IMDB

5000

Movie

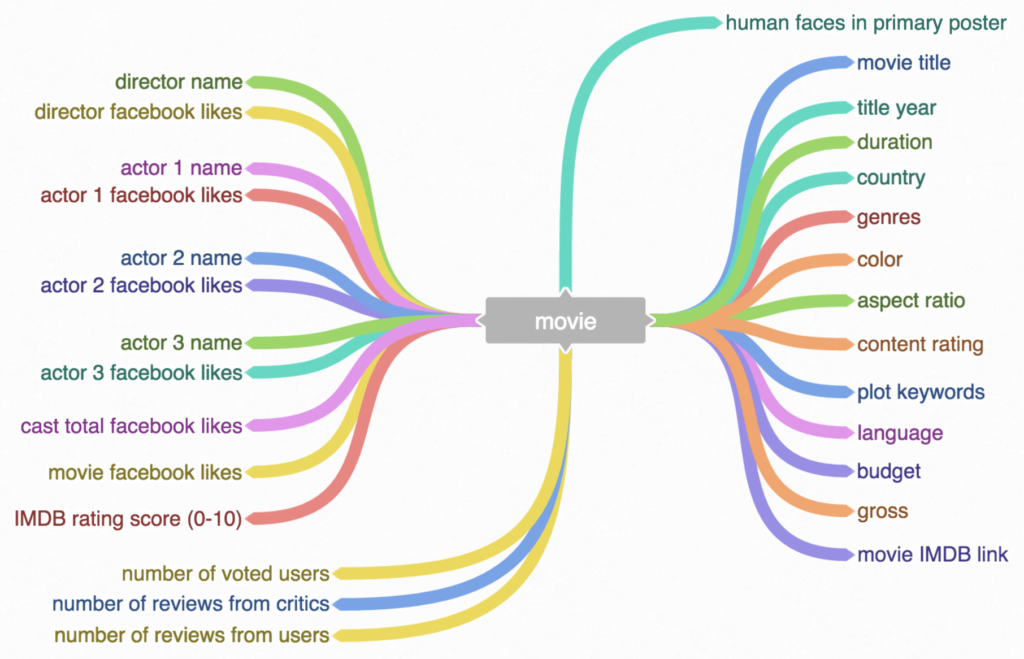
Dataset

Qiman Wang, Christian Colon, Michael Persaud

Visualization

This year famous Chinese director Stephen Chow released a movie called “The Mermaid”. The movie is about a Mermaid named Shan, is send to assassinate a real estate developer who threaten the ecosystem of her race, but ends up falling in love with him instead. The movie is very successful in china and is in movie theater worldwide. There seems to be a lot of analysis that can be done and our group decided to apply our machine learning skills to the test. We first went on kaggle and found an IMDB dataset. Some nice guy Used Scrapy tool in python to web scrape the data off IMDB’s website.

We first used R to do some data visualization and analysis. We read in the file as a csv file (Excel), then we ran the basic summary, structure and head methods to get a better understanding of the data. The dataset contains 5,043 data points and 28 variables. The variables contain information like director name, number of Facebook likes, budget, genre, etc. we were extremely interested in what makes a movie good. Different people have different taste as to if a movie is good or not. The diagram below shows on the left side, variables that are involved in the production of the movie. On the right are variables that directly relate to the movie.



color - black and white or color

director\_name – Name of the director

num\_critic\_for\_reviews – number of review

duration - Time length of movie (in minutes)

director\_facebook\_likes – number of director’s Facebook likes

actor\_3\_facebook\_likes - number of actor\_3’s Facebook likes

actor\_2\_facebook\_likes - number of actor\_2’s Facebook likes

actor\_1\_facebook\_likes - number of actor\_1’s Facebook likes

actor\_3\_name - actor 3’s name

actor\_2\_name - actor 2’s name

actor\_1\_name - actor 1’s name

gross – total revenue

genre – type of movie (Action, Adventure,Fantasy,….)

movie\_title – title of movie

num\_voted\_users – total number of voted users

cast\_total\_facebook\_likes - total number of Facebook likes of the whole cast team

