Project Part 1

Rodrigo De Luna Lara

10 October 2016

Dataset Description

The data used for the analysis requested in this project is the movie_merged dataframe. This dataframe has 40,000 observations of 39 variables. Each observartion corresponds to a different movie with each column describing a different characteristic of the movie. The dataset seems to be the merge of several dataframes, the first of which seems to be the API from the Open Movie Database. A default RESTFUL request to the API returns the following variables in the JSON response:

Table 1: Default OMDB API Response Variables

Variable name	**Description**	**Type**
Title	Title of the film	String
Year	Year of release	Numeric
Rated	Rating of the film in MPAA film rating system	String
Released	Full date of release	Date
Runtime	Duration of the film in minutes	String
Genre	Genre(s) applicable to the film	String
Director	Full name of the director of the film	String
Writer	Full name of the writer of the script	String
Actors	List of principal actors in the film	String
Plot	Summary of the plot of the film	String
Language	List of languages in which the film was released	String
Country	List of countries in which the film was released	String
Awards	Number of awards the film won	String
Poster	Link to an image of the film's poster	String
Metascore	Numeric score from Metacritic	String
imdbRating	Numeric rating in IMDB	Numeric
imdbVotes	Number of votes received in IMDB	Numeric
imdbID	Unique id of the film in IMDB	String
Type	Type of media (movie, series or game)	String
Response	Indication if the API request was sucessful	String

These variables include information from both Metacritic and IMDB. Additional information from Rotten Tomatoes can be added to the request by including the "tomatoes" parameter in the RESTFUL request. The additional variables included in the dataframe from the response JSON can be seen in Table 2. With these variables there is combined information from 3 different sources comprising the data in the dataframe.

Table 2: Rotten Tomatoes OMDB API Response Variables

Variable name	Description	Type
tomatoMeter	Positive reviews by critics (%)	Integer
tomatoImage	Rotten Tomatoes recommendation label	String
tomatoRating	Average rating given by critics on a 10.0 scale	Numeric
tomatoReviews	Total number of reviews by critics	Integer
tomatoFresh	Number of positive reviews by critics	Integer
tomatoRotten	Number of negative reviews by critics	Integer
tomatoConsensus	Consensus of the critics on the film	String
tomatoUserMeter	Percentage of users that liked the film	Numeric
tomatoUserRating	Average rating given by audience on a 5.0 scale	Numeric
tomatoUserReviews	Number of reviews by users	Integer
tomatoURL	Link to the Rotten Tomatoes review	String
DVD	Date of release to DVD	Date
BoxOffice	Revenue at box office in USD	String
Production	Company that produced the film	String
Website	URL of the film's website	String

There are 4 additional numeric variables of unknown source (although the information seems to have come from the website http://www.the-numbers.com/movie/budgets/all) related to financials of the films. These 4 variables can be seen in Table 3.

Table 3: Additional Dataframe Variables

Variable name	Description	Type
Budget	Budget used by the film in USD	Numeric
Domestic Gross	Gross revenue of the film in the USA in USD	Numeric
Gross	Gross worldwide revenue of the film in USD	Numeric
Date	Date of release of the film	Numeric

Type Filtering (Assignment 1)

As the objects of analysis are films, any observation in the dataframe that doesn't correspond to a movie has to be removed, this filtering can be applied with the Type variable.

```
# Determine number of observations that are not movies
length(which(movies_merged$Type != "movie"))
```

```
## [1] 789
```

```
#  Remove them from the dataframe
movies_merged = movies_merged[which(movies_merged$Type == "movie"),]
nrow(movies_merged)
```

```
## [1] 40000
```

Only 789 observations were removed from the dataset, which is 1.93% of the original size of the dataframe.

Missing Values

It was noted that the dataframe contains strings indicating missing values ("N/A") instead of R's NA, so any occurences of this string were replaced with the appropriate value for R.

```
# Replace "N/A" with R's NA
dataframe.aslist <- lapply(movies_merged,gsub,pattern="N/A",replacement=NA)
```

The dataframe is then reconstructed coercing specific types to the variables (according to the original dataframe) and dropping unnecesary variables. The variables that were dropped are:

- Poster: the image of the poster is not relevant for this analysis
- imdbID: this variable only indicates the id of the movie in IMDB
- Type: as only movies were filtered, there is no point in keeping this variable
- Response: this variable is only indicative of a successful JSON reponse
- tomatoURL: the link to the Rotten Tomatoes page is not relevant for this analysis
- Website: the URL for the website of the film is not relevant for this analysis

Table 4: Coerced Variable Types in the Dataframe

Variable name	**Coerced Type**	**Variable name**	**Coerced Type**
Title	Character array	Year	Numeric
Rated	Factor	Released	Date
Runtime	Character array	Genre	Character array
Director	Character array	Writer	Character array
Actors	Character array	Plot	Character array
Language	Character array	Country	Character array
Awards	Character array	Metascore	Numeric
imdbRating	Numeric	imdbVotes	Numeric
tomatoMeter	Numeric	tomatoImage	Factor
tomatoRating	Numeric	tomatoReviews	Numeric
tomatoFresh	Numeric	tomatoRotten	Numeric
tomatoConsensus	Character array	tomatoUserMeter	Numeric
tomatoUserRating	Numeric	tomatoUserReviews	Numeric
DVD	Date	BoxOffice	Character array
Production	Factor	Budget	Numeric
Domestic Gross	Numeric	Gross	Numeric
Date	Numeric		

```
# Calculate the number of complete observations
n.complete.obs = sum(complete.cases(movies_merged)) / nrow(movies_merged) * 100
print(n.complete.obs)
```

```
## [1] 0.0125
```

Less than 1% of the dataframe has complete observations, this is expected due to the different sources of the data and the type of data (most of the data is not quantitative).

Year Variable Analysis

It is also a good idea to familiarize with the distribution of the data on certain variables, starting with the year of release.

```
# Get the unique values for the Year variable
years.indata = sort(unique(movies_merged$Year))
print(years.indata)
```

```
##   [1] 1888 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905
##  [15] 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919
##  [29] 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933
##  [43] 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947
##  [57] 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961
##  [71] 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975
##  [85] 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989
##  [99] 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003
## [113] 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017
## [127] 2018
```

The frequency of movies in the dataframe by year can be seen in Figure 1.

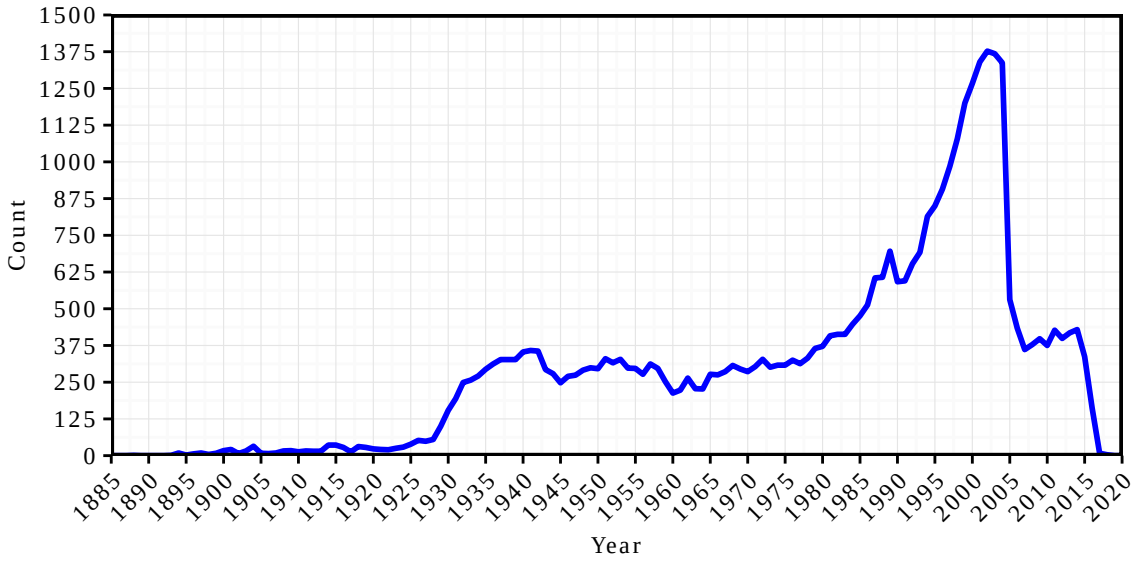

Figure 1: Frequency of Films by Year

At first sight it is not easy to determine if there are missing years, so this can be explicitly determined:

```
# Create a sequence of years with the minimum and maximum values in the dataframe
years.sequence = seq(min(movies_merged$Year),max(movies_merged$Year))
# Determine which years in the range are missing
missing.years = years.sequence[!(years.sequence %in% years.indata)]
print(missing.years)
```

```
## [1] 1889 1890 1891 1892
```

The dataframe lacks information from movies from 1889 to 1892. It is also of interest that there are unreleased films in the dataset, which are not of interest to the study as they don't contain any ratings or financial information, so these observations are removed from the dataframe.

```
# Determine which movies have a date greater than 2016
films.toremove = movies_merged[which(movies_merged$Year > 2016),1:2]
# Print the titles of the films to be removed from the dataset
print(films.toremove)
```

```
##                                                 Title Year
## 4495                          Killer Villa, the [Gb] 2017
## 19049                      Oracle: The Last Judgment 2018
## 22069                           Warlords of Atlantis 2018
## 24361                                     Utvandrarna 2018
## 25922 The Art of Love: The Story of Michalina Wislocka 2017
## 31112                        Red Shoes & the 7 Dwarfs 2017
## 37188                                     Sky Sharks 2017
## 37189                                The Thousand Miles 2017
## 37190      Valerian and the City of a Thousand Planets 2017
## 37191                                     Tulip Fever 2017
## 37192                                T2: Trainspotting 2017
```

```
# The dataset is refactored with only films up to 2016
movies_merged = movies_merged[which(movies_merged$Year <= 2016),]
```

There are only 11 films with dates posterior to 2016, so only 0.0275% of films in the dataframe are removed, having no significant impact on the number of observations; however, it is a grood practice to cleanup the dataframes removing any unnecessary variables and observations which act just as padding and could even influence some of the analyses.

Runtime Variable Analysis (Assignment 2)

For commidity in data processing, the Runtime variable was parsed into a numeric array from the string. Then its distribution was plotted as seen in Figure 2.

```
movies_merged$Runtime = as.numeric(lapply(movies_merged$Runtime, function(x) {
    time = NA
    if (length(grep("min",x)) != 0){
        if (length(grep("h",x)) != 0){
            pos = gregexpr("h",x)[[1]][1]
            hrs = as.numeric(substr(x,1,pos-1)) * 60
            x = substr(x,pos+1,nchar(x))
        } else {
            hrs = 0
        }
        pos = gregexpr("m",x)[[1]][1]
        min = as.numeric(substr(x,1,pos-1))
        time = hrs + min
    }
    return(time)
}))
```

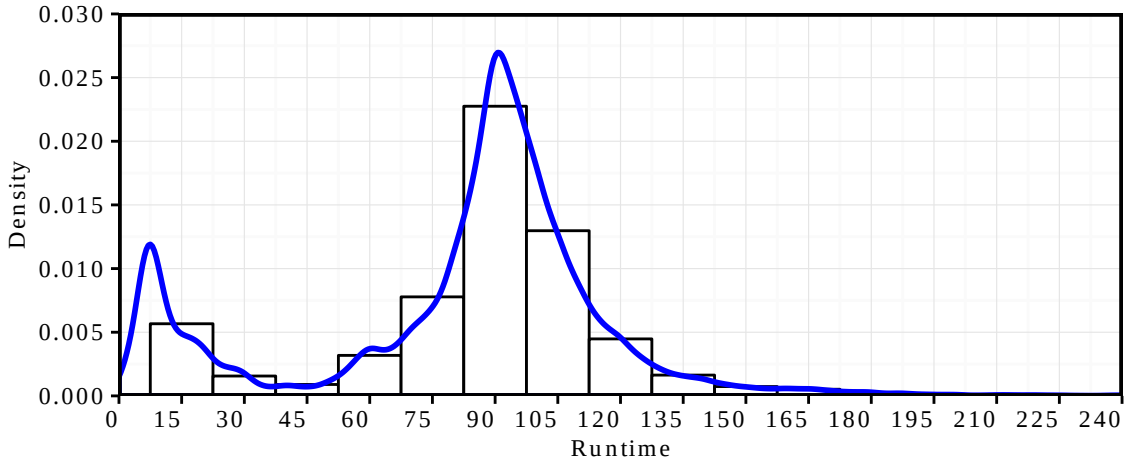

Figure 2: Histogram of Film Runtime

It can be seen that the data is bimodal, there is one mode at 15 minutes and a second mode at 90 minutes. The films around the 15 minute range should correspond to short films, which could be considered a separate category of films, the great majority of films have a runtime between 45 and 150 minutes.

The distribution of runtimes by year can be visualized in several forms, one of which is a radial plot in which the angle is defined by the year and the runtime by the magnitude, this plot can be seen in Figure 3. The second, simpler, form is a trend plot of the median of runtimes by year, which can be seen in Figure 4.

