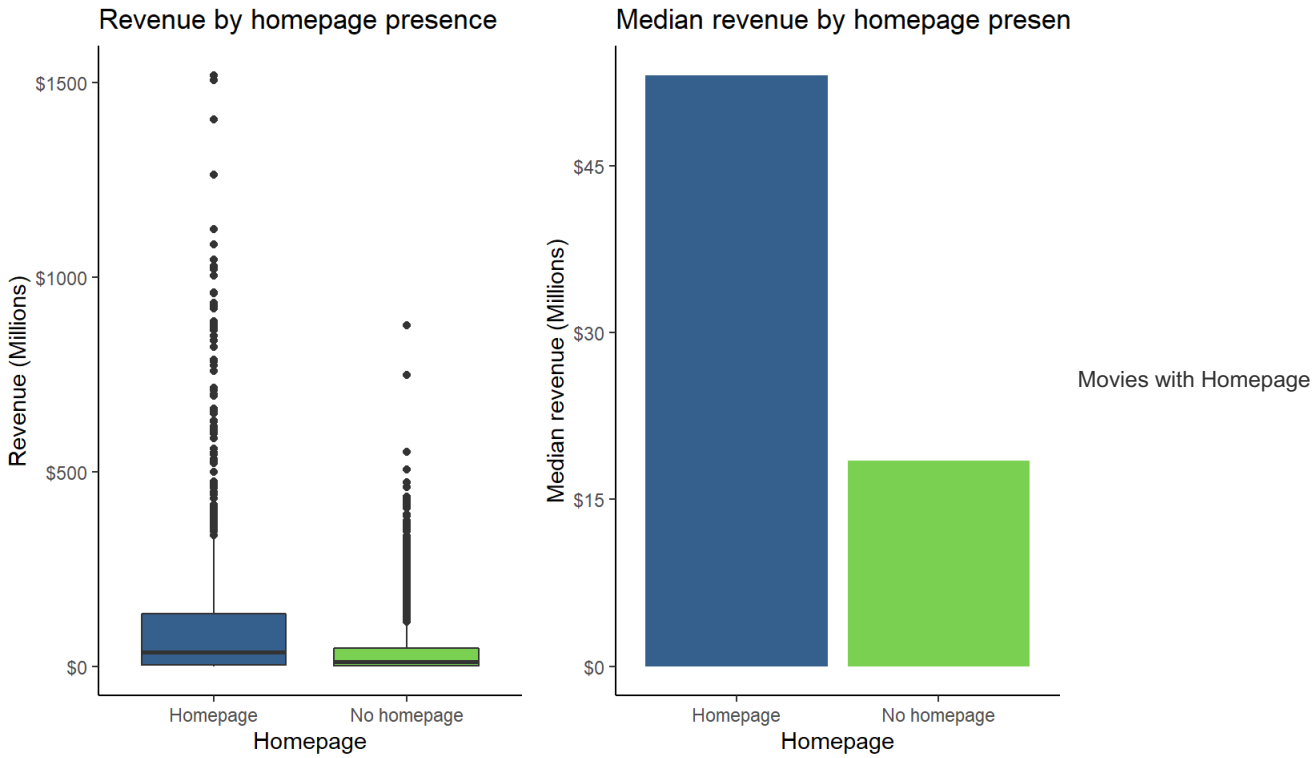
Box plot

```
ggplot(full_data[1:3000,], aes(x = homepage_presence, y=revenue, fill=homepage_presence)) +
  #  geom_boxplot(fill = c('grey50', 'red3')) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by homepage presence', x = 'Homepage', y = 'Revenue (Millions)') -> p5
```

Bar plot

```
ggplot(full_data[1:3000,], aes(x = homepage_presence, y=revenue, fill=homepage_presence)) +
  # stat_summary_bin(fun.y = median, geom = "bar", fill = c('grey50', 'red3')) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 10000000, 20000000, 30000000),
                     labels = c('$0', '$15', '$30', '$45')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title='Median revenue by homepage presence', x='Homepage', y='Median revenue (Millions)') -> p6
```

```
grid.arrange(p5, p6, ncol = 2)
```



generates higher revenue than movies with no Homepage. Movies with homepages seem to be making on average 3 times as much as movies without a homepage.

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.

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

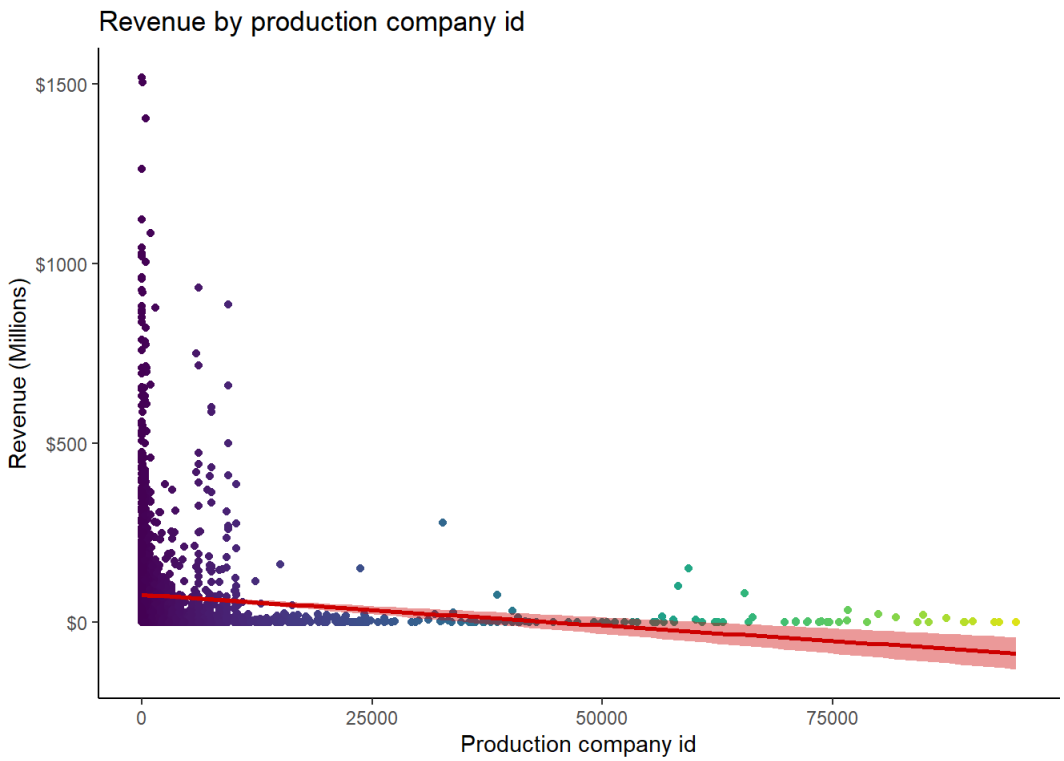
Create a scatter plot: revenue by prod_comp_id

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

```
ggplot(full_data[1:3000,], aes(x = prod_comp_id, y = revenue, color=prod_comp_id)) +
  geom_point() +
  # scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by production company id', x = 'Production company id', y = 'Revenue (Millions)')
```

```
## Warning: Removed 156 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 156 rows containing missing values (geom_point).
```



Correlation of revenue vs production company id

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

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

Production companies with lower numbered id's seem to be making more revenue compared to the ones with higher id's. There is small negative correlation present.

Top production companies

Lets extract the main production company name from production_companies and Check the top countries by number of movies produced.

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

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

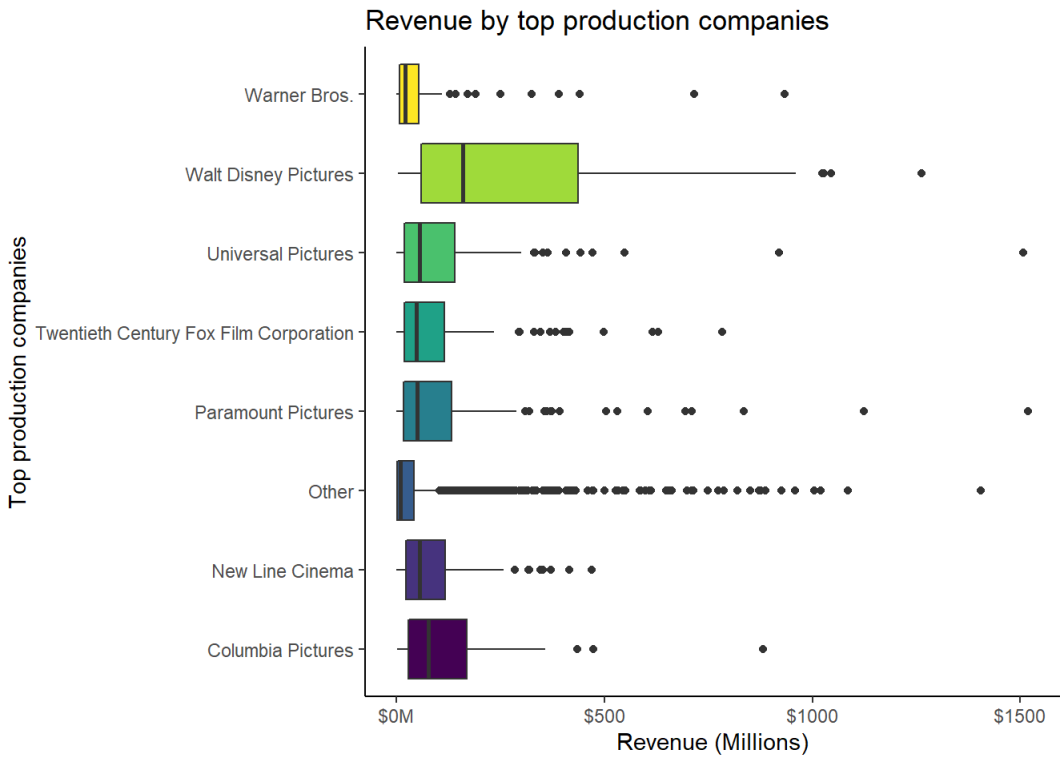
```
full_data$top_prod_comp[full_data$prod_comp_name=='Universal Pictures'] <- 'Universal Pictures'
full_data$top_prod_comp[full_data$prod_comp_name=='Paramount Pictures'] <- 'Paramount Pictures'
full_data$top_prod_comp[full_data$prod_comp_name=='Twentieth Century Fox Film Corporation'] <- 'Twentieth Ce
ntury Fox Film Corporation'
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'
```

Box plot of revenue by 'top_prod_comp'