facenumber\_in\_poster – number of face on the movie poster

plot\_keywords – keywords about the movie EX: war, pirate, love

movie\_imbd\_link – link of the review on IMDB

num\_user\_for\_reviews – users who wrote a review

language – language the movie was shot in

Country – Director country

Content\_rating – EX: PG-13, Rated – R

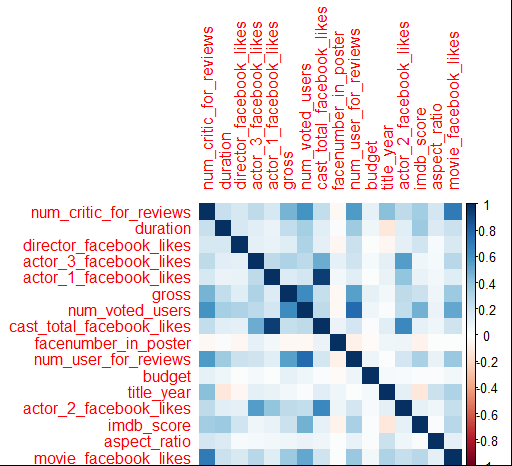
Budget – cost to make the movie

Title\_year – year the movie was released

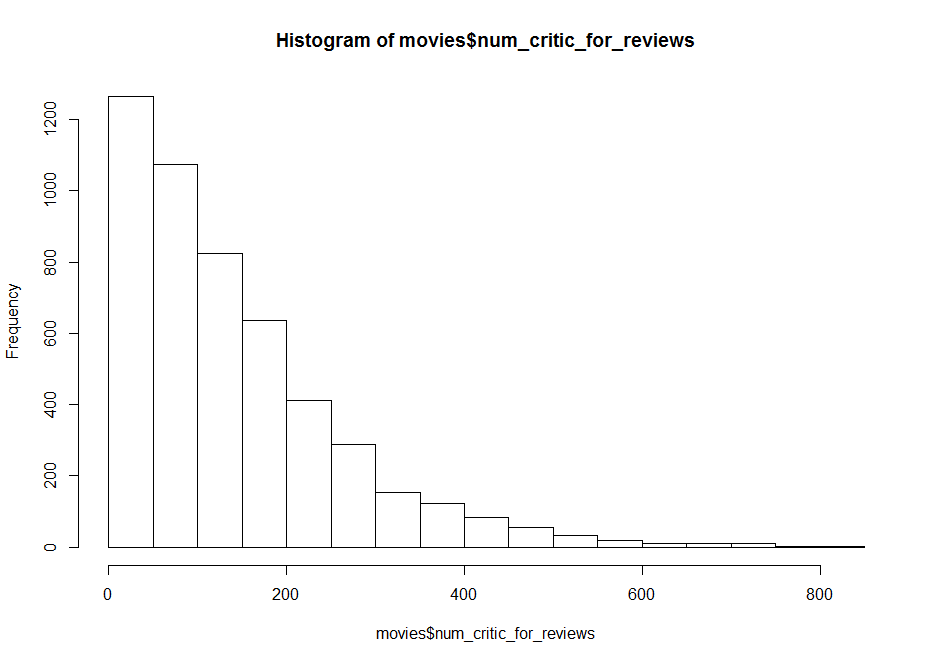
IMBD score – how good is the movie, 10 being the highest

Aspect\_ratio – width and height ratio

Movie\_facebook\_likes – how much likes does the movie have on Facebook



The diagram above shows the correlation of all numerical variables. Dark blue means very correlated and red means inversely correlated. The diagonal is dark blue because every variable is correlated to itself. One interesting observation is that the cast total like is correlated to actor 1 Facebook like. This means that the main actor contributes most of the cast likes and that actor 2 and 3 don’t have as much Facebook likes. So maybe this means the main actor can promote the movie on his/her Facebook and therefore people will go to the movie page and like it.



The diagram above shows the distribution of how many critics do movies get. We see that about 1200 movies have 0 to 50 reviews. And only about 30 movies have more than 600 reviews. This means that most movie gets less than 200 reviews on average and it’s rare to have a movie with over 600 reviews. This makes sense because only super popular movies will attract reviews. Movies like titanic in the top 250 would be in this list. This distribution follows a fractal pattern. Which is how most of the distributions are in real world. Plotting this on a log log plot and we can see a consistent pattern.

Variables sorted by number of missings:

