Data Mining Project

Internet Movie DataBase Study (IMDB)

Makhlouf Slimane - 23 mars 2017

Introduction

For this project, we had to gather datas and process some treatments in order to extract interesting informations from the dataset.

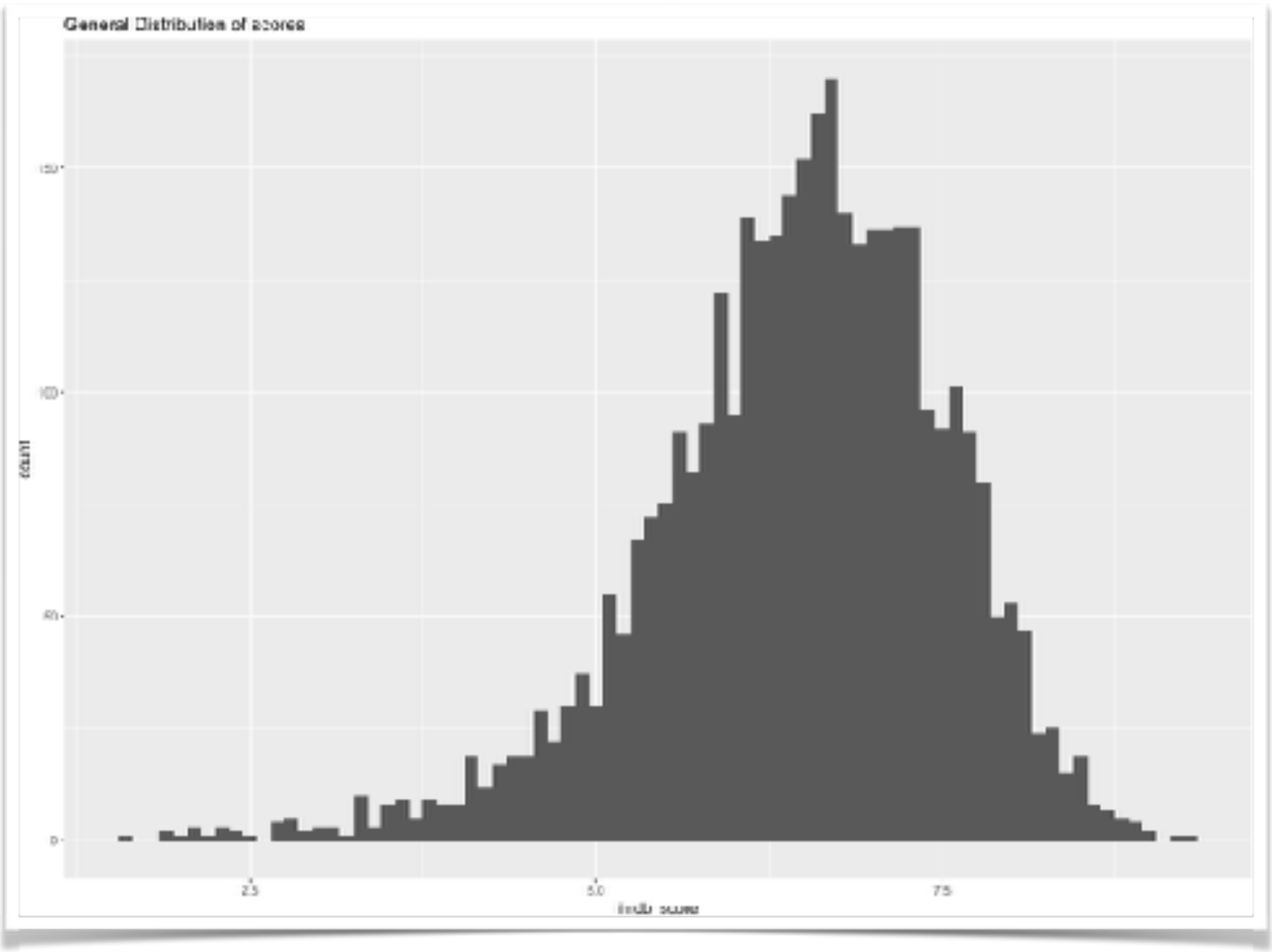
I first wanted to work on datas from League of Legend, the most played online video game in the world. I have found a way to retrieve a lot of datas but the problem was about organizing these datas. Indeed, I had to send HTTP request to their API, the response was a JSON file, but the format of these JSON responses were very hard to melt with each others. It was so inconvenient, that I had to give up on this dataset.