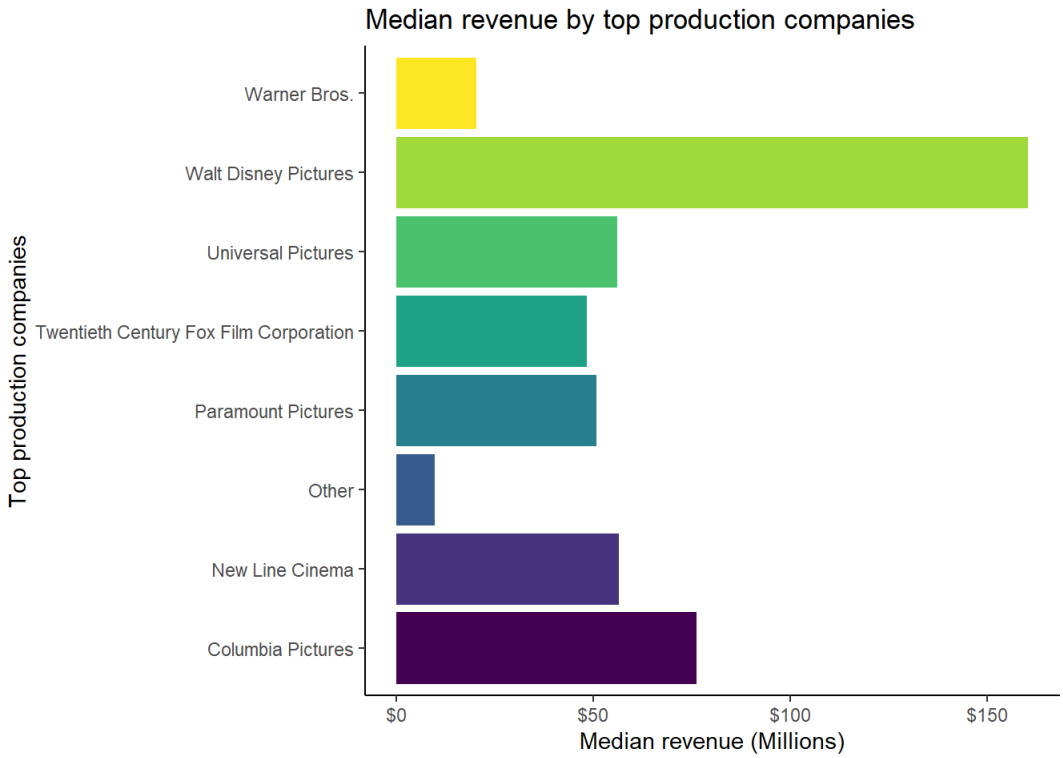
Plot our new top_prod_country variable.

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

```
ggplot(full_data[1:3000,], aes(x = top_prod_comp, y = revenue, fill=top_prod_comp)) +
  geom_boxplot() +
  #  scale_fill_brewer(palette = 'RdGy') +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0M', '$500', '$1000', '$1500')) +
  coord_flip() +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by top production companies',
       x = 'Top production companies', y = 'Revenue (Millions)')
```

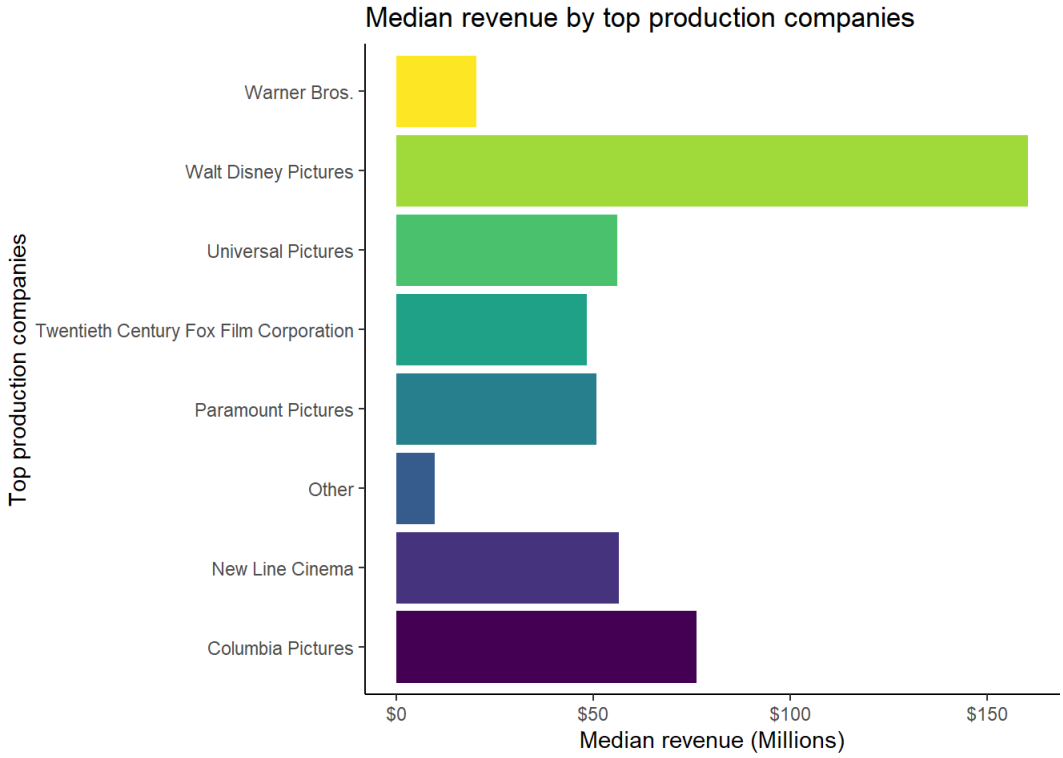



```
ggplot(full_data[1:3000,], aes(x = top_prod_comp, y = revenue, fill = top_prod_comp)) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  #  scale_fill_brewer(palette = 'RdGy') +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  scale_y_continuous(breaks = c(0, 50000000, 100000000, 150000000),
                     labels = c('$0', '$50', '$100','$150')) +
  coord_flip() +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median revenue by top production companies',
       x = 'Top production companies', y = 'Median revenue (Millions)')
```



Bar plot of median revenue by top_prod_comp

```
ggplot(full_data[1:3000,], aes(x = top_prod_comp, y = revenue, fill = top_prod_comp)) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  #  scale_fill_brewer(palette = 'RdGy') +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  scale_y_continuous(breaks = c(0, 50000000, 100000000, 150000000),
                     labels = c('$0', '$50', '$100','$150')) +
  coord_flip() +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median revenue by top production companies',
       x = 'Top production companies', y = 'Median revenue (Millions)')
```



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 produce
r'
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

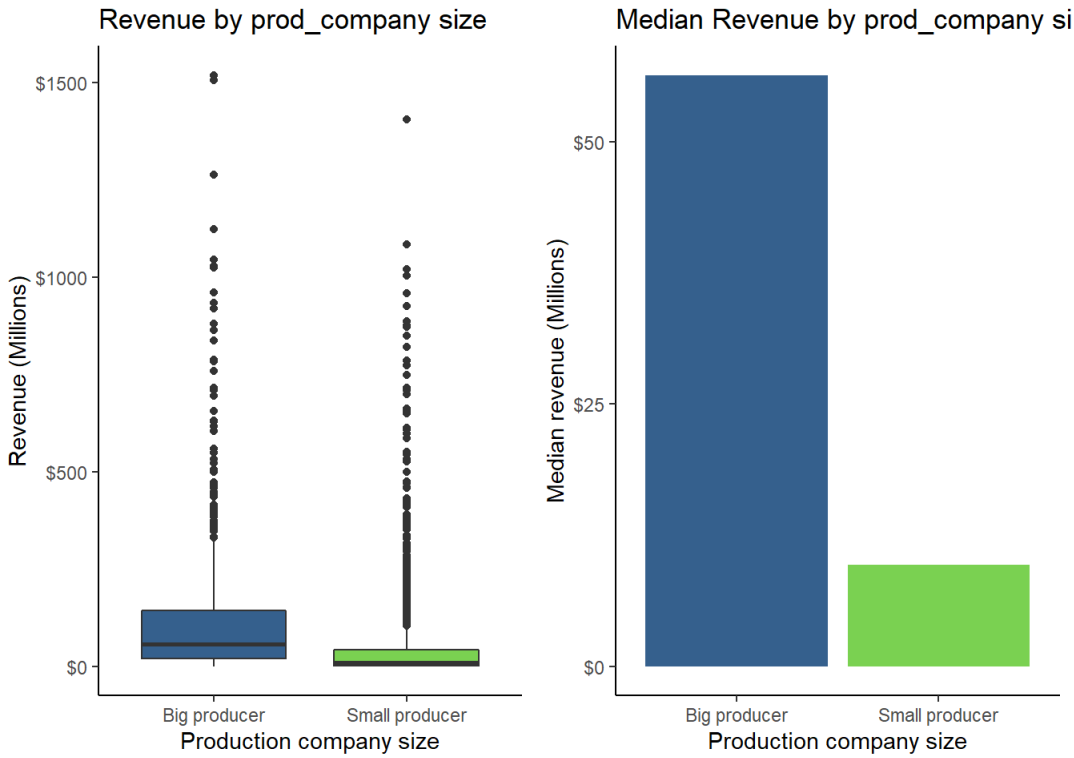
Box plot of revenue by ‘prod_comp_size’

```
ggplot(full_data[1:3000,], aes(x = prod_comp_size, y=revenue, fill=prod_comp_size)) +
  #  geom_boxplot(fill = c('grey50', 'red3')) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by prod_company size',
       x = 'Production company size', y = 'Revenue (Millions)') -> p7
```

Bar plot of median revenue by prod_comp_size

```
ggplot(full_data[1:3000,], aes(x = prod_comp_size, y=revenue, fill=prod_comp_size)) +
  #  stat_summary_bin(fun.y = median, geom = "bar", fill = c('grey50', 'red3')) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 25000000, 50000000, 70000000),
                     labels = c('$0', '$25', '$50', '$70')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median Revenue by prod_company size',
       x = 'Production company size', y = 'Median revenue (Millions)') -> p8
```

```
grid.arrange(p7, p8, ncol = 2)
```



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. Check the top countries by number of movies produced.

```
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
```