Variable Count

gross 884

budget 492

aspect\_ratio 329

title\_year 108

director\_facebook\_likes 104

num\_critic\_for\_reviews 50

actor\_3\_facebook\_likes 23

num\_user\_for\_reviews 21

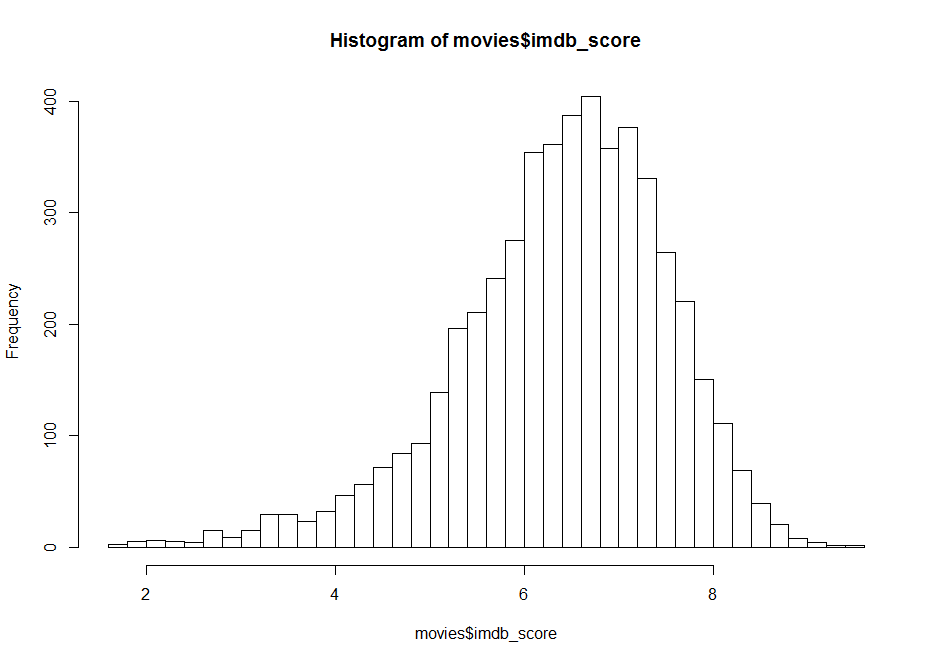
duration 15

facenumber\_in\_poster 13

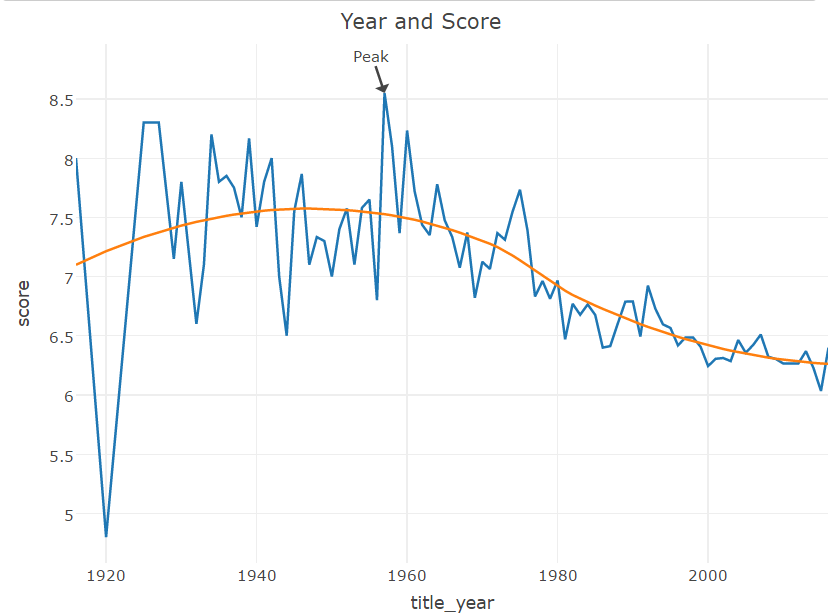
actor\_2\_facebook\_likes 13

actor\_1\_facebook\_likes 7

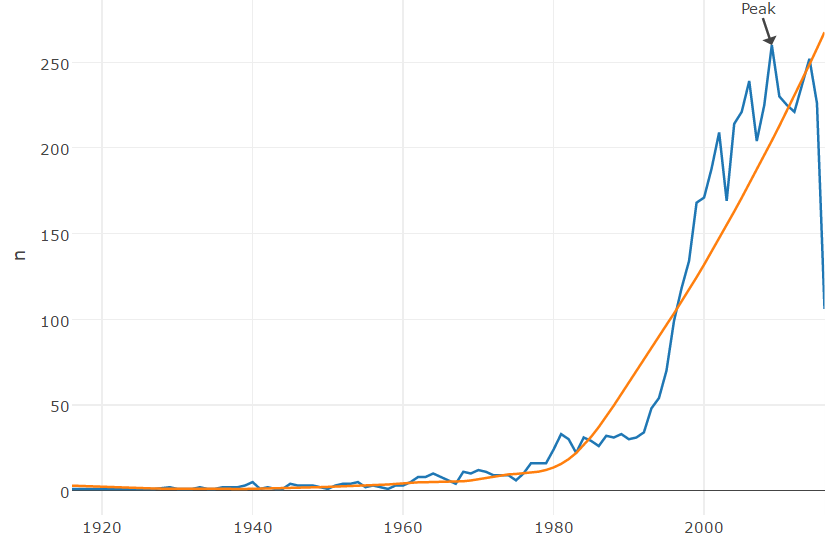
This is copied straight from the terminal and we see the variables missing the most data are gross. Which means we don’t know how much a movie profited. The next variable is budget, so it seems that the money variables are likely to be missing. Maybe the director doesn’t want people to know how good or bad a movie is.

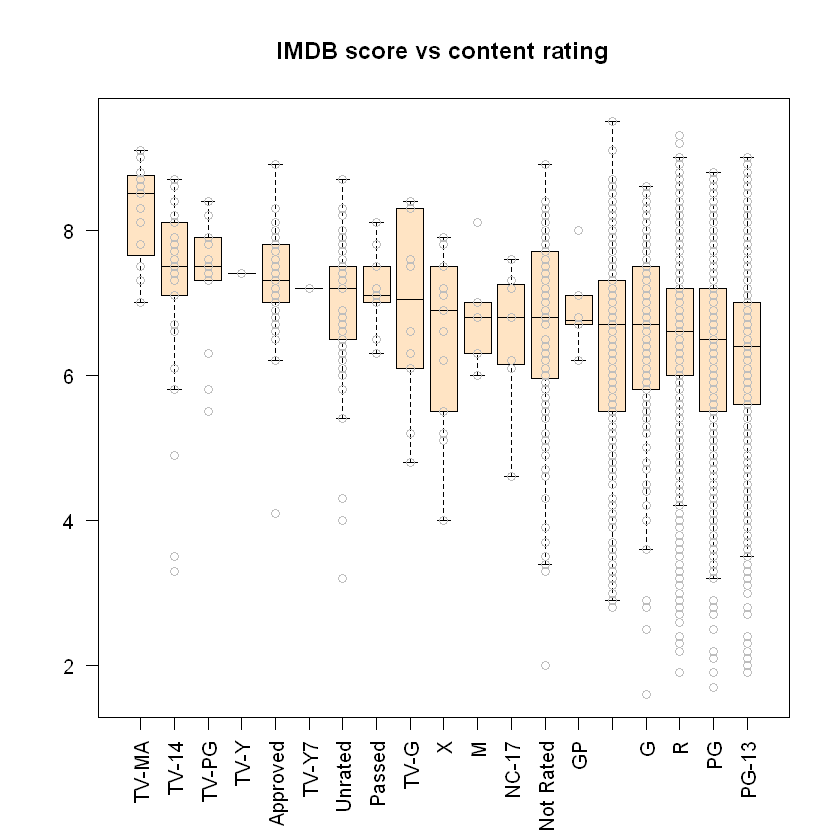
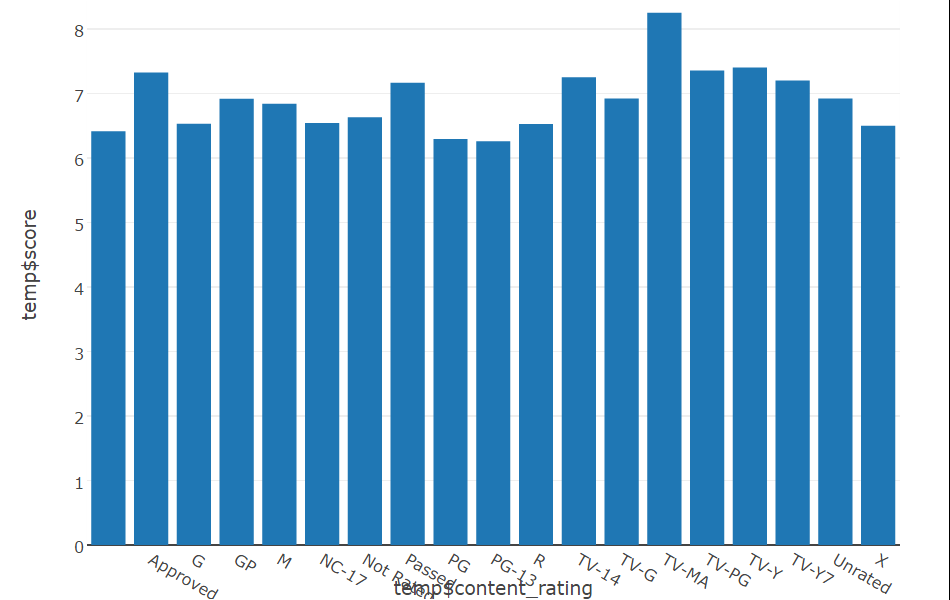


The distribution of rating of movies looks like a normal curve. Most movie are between 5 and 8. Look at the tails we see that there’s a longer tail on the left side ranging from 2 to 4. This means that it’s more likely to get bad movies than really good movies. Movies higher than 8 are listed in the IMDB top 250.