After this failure, I found the dataset about Internet Movie Database (also known as IMDB) which was way more easy to manipulate.

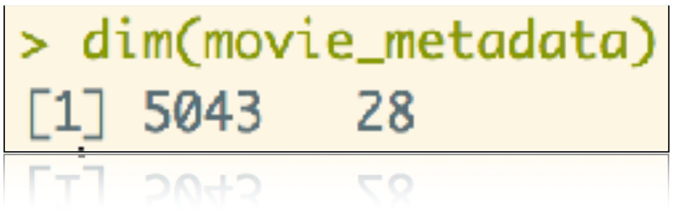
Problem Understanding

I am often using IMDB to find some movies to watch, and generally, I rely on the IMDB score to know if a movie is going to be good or not. This IMDB score is acknowledged to be quite fair.

I wanted to know if we could find some factors that have influence on this score.

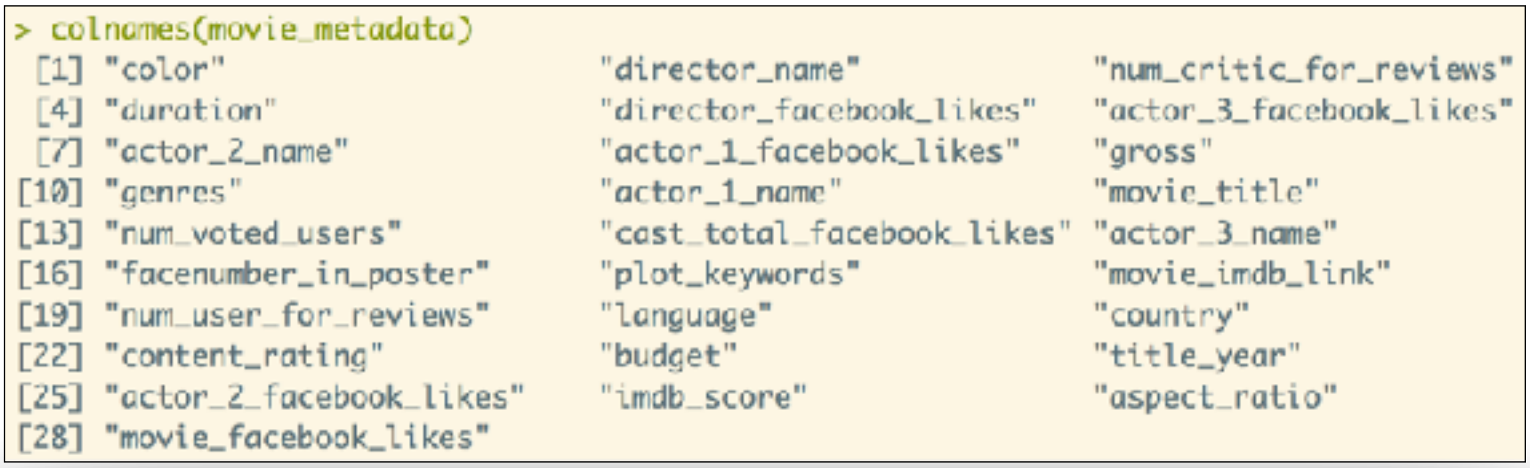


Data Understanding



```
> dim(movie_metadata)
[1] 5043   28
```

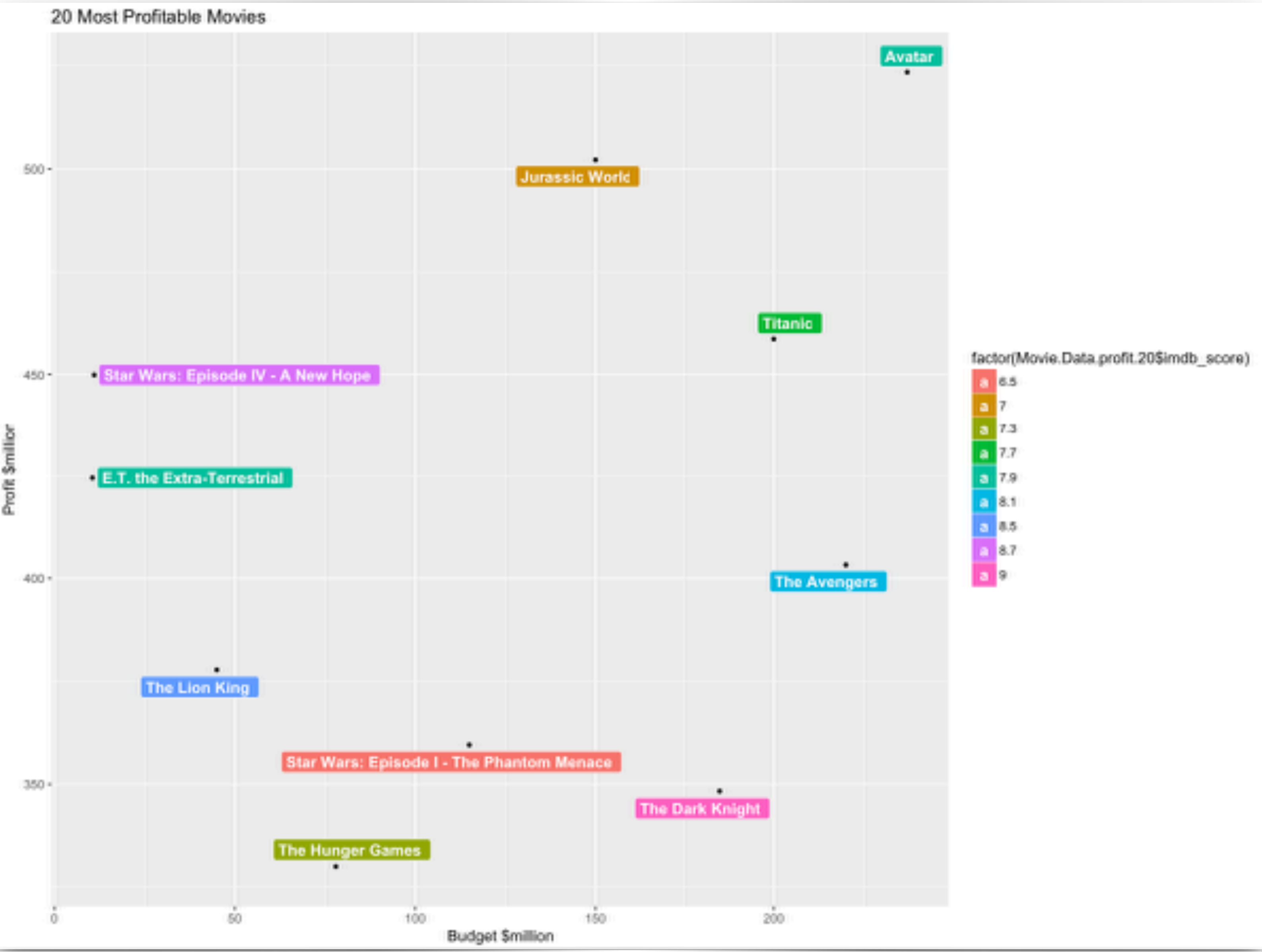
This dataset contains 28 columns for 5043 movies. The features are for example the title of the movie, its score, the type of movie it is (Documentary , Action, Adventure…).



```
> colnames(movie_metadata)
 [1] "color"                     "director_name"             "num_critic_for_reviews"
 [4] "duration"                  "director_facebook_likes"   "actor_3_facebook_likes"
 [7] "actor_2_name"              "actor_1_facebook_likes"    "gross"
[10] "genres"                    "actor_1_name"              "movie_title"
[13] "num_voted_users"           "cast_total_facebook_likes" "actor_3_name"
[16] "facenumber_in_poster"      "plot_keywords"             "movie_imdb_link"
[19] "num_user_for_reviews"      "language"                  "country"
[22] "content_rating"            "budget"                    "title_year"
[25] "actor_2_facebook_likes"    "imdb_score"                "aspect_ratio"
[28] "movie_facebook_likes"
```