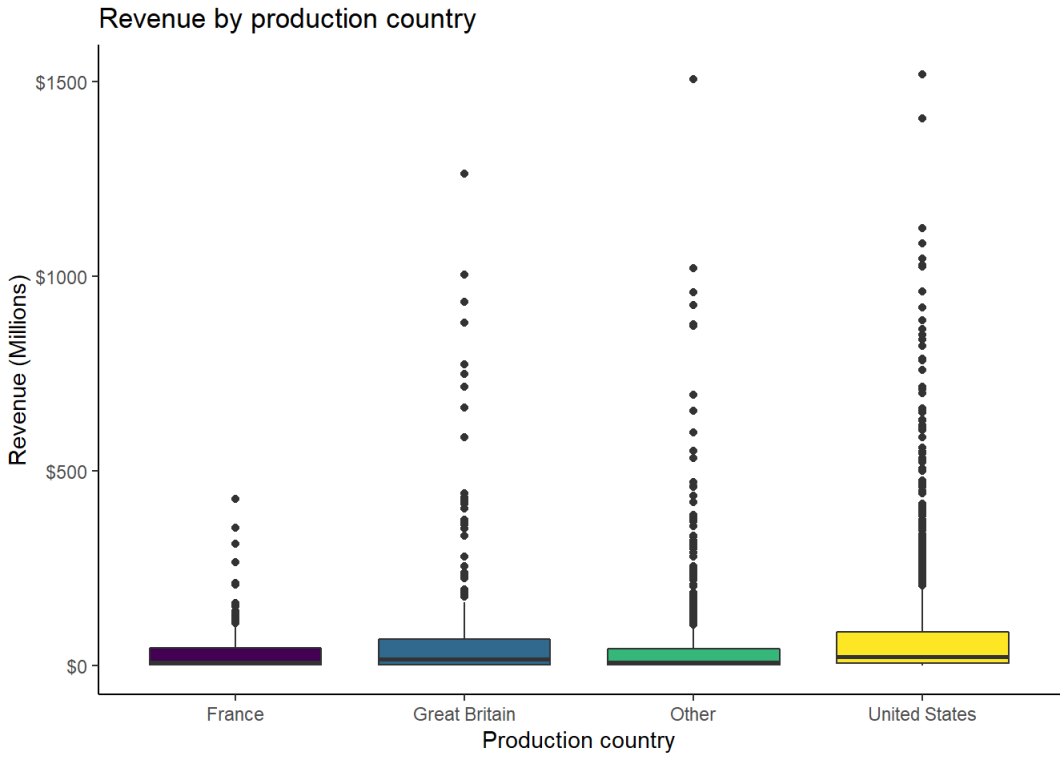
Separate into top production countries (criteria: 100+ movies) and 'other'.

```
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.

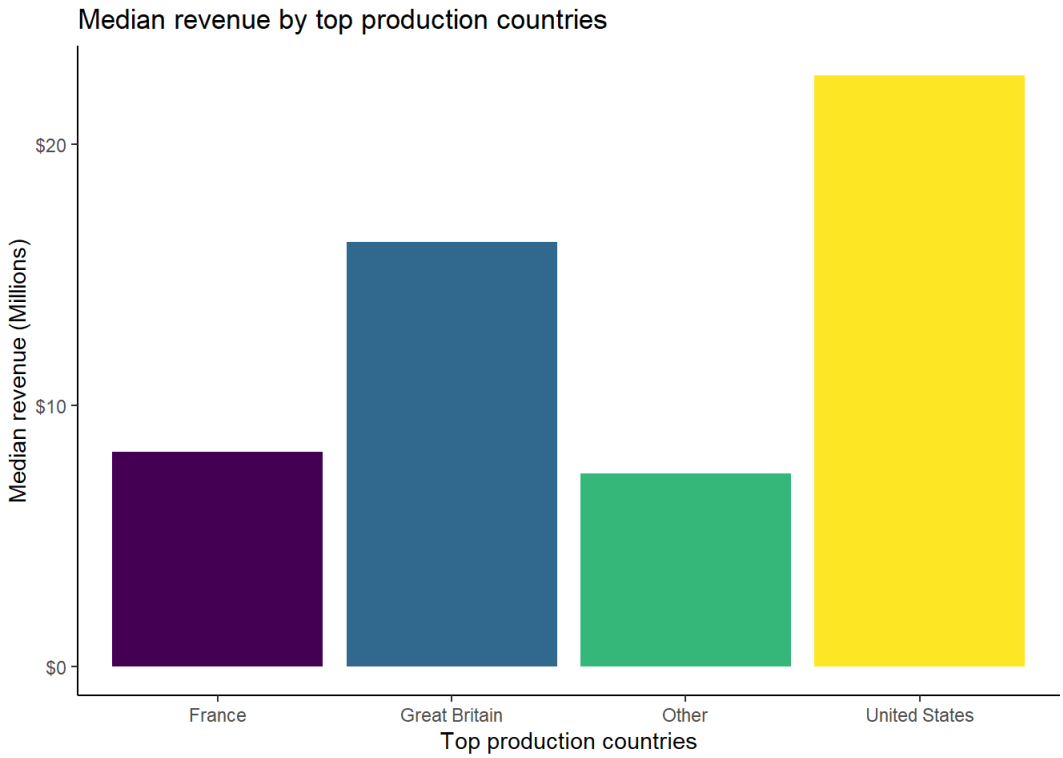
Box plot

```
ggplot(full_data[1:3000,], aes(x = top_prod_country, y=revenue, fill=top_prod_country)) +
  geom_boxplot() +
  #  scale_fill_brewer(palette = 'RdGy') +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title='Revenue by production country', x='Production country', y='Revenue (Millions)')
```



Bar plot

```
ggplot(full_data[1:3000,], aes(x=top_prod_country, y=revenue, fill=top_prod_country)) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  #  scale_fill_brewer(palette = 'RdGy') +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0, end = 1) +
  scale_y_continuous(breaks = c(0, 10000000, 20000000),
                     labels = c('$0', '$10', '$20')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median revenue by top production countries',
       x = 'Top production countries', y = 'Median revenue (Millions)')
```



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]+')
```

The format of the extracted value from imdb_id is in string. To create it's scatter plot we will parse the string values as in Integer.

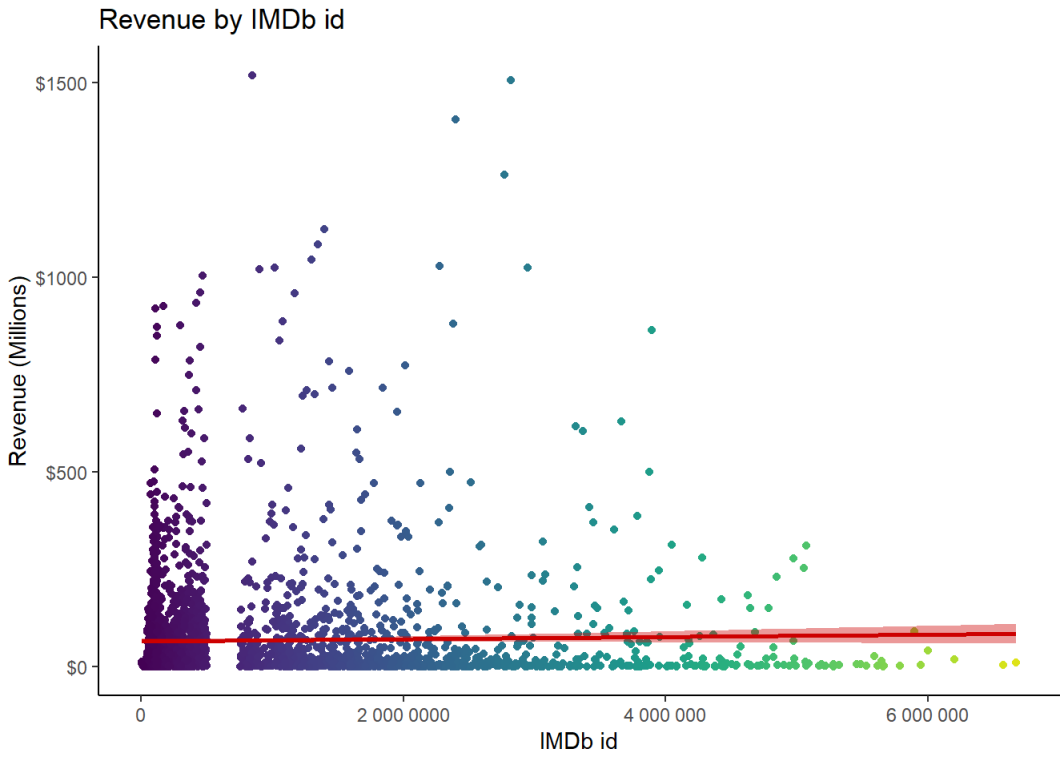
Set imdb_id_2 as an integer

```
full_data$imdb_id_2 <- as.integer(full_data$imdb_id_2 )
```

Plot this new variable.

Create scatter plot of revenue by imdb_id_2

```
ggplot(full_data[1:3000,], aes(x = imdb_id_2, y = revenue, color = imdb_id_2)) +
  geom_point() +
  #  scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  scale_x_continuous(breaks = c(0, 2000000, 4000000, 6000000),
                     labels = c('0', '2 000 0000', '4 000 000', '6 000 000')) +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by IMDb id', x = 'IMDb id', y = 'Revenue (Millions)')
```



Correlation

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

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

The correlation is very low (say threshold be 0.1. Here the correlation is less than 0.1) 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.

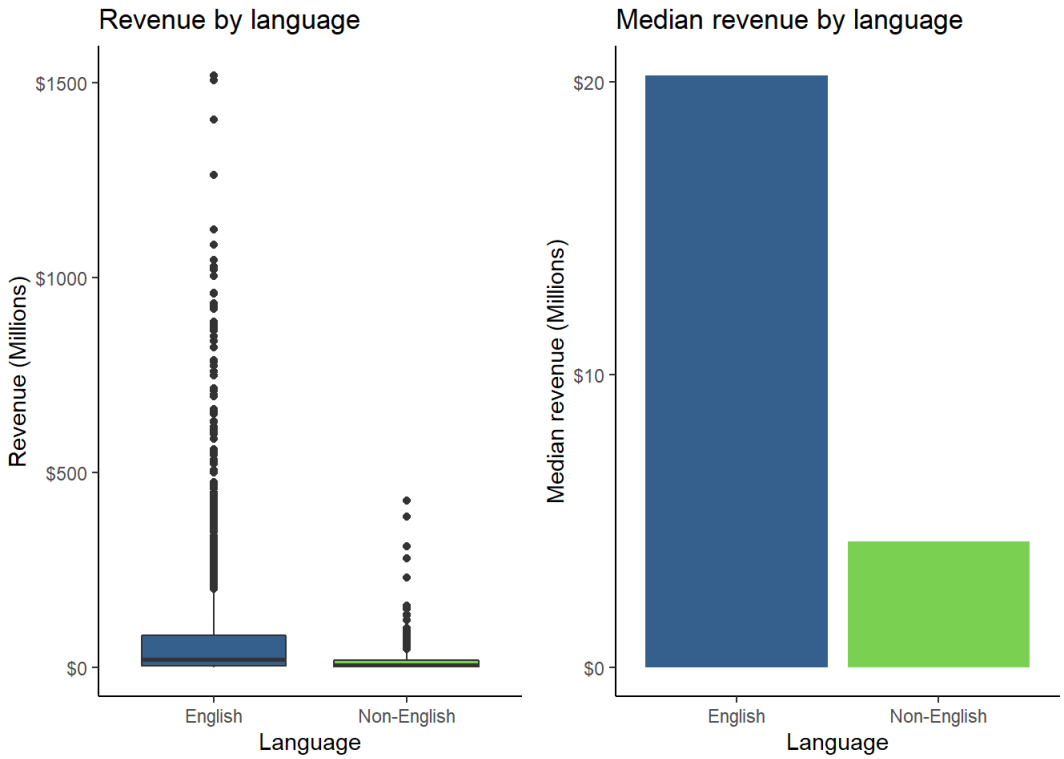
Box plot.

```
ggplot(full_data[1:3000,], aes(x = language, y = revenue, fill = language)) +
  #  geom_boxplot(fill = c('grey50', 'red3')) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by language', x='Language', y='Revenue (Millions)') -> p9
```