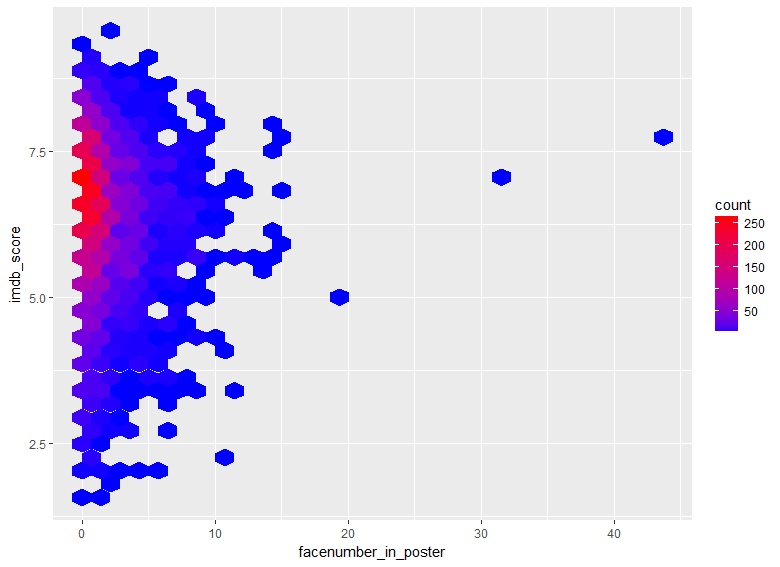


The diagram above shows the average rating of movies for the past century. We see that the average rating of movie was highest during 1950’s and is declining according to the trendline. Does this means the movies made in the past few years are really that bad? One explanation why movies in the mid 1900’s is higher is because people are more likely to watch good movies in the past. People don’t want to waste their time watching a bad movie made in the past, so most likely family or friend recommend good movies. Or maybe the person searched up on the internet resources like IMDB and watch the top 250 movies. So, the old good movie’s rating will keep getting higher. Another possibility is there’s so much movies produced recently that there’s more bad movies than good movies. The diagram below shows the number of movies produced every year. We see that there is an exponential growth in the number of movies produced. Because there’s more demand for movies and more directors, cheaper film equipment that contribute to this increase. With streaming services like Netflix and online streaming, movies are in higher demand and more people will watch movies.