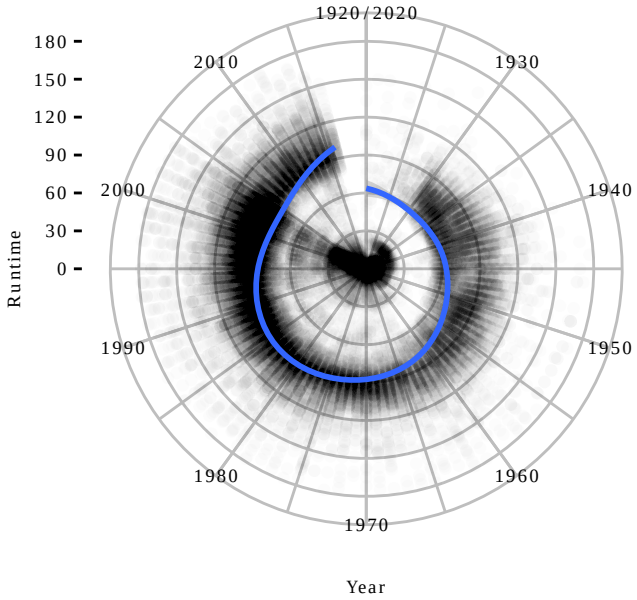

Figure 3: Radial plot of Runtime by Year

It can be seen that bulk of the values are distributed between the 60 and 120 minutes mark, in agreement with the histogram in Figure 2. The distribution around the first mode, also seen in Figure 2, is also clearly visible as a cluster in the central part of the plot, confirming the bimodal distribution of the runtime. The regression line shows an increasing trend in runtime from 60 to 90 minutes from 1920 to 1975, after which the runtime remains stable at 90 minutes until 1995. After 1995 the runtime is reduced slightly to about 80 minutes, rising again to the 90 minutes mark near the year 2005.

Figure 4 shows the same overall behavior with the median of the runtime by year, showing a rising trend from 1920 to 1970, stable behavior from 1970 to 1995, a slight decrease from 1995 to 2005 and a rising trend back to the 90 minute mark after 2005, confirming the observations made in the radial plot in Figure 3.

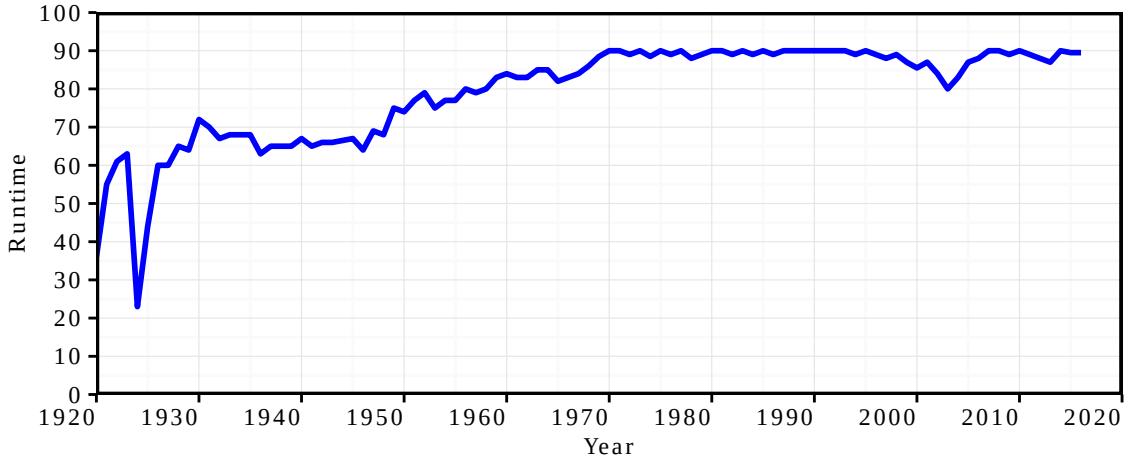

Figure 4: Median Runtime by Year

Relationship Between Runtime and Budget (Assignment 2)

To analyze the relationship between runtime and budget, the distribution of the budgets was plotted first to determine the range in which most of the data lies. As seen in Figure 5 an acceptable range for analysis is up to 150 million dollars, encompassing 97.60% of the films with budget information.

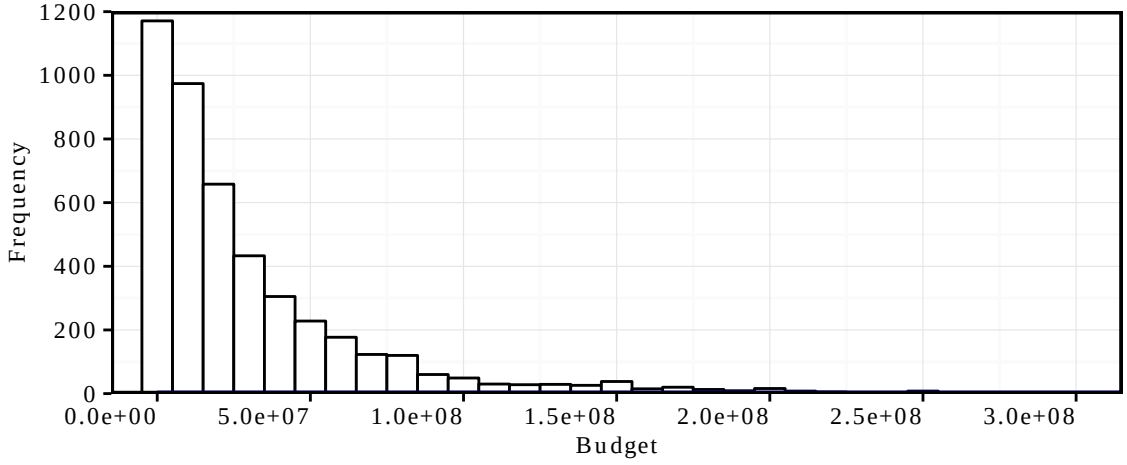

Figure 5: Histogram of Film Budget

```r
# Calculate percentage of films with budgets less or equal to 150 million dollars
print(
   sum(as.numeric(movies_merged$Budget[!is.na(movies_merged$Budget)] <= 1.5e08))
  /sum(!is.na(as.numeric(movies_merged$Budget))))
```

```
## [1] 0.9760597
```

The relationship between both variables was then analyzed by grouping the data by decade. The grouping by decade was done by parsing the Year variable as a string to create a factor with a level for each decade. This was done to facilitate grouping of the data by decades for use as facets in the creation of the plot.

```r
# Use the Year variable to make a factor indicating the decade
movies_merged$Decade = as.factor(as.character(
  lapply(movies_merged$Year, function(x) {
    # The year can be used directly to assing the factor
    return(paste0(substr(x,1,3),"0's"))
  })
))
# Print the levels of the resulting factor
print(levels(movies_merged$Decade))
```

```
##  [1] "1880's" "1890's" "1900's" "1910's" "1920's" "1930's" "1940's"
##  [8] "1950's" "1960's" "1970's" "1980's" "1990's" "2000's" "2010's"
```

The resulting factors were used to group the data by decade and then create a plot between the runtime of the film and the logarithm of the budget of the film. The decades of 1880 through 1920 were not included in the plots as they have an extremely low amount of data as evidenced by the distribution in Figure 1.

The resulting plot in Figure 6 evidences several things. First, it can be seen that the budget has increased over the decades, peaking in the 2000's, the decade of the 2010's shows less movies with high budgets but that could be due to the fact that the dataframe has data up to 2016. It can also be seen that as each decade passes the regression becomes more logarithmic, flattening out quickly, this suggests that for recent decades the budget of the film is non-determinant for the runtime.

Said behavior is also evidenced by the presence of clearly visible vertical clusters at specific budgets, indicating that for a given budget there can be varying runtimes which are most definitely function of other factors such as the actors, director, writers and genre of the films. However, this is the case just recently, as it seems that up to 1980's the budget did have a strong relationship with the runtime of the film, increased budget resulted in increased runtime, if a log transformation of the budget were applied there would be decreasing slope lines for each decade, becoming almost flat at recent decades.

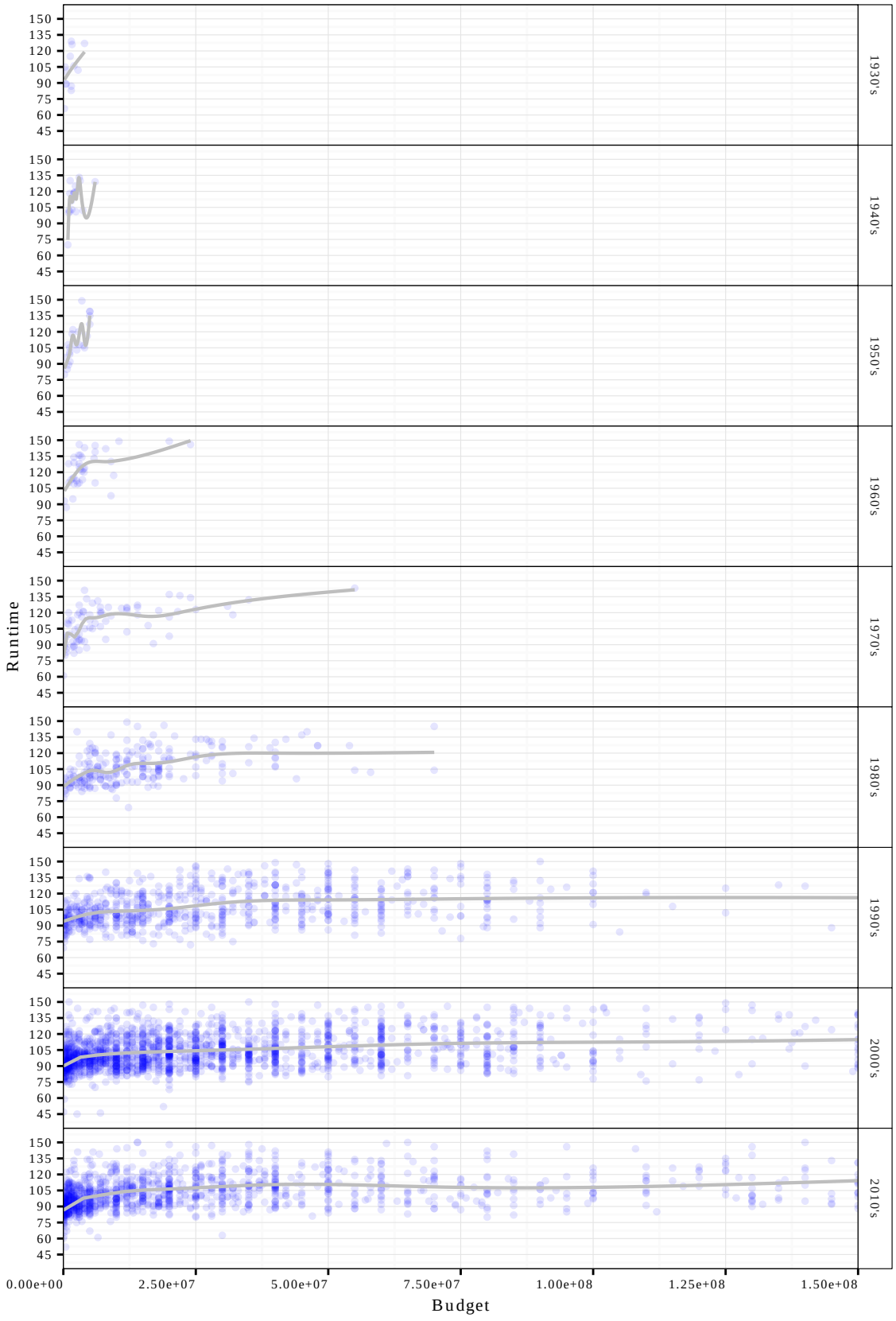

Figure 6: Trends for Budget by Runtime by Decade

Genre Variable Analysis (Assignment 3)

The next variable to analyze is the genre. To determine which unique genres are located in the Genre variable, the package abind is extremely useful. Each string is split by commas to create an array of genres for each entry, then they're bound in a single array to determine the unique values:

```
library(abind)
library(stringr)
genres.split = lapply(strsplit(movies_merged$Genre,split=","),str_trim)
unique.genres = str_trim(sort(unique(abind(genres.split))))
print(unique.genres)
```

```
##  [1] "Action"      "Adult"       "Adventure"   "Animation"   "Biography"
##  [6] "Comedy"      "Crime"       "Documentary" "Drama"       "Family"
## [11] "Fantasy"     "Film-Noir"   "Game-Show"   "History"     "Horror"
## [16] "Music"       "Musical"     "Mystery"     "News"        "Reality-TV"
## [21] "Romance"     "Sci-Fi"      "Short"       "Sport"       "Talk-Show"
## [26] "Thriller"    "War"         "Western"
```

Once having the dictionary of genres, a binary vector is created for each unit indicating the presence of each genre for each movie, the result is converted to a matrix to obtain the column sums to measure the most frequent genres.

```
movies_merged$Genre.binary = (lapply(genres.split,function (x){
  as.numeric(unique.genres %in% x)
}))

genres.matrix = do.call(rbind, movies_merged$Genre.binary)
colnames(genres.matrix) = unique.genres
row.names(genres.matrix) = movies_merged$Title

# Get the top 10 genres
genres.top10 = stack(sort(colSums(genres.matrix),decreasing=TRUE)[1:10])
colnames(genres.top10) = c("Frequency","Genre")
print(genres.top10)
```

```
##    Frequency       Genre
## 1      15856       Drama
## 2      12848      Comedy
## 3       6516       Short
## 4       4974     Romance
## 5       4412      Action
## 6       4062       Crime
## 7       3379    Thriller
## 8       3051 Documentary
## 9       2927   Adventure
## 10      2786   Animation
```