Bar plot.

```
ggplot(full_data[1:3000,], aes(x = language, y = revenue, fill=language)) +
  # stat_summary_bin(fun.y = median, geom = "bar", fill = c('grey50', 'red3')) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 10000000, 20000000),
                     labels = c('$0', '$10', '$20')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Median revenue by language', x = 'Language', y = 'Median revenue (Millions)') -> p10
```

```
grid.arrange(p9, p10, ncol = 2)
```



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

DATE

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

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

```
## [1] 3829
```

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

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

```
full_data$release_date[3829] <- '3/20/01'
full_data$release_date_mod <- parse_date_time2(full_data$release_date, "mdy",
                                               cutoff_2000 = 20)
full_data$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.
```

Year released

```
year_plot <- ggplot(full_data[1:3000,], aes(x = year_released, y = revenue,
                                            color=year_released)) +
  geom_point() +
  # scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_light() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by year released', x = 'Release year', y = 'Revenue (Millions)')
```

Quarter released

```
quarter_plot <- ggplot(full_data[1:3000,], aes(x = factor(quarter_released),
                                               y = revenue, fill = factor(quarter_released))) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  # scale_fill_grey() +
  scale_fill_viridis(begin = 0, end = .95, option = 'D', discrete = TRUE) +
  scale_y_continuous(breaks = c(0, 10000000, 20000000),
                     labels = c('$0', '$10', '$20')) +
  theme_light() +
  theme(legend.position = 'none', axis.text.x = element_text(angle = 90)) +
  labs(title='Revenue by quarter released', x='Release quarter', y='Median revenue (Millions)')
```

Month released

```
month_plot <- ggplot(full_data[1:3000,], aes(x = month_released, y = revenue,
                                             fill = month_released)) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  # scale_fill_grey() +
  scale_fill_viridis(begin = 0, end = .95, option = 'D', discrete = TRUE) +
  scale_y_continuous(breaks = c(0, 10000000, 20000000, 30000000),
                     labels = c('$0', '$10', '$20', '$30')) +
  theme_light() +
  theme(legend.position = 'none', axis.text.x = element_text(angle = 10)) +
  labs(title='Median revenue by month released', x='Release month', y='Median revenue (Millions)')
```

Week released

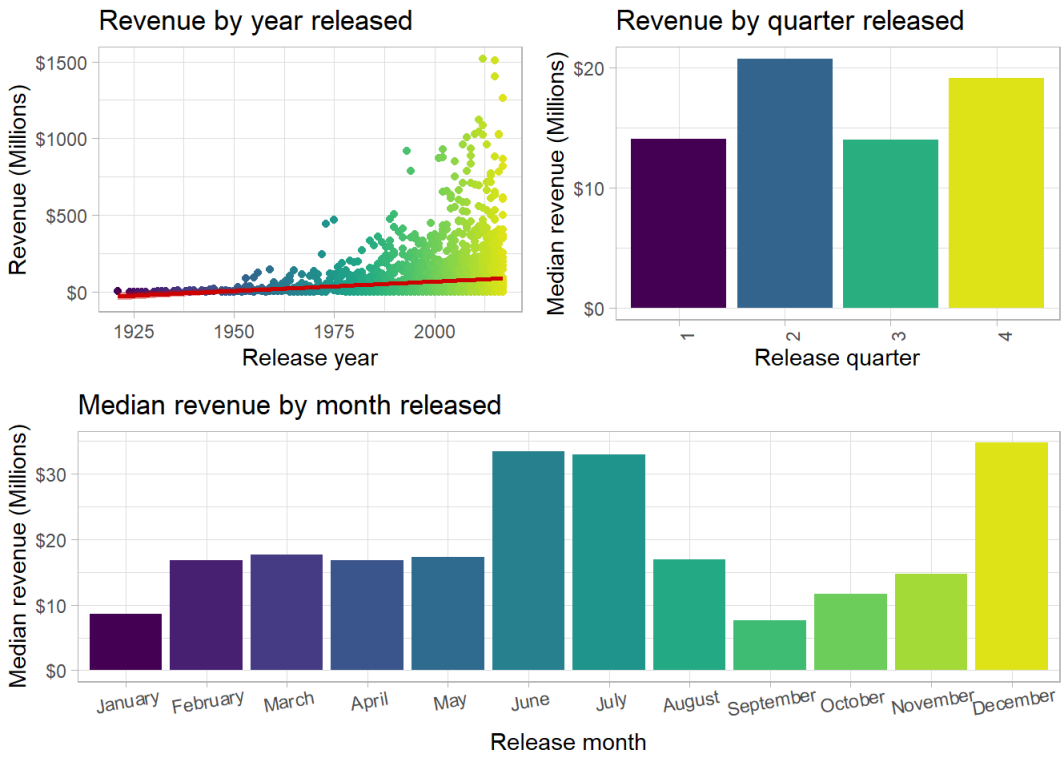
```
week_plot <- ggplot(full_data[1:3000,], aes(x = factor(week_released),
                                            y = revenue, fill = factor(week_released))) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  #  scale_fill_grey() +
  scale_fill_viridis(begin = 0, end = .95, option = 'D', discrete = TRUE) +
  scale_y_continuous(breaks = c(0, 20000000, 40000000, 60000000),
                     labels = c('$0', '$20', '$40', '$60')) +
  theme_light() +
  theme(legend.position = 'none', axis.text.x = element_text(angle = 90)) +
  labs(title='Revenue by week released', x='Release week', y='Median revenue (Millions)')
```

Weekday released

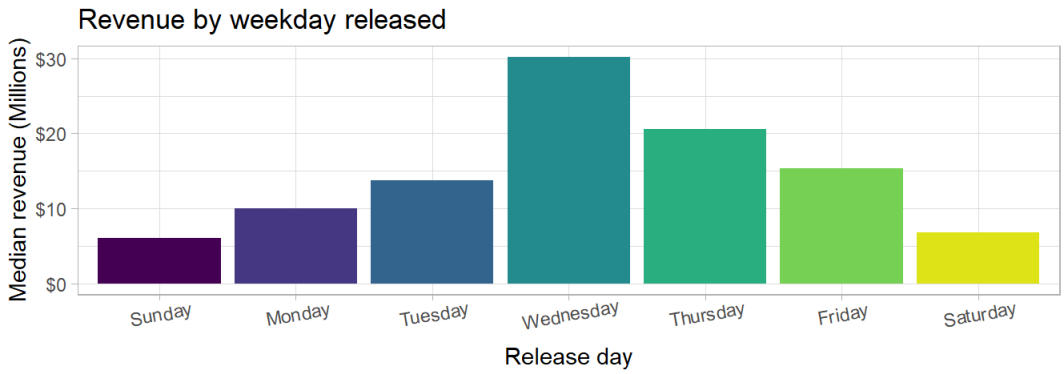
```
weekday_plot <- ggplot(full_data[1:3000,], aes(x = weekday_released, y = revenue,
                                               fill = weekday_released)) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  #  scale_fill_grey() +
  scale_fill_viridis(begin = 0, end = .95, option = 'D', discrete = TRUE) +
  scale_y_continuous(breaks = c(0, 10000000, 20000000, 30000000),
                     labels = c('$0', '$10', '$20', '$30')) +
  theme_light() +
  theme(legend.position = 'none', axis.text.x = element_text(angle = 10)) +
  labs(title = 'Revenue by weekday released', x='Release day', y='Median revenue (Millions)')
```

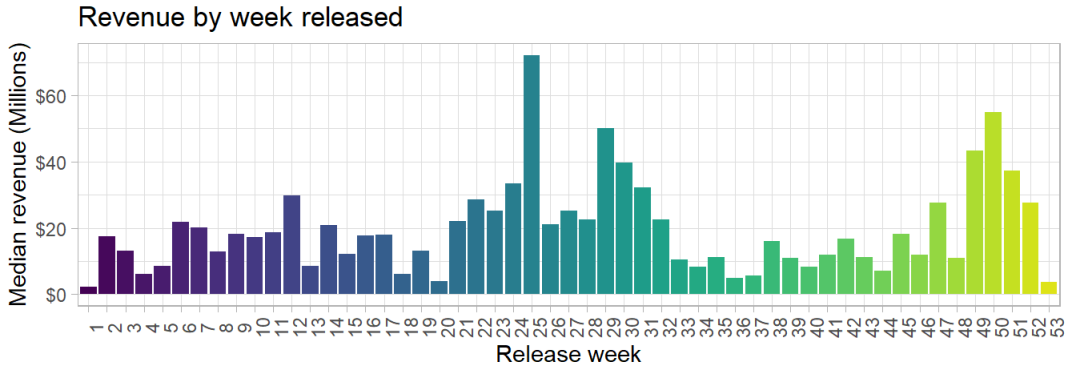
Create a grid of the plots.

```
grid.arrange(year_plot, quarter_plot, month_plot,
             layout_matrix = rbind(c(1, 2),
                                   c(3)))
```




```
grid.arrange(weekday_plot, week_plot,
             layout_matrix = rbind(c(1),
                                   c(2)))
```





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.