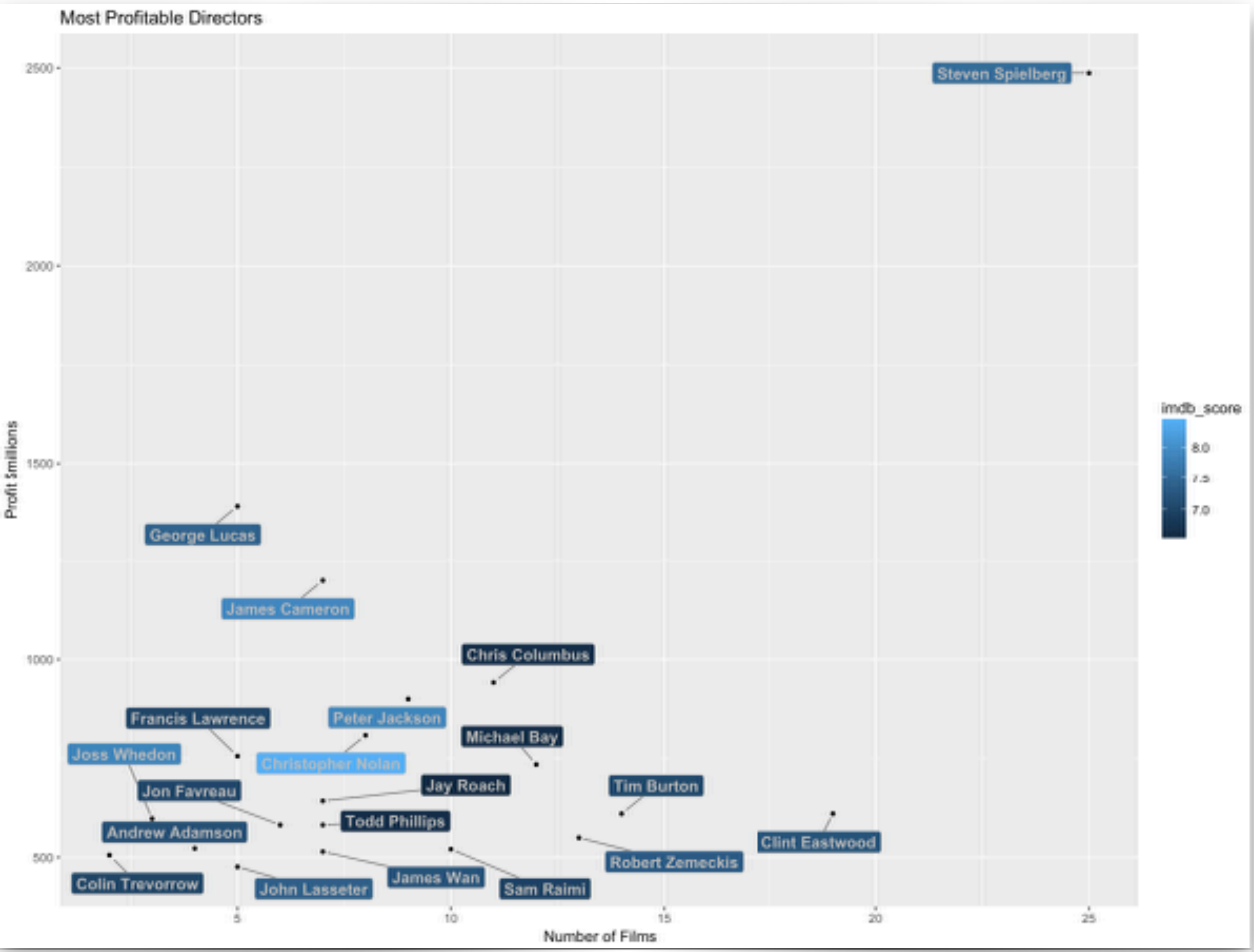
Data Preparation

To prepare data, I first removed duplicated films, divided gross and budget by 1 000 000 to have more understandable plots. I added column for profit and generated some subsets like the 20s most profitable movies or the 20s most profitable directors.