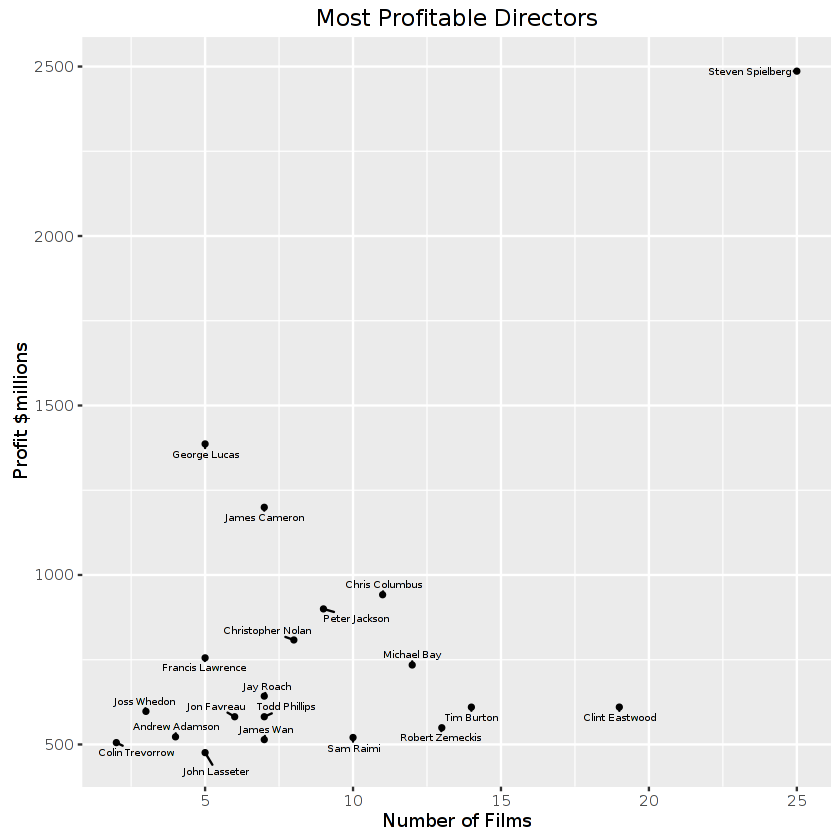




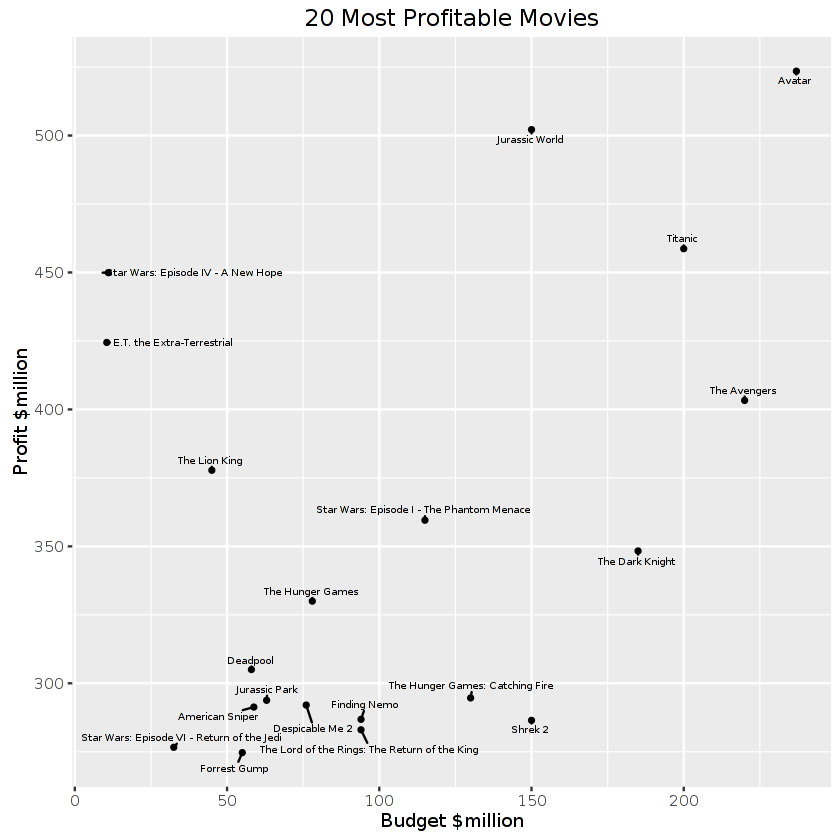
Based on this diagram we see that PG-13 have the lowest rated score. The highest genre is TV-MA which is any material that is suited for public tevelivison.



This diagram shows that movies that do good generally have less than 5 face on the poster. But of course this is not true because movies like 300, which have like 30 faces on still did really good. However it’s safe to say that movies with less face on poster are more likely to get a good rating.



Looking at the above diagram you see that Steven Spielberg made 2.5 billion dollars from being a director of 25 movies. That’s 100 million dollars per movie. On the other hand we have Clint Eastwood who has made almost 20 movies and only made about 600 million. That’s about 30 million dollars per movie. Which is still really good comparing to most other directors. The most profitable director is George lucas who made 1.3 billion in 5 movies. That’s about 260 million dollars per movie. If you want to learn how to be a good director, definitely talk to Lucas George.

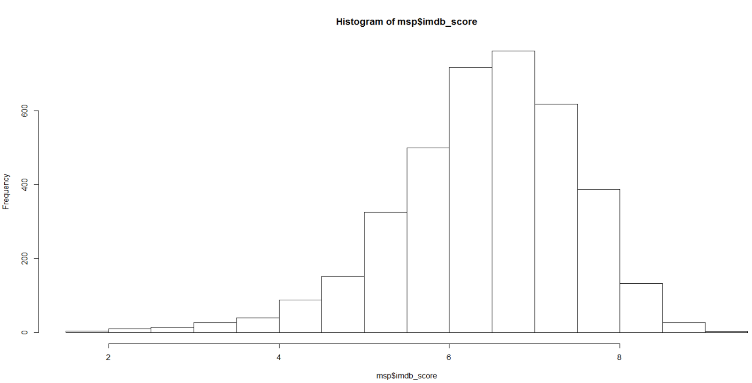


The diagram shows the highest profit movies. We see that avatar is the most expensive movie to be created. Both avengers and avatar came out in the recents years, Before that titanic was the highest budget movie ever being produced. The most profitable movie ever produced would be Star Wars 4 – a new hope. This movie’s budget is only 20 million and the profit was 450 million. So If you want to make some money, you can make a similar movie.

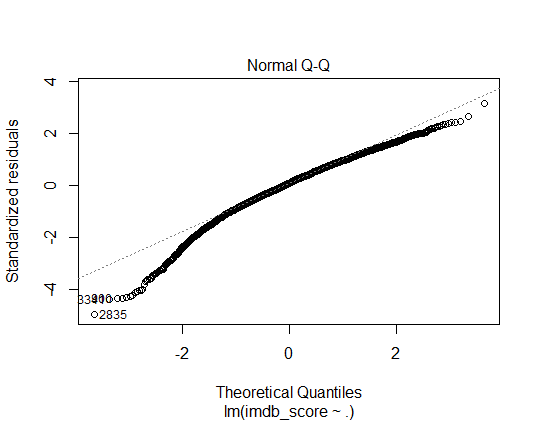
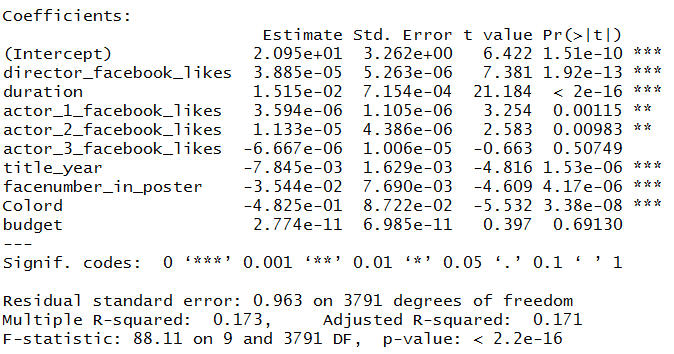
Regression Analysis

We are interested in regression for IMDB score in our dataset. Being able to predict IMDB would be very useful. Moviegoers would not have to wait for an extended period for a fair amount of movie critics to review a movie they are interested in seeing. Well to start of the analysis we take out all data points with missing attributes we also only look at movies with USA as country of origin. This leaves us with 3800 data points. The features "director\_facebook\_likes", "duration”, “actor\_1\_facebook\_likes", "actor\_2\_facebook\_likes", "actor\_3\_facebook\_likes", "title\_year", "facenumber\_in\_poster", "Colord", "budget” were chosen to regress on IMDB score. Colord is a dummy variable telling us if the movie is in color or not. These variables were chosen because this is likely the information a user has before a movie has been released for an extended period. Features like gross, number of user reviews and number of critic reviews are not useful to us. Cast total Facebook likes was not included because of its high correlation with actor 1,2 and 3 Facebook likes, excluding it helps us avoid multicollinearity.