Gender of cast & crew

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

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'))
```

Revenue by number_of_cast

```
ggplot(full_data[1:3000,], aes(x = number_of_cast, y = revenue, color = number_of_cast)) +
  geom_point() +
  scale_color_gradient(low = "grey10", high = "grey75") +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by cast count', x = 'Cast count', y = 'Revenue (Millions)') -> n1
```

Revenue by female_cast

```
ggplot(full_data[1:3000,], aes(x = female_cast, y = revenue, color=female_cast)) +
  geom_point() +
  #  scale_color_gradient(low = "brown4", high = "red") +
  scale_color_viridis(begin = 0.3, end = .95, option = 'A') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by female cast count', x = 'Female cast count', y = 'Revenue (Millions)') -> n2
```

Revenue by male_cast

```
ggplot(full_data[1:3000,], aes(x = male_cast, y = revenue, color = male_cast)) +
  geom_point() +
  #  scale_color_gradient(low = "midnightblue", high = "aquamarine") +
  scale_color_viridis(begin = 0.2, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by male cast count', x = 'Male cast count', y = 'Revenue (Millions)') -> n3
```

Revenue by number_of_crew

```
ggplot(full_data[1:3000,], aes(x = number_of_crew, y = revenue, color = number_of_crew)) +
  geom_point() +
  scale_color_gradient(low = "grey10", high = "grey75") +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by Crew count', x = 'Crew count', y = 'Revenue (Millions)') -> n4
```

Revenue by female_crew

```
ggplot(full_data[1:3000,], aes(x = female_crew, y = revenue, color=female_crew)) +
  geom_point() +
  #  scale_color_gradient(low = "brown4", high = "red") +
  scale_color_viridis(begin = 0.3, end = .95, option = 'A') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by female crew count', x = 'Female crew count', y = 'Revenue (Millions)') -> n5
```

Revenue by male_crew

```
ggplot(full_data[1:3000,], aes(x = male_crew, y = revenue, color = male_crew)) +
  geom_point() +
  #  scale_color_gradient(low = "midnightblue", high = "aquamarine") +
  scale_color_viridis(begin = 0.2, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by male crew count', x = 'Male crew count', y = 'Revenue (Millions)') -> n6
```

```
grid.arrange(n1, n2, n3, n4,
             layout_matrix = rbind(c(1, 2),
                                   c(3, 4)))
```

```
## Warning: Removed 26 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 26 rows containing missing values (geom_point).
```

```
## Warning: Removed 26 rows containing non-finite values (stat_smooth).
```

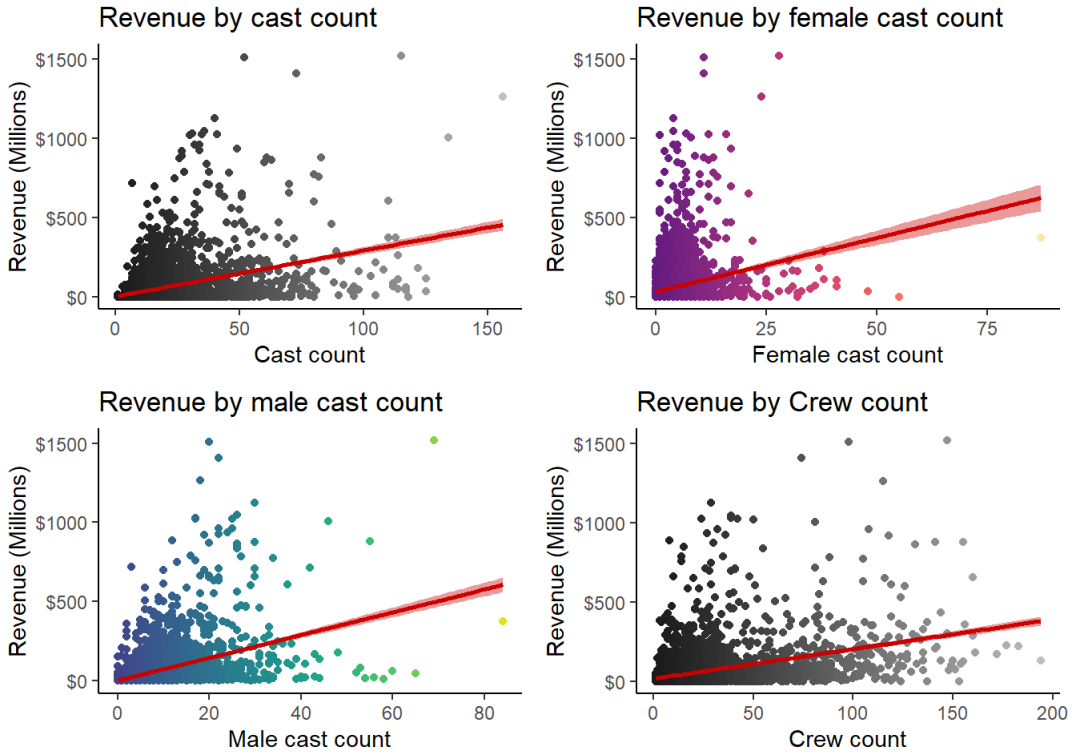
```
## Warning: Removed 26 rows containing missing values (geom_point).
```

```
## Warning: Removed 26 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 26 rows containing missing values (geom_point).
```

```
## Warning: Removed 16 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 16 rows containing missing values (geom_point).
```



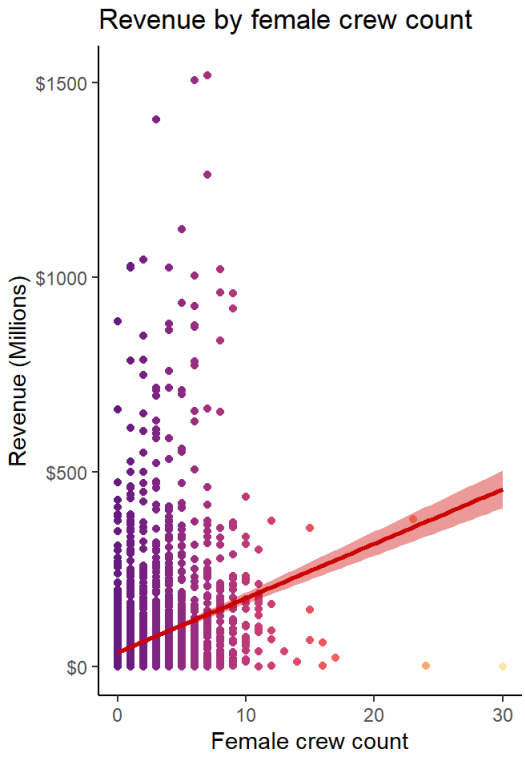

```
grid.arrange(n5, n6,
             layout_matrix = rbind(c(1, 2)))
```

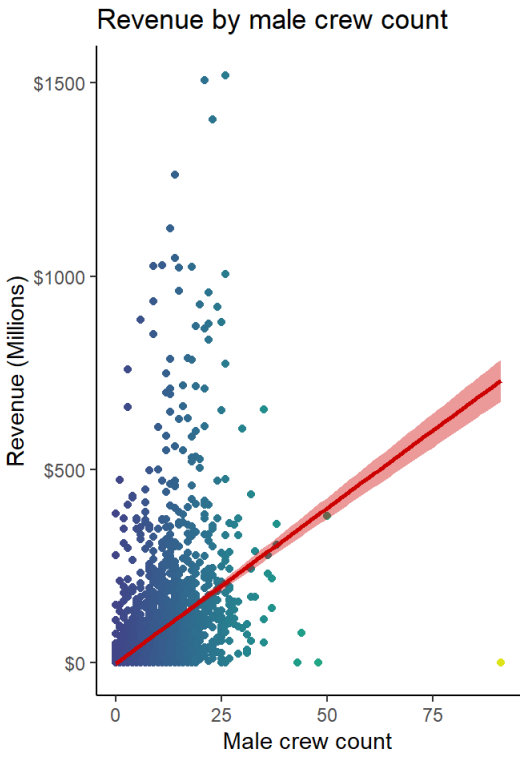
```
## Warning: Removed 16 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 16 rows containing missing values (geom_point).
```

```
## Warning: Removed 16 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 16 rows containing missing values (geom_point).
```





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…

```
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')
```

number_of_genres

```
ggplot(full_data[1:3000,], aes(x = as.factor(number_of_genres), y = revenue,
                               fill=number_of_genres)) +
  geom_boxplot() +
  #  scale_fill_gradient(low = "grey40", high = "red3") +
  scale_fill_viridis(begin = 0, end = .95, option = 'D') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Genre count by revenue', x = 'Genre count', y = 'Revenue (Millions)') -> num1
```

number_of_prod_companies

```
ggplot(full_data[1:3000,],  aes(x = as.factor(number_of_prod_companies),
                                y = revenue, fill = number_of_prod_companies)) +
  geom_boxplot() +
  #  scale_fill_gradient(low = "grey40", high = "red3") +
  scale_fill_viridis(begin = 0, end = .95, option = 'D') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title='Prod. company count by revenue', x='Production company count', y='Revenue (Millions)') -> num2
```

number_of_prod_countries

```
ggplot(full_data[1:3000,], aes(x = as.factor(number_of_prod_countries), y=revenue,
                               fill = number_of_prod_countries)) +
  geom_boxplot() +
  #  scale_fill_gradient(low = "grey40", high = "red3") +
  scale_fill_viridis(begin = 0, end = .95, option = 'D') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title='Prod. country count by revenue', x='Production country count', y='Revenue (Millions)') -> num3
```

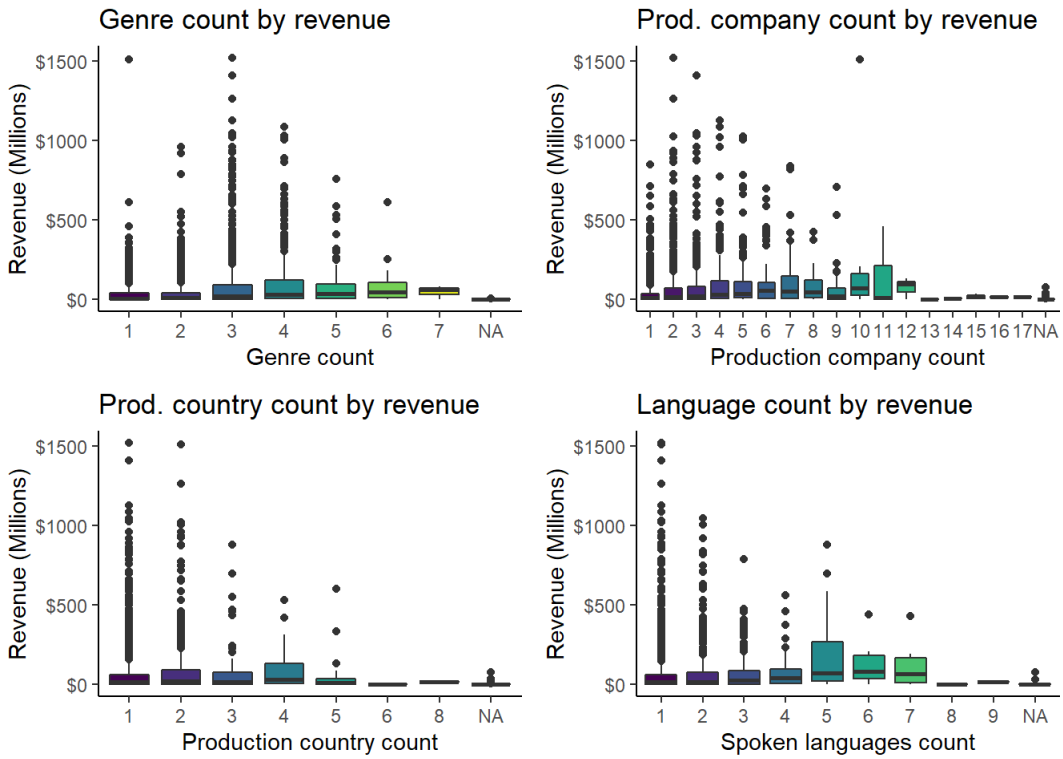
number_of_spoken_languages