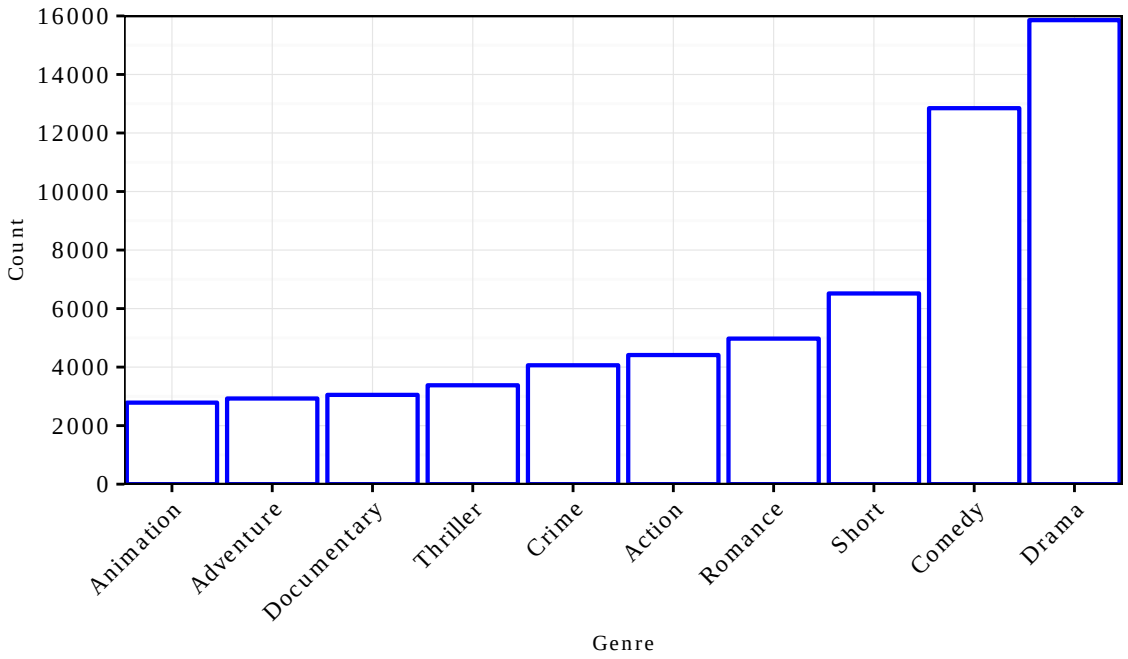

Figure 7: Frequency of Top 10 Genres

It can be seen in Figure 7 that Comedy and Drama are the most recurrent genres, having more than double the films than the 3rd most common genre, which is the Short film genre, the rest of the genres have a pretty uniform distribution, varying by less than 1000 films between each. The remaining 18 genres are not statistically significant because they comprise a very small subset of the original dataframe.

```
library(reshape2)
topgenres.index = rowSums(genres.matrix[,colnames(genres.matrix)
                  %in% genres.top10$Genre]) >= 1
# Refactor the dataset with only movies in the top 10 genres
movies_topgenres = movies_merged[topgenres.index,]
topgenres.matrix = genres.matrix[topgenres.index,
                   colnames(genres.matrix) %in% genres.top10$Genre]
# Get the number of dropped rows
print(nrow(movies_merged)-nrow(movies_topgenres))
```

```
## [1] 3568
```

To observe the relationship between the Gross revenue and the Genre, the dataset was first filtered to only include observations with the top 10 genres, the genre matrix was also modified in this manner. Given that the matrix is binary, a multiplication by any other factor can be made to use the matrix for further data analysis, in this case the matrix was multiplied by the corresponding Gross revenue.

```
gross.matrix = topgenres.matrix;
for (i in 1:10){
  gross.matrix[,i] = topgenres.matrix[,i]*movies_topgenres$Gross
}

gross.matrix[gross.matrix == 0] = NA
```

```
gross.matrixmelt = melt(gross.matrix)
gross.matrixmelt[,1] = NULL
colnames(gross.matrixmelt) = c("Genre","value")
```

The data was then summarized in boxplots by genre, the outliers are not being shown to have a cleaner plot, although there are a significant number of outliers for each genre. The medians are calculated to sort the plot and for further analysis:

```
library(plyr)
topgenre.medians = ddply(gross.matrixmelt[!is.na(gross.matrixmelt$value),],
                         c("Genre"),
                         summarise,
                         median=median(value))
print(topgenre.medians)
```

```
##          Genre    median
## 1       Action  82145379
## 2    Adventure 110623265
## 3    Animation 204774690
## 4       Comedy  32919473
## 5        Crime  31070412
## 6  Documentary   3177636
## 7        Drama  22984919
## 8      Romance  25059640
## 9        Short  39686345
## 10    Thriller  43905746
```

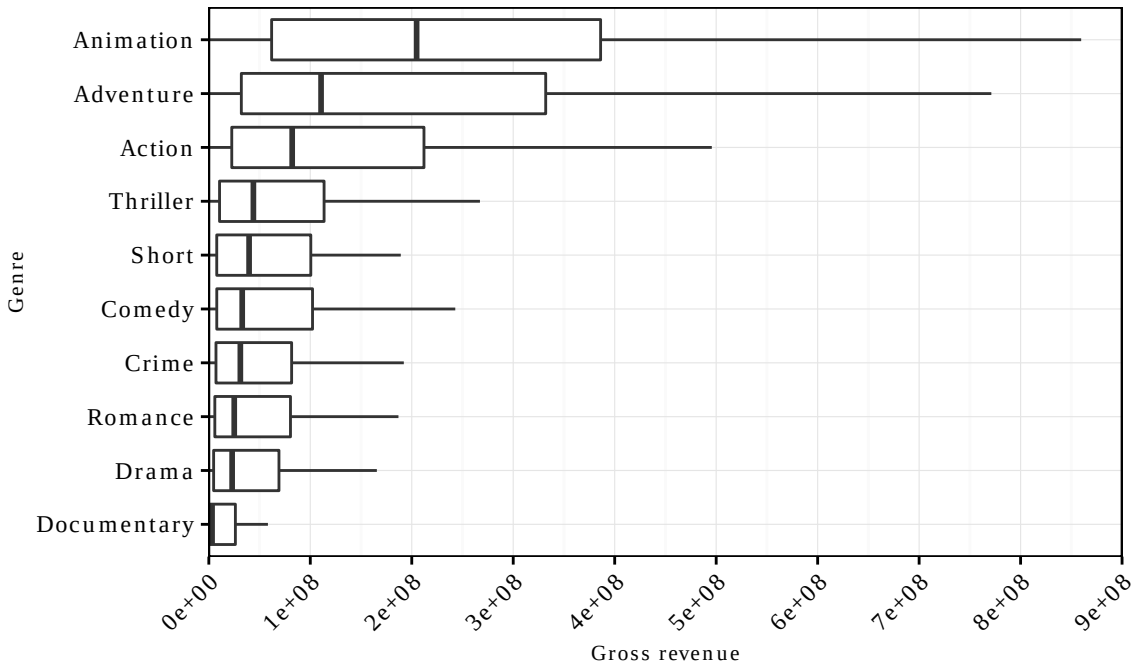

Figure 8: Boxplots of Gross Revenue by Genre

As seen in Figure 8 the top 3 genres by revenue are Animation, Adventure and Action. It is interesting to note that they are the 10th, 9th and 5th most common genres in the dataframe, which shows that not because a genre has more movies the expected gross revenue is expected to be higher. In contrast, Drama has the 2nd lowest gross revenue of these genres, and it is the most frequent genre.

The Animation genre has both the highest median and highest spread, showing the most variable revenue. It is also noticeable that it's first quartile is at a higher value than the median of the bottom 7 genres. Adventure and Action genres both have high spreads as evidenced by their whiskers in the boxplots, but have a comparatively shorter interquartile range than the Animation genre.

On the other end of the spectrum, Documentary films have the lowest spread, lowest median and shortest interquartile range, even if they are more common than both the Animation and Adventure genres in the dataframe. The remaining genres (Drama, Romance, Crime, Comedy, Short and Thriller) all have pretty similar interquartile ranges and their medians are relatively close to each other, showing similar distributions amongst them.

The difference between the medians of each Genre based on their order in the boxplots can be calculated to have a better idea of the distribution of the gross revenue.

```
topgenre.medians.diff =
  apply(as.matrix(sort(topgenre.medians$median,decreasing = TRUE)),2,diff)
# Print the difference between each consecutive genre in millions
print(topgenre.medians.diff/1e06)
```

```
##             [,1]
##  [1,] -94.151425
##  [2,] -28.477886
##  [3,] -38.239633
##  [4,]  -4.219401
##  [5,]  -6.766872
##  [6,]  -1.849060
##  [7,]  -6.010772
##  [8,]  -2.074721
##  [9,] -19.807282
```

Year and Release Variables Mismatch (Assignment 4)

The dataframe was created by merging two sources by the Title of the movie; however, there can be mismatch if two movies have the same title. The first source's release time is represented by the Year variable, while the second source's release time is represented by the Release variabl. These two variables can be used to detect possible mismatchs and remove the related observations.

First the number of observations with non-NA values in the dataframe was calculated.

```
nobs.gross = sum(!is.na(movies_merged$Gross))
print(nobs.gross)
```

```
## [1] 4553
```

Then the comparison between the Year variable is made with the year portion of the Released variable, first for observations which have NA's in the Gross variable

```
mismatch.gross.na = which(
  ((as.numeric(format(movies_merged$Released,"%Y")) - movies_merged$Year) != 0)
  & is.na(movies_merged$Gross)
  )

# The number of observations to be removed is printed and they are removed
print(length(mismatch.gross.na))
```

```
## [1] 4957
```

```
movies_merged = movies_merged[-mismatch.gross.na,]
```

The same logic is applied to observations with valid values in the Gross variable

```
mismatch.gross = which(
  ((as.numeric(format(movies_merged$Released,"%Y")) - movies_merged$Year) != 0)
  & !is.na(movies_merged$Gross)
  )
```

If the number of observations to be removed is greater than 10% of the observations with valid values in the Gross variable, they are resampled to remove only 10%

```
if (length(mismatch.gross) > nobs.gross*0.10){
  set.seed(42)
  mismatch.gross = sample(mismatch.gross,nobs.gross*0.10)
}

# The number of observations to be removed is printed and they are removed
print(length(mismatch.gross))
```

```
## [1] 455
```

```
movies_merged = movies_merged[-mismatch.gross,]
```

Finally, the total number of removed observations is printed

```
print(length(mismatch.gross) + length(mismatch.gross.na))
```

```
## [1] 5412
```

Relationship Between Release Date and Gross Revenue (Assignment 5)

To observe the relationship between release date and gross revenue with the latest dataframe in the analysis, the month and day for each release date was extracted and used as a factor on separate variables, a separate dataframe was then created with the median gross revenue for each day in the year for all available data. The result was plotted in Figure 9, having a separate panel in the grid for each month.

```
movies_merged$Day = as.factor(format(movies_merged$Released,"%d"))
movies_merged$Month = as.factor(format(movies_merged$Released,"%m"))
months.string.EN = c("01"="Jan","02"="Feb","03"="Mar","04"="Apr",
                     "05"="May","06"="Jun","07"="Jul","08"="Aug",
                     "09"="Sep","10"="Oct","11"="Nov","12"="Dec")
movies_merged$Month = revalue(movies_merged$Month,months.string.EN)
day.bymonth = ddply(movies_merged[!is.na(movies_merged$Gross),],
                    c("Month","Day"),summarise,median=median(Gross))
day.bymonth = day.bymonth[-365,]
```

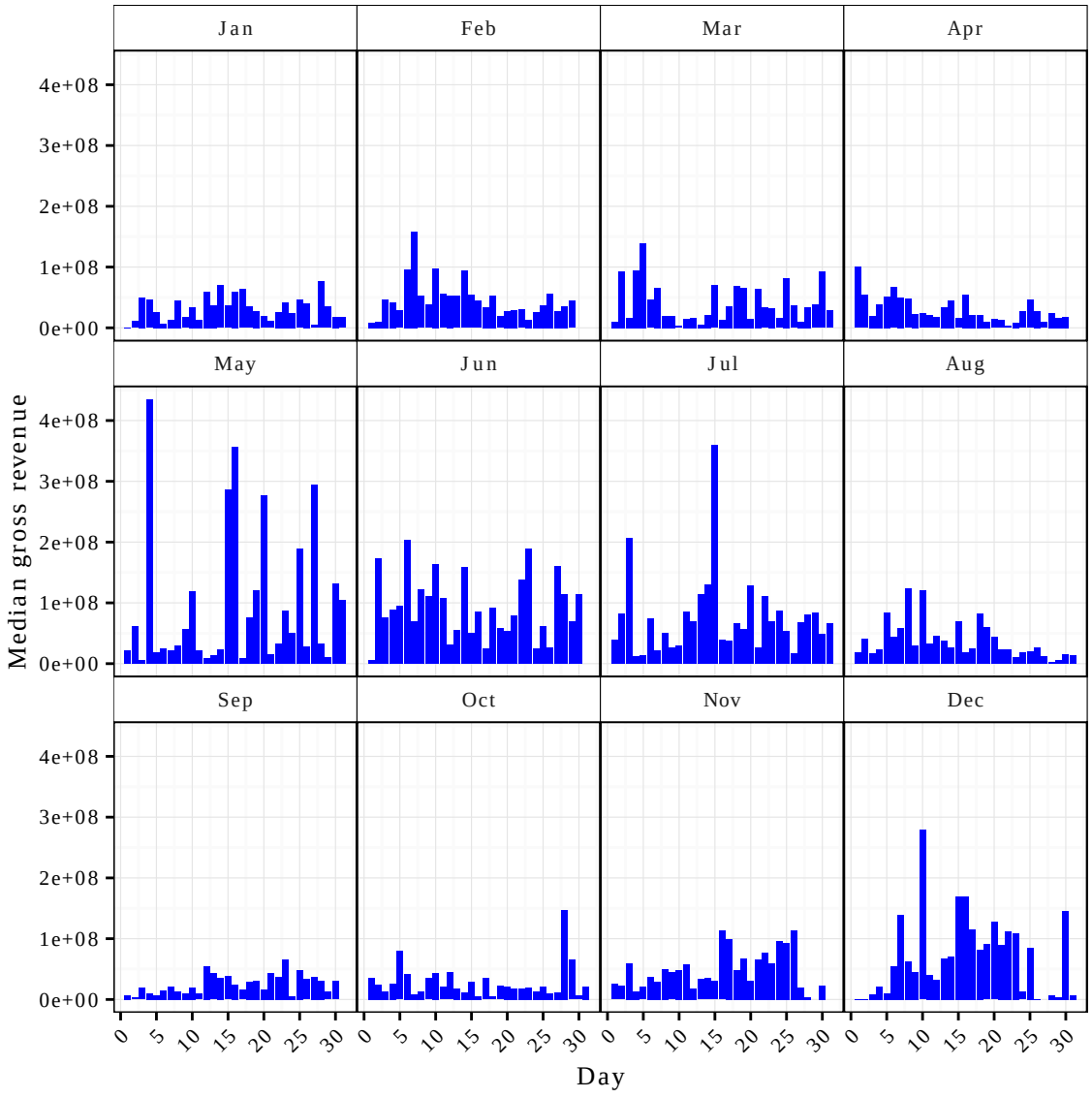

Figure 9: Median Gross Revenue by Day and Month

Conveniently, the panels are placed in 4 month intervals, dividing the year in 3 parts. It can be seen that the middle part of the year shows the highest median revenues, from May to July the medians are in overall the highest levels in the year, so summer releases tend to have a higher revenue. The highest value in these months is in early May, which is also the maximum of all days in the year.