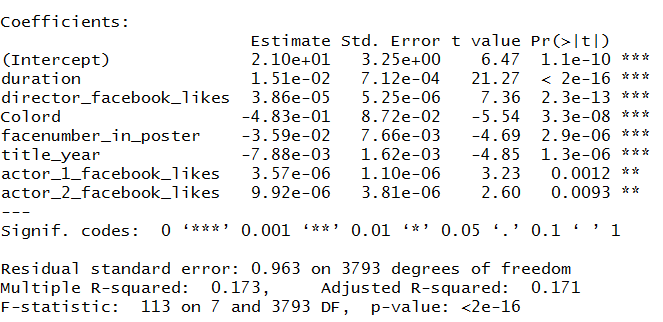
Multiple Linear Regression



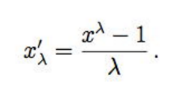
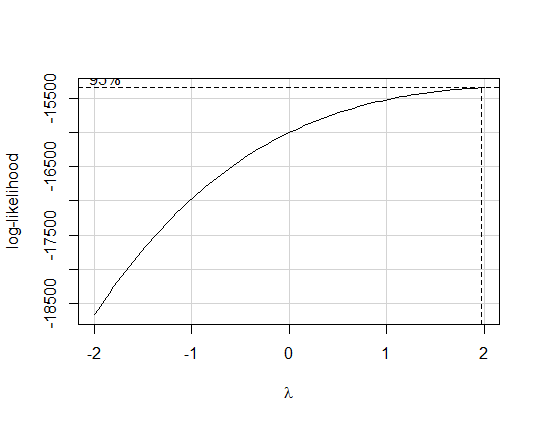
The above figure shows show that the distribution of IMDB score is left skewed. The normality of errors is most likely violated. But we still try multiple linear regression anyway. If we just regress IMDB on all features, we get this model:



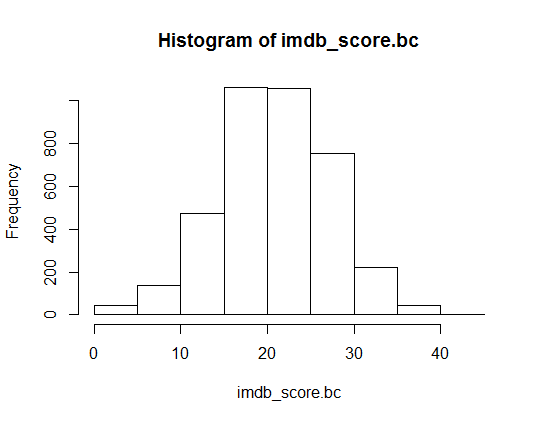
The variance explained for the full model is 17% and normality is not satisfied. We will try to use stepwise regression to make a multiple regression model. The process we will use is forward selection. Our forward selection will select models based on Akaike information criterion. After the the step regression we get :

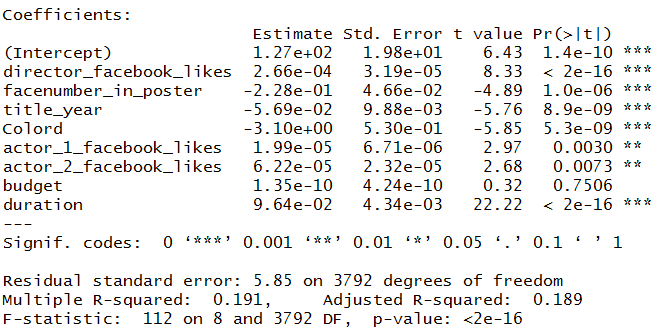


The step regression removed budget and actor 3 facebook likes. To get the normality assumption we will use the boxcox transformation on the dependent variable. The boxcox transformation can be used because our dependent variable is always positive.

 This the transformtion with a choice of lambda.

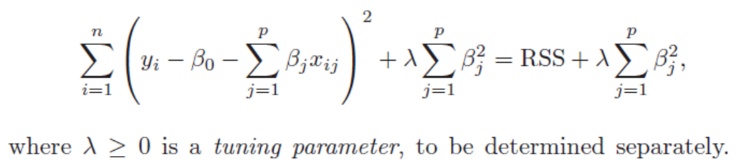
The most likely lambda is 2 from the graph.

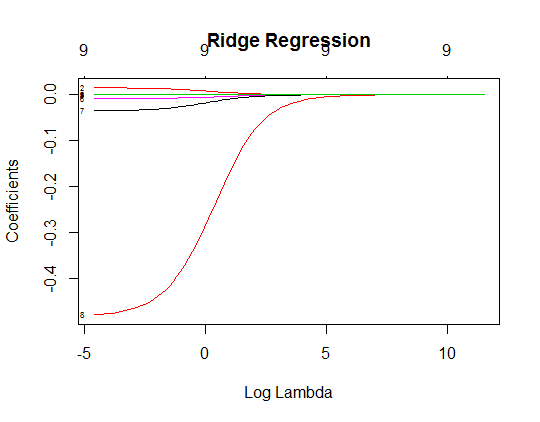


After applying the transformation on IMDB score the distribution of the data became normal. The normal QQ plot also shows that the data has become normal. Although the model is uninterpretable we can still see which variables are relevant and their effects on IMDB. From the model we can see budget is not a significant variable. Also we see that number of faces in poster and title year have a negative effect of IMDB score. This agrees with the plots of IMDB v. Number of faces in poster and IMDB v. Year. Also the movie being in color has a negative effect on IMDB rating. The other variables like director Facebook likes and actor Facebook have a postive effect on IMBD score. This model explains 19% of the variability.