```
ggplot(full_data[1:3000,], aes(x = as.factor(number_of_spoken_languages),
                               y = revenue, fill = number_of_spoken_languages)) +
  geom_boxplot() +
  # scale_fill_gradient(low = "grey40", high = "red3") +
  scale_fill_viridis(begin = 0, end = .95, option = 'D') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title='Language count by revenue', x='Spoken languages count', y='Revenue (Millions)') -> num4
```

number_of_keywords

```
ggplot(full_data[1:3000,], aes(x = number_of_keywords, y=revenue, color=number_of_keywords)) +
  geom_point() +
  #  scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by keyword count', x = 'Keyword count', y = 'Revenue (Millions)') -> num5
```

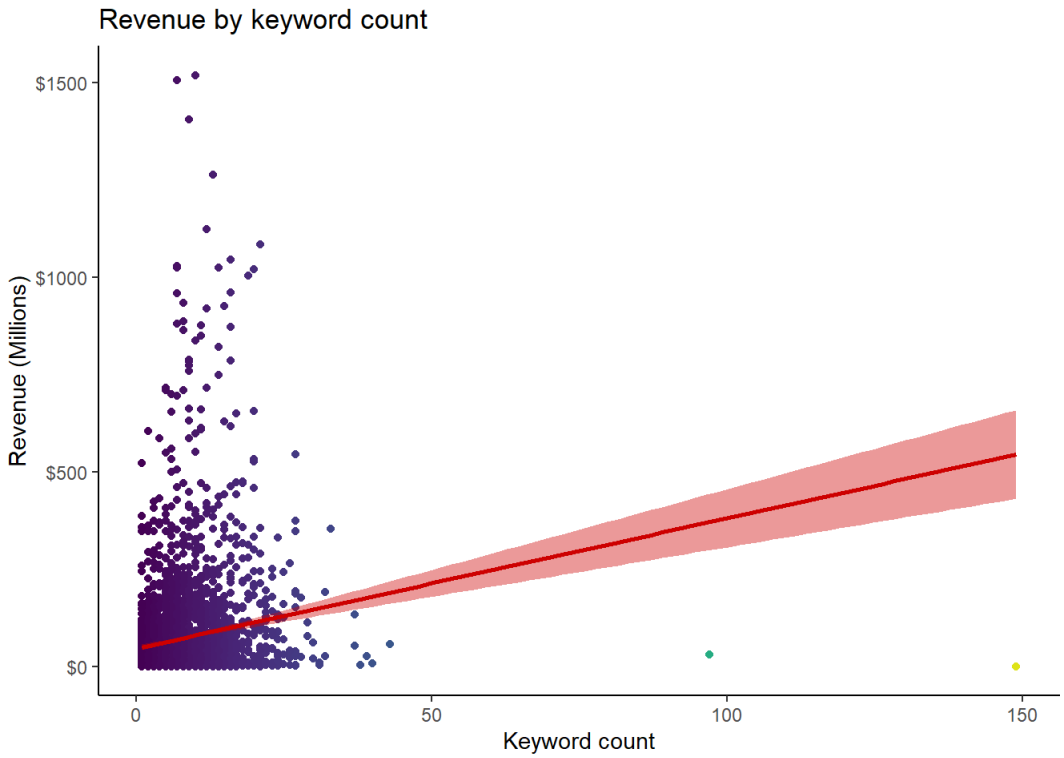
```
grid.arrange(num1, num2, num3, num4,
             layout_matrix = rbind(c(1, 2),
                                   c(3,4)))
```



```
grid.arrange(num5,
             layout_matrix = rbind(c(1)))
```

```
## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 276 rows containing missing values (geom_point).
```

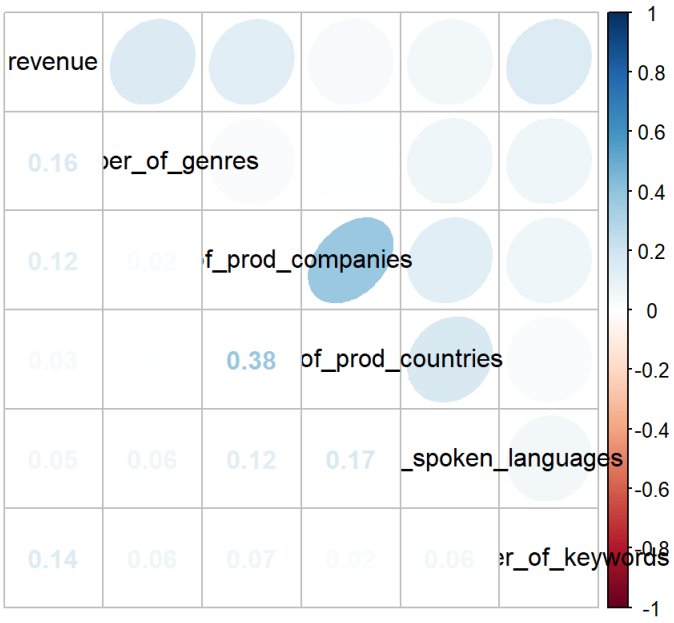


Here we can see that:

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.

```
corrplot.mixed(
  corr = cor(full_data[c('revenue','number_of_genres', 'number_of_prod_companies', 'number_of_prod_countries
', 'number_of_spoken_languages', 'number_of_keywords')], use = 'complete.obs'),
  tl.col = "black",
  upper = "ellipse")
```



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.

Tagline presence

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

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

Box plot

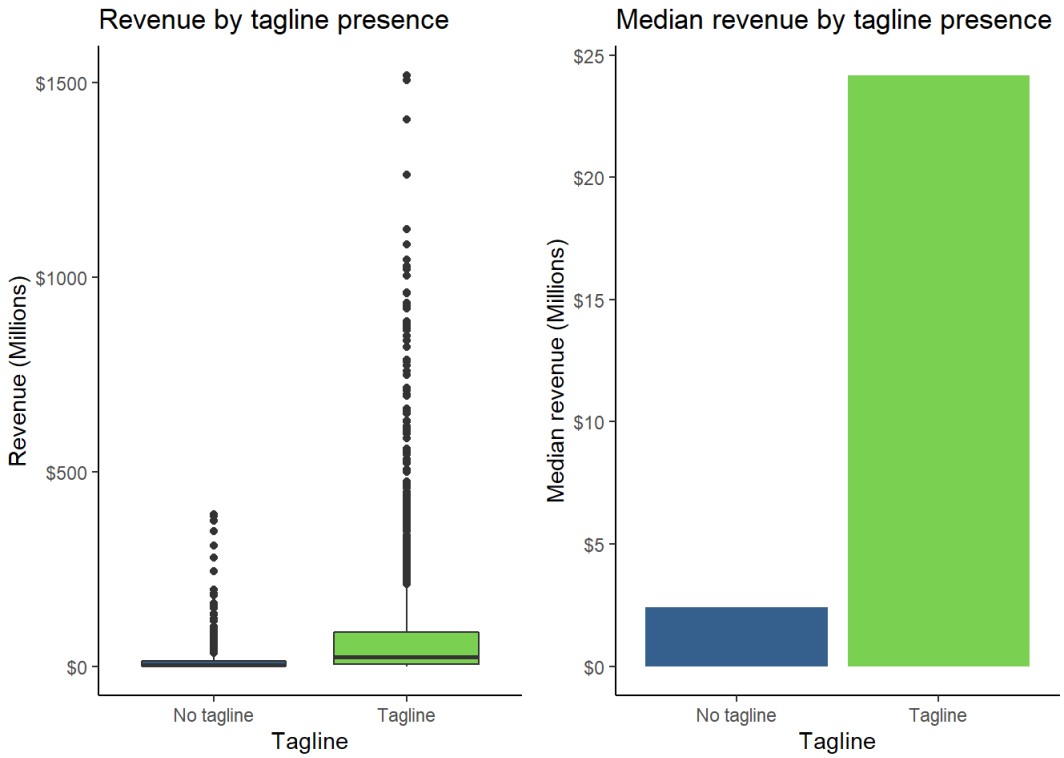
```
ggplot(full_data[1:3000,], aes(x = tagline_presence, y = revenue, fill = tagline_presence)) +
  #  geom_boxplot(fill = c('grey50', 'red3')) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by tagline presence', x = 'Tagline', y = 'Revenue (Millions)') -> tagPlot1
```

Bar plot

```
ggplot(full_data[1:3000,], aes(x = tagline_presence, y=revenue, fill=tagline_presence)) +
  #  stat_summary_bin(fun.y = median, geom = "bar", fill = c('grey50', 'red3')) +
  stat_summary_bin(fun.y = median, geom = "bar") +
  scale_fill_viridis(discrete = TRUE, option = 'D', begin = 0.3, end = .8) +
  scale_y_continuous(breaks = c(0,5000000, 10000000, 15000000, 20000000, 25000000),
                     labels = c('$0', '$5', '$10', '$15', '$20', '$25')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title='Median revenue by tagline presence', x='Tagline', y='Median revenue (Millions)') -> tagPlot2
```

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

```
grid.arrange(tagPlot1, tagPlot2,
             layout_matrix = rbind(c(1,2)))
```



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

Length of title_length, overview_length and tagline

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 # title_length

```
ggplot(full_data[1:3000,], aes(x = title_length, y = revenue, color = title_length)) +
  geom_point() +
  # scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by title length', x = 'Title length', y = 'Revenue (Millions)') -> length_title
```

tagline_length

```
ggplot(full_data[1:3000,], aes(x=tagline_length, y=revenue, color=tagline_length)) +
  geom_point() +
  #  scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by tagline length', x = 'Tagline length', y = 'Revenue (Millions)') -> length_taglin
e
```

overview_length