From January to April the distribution of median gross revenue is pretty uniform, having some peaks in early February and March, but without having noticeably larger magnitudes than the rest of the days in the month. Finally, from September to December the only month with significant revenues is December, September and October are the months which appear to have the least median revenues of the year.

Based on these observations a good recommendation would be to release a movie in the summer, from May to July, or tentatively December, the rest of the months don't have comparable medians in gross revenue. Based on this analysis, without considering any additional factors or subsetting this would be the best recommendation.

Figure 9 also shows an interesting trend, the first half of the month appears to have higher median revenues than the second half, most peak medians are in the first half. So the recommendation could be expanded to releasing movies during the summer within the first half of the month to maximize the chance of a higher gross revenue, without consdering the genre of the film.

If a more precise recommendation by genre were required, more data processing would be required to have the revenues by genre and by month and day. Therefore, considering the same top 10 genres that were determined in the previous parts of the analysis, a long format datatable is created with the gross revenue of each film to summarise by the mean gross by genre.

```
# Use only the subset of data with gross revenue information
movies_gross = movies_merged[!is.na(movies_merged$Gross),]
# Create the genres matrix from the binary vector
genres.matrix = do.call(rbind, movies_gross$Genre.binary)
row.names(genres.matrix) = movies_gross$Title
colnames(genres.matrix) = unique.genres
# Delete genres without films
genres.matrix = genres.matrix[,colSums(genres.matrix) > 0]
# Subset only the top 10 genres
genres.matrix = genres.matrix[,colnames(genres.matrix) %in% genres.top10$Genre]

# Create the long dataframe by adding data for each genre
long.df = data.frame()
l = nrow(genres.matrix)
for (i in 1:ncol(genres.matrix)){
    long.df[(l*i-(l-1)):(l*i),"Genre"] = colnames(genres.matrix)[i]
    long.df[(l*i-(l-1)):(l*i),"Month"] = movies_gross$Month
    long.df[(l*i-(l-1)):(l*i),"Day"]   = movies_gross$Day
    long.df[(l*i-(l-1)):(l*i),"Gross"] = genres.matrix[,i]*movies_gross$Gross
}

plot.df = ddply(long.df,c("Month","Day","Genre"),summarise,mean=mean(Gross))
plot.df = plot.df[-which(is.na(plot.df$Month)),]
#plot.df = plot.df[-which(plot.df$Genre == "Short" | plot.df$Genre == "Documentary"),]
```

A facet grid is created with genre by row and month by column, allowing a piecewise approximation of the mean gross revenue of each genre along the year, allowing better identification of the best release months by genre. It also evidences that the cleanup by removing observations with a mismatch between the Year and Release variable reduced significantly the number of available data for some genres, most notably the Documentary and Short film genre, as seen in Figure 10.

A genre-based recommendation for the top 10 genres could be made with Figure 10, with the disclaimer that it is based on mean gross revenue across all years. For some genres, like Crime, Documentary and Short there is not enough information to make an acceptable recommendation.

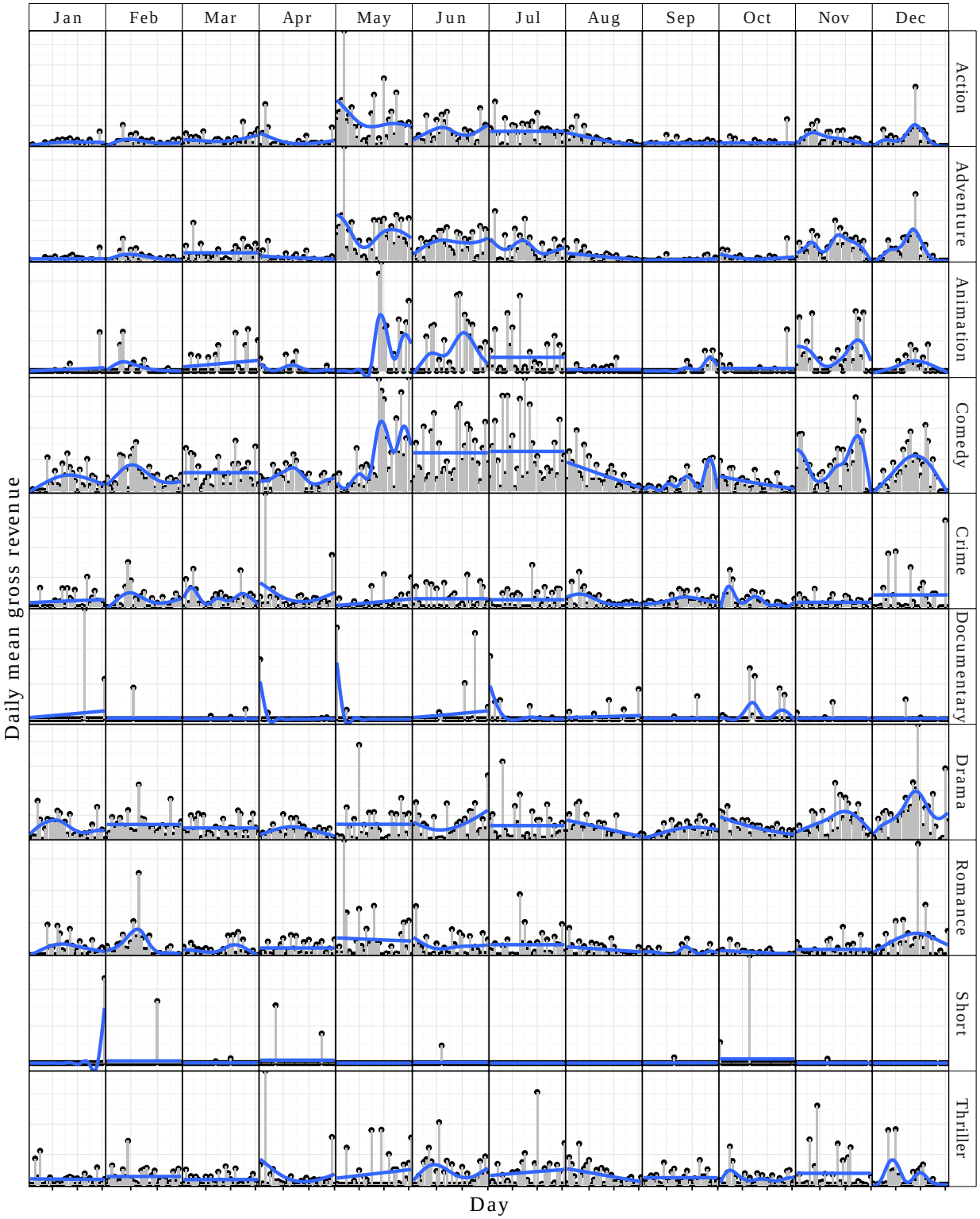

Figure 10: Mean Gross Revenue Throughout the Year by Genre

The recommendations for the top 10 genres can be summarized as follows:

1. Action: it is best to release the movie between May and July, preferably May. During the winter it is best to release the movie in mid-December.
2. Adventure: the same recommendation for Action, it is best to release it between May and July or alternatively in December during the winter.
3. Animation: the same three months from May to July seem to be the best, if the movie is to be released in the winter it would be best to release it in November.
4. Comedy: May to July keep being the best months for release, for the rest of the year November and December would also be recommended.
5. Crime: there is not enough information to give a good recommendation, the trend is pretty flat all across the year.
6. Documentary: there is not enough information to give a recommendation.
7. Drama: the best months to release a Drama film are November and December, the rest of the year is pretty flat on the trend.
8. Romance: the best months for release are February and December.
9. Short: there is not enough information to give a good recommendation.
10. Thriller: from June to August seem to be the best months to release a movie from this genre, although by no significant difference to the rest of the months.

Ratings Correlation Analysis (Assignment 6)

It was then requested to determine the relationship between the several rating variables from IMDB and Rotten Tomatoes, investigating the pairwise relationships between these two variables, also relating them with the Gross revenue. The pairwise relationships can be visualized with a matrix of scatter plots, this can be easily accomplished with the GGally package.

First a subset of data is obtained with only the information related to rating variables, and only the complete observations are kept, as to have only films with all ratings and have a better understanding of the correlation between the related variables. A correlation matrix is then calculated and visualized using the corrplot library. Figure 11 shows the correlation between all variables in an intuitive way.

```
movies_ratings = movies_merged[,c("imdbRating","imdbVotes",
                                  "tomatoMeter","tomatoRating",
                                  "tomatoReviews","tomatoFresh",
                                  "tomatoRotten","tomatoUserMeter",
                                  "tomatoUserRating","tomatoUserReviews",
                                  "Gross")]
movies_ratings = movies_ratings[complete.cases(movies_ratings),]
corr_matrix = cor(movies_ratings)
```

The graphical relationship between each pair of variables can be seen in the matrix of scatter plots in Figure 12, in which the diagonal also shows the distribution of each variable. Linear regression lines were added to display a general measure of the correlation between variables. As seen in both Figures 11 and 12, there is a strong positive correlation between the ratings from the users in imdb (imdbRating) and the critic and user ratings in Rotten tomatoes (tomatoMeter, tomatoRating, tomatoUserMeter and tomatoUserRating). Also, as expected, there is a negative correlation between the tomatoRotten variable and the overall rating of the movie in both imdb and Rotten Tomatoes.