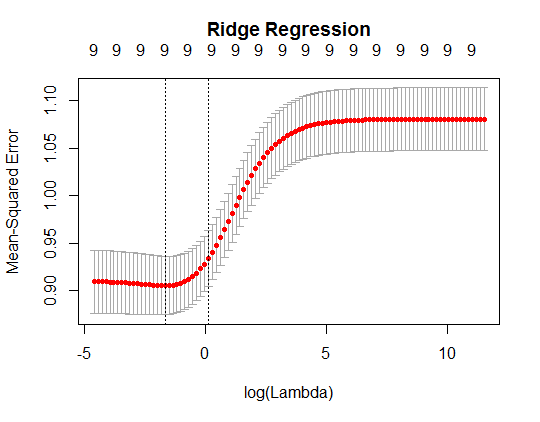
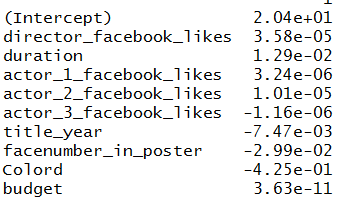
Ridge Regression

We would like to use Ridge Regression because normality is not assumed. From the Normal Q-Q plot we saw that normality is not satisfied. Ridge Regression adds a penaly term to our minimalization problem. Ridge Regression will penalize large coeffiecients due to its L2 regularization.

This is the Error we are trying to minimize.

We give a sequence lambdas to choose from. We have 100 lambdas ranging from 10000 to .01.

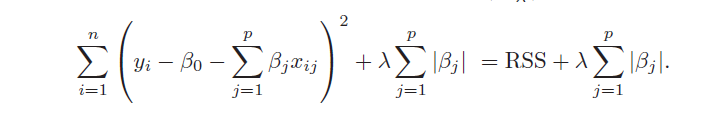
This shows the effect of lamda on the coefficients. As expected the coeffiecients shrink as lambda gets larger. The top of the plot just tells us how many variables are in the model. To select the best lambda we are going to use 10-fold cross validation. We will train on 70 % of the data and test on 30% of data. The folds are split randomly. The best MSE we get is .92. This is for when lambda is .187. Below are the coefficients to Ridge model.



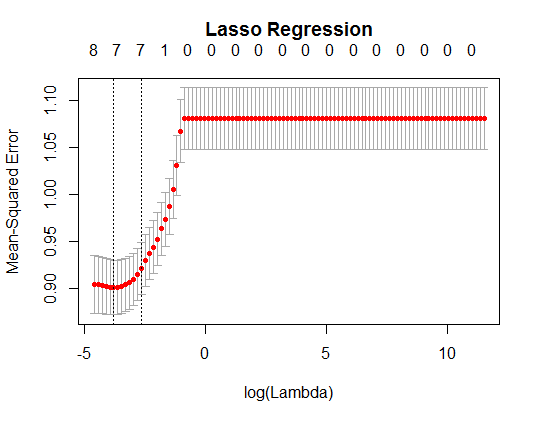
When we apply the model on the whole dataset we get a MSE of 1.207. This gives us an R squared of .23.

LASSO Regression

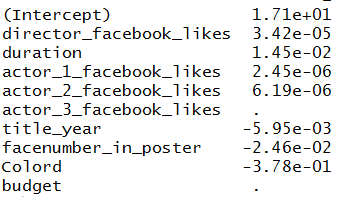
LASSO regression also adds a penalty term to the error but LASSO uses L1 regularization. It also does not assume normality. This allows coefficients to be shrunk to zero, hence LASSO regression is capable of model selection.



Again we use 100 lambdas ranging from 10000 to .01. We 10-fold cross validation to obtain best lambda.

From the plot the best lambda is .0226 and in the model we have 7 regresssors.

We can see below that the regression removed budget and actor 3 facebook likes. With the model we get an MSE of .927. The LASSO regression gives us a better fit than the Ridge Regression because although there MSE are similar, the LASSO regression will have a better fit based on its Adjusted R square. Adjusted R squared penalized for more parameters.



Movie Recommendation System

Using our dataset, we try to build a recommender system that will take in an input list of movies a user like and output a list of movie recommendation that a user might like.

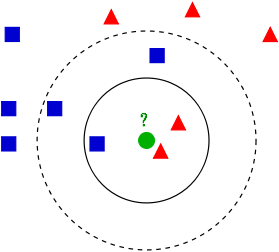
A Recommender System is a system that tries to predict a user’s preference or likeness on a given item. Using this system, a user “discover” new items that wouldn’t be previously known to them. This provides a useful alternative to just using a search algorithm where the system might have hundreds to thousands of choices. Generally, to build a recommender system there are three possible choices: Collaborative Filtering, Content-Based Filtering, and Hybrid Filtering.

Collaborative Filtering takes a user and their past preference and predicts an item given that the item is in another user’s preference. Content-Based Filtering takes a user and list of preferences and gives an item that has similar features to the item’s in preference. Hybrid Filtering is a mixture of the two filtering methods and used in many modern systems and services such as Netflix and Spotify.

Collaborative Filtering has many potential issues/problems. Three of these problems are Cold Start, Scalability, and Sparsity. Cold Starting is that to create accurate recommendation, a large initial user base is need to prime the system. Scalability is that given a large set of users and preferences, computation becomes a very important issue in queries of the system. Sparsity is that a user has only a defined preference for a relatively small number of item data. An example of this is a person only watches relatively a small part of a database of movies.

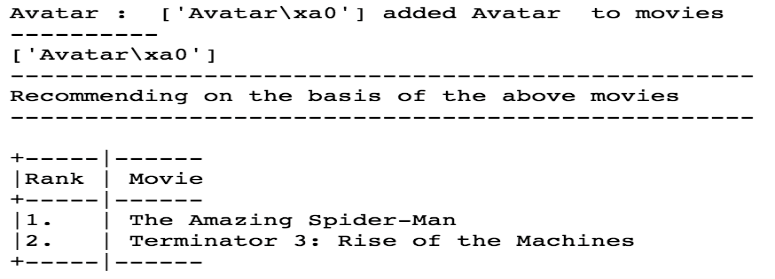
Along with this content-based filtering also has its share of issue. A key issue with content-based filtering is whether the system can learn user preferences from users' actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful, but would be much more useful when music, videos, products, discussions etc. from different services can be recommended based on news browsing.