```
ggplot(full_data[1:3000,], aes(x=overview_length, y=revenue, color=overview_length))+
  geom_point() +
  #  scale_color_gradient(low = "grey10", high = "grey75") +
  scale_color_viridis(begin = 0, end = .95, option = 'D') +
  geom_smooth(method = 'lm', color = 'red3', fill = 'red3') +
  scale_y_continuous(breaks = c(0, 500000000, 1000000000, 1500000000),
                     labels = c('$0', '$500', '$1000', '$1500')) +
  theme_classic() +
  theme(legend.position = 'none') +
  labs(title = 'Revenue by overview length', x = 'Overview length', y = 'Revenue (Millions)') -> length_over
view
```

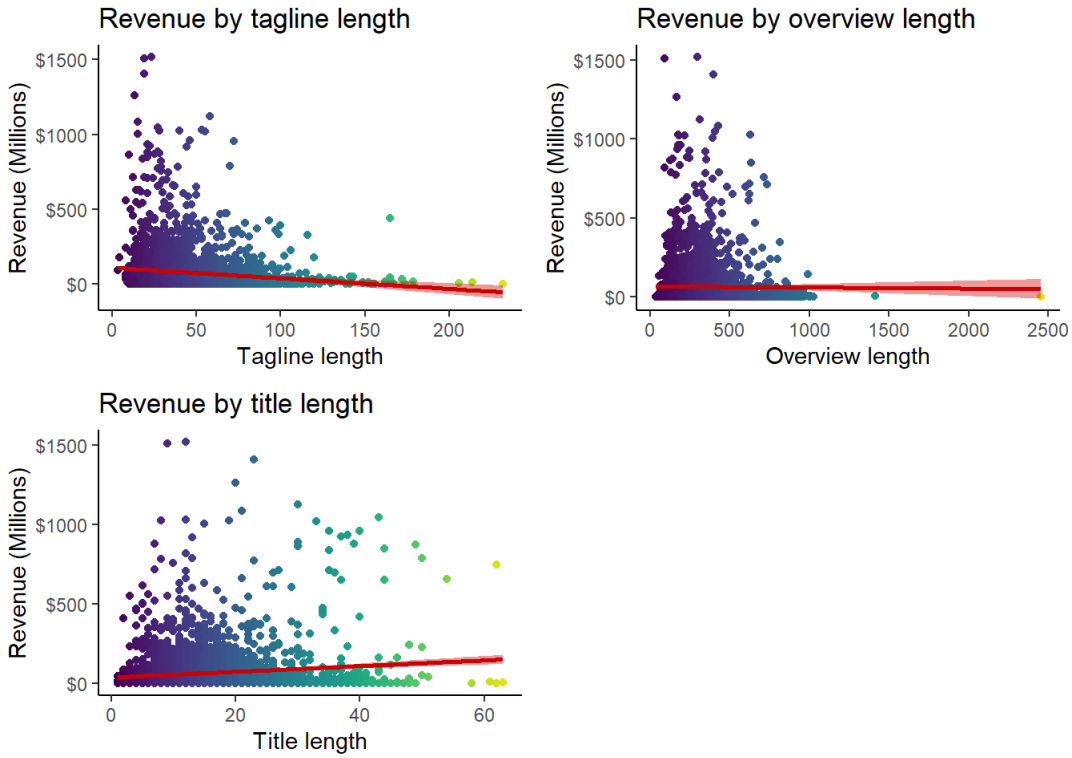
```
grid.arrange(length_title, length_tagline, length_overview,
             layout_matrix = rbind(c(1, 2),
                                   c(0, 3)))
```

```
## Warning: Removed 597 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 597 rows containing missing values (geom_point).
```

```
## Warning: Removed 8 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```



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

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

```
## [1] 0.1087646
```

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

```
## [1] -0.1206457
```

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

```
## [1] -0.008616267
```

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

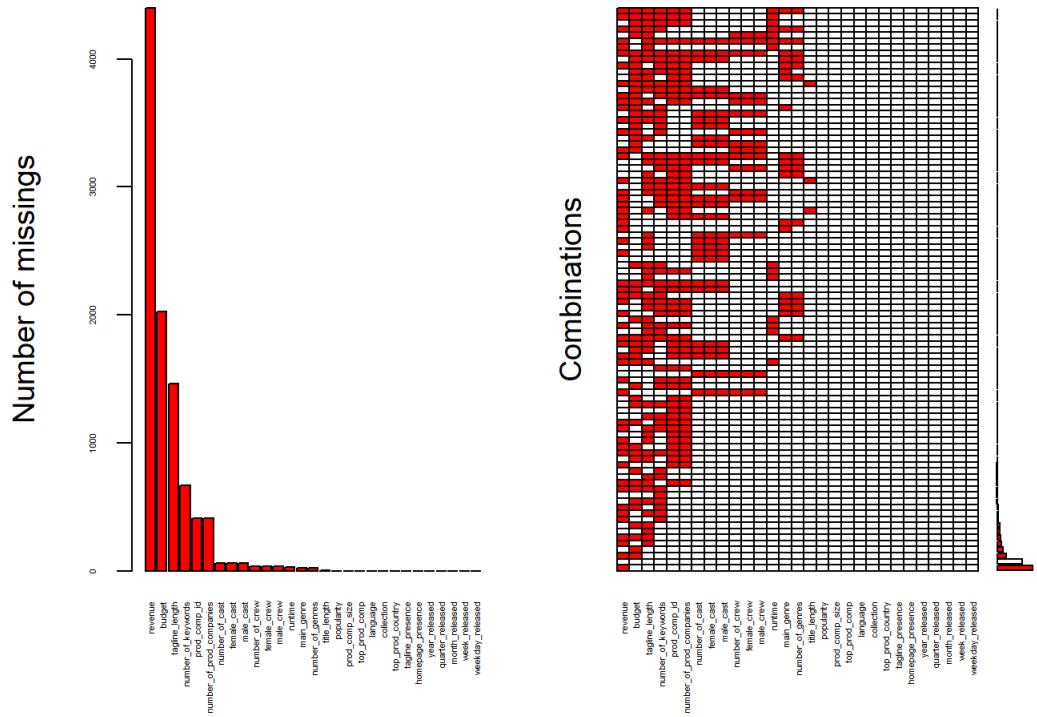
Lets first create a subset containing all the variables we want to keep for our machine learning model. We removed : number_of_prod_countries, number_of_spoken_languages, imdb_id_2, overview_length, unspecified_cast, unspecified_crew.

```
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?

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

```
## Warning in plot.aggr(res, ...): not enough vertical space to display
## frequencies (too many combinations)
```



```
##
##  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
```

```
full_data_subset$runtime[is.na(full_data_subset$runtime)] <- mean(full_data_subset$runtime, na.rm = TRUE)
full_data_subset$number_of_cast[is.na(full_data_subset$number_of_cast)] <- mean(full_data_subset$number_of_c
ast, na.rm = TRUE)
full_data_subset$number_of_crew[is.na(full_data_subset$number_of_crew)] <- mean(full_data_subset$number_of_c
rew, na.rm = TRUE)
full_data_subset$tagline_length[is.na(full_data_subset$tagline_length)] <- mean(full_data_subset$tagline_len
gth, na.rm = TRUE)
full_data_subset$title_length[is.na(full_data_subset$title_length)] <- mean(full_data_subset$title_length, n
a.rm = TRUE)
full_data_subset$female_cast[is.na(full_data_subset$female_cast)] <- mean(full_data_subset$female_cast, na.r
m = TRUE)
full_data_subset$male_cast[is.na(full_data_subset$male_cast)] <- mean(full_data_subset$male_cast, na.rm = TR
UE)
full_data_subset$female_crew[is.na(full_data_subset$female_crew)] <- mean(full_data_subset$female_crew, na.r
m = TRUE)
full_data_subset$male_crew[is.na(full_data_subset$male_crew)] <- mean(full_data_subset$male_crew, na.rm = TR
UE)
full_data_subset$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

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

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)
```

```
## Warning in full_data_subset$budget[is.na(full_data_subset$budget)] <-
## predict(lm_budget): number of items to replace is not a multiple of
## replacement length
```

Final preparations

```
full_data_subset$budget_year_ratio <- full_data_subset$budget/full_data_subset$year_released
full_data_subset <- full_data_subset %>% mutate_if(is.character, as.factor)
full_data_subset$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)
train <- full_data_subset[1:3000,]
test <- full_data_subset[3001:7398,]
```

Machine Learning Model

Model 1: Linear Regression

We have splited data in 80-20 ratio as Train data and Validation data.

```
train1 <- train[1:2400,]
valTest<- train[2401:3000,]
```

```
model1<-lm(revenue~popularity+ runtime+ budget,train1)
summary(model1)
```

```
##
## Call:
## lm(formula = revenue ~ popularity + runtime + budget, data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3526 -0.3490  0.2178  0.6204  4.0149
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.046076   0.205882   5.081 4.05e-07 ***
## popularity  1.771211   0.069599  25.449  < 2e-16 ***
## runtime     0.005121   0.001021   5.014 5.72e-07 ***
## budget      0.537184   0.027744  19.362  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.049 on 2396 degrees of freedom
## Multiple R-squared:  0.3984, Adjusted R-squared:  0.3976
## F-statistic: 528.9 on 3 and 2396 DF,  p-value: < 2.2e-16
```

For numerical attributes: Popularity, Runtime, Budget to predict revenue

Residual standard error: 1.049 on 2396 degrees of freedom

Multiple R-squared: 0.3984, Adjusted R-squared: 0.3976

F-statistic: 528.9 on 3 and 2396 DF, p-value: < 2.2e-16

```
model2<- lm(revenue~popularity+ runtime+ budget+ prod_comp_size+ prod_comp_id+collection+male_crew,train1)
summary(model2)
```

```
##
## Call:
## lm(formula = revenue ~ popularity + runtime + budget + prod_comp_size +
##     prod_comp_id + collection + male_crew, data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7709 -0.3304  0.1870  0.5702  3.7682
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.610e+00  2.182e-01  11.959  < 2e-16 ***
## popularity                    1.222e+00  7.611e-02  16.055  < 2e-16 ***
## runtime                       4.650e-03  9.828e-04   4.731 2.36e-06 ***
## budget                        4.522e-01  2.694e-02  16.785  < 2e-16 ***
## prod_comp_sizeSmall producer -3.947e-01  4.973e-02  -7.938 3.13e-15 ***
## prod_comp_id                 -1.132e-05  2.015e-06  -5.620 2.13e-08 ***
## collectionNo collection      -4.035e-01  5.230e-02  -7.715 1.76e-14 ***
## male_crew                     2.773e-02  3.696e-03   7.502 8.83e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9956 on 2392 degrees of freedom
## Multiple R-squared:  0.459,  Adjusted R-squared:  0.4574
## F-statistic: 289.9 on 7 and 2392 DF,  p-value: < 2.2e-16
```

Now we have introduced some more attributes (prod_comp_size,

prod_comp_id, collection, male_crew) along with popularity, runtime, budget to ### check whether adjusted r squared value will increase or decrease by considering

more attributes to our Linear Regression model. We got following values:

Residual standard error: 0.9956 on 2392 degrees of freedom

Multiple R-squared: 0.459, Adjusted R-squared: 0.4574

F-statistic: 289.9 on 7 and 2392 DF, p-value: < 2.2e-16

Since the value of adjusted R squared value has increased from 0.3976 to 0.4574 ##### so we have decided to consider all attributes after removing attributes having

correlation value greater than 0.1 with revenue.

Let apply the model

```
model3<- lm(revenue~.,train1)

summary(model3)
```

```
##
## Call:
## lm(formula = revenue ~ ., data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1284 -0.3290  0.1370  0.5409  3.6552
##
## Coefficients: (4 not defined because of singularities)
##                                                    Estimate Std. Error
## (Intercept)                                      -1.318e+02  1.179e+02
## popularity                                        1.059e+00  8.459e-02
## runtime                                           5.930e-03  1.088e-03
## budget                                           -2.606e+01  1.774e+01
## prod_comp_sizeSmall producer                     -3.992e-01  1.236e-01
## top_prod_compNew Line Cinema                     -1.299e-02  1.821e-01
## top_prod_compOther                                       NA         NA
## top_prod_compParamount Pictures                  -1.400e-01  1.489e-01
## top_prod_compTwentieth Century Fox Film Corporation -7.628e-02  1.548e-01
```