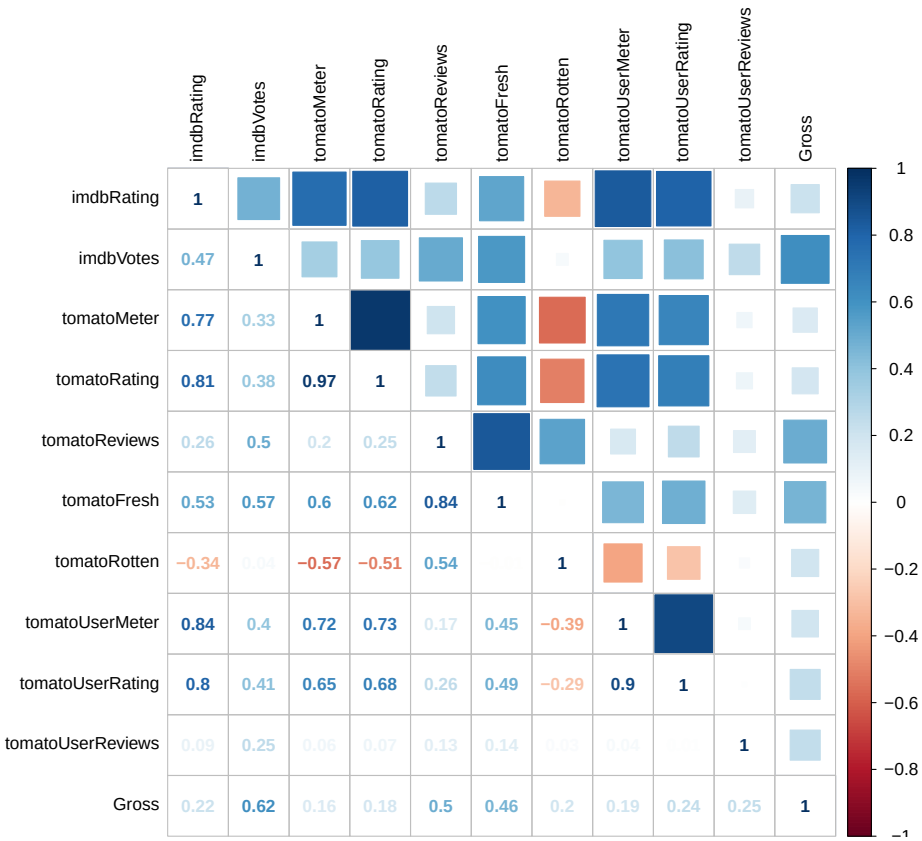

Figure 11: Correlation Plot for Selected Rating Variables

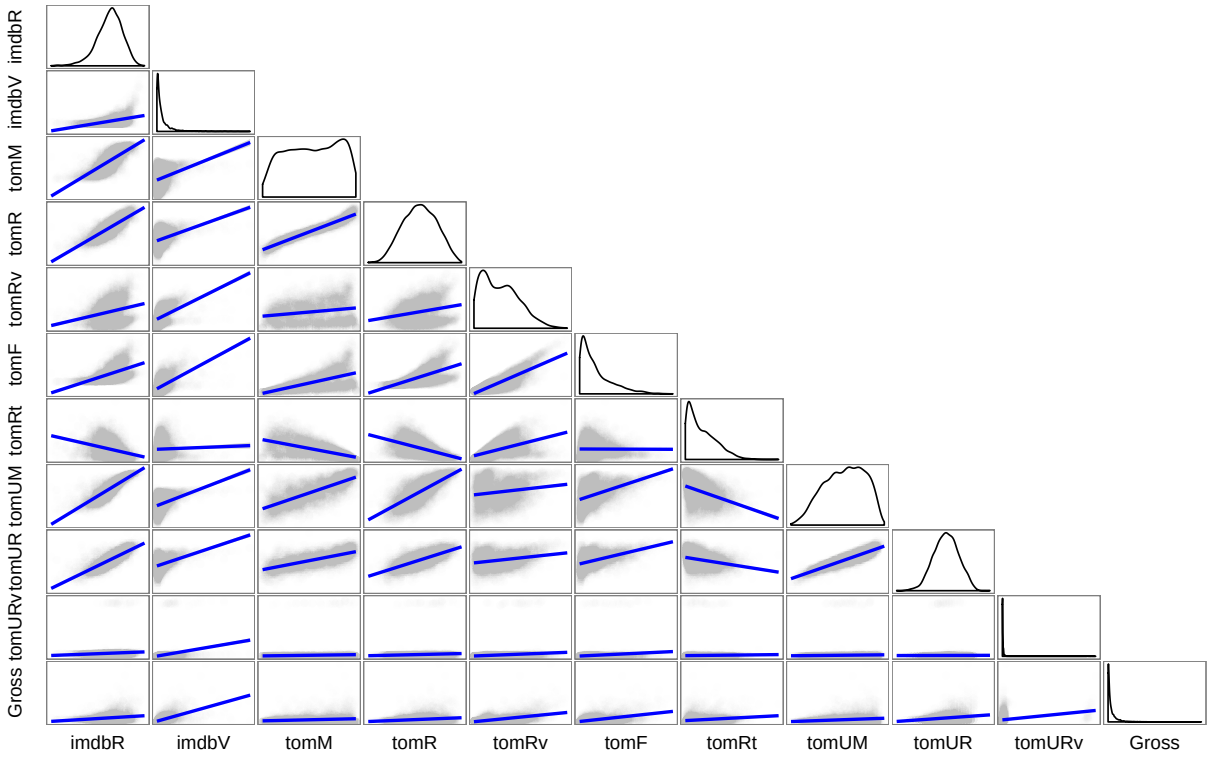

Figure 12: Matrix Plot for Selected Rating Variables

On the Figure, the variable names are shortened: imdbR = imdbRatings, imdbV = imdbVotes, tomM = tomatoMeter, tomR = tomatoRating, tomRv = tomatoReviews, tomF = tomatoFresh, tomRt = tomatoRotten, tomUM = tomatoUserMeter, tomUR = tomatoUserRating, tomURv = tomatoUserReviews.

There are also some variables that just show the number of votes received, like imdbVotes and tomatoReviews, which both seem to have the highest correlation with the Gross variable, closely followed by the tomatoFresh variable. This indicates that the gross revenue of a film is more correlated to the amount of viewers it had (which most likely reflects on the number of ratings made), the number of critics that reviewed the film and the amount of positive reviews received by critics (tomatoFresh).

Surprisingly, both plots also show a weak positive correlation with the tomatoRotten variable, which seemingly indicates that the more reviewed a movie is, either positively or negatively, there is more correlation with the gross revenue. Users reviews also seem to have a higher impact than critics reviews, as the votes by users in IMDB shows the highest correlation with the gross revenue, as seen in both Figures 11 and 12.

Figure 12 also shows interesting behaviors in the distribution of each variable, some variables show a more normal distribution, like imdbRating, tomatoRating and tomatoUserRating. This indicates that the ratings are normally distributed, most films have an average score in both IMDB and Rotten Tomatoes, while there are few movies with near-perfect ratings and few movies with very bad ratings.

Other variables, like imdbVotes, tomatoReviews, tomatoFresh, tomatoRotten and tomatoUserReviews have a highly right-skewed distribution. This just indicates that most films share a very near number of reviews, with just a few films having significantly more reviews than others. The number of critic and user reviews thus seems to be centered around a very specific mode. Finally, the variables tomatoMeter and tomatoUserMeter have a pretty uniform distribution, showing that for Rotten Tomatoes ratings the scores are pretty evenly distributed along the scale, showing just a slight increase towards the best ratings on the right of the scale.

Awards Variable Analysis (Assignment 7)

It was requested to create a three dimensional binary vector whose first component represents if the movie has no nominations or awards, the second component representes some nominations and awards and the third component represents many nominations or awards. As that binary vector would indicate three levels of a factor, it is easier to define it declare it directly as a factor, which makes it easier to use for visualization with ggplot.

To do this, the Awards variable has to be processed to extract only the total number of nominations and awards for each movie. Considering that in the strings there are only numbers for the several types of awards and nominations, the strings can be split to an array of strings, then converted to numbers (the text will become NA), and finally summed to get the total number of awards and nominations for each movie.

```
awards.split = strsplit(movies_merged$Awards,split = " ")
awards.numeric = lapply(awards.split,as.numeric)
movies_merged$Awards.sum = as.numeric(lapply(awards.numeric, function(x){
    return(sum(x[which(!is.na(x))]))
}))
```

To empirically examine the distribution of the gross revenue with the number of awards, 3 categories are defined: “None”, with movies with no awards or nominations; “Few”, wtih movies with less than 5 total awards and nominations; and “Many”, with movies with more than 5 total awards and nominations.

```
movies_merged$Awards.binary =
  as.factor(as.character(lapply(movies_merged$Awards.sum, function(x){
    if(x == 0){ return("None") }
    else if (x < 5) { return("Few") }
    else { return("Many") }
})))
```

These fuzzy factors are then used to visualize the distribution of the gross revenue.

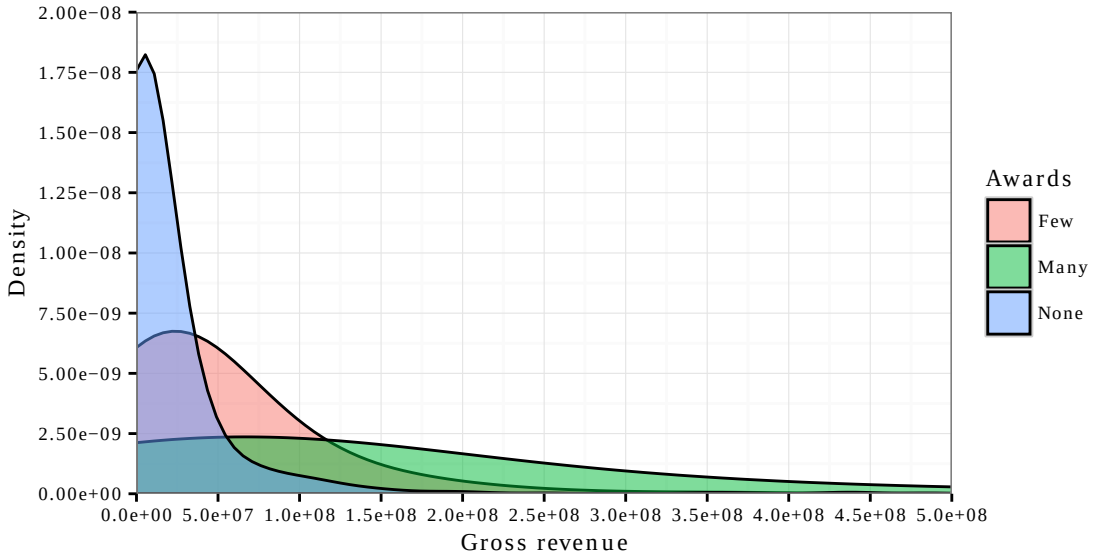

Figure 13: Distribution of Gross Revenue by Number of Awards

The number of ocurrences of each is also of use

```r
# Print the number of occurences of each
table(movies_merged$Awards.binary)
```

```
##
##   Few  Many  None
##  7356  4595 22626
```

The distribution can be alternatively visualized with boxplots.

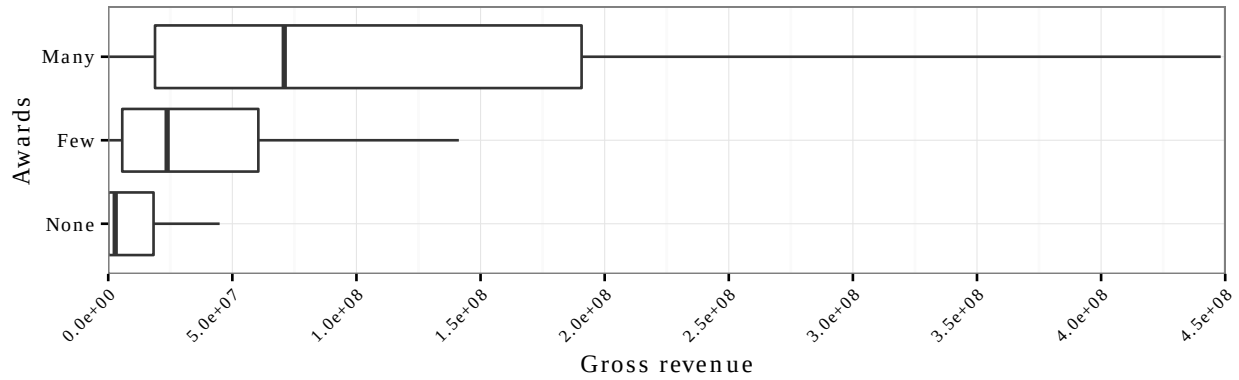

Figure 14: Boxplots of Gross Revenue by Number of Awards

As can be seen from Figures 13 and 14, there is a clear separation between the distribution of the gross revenue of the 3 groups. Movies with many awards have significantly higher gross revenue than movies with few or none awards. The median for the group of movies with many awards is beyond the third quartile of the group of movies with few awards, the same happens between the latter and the group of movies with no awards. Therefore, the amount of nomination and awards does seem to impact the gross revenue of the film.

As the total number of nomination and awards was calculated and included in the dataset, the correlation between the number of nominations and awards and the gross revenue can be visualized in a scatter plot (Figure 15). The regression line was made with LOESS (local polynomial regression), and shows a quadratic trend, the gross revenue increases with increasing number of awards until a maximum at about 250 awards, then it decays slowly.

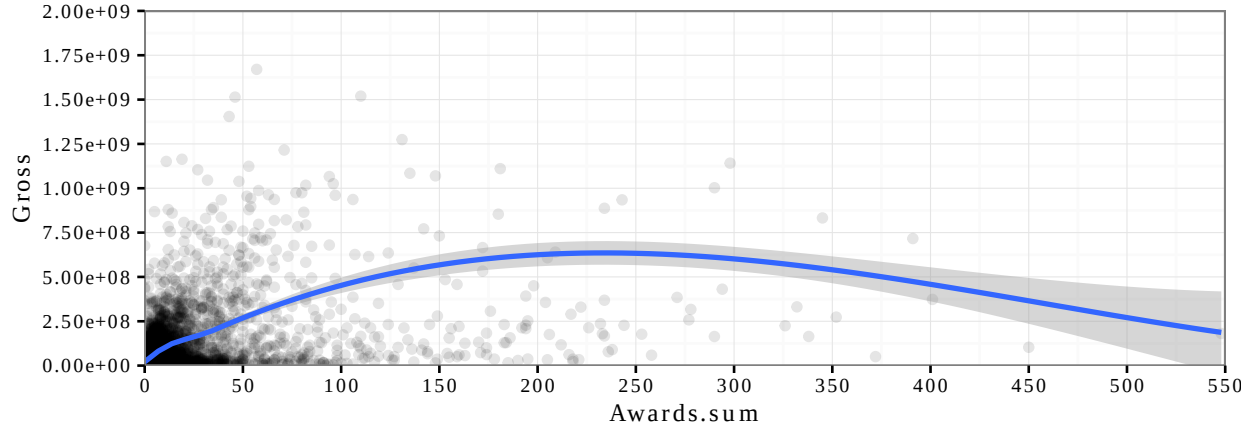

Figure 15: Gross Revenue vs Total Number of Nominations and Awards

Based on the figures presented it can be concluded that the movies with many awards tend to have higher gross revenues, in Figure 13 this is evidenced by the longer right tail and higher density with respect to the other 2 categories at higher revenues; in Figure 14, it is evidenced by the significantly larger interquartile range and evidently higher median; finally, in Figure 15 it is shown that for increasing number of awards, the gross revenue tends to be higher for most of the data, and for the few movies with an extremely large number of awards there is a very large spread of gross revenues.