In this project, we will be using content-based filtering because our dataset does not support individual user ratings. We’ll use a k-d tree implementation which partitions each feature of the data and creates a tree structure. This is a very common approach to building a recommender system. A k-d tree itself is a binary tree that exists in k-dimensional space. Each dimension deals with a different feature and this allows use to have a structure to group items that are similar in nature. This can be thought of as a k-dimensional tree. Searches through the tree uses the popular k-nearest neighbor’s algorithm.

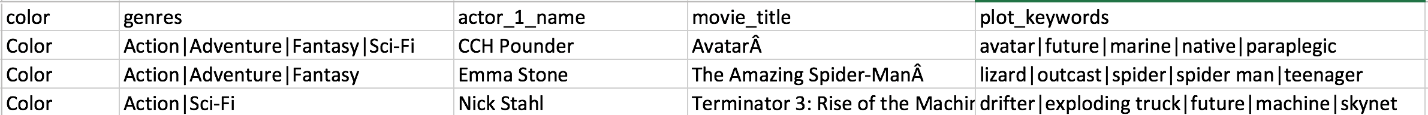


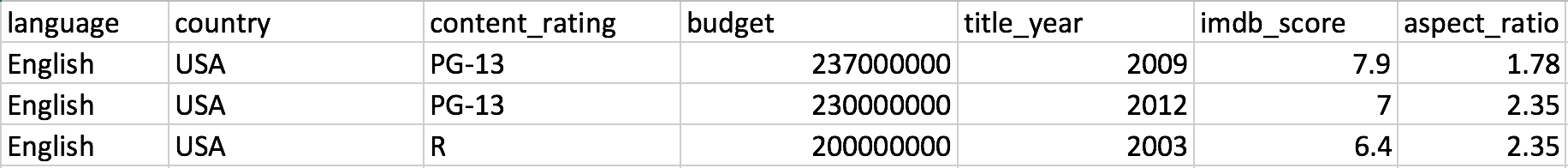
*Figure A*

In *Figure A*, the solid line refers to k=3 and the dotted line refers to k =5. If we choose k=3, the point in question will be classified as a triangle while if we choose k=5, the point in question will be classified as a square. In our use of k-NN we chose our k to equal 2. Given an input movie, we pull the two movies that are closest by computing some distance function.



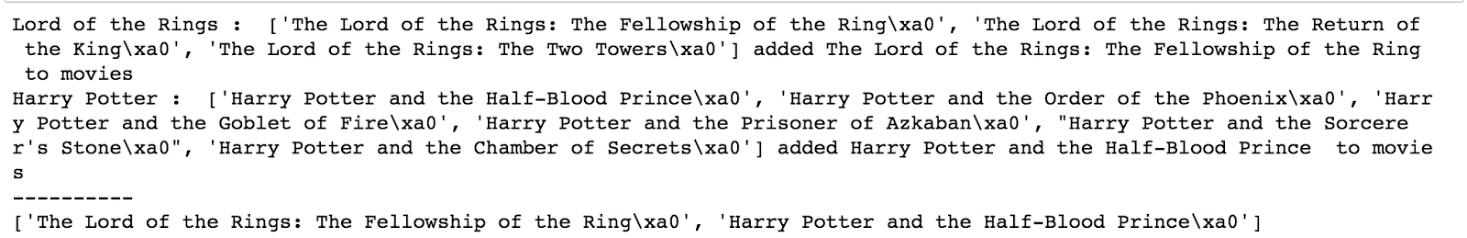
*Figure B*

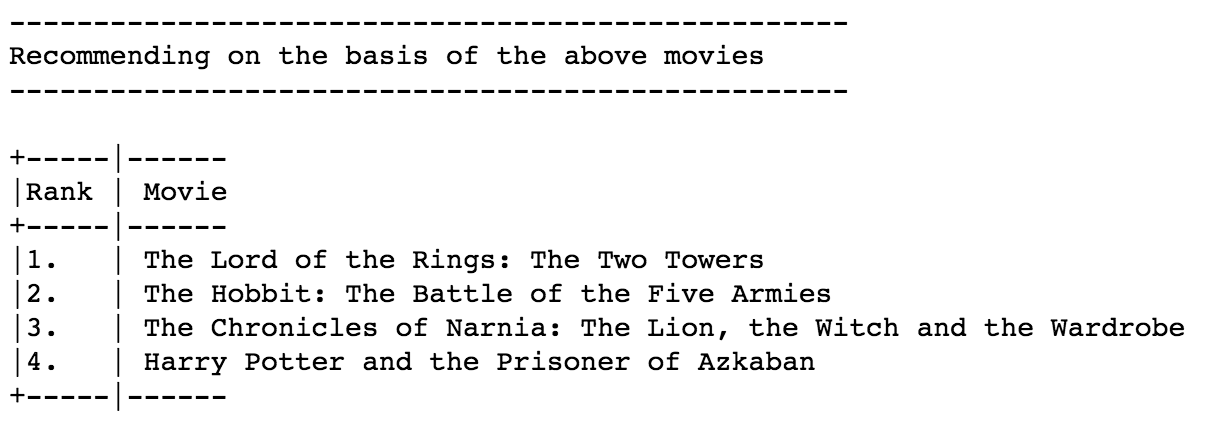




*Figure B*

As an example, we inputted the movie “Avatar and get two outputs for it: The Amazing Spider-Man and Terminator 3: Rise of the Machines. In *Figure B*, this is what outputs when we are input “Avatar” it into our program. *Figure C* is a listing of some of the features of the three movies that should correlated the most. We see they all have a strong correlation in genre, imdb score, and plot keywords which is what a person would usually use to pick a new movie to watch. This verifies that the system is working accordingly.





*Figure C*

As another example, we inputted “Lord of the Rings” and “Harry Potter” into the list. Because there are many of these types of movies, the program only pulls the first one. The recommended movies that are given back are some of the sequels and different movies which satisfies the serendipity condition of Recommender systems. Serendipity refers to the condition of recommending only new things and does not repeat itself. As we increase the list of input movies, we get better, more accurate results and more movies to rank.