```
## top_prod_compUniversal Pictures                       -1.103e-01  1.453e-01
## top_prod_compWalt Disney Pictures                     -1.190e-01  1.884e-01
## top_prod_compWarner Bros.                             -2.867e-01  1.755e-01
## prod_comp_id                                          -9.659e-06  2.031e-06
## main_genreAdventure                                    1.480e-01  9.527e-02
## main_genreAnimation                                    2.372e-01  1.320e-01
## main_genreComedy                                       6.868e-02  7.217e-02
## main_genreDocumentary                                 -6.305e-01  1.411e-01
## main_genreDrama                                       -2.204e-01  6.693e-02
## main_genreFantasy                                     -5.091e-02  1.367e-01
## main_genreForeign                                     -1.915e-01  9.760e-01
## main_genreHistory                                     -3.003e-01  2.977e-01
## main_genreHorror                                      -6.474e-02  9.811e-02
## main_genreMusic                                       -2.219e-02  2.748e-01
## main_genreMystery                                     -1.691e-01  1.839e-01
## main_genreRomance                                      6.640e-02  1.424e-01
## main_genreScience Fiction                             -2.443e-01  1.692e-01
## main_genreThriller                                    -2.508e-01  1.087e-01
## main_genreTV Movie                                     1.114e+00  9.762e-01
## main_genreWar                                         -1.947e-01  2.423e-01
## main_genreWestern                                     -1.542e-01  2.889e-01
## languageNon-English                                    1.801e-01  7.745e-02
## collectionNo collection                               -3.207e-01  5.593e-02
## top_prod_countryGreat Britain                          1.045e-01  1.185e-01
## top_prod_countryOther                                  1.597e-01  9.777e-02
## top_prod_countryUnited States                          3.059e-01  9.975e-02
## tagline_presenceTagline                                2.404e-01  5.920e-02
## homepage_presenceNo homepage                          -8.817e-02  4.848e-02
## year_released                                          4.051e+01  3.573e+01
## quarter_released2                                     -7.916e-01  3.566e-01
## quarter_released3                                     -1.572e+00  5.568e-01
## quarter_released4                                     -1.947e+00  7.706e-01
## month_releasedFebruary                                 1.163e-02  1.185e-01
## month_releasedMarch                                   -2.348e-01  1.629e-01
## month_releasedApril                                    4.398e-01  1.742e-01
## month_releasedMay                                      9.840e-02  1.241e-01
## month_releasedJune                                            NA         NA
## month_releasedJuly                                     6.564e-01  1.683e-01
## month_releasedAugust                                   3.769e-01  1.145e-01
## month_releasedSeptember                                       NA         NA
## month_releasedOctober                                  1.268e-01  1.713e-01
## month_releasedNovember                                -2.726e-02  1.273e-01
## month_releasedDecember                                        NA         NA
## week_released                                          4.796e-02  1.607e-02
## weekday_releasedMonday                                -9.545e-02  1.436e-01
## weekday_releasedTuesday                               -5.427e-02  1.328e-01
## weekday_releasedWednesday                              1.151e-01  1.194e-01
## weekday_releasedThursday                               1.243e-01  1.160e-01
## weekday_releasedFriday                                 6.637e-02  1.116e-01
## weekday_releasedSaturday                              -2.319e-01  1.368e-01
## number_of_keywords                                     4.647e-03  3.362e-03
## number_of_prod_companies                               1.662e-02  1.174e-02
## number_of_genres                                       1.462e-02  2.057e-02
## title_length                                           1.503e-03  2.555e-03
## tagline_length                                         5.855e-04  9.248e-04
## number_of_cast                                        -3.170e-03  2.378e-03
## number_of_crew                                        -1.835e-03  1.328e-03
## female_cast                                            1.444e-02  6.106e-03
## male_cast                                              4.711e-03  4.838e-03
## female_crew                                            1.113e-02  1.065e-02
## male_crew                                              2.273e-02  6.122e-03
## budget_year_ratio                                      8.747e+01  5.853e+01
##                                                      t value Pr(>|t|)
## (Intercept)                                           -1.118  0.26348
## popularity                                            12.521  < 2e-16 ***
## runtime                                                5.451 5.52e-08 ***
## budget                                                -1.469  0.14194
## prod_comp_sizeSmall producer                          -3.230  0.00125 **
## top_prod_compNew Line Cinema                          -0.071  0.94313
## top_prod_compOther                                        NA       NA
## top_prod_compParamount Pictures                       -0.940  0.34721
## top_prod_compTwentieth Century Fox Film Corporation   -0.493  0.62216
## top_prod_compUniversal Pictures                       -0.759  0.44773
```

```
## top_prod_compUniversal Pictures                    0.759  0.44773
## top_prod_compWalt Disney Pictures                 -0.632  0.52759
## top_prod_compWarner Bros.                         -1.633  0.10259
## prod_comp_id                                      -4.755 2.10e-06 ***
## main_genreAdventure                                1.553  0.12054
## main_genreAnimation                                1.797  0.07241 .
## main_genreComedy                                   0.952  0.34135
## main_genreDocumentary                             -4.470 8.21e-06 ***
## main_genreDrama                                   -3.292  0.00101 **
## main_genreFantasy                                 -0.372  0.70958
## main_genreForeign                                 -0.196  0.84445
## main_genreHistory                                 -1.009  0.31326
## main_genreHorror                                  -0.660  0.50938
## main_genreMusic                                   -0.081  0.93563
## main_genreMystery                                 -0.919  0.35802
## main_genreRomance                                  0.466  0.64109
## main_genreScience Fiction                         -1.444  0.14883
## main_genreThriller                                -2.307  0.02114 *
## main_genreTV Movie                                 1.142  0.25370
## main_genreWar                                     -0.804  0.42174
## main_genreWestern                                 -0.534  0.59363
## languageNon-English                                2.326  0.02013 *
## collectionNo collection                           -5.733 1.11e-08 ***
## top_prod_countryGreat Britain                      0.881  0.37818
## top_prod_countryOther                              1.634  0.10243
## top_prod_countryUnited States                      3.067  0.00219 **
## tagline_presenceTagline                            4.060 5.07e-05 ***
## homepage_presenceNo homepage                      -1.819  0.06907 .
## year_released                                      1.134  0.25704
## quarter_released2                                 -2.220  0.02654 *
## quarter_released3                                 -2.823  0.00480 **
## quarter_released4                                 -2.527  0.01158 *
## month_releasedFebruary                             0.098  0.92184
## month_releasedMarch                               -1.442  0.14950
## month_releasedApril                                2.525  0.01165 *
## month_releasedMay                                  0.793  0.42797
## month_releasedJune                                    NA       NA
## month_releasedJuly                                 3.901 9.85e-05 ***
## month_releasedAugust                               3.292  0.00101 **
## month_releasedSeptember                               NA       NA
## month_releasedOctober                              0.740  0.45950
## month_releasedNovember                            -0.214  0.83053
## month_releasedDecember                                NA       NA
## week_released                                      2.985  0.00287 **
## weekday_releasedMonday                            -0.665  0.50632
## weekday_releasedTuesday                           -0.409  0.68277
## weekday_releasedWednesday                          0.964  0.33509
## weekday_releasedThursday                           1.072  0.28403
## weekday_releasedFriday                             0.595  0.55221
## weekday_releasedSaturday                          -1.696  0.09011 .
## number_of_keywords                                 1.382  0.16707
## number_of_prod_companies                           1.415  0.15708
## number_of_genres                                   0.711  0.47733
## title_length                                       0.588  0.55636
## tagline_length                                     0.633  0.52674
## number_of_cast                                    -1.333  0.18263
## number_of_crew                                    -1.381  0.16730
## female_cast                                        2.365  0.01810 *
## male_cast                                          0.974  0.33025
## female_crew                                        1.046  0.29584
## male_crew                                          3.712  0.00021 ***
## budget_year_ratio                                  1.494  0.13523
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9638 on 2333 degrees of freedom
## Multiple R-squared:  0.5055, Adjusted R-squared:  0.4915
## F-statistic: 36.14 on 66 and 2333 DF,  p-value: < 2.2e-16
```

```
summary(train$budget)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.301   6.778   7.176   7.079   7.544   9.332
```

Residual standard error: 0.9638 on 2333 degrees of freedom

Multiple R-squared: 0.5055, Adjusted R-squared: 0.4915

F-statistic: 36.14 on 66 and 2333 DF, p-value: < 2.2e-16

Predicting Revenue for validation test using Model3 and calculating RMSE value

```
valTest$Lm_prediction <- predict(model3, valTest)
```

```
## Warning in predict.lm(model3, valTest): prediction from a rank-deficient
## fit may be misleading
```

```
rmse(valTest$revenue, valTest$Lm_prediction)
```

```
## [1] 0.9133189
```

RMSE Value for Linear Regression model (Model3) = 0.9133

Let's apply another Machine Learning model: Random Forest

Model 2: Random Forest

```
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, import
ance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 501
## No. of variables tried at each split: 9
##
##           Mean of squared residuals: 0.8591925
##                     % Var explained: 51.38
```

Random Forest

Call:

randomForest(formula = revenue ~ ., data = train, ntree = 501, replace = TRUE, nodesize = 9, importance = TRUE)

Type of random forest: regression

Number of trees: 501

No. of variables tried at each split: 9

Mean of squared residuals: 0.8591925

% Var explained: 51.38

Importance of variables

Create an object for importance of variables

```
importance <- importance(rf_model)
```

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())
```

Prediction

```
prediction <- predict(rf_model, valTest)
valTest$RF_prediction<-(prediction)
view(test$revenue)
solution <- as_tibble(test) %>%
  mutate(revenue = 10^revenue)
solution$id<-test_data$ï..id
Solution1<- subset(solution, select = c(id,revenue))
```

```
rmse(valTest$revenue, valTest$RF_prediction)
```

```
## [1] 0.4210502
```

```
set.seed(222)

rf_model <- randomForest(revenue ~ .,
                         data = train1,
                         ntree = 501,
                         replace = FALSE,
                         nodesize = 9,
                         importance = TRUE);
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = revenue ~ ., data = train1, ntree = 501,      replace = FALSE, nodesize = 9, impo
rtance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 501
## No. of variables tried at each split: 9
##
##           Mean of squared residuals: 0.8968137
##                     % Var explained: 50.89
```

```
write.csv(Solution1 ,file = 'Box_office_prediction.csv', row.names = F)


#summary(rf_model)
```