Expected and Unexpected Insights (Assignment 8)

Expected Insight 1: Movies in English

Based on the movies in the data it is expected that English is the dominant language as the market is highly driven by the movies produced in the United States. It is possible to visualize this clearly by creating a matrix similar to the one used in previous sections for the Genres, but in this case for the languages of the movies.

```
languages.split = lapply(strsplit(movies_merged$Language,split=","),str_trim)
unique.languages = str_trim(sort(unique(abind(languages.split))))

movies_merged$Language.binary = (lapply(languages.split,function (x){
    as.numeric(unique.languages %in% x)
}))

languages.matrix = do.call(rbind, movies_merged$Language.binary)
colnames(languages.matrix) = unique.languages
row.names(languages.matrix) = movies_merged$Title

languages.sum = stack(sort(colSums(languages.matrix),decreasing=TRUE))
```

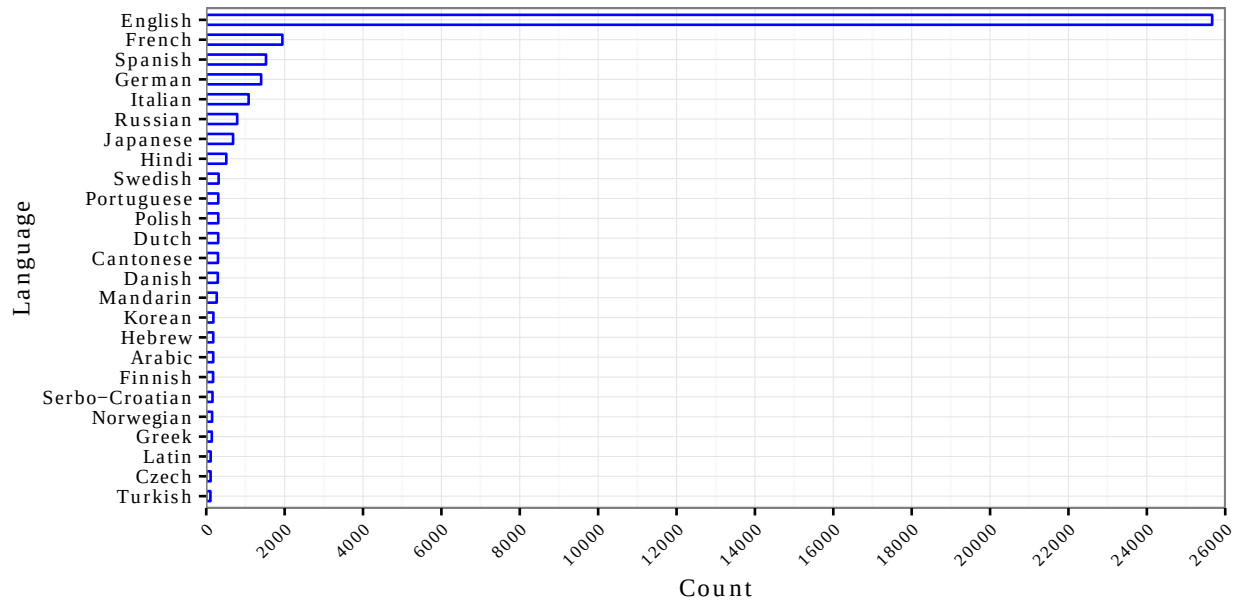

Figure 16: Top 25 Languages in the Data

As expected, English completely dominates the rest of the languages by a huge margin, having about 13 times more movies than the French, which is the second most common language in the movies. This was expected as the USA dominates the film market, or the source of the data is heavily biased towards acquisition of information related to American movies only. The same can be done with the country variable to confirm this insight.

```
countries.split = lapply(strsplit(movies_merged$Country,split=","),str_trim)
unique.countries = str_trim(sort(unique(abind(countries.split))))

movies_merged$Country.binary = (lapply(countries.split,function (x){
    as.numeric(unique.countries %in% x)
}))

countries.matrix = do.call(rbind, movies_merged$Country.binary)
colnames(countries.matrix) = unique.countries
row.names(countries.matrix) = movies_merged$Title

countries.sum = stack(sort(colSums(countries.matrix),decreasing=TRUE))
```

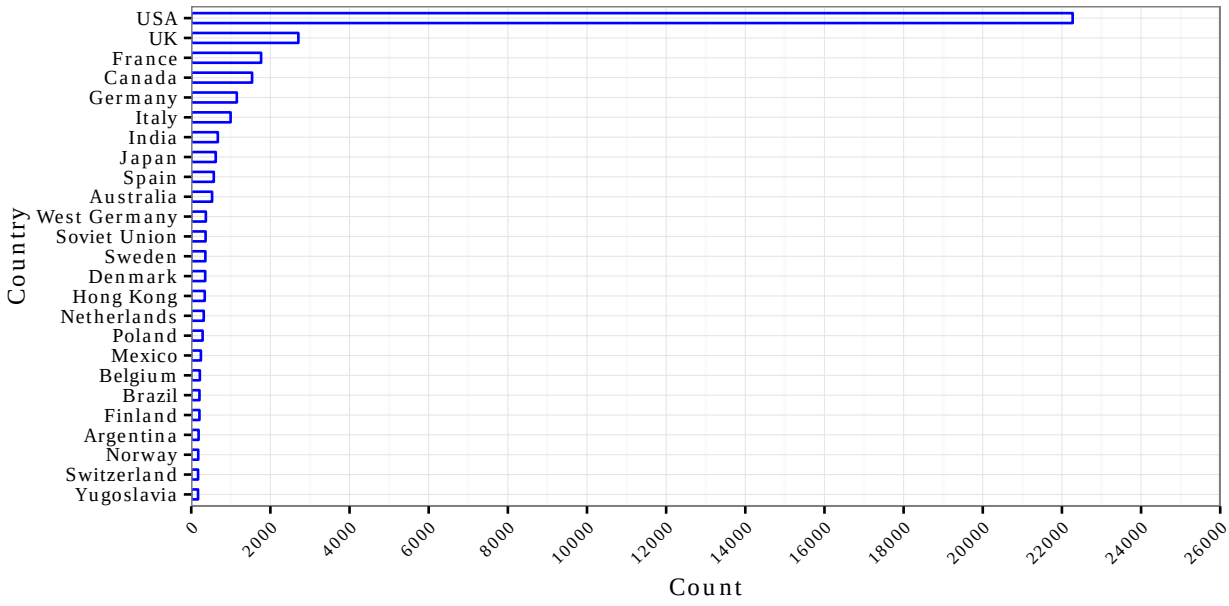

Figure 17: Top 25 Countries in the Data

This confirms that the USA dominates the market for the movies in the dataset. As mentioned before this may be due to the process for data acquisition, as IMDB and Rotten Tomatoes are both websites in English it is likely that several movies from other countries are not included in the data, which may be a bias. However, both figures (16 and 17) confirm the expected distribution of movies by country and language.

Expected Insight 2: Movie Franchises in Top by Gross Revenue

The title of the movies can be mined to get the most common words, which provide an overview of the most recurring themes. Text mining is used to generate a wordcloud of the most common words in the titles, for this only movies with gross revenue information were used, in order to get two different wordclouds, one for all the movies and one for the top 10% of those movies by gross revenue. This processing is done with R's "tm" and "wordcloud" library.

```
library(tm)
library(wordcloud)
# Create a corpus from the titles with revenue information
mineText = function(data,column.name){
  text = Corpus(VectorSource(data[,column.name]))
  # Convert all words to lower case
  text = tm_map(text, content_transformer(tolower))
  # Remove any numbers
  text = tm_map(text, removeNumbers)
  # Remove punctuation
  text = tm_map(text, removePunctuation)
  # Remove common stopwords in English
  text = tm_map(text, removeWords, stopwords("english"))
  # Remeove unnecessary whitespace
  text = tm_map(text, stripWhitespace)
  # Build a word frequency table
  text.matrix = TermDocumentMatrix(text)
  # Remove sparse terms
  # Convert it to a matrix
  text.matrix = as.matrix(text.matrix)
  # Calculate the row sums
  text.sum = sort(rowSums(text.matrix),decreasing=TRUE)
  # Create a data frame from the sums
  text.df = data.frame(word = names(text.sum),freq=text.sum)
}
data = movies_merged[which(!is.na(movies_merged$Gross)),]
wordcloud.data.all = mineText(data,"Title")
```

Figure 18 shows the word cloud for all of the movies with non-NA values in the Gross variable. The same procedure was repeated with the data for the top 10% movies by revenue. The resulting wordlcouds can be seen in Figures 18 and 19.

It can be seen in Figure 18 that love is a common word in the titles for these movies, which is somehow expected as a large amount of movies are related to this topic. Surprisingly, it can also be seen that the word movie itself is very recurrent in the titles. Looking at the worldcloud allows for an overview of the most recurring themes in the movies used to generate the cloud, themes like war and love are pretty dominant.

On the other hand, Figure 19 shows that in the top 10% movies by revenue, movie franchises are very common. We can see words for franchises such as Harry Potter, Batman, X-men, Twilight, Pirates of the Caribbean, Spiderman, Shrek, Terminator and Indiana Jones, amongst others. It was expected that some of these movie franchises were very common in the top 10% movies by revenue. Also, the word "part" is by itself very common, confirming that in the top movies by gross there are not one or two franchises but several movie franchises, as expected.

Figure 18: Title Wordcloud for Movies with Revenue Information

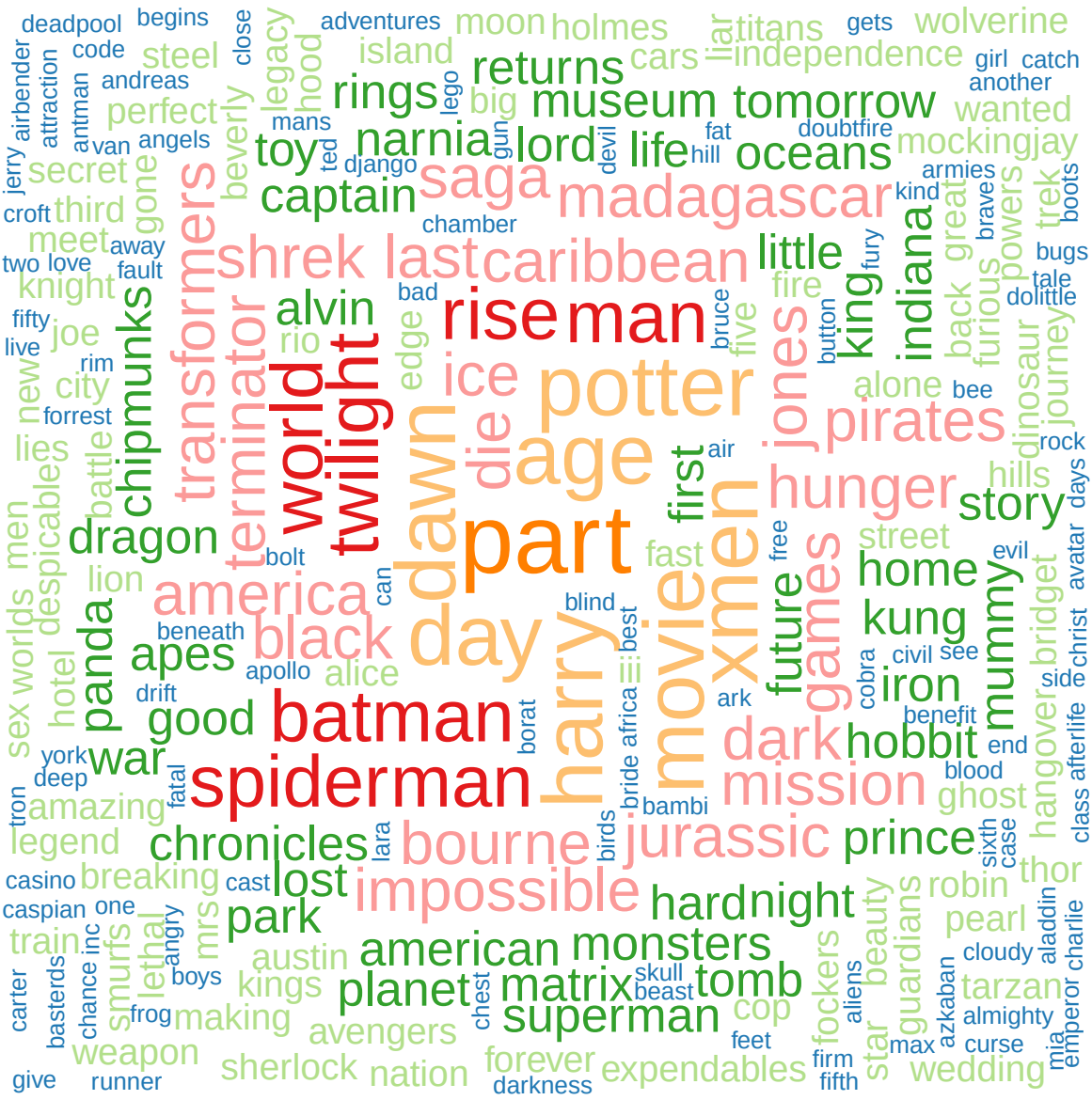

Figure 19: Title Wordcloud for Top 500 Movies by Revenue

Unexpected Insight 1: Plot Is Not Very Descriptive

It could be expected that in some months of the year there are certain recurring themes in the plots of the movies. For example, love themes for February, horror for November because of Halloween and Christmas for December. To analyze this, the plot text for the movies with release date at these months was text mined, creating the three wordclouds seen in Figures 20, 21 and 22.

```
movies.February = movies_merged[which(movies_merged$Month == "Feb"),]
wordcloud.data.february = mineText(movies.February,"Plot")

movies.November = movies_merged[which(movies_merged$Month == "Nov"),]
wordcloud.data.november = mineText(movies.November,"Plot")

movies.December = movies_merged[which(movies_merged$Month == "Dec"),]
wordcloud.data.december = mineText(movies.December,"Plot")
```

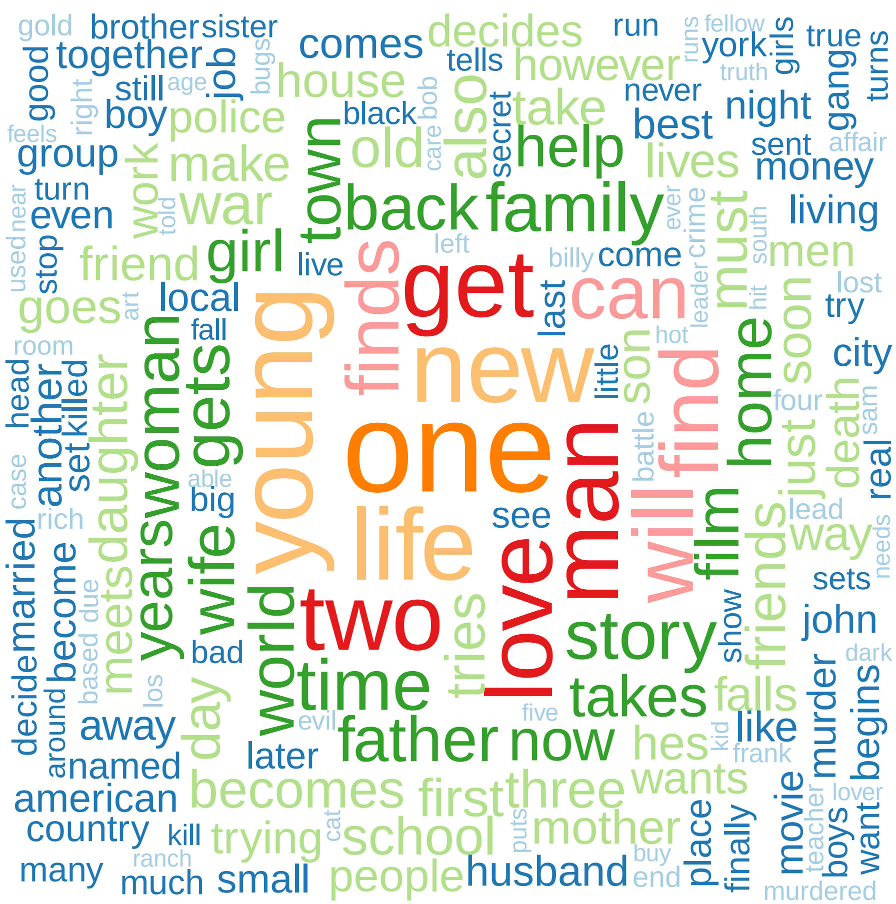

Figure 20: Wordcloud for Plot for Movies in February

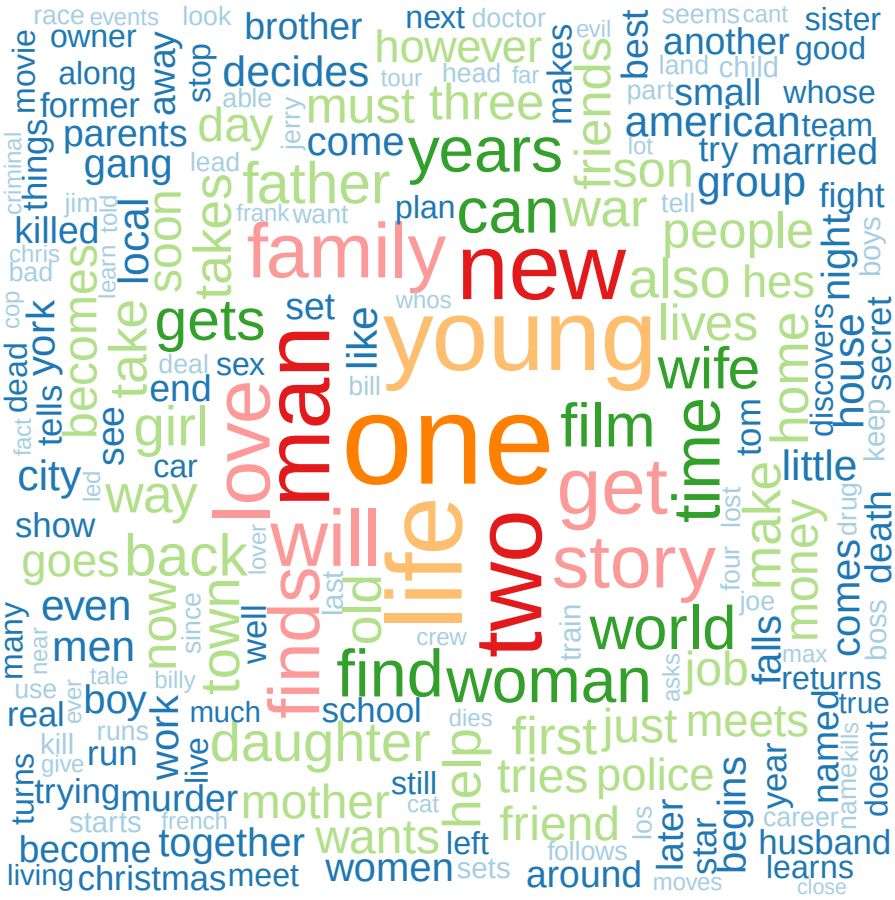

Figure 21: Wordcloud for Plot for Movies in November

Figure 22: Wordcloud for Plot for Movies in December

Looking at the wordcloud for each month, it is really hard to distinguish if there are specific seasonal trends in the plots for the movies, none of the wordclouds seem to convey any specific information on the most common theme during these 3 months, which can be further confirmed by looking at the 30 most common words by frequency.

```
plot.topwords = cbind(wordcloud.data.february[1:30,],
                      wordcloud.data.november[1:30,],
                      wordcloud.data.december[1:30,])
rownames(plot.topwords) = seq(1,30)
colnames(plot.topwords) = c("Feb.Word","Feb.Freq",
                            "Nov.Word","Nov.Freq",
                            "Dec.Word","Dec.Freq")
print(plot.topwords)
```

```
##    Feb.Word Feb.Freq Nov.Word Nov.Freq Dec.Word Dec.Freq
## 1       one      306      one      451      one      390
## 2     young      246     life      363     life      355
## 3      life      234    young      339      new      325
## 4       new      233      two      323      two      309
## 5       man      222      man      315    young      305
## 6       get      215      new      311     love      298
## 7      love      213     will      261      man      271
## 8       two      209      get      260      get      255
## 9      will      187     love      260     will      240
## 10     find      183   family      251     find      215
## 11    finds      163    story      239     time      206
## 12      can      160    finds      226    finds      189
## 13     time      150     find      205    story      188
## 14   family      148     time      202      can      187
## 15    story      144    woman      192   family      187
## 16     gets      137      can      189    years      181
## 17    woman      133    years      188     back      179
## 18     wife      132    world      183      old      175
## 19    world      132     film      181     help      174
## 20     back      130     wife      177     wife      166
## 21   father      128     gets      170     gets      163
## 22     film      123     back      169     home      162
## 23     home      123   father      169      way      159
## 24     town      122     girl      165    world      158
## 25     girl      120     town      161    woman      157
## 26    years      119     also      159   father      154
## 27      now      118 daughter      158     girl      154
## 28     help      115      war      155      son      153
## 29    takes      115     make      153     also      149
## 30     also      114      old      152 daughter      147
```

This is an unexpected finding as it seems that the plots are not very descriptive, this could be further confirmed by looking at the wordclouds for movies with three genres that should be significantly different in plots, to determine if this behavior remains. The chosen genres for this analysis were Drama, Action and Fantasy

```
movies.Action =
  movies_merged[which(genres.matrix[,which(colnames(genres.matrix) == "Action")] == 1),]
```

```
wordcloud.data.action = mineText(movies.Action,"Plot")

movies.Drama =
  movies_merged[which(genres.matrix[,which(colnames(genres.matrix) == "Drama")] == 1),]
wordcloud.data.drama = mineText(movies.Drama,"Plot")

movies.Fantasy =
  movies_merged[which(genres.matrix[,which(colnames(genres.matrix) == "Fantasy")] == 1),]
wordcloud.data.fantasy = mineText(movies.Fantasy,"Plot")
```

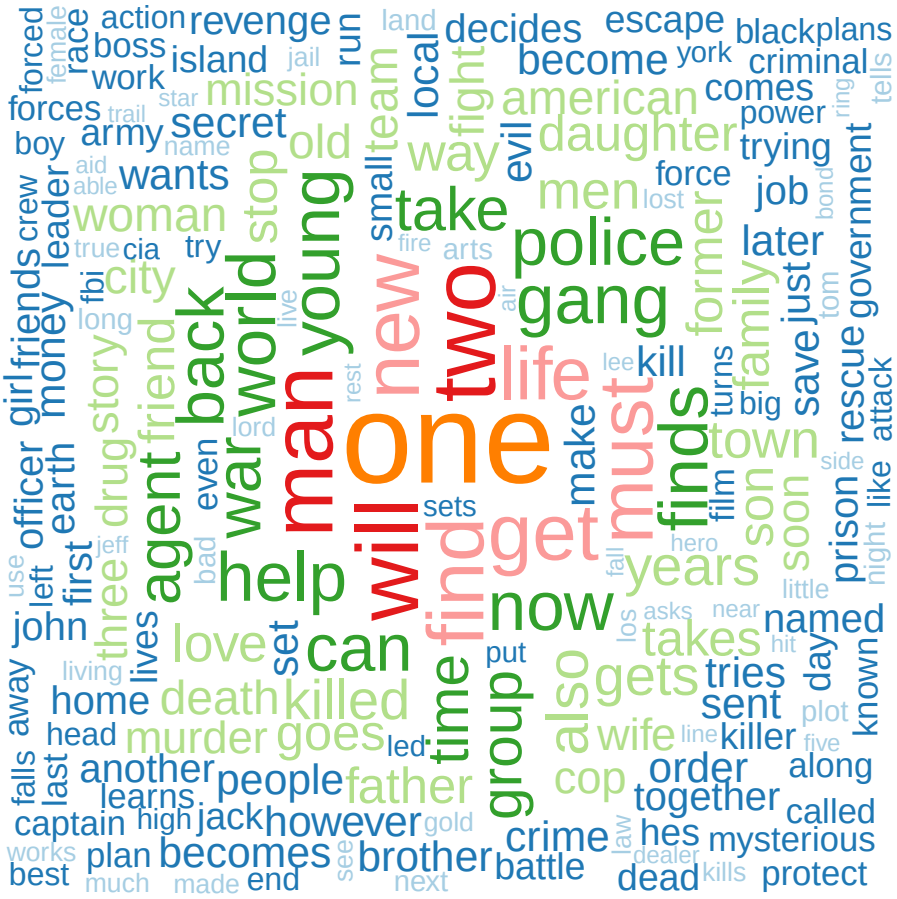

Figure 23: Wordcloud for Plot for Action Movies

Figure 24: Wordcloud for Plot for Drama Movies

Figure 25: Wordcloud for Plot for Fantasy Movies

The first 30 words for each genre are also displayed.

```
plot.topwords = cbind(wordcloud.data.action[1:30,],
                      wordcloud.data.drama[1:30,],
                      wordcloud.data.fantasy[1:30,])
rownames(plot.topwords) = seq(1,30)
colnames(plot.topwords) = c("Action.Word","Action.Freq",
                            "Drama.Word","Drama.Freq",
                            "Fantasy.Word","Fantasy.Freq")
print(plot.topwords)
```

```
##    Action.Word Action.Freq Drama.Word Drama.Freq Fantasy.Word Fantasy.Freq
## 1          one         731       life       2415          one          226
## 2          two         479        one       2189         life          207
## 3          man         475      young       2139        young          189
## 4         will         472       love       1850          man          173
## 5          new         426        man       1663         find          152
## 6         find         423        new       1622         will          150
## 7          get         413        two       1523        world          149
## 8         must         398       will       1468         love          138
## 9         life         379     family       1396          new          136
## 10       young         350      story       1261         time          135
## 11        gang         346        get       1146          can          125
## 12        help         342      woman       1143         back          121
## 13         now         342     father       1138          two          120
## 14       world         342       wife       1118        finds          118
## 15       finds         340      finds       1087         must          114
## 16         can         337       find        989        woman          113
## 17      police         330      years        960         evil          109
## 18        back         328      lives        950       family          104
## 19         war         317       time        948         help          101
## 20        take         295      world        927        story          101
## 21       agent         291        son        917       father           89
## 22        time         280       girl        897        years           87
## 23       group         276        war        896          get           85
## 24        also         274        can        890          old           85
## 25       years         268       home        885          day           82
## 26      killed         266        old        853         girl           82
## 27        gets         261     mother        851        death           79
## 28         men         247       back        835          now           73
## 29         way         247   daughter        828         town           72
## 30        love         246       film        826   mysterious           70
```

As can be deducted from these lists, the description for the plots are very vague and general. It is expected to find that the word love appears in all of them (also confirming that this is a very common theme), but other than that these 30 top words don't show much correlation to the genre, and don't covey much of the plot of the films with respect to the genre. This is somehow unexpected because the plots are supposed to convey the setting of the film in as few words as possible, but the text mining as is done here doesn't seem to show that.

Perhaps a deeper, much more complex text mining analysis could be done to try to determine if the words in the plot correlate with the genre of the movie, but at this level of text analysis, removing stop words and numbers it doesn't seem like that is the case. Perhaps the purpose of the plot is not to spoil the movie, but at least it was expected to see that for certain genres there are several ocurrences of catch-words or phrases related to those genres.

Appendices

Code for Figure 1

```
library(ggplot2)
ggplot(movies_merged,aes(Year,..count..))  +
  geom_freqpoly(binwidth=1,color="blue",size=1) +
  scale_x_continuous(limits=c(1885,2020),expand=c(0,0),
                     breaks=round(seq(1885,2020,by=5))) +
  scale_y_continuous(limits=c(0,1500),expand=c(0,0),
                     breaks=seq(0,1500,by=125)) +
  theme_bw() +
  theme(text=element_text(size=9,family="Bookman"),
        axis.text.x=element_text(angle=45,size=9,hjust=1),
        axis.text.y=element_text(size=9,hjust=1),
        panel.border=element_rect(colour ="black",size=1,fill=NA)) +
  labs(y="Count")
```

Code for Figure 2

```
ggplot(movies_merged,aes(Runtime,..density..)) +
  geom_histogram(fill="white",color="black",binwidth = 15) +
  scale_x_continuous(limits=c(0,240),expand = c(0,0),breaks=seq(0,240,by=15)) +
  theme_bw() +
  geom_density(color="blue",size=1) +
  scale_y_continuous(expand=c(0,0),limits=c(0,0.03),
                     breaks=seq(0,0.03,by=0.005)) +
  theme(text=element_text(size=9,family="Bookman"),
        axis.text.x=element_text(size=9,hjust=1),
        axis.text.y=element_text(size=9,hjust=1),
        panel.border=element_rect(colour ="black",size=1,fill=NA)) +
  labs(y="Density")
```

Code for Figure 3

```
mm = movies_merged[which(movies_merged$Runtime < 180),]
ggplot(mm,aes(x=Year,y=Runtime)) +
  geom_point(alpha=1/75) +
  coord_polar() +
  geom_smooth() +
  scale_x_continuous(limits=c(1920,2020),breaks=seq(1920,2020,by=10)) +
  scale_y_continuous(limits=c(0,180),breaks=seq(0,180,by=30))+theme_bw() +
  theme(text=element_text(size=7,family="Bookman"),
        axis.text.x=element_text(size=7,hjust=1),
        axis.text.y=element_text(size=7,hjust=1),
        panel.border=element_rect(colour ="white",size=1,fill=NA),
        panel.grid.minor = element_line(colour = "grey", size=0.5),
        panel.grid.major = element_line(colour = "grey", size=0.5))
```

Code for Figure 4

```
ggplot(movies_merged,aes(x=Year,y=Runtime)) +
  stat_summary(aes(x=Year,y=movies_merged$Runtime),
               fun.y="median",geom="line",size=1,color="blue") +
  scale_x_continuous(limits=c(1920,2020),expand=c(0,0),breaks=seq(1920,2020,by=10)) +
  scale_y_continuous(limits=c(0,100),expand=c(0,0),breaks=seq(0,100,by=10)) +
  theme_bw() +
  theme(text=element_text(size=9,family="Bookman"),
        axis.text.x=element_text(size=9,hjust=1),
        axis.text.y=element_text(size=9,hjust=1),
        panel.border=element_rect(colour ="black",size=1,fill=NA))
```

Code for Figure 5

```
ggplot(movies_merged,aes(Budget,..count..)) +
    geom_histogram(fill="white",color="black",binwidth = 10e06) +
    coord_cartesian(xlim=c(0,3e08),ylim=c(0,1200)) +
    scale_x_continuous(breaks=seq(0,3e08,by=50e06)) +
    scale_y_continuous(expand=c(0,0), breaks=seq(0,1200,by=200)) +
    theme_bw() +
    geom_density(color="blue",size=1) +
    theme(text=element_text(size=9,family="Bookman"),
          axis.text.x=element_text(size=9,hjust=1),
          axis.text.y=element_text(size=9,hjust=1),
          panel.border=element_rect(colour ="black",size=1,fill=NA)) +
    labs(y="Frequency")
```

Code for Figure 6

```
ggplot(movies_merged[as.integer(movies_merged$Decade) >= 6
                     & !is.na(movies_merged$Budget),],
       aes(x = Budget, y = Runtime))+facet_grid(Decade ~ .) +
    geom_point(alpha=1/10,color="blue",size=1) +
    geom_smooth(size=0.75,color="grey",alpha=0.50,se=FALSE) +
    coord_cartesian(ylim=c(45,150),xlim=c(0,1.5e08)) +
    scale_y_continuous(limits=c(45,150),breaks=seq(45,150,by=15),expand=c(0.125,0.125)) +
    scale_x_continuous(expand=c(0,0),breaks=seq(0,1.5e08,by=25e06)) +
    theme_bw() +
    theme(text=element_text(size=8,family="Bookman"),
          axis.text.x=element_text(size=6,hjust=1),
          axis.text.y=element_text(size=6,hjust=1),
          panel.border=element_rect(colour ="black",size=0.25,fill=NA),
          panel.margin = unit(c(0),"cm"),
          strip.background = element_rect(color="black",fill="white"))
```

Code for Figure 7

```
ggplot(genres.top10, aes(x = reorder(Genre,Frequency), y = Frequency)) +
  geom_bar(stat="identity",fill="white",color="blue",size=0.75) +
  scale_x_discrete(expand=c(0.0025,0.0025)) +
  scale_y_continuous(expand=c(0,0),limits=c(0,16000),breaks=seq(0,16000,by=2000)) +
  theme_bw() +
  labs(x = "Genre", y = "Count") +
  theme(text=element_text(size=8,family="Bookman"),
        axis.text.x=element_text(angle=45,size=9,hjust=1),
        axis.text.y=element_text(size=9,hjust=1),
        panel.border=element_rect(colour ="black",size=0.5,fill=NA),
        panel.margin = unit(c(0),"cm"),
        strip.background = element_rect(color="black",fill="white"))
```

Code for Figure 8

```
gross.matrixmelt$Genre<- factor(gross.matrixmelt$Genre,levels =
                  topgenre.medians$Genre[order(topgenre.medians$median)])

ggplot(gross.matrixmelt,aes(x=Genre,y=value)) +
  geom_boxplot(outlier.shape = NA) +
  coord_flip(ylim=c(0,9e08)) +
  scale_y_continuous(breaks=seq(0,9e08,length.out = 10),expand=c(0,0)) +
  theme_bw() +
  labs(x = "Genre", y = "Gross revenue") +
  theme(text=element_text(size=8,family="Bookman"),
        axis.text.x=element_text(angle=45,size=9,hjust=1),
        axis.text.y=element_text(size=9,hjust=1),
        panel.border=element_rect(colour ="black",size=0.5,fill=NA),
        panel.margin = unit(c(0),"cm"),
        strip.background = element_rect(color="black",fill="white"))
```

Code for Figure 9

```
ggplot(day.bymonth,aes(x = as.numeric(Day), y = median)) +
    geom_bar(stat="identity",fill="blue") +
    facet_wrap(~ Month) +
    theme_bw() +
    scale_x_continuous(breaks=seq(0,30,by=5)) +
    labs(y = "Median gross revenue", x = "Day") +
    theme(text=element_text(size=10,family="Bookman"),
          axis.text.x=element_text(angle=45,size=8,hjust=1),
          axis.text.y=element_text(size=8,hjust=1),
          panel.border=element_rect(colour ="black",size=0.5,fill=NA),
          panel.margin = unit(c(0),"cm"),
          strip.background = element_rect(color="black",fill="white"))
```

Code for Figure 10

```
ggplot(plot.df,aes(x = as.numeric(Day), y = mean)) +
    geom_point(size=1.5) +
    geom_bar(stat="identity",position = "dodge", fill="grey") +
    facet_grid(Genre~Month,scales="free") +
    geom_smooth(se=FALSE,size=1.25) +
    labs(y = "Daily mean gross revenue", x = "Day") +
    scale_y_continuous(expand = c(0,0)) +
    scale_x_continuous(expand = c(0,0)) +
    theme_bw() +
    theme(text=element_text(size=16,family="Bookman"),
          axis.text.x=element_blank(),
          axis.text.y=element_blank(),
          axis.ticks.y = element_blank(),
          panel.border=element_rect(colour ="black",size=0.5,fill=NA),
          panel.margin = unit(c(0),"cm"),
          strip.background = element_rect(color="black",fill="white"))
```

Code for Figure 11

```
library(corrplot)
par(mar=c(3,3,3,3))
corrplot.mixed(corr_matrix,upper="square",tl.pos="lt",diag="l",tl.cex=1,
               tl.col="black",number.cex=1,cl.cex=1)
colnames(movies_ratings) = c("imdbR","imdbV","tomM","tomR",
                             "tomRv","tomF","tomRt","tomUM",
                             "tomUR","tomURv","Gross")
```

Code for Figure 12

```
library(GGally)

lowerFn <- function(data, mapping, method = "lm", ...) {
  p <- ggplot(data = data, mapping = mapping) +
    geom_point(colour = "grey", alpha = 1/75) +
    geom_smooth(method = method, color = "blue", ...)
  p
}

ggpairs(movies_ratings,lower=list(continuous=wrap(lowerFn, method="lm")),
        upper=list(continuous=wrap("blank"))) + theme_bw() +
    theme(text=element_text(size=16,family="Bookman"),
          axis.text.x=element_blank(),
          axis.text.y=element_blank(),
          panel.grid.major=element_blank(),
          panel.grid.minor=element_blank(),
          axis.ticks.y=element_blank(),
          axis.ticks.x=element_blank())
```

Code for Figure 13

```
ggplot(movies_merged[which(!is.na(movies_merged$Gross)),],
       aes(Gross,fill=factor(Awards.binary))) +
  geom_density(position="dodge",alpha=0.5,adjust=5) +
  scale_y_continuous(expand=c(0,0),breaks=seq(0,2e-08,by=2.5e-09)) +
  scale_x_continuous(expand=c(0,0),breaks=seq(0,5e08,by=0.5e08)) +
  coord_cartesian(xlim=c(0,5e08),ylim=c(0,2e-08)) +
  labs(x="Gross revenue",y="Density",fill="Awards") +
  theme_bw() +
  theme(text=element_text(size=9,family="Bookman"))
```

Code for Figure 14

```
ggplot(movies_merged[which(!is.na(movies_merged$Gross)),],
       aes(y=Gross,x=reorder(factor(Awards.binary),Gross,median))) +
  geom_boxplot(outlier.color = NA) +
  coord_flip(ylim=c(0,4.5e08)) +
  scale_y_continuous(expand=c(0,0),breaks=seq(0,4.5e08,by=0.5e08)) +
  labs(x="Awards",y="Gross revenue") +
  theme_bw() +
  theme(text=element_text(size=10,family="Bookman"),
        axis.text.x=element_text(angle=45,size=7,hjust=1))
```

Code for Figure 15

```
ggplot(movies_merged[which(!is.na(movies_merged$Gross)),],aes(x=Awards.sum,y=Gross)) +
    geom_point(alpha=1/10) +
    geom_smooth(method="loess") +
    coord_cartesian(xlim=c(0,550),ylim=c(0,2e09)) +
    scale_x_continuous(expand=c(0,0),breaks=seq(0,550,by=50)) +
    scale_y_continuous(expand=c(0,0),breaks=seq(0,2e09,by=2.5e08)) +
    theme_bw() +
    theme(text=element_text(size=10,family="Bookman"))
```

Code for Figure 16

```
ggplot(languages.sum[1:25,],aes(x = reorder(ind,values), y = values)) +
  geom_bar(stat="identity",
           position = position_dodge(width = 0.8), width=0.5, fill=NA, color="blue") +
  coord_flip(ylim = c(0,26000)) +
  scale_y_continuous(expand=c(0,0),breaks=seq(0,26000,by=2000)) +
  theme_bw() +
  labs(x = "Language", y = "Count") +
  theme(text=element_text(size=10,family="Bookman"),
        axis.text.x=element_text(angle=45,size=7,hjust=1))
```

Code for Figure 17

```
ggplot(countries.sum[1:25,],aes(x = reorder(ind,values), y = values)) +
  geom_bar(stat="identity",
           position = position_dodge(width = 0.8), width=0.5, fill=NA, color="blue") +
  coord_flip(ylim = c(0,26000)) +
  scale_y_continuous(expand=c(0,0),breaks=seq(0,26000,by=2000)) +
  theme_bw() +
  labs(x = "Country", y = "Count") +
  theme(text=element_text(size=10,family="Bookman"),
        axis.text.x=element_text(angle=45,size=7,hjust=1))
```

Code for Figure 18

```
wordcloud(words = wordcloud.data.all$word,
          freq = wordcloud.data.all$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 19

```
set.seed(42)
wordcloud(words = wordcloud.data.top$word,
          freq = wordcloud.data.top$freq, min.freq = 1,
          max.words=1000, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 20

```
set.seed(42)
wordcloud(words = wordcloud.data.february$word,
          freq = wordcloud.data.february$freq, min.freq = 5,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 21

```
set.seed(42)
wordcloud(words = wordcloud.data.november$word,
          freq = wordcloud.data.november$freq, min.freq = 5,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 22

```
set.seed(42)
wordcloud(words = wordcloud.data.december$word,
          freq = wordcloud.data.december$freq, min.freq = 5,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 23

```
set.seed(42)
wordcloud(words = wordcloud.data.action$word,
          freq = wordcloud.data.action$freq, min.freq = 5,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 24

```
set.seed(42)
wordcloud(words = wordcloud.data.drama$word,
          freq = wordcloud.data.drama$freq, min.freq = 5,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```

Code for Figure 25

```
set.seed(42)
wordcloud(words = wordcloud.data.fantasy$word,
          freq = wordcloud.data.fantasy$freq, min.freq = 5,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Paired"))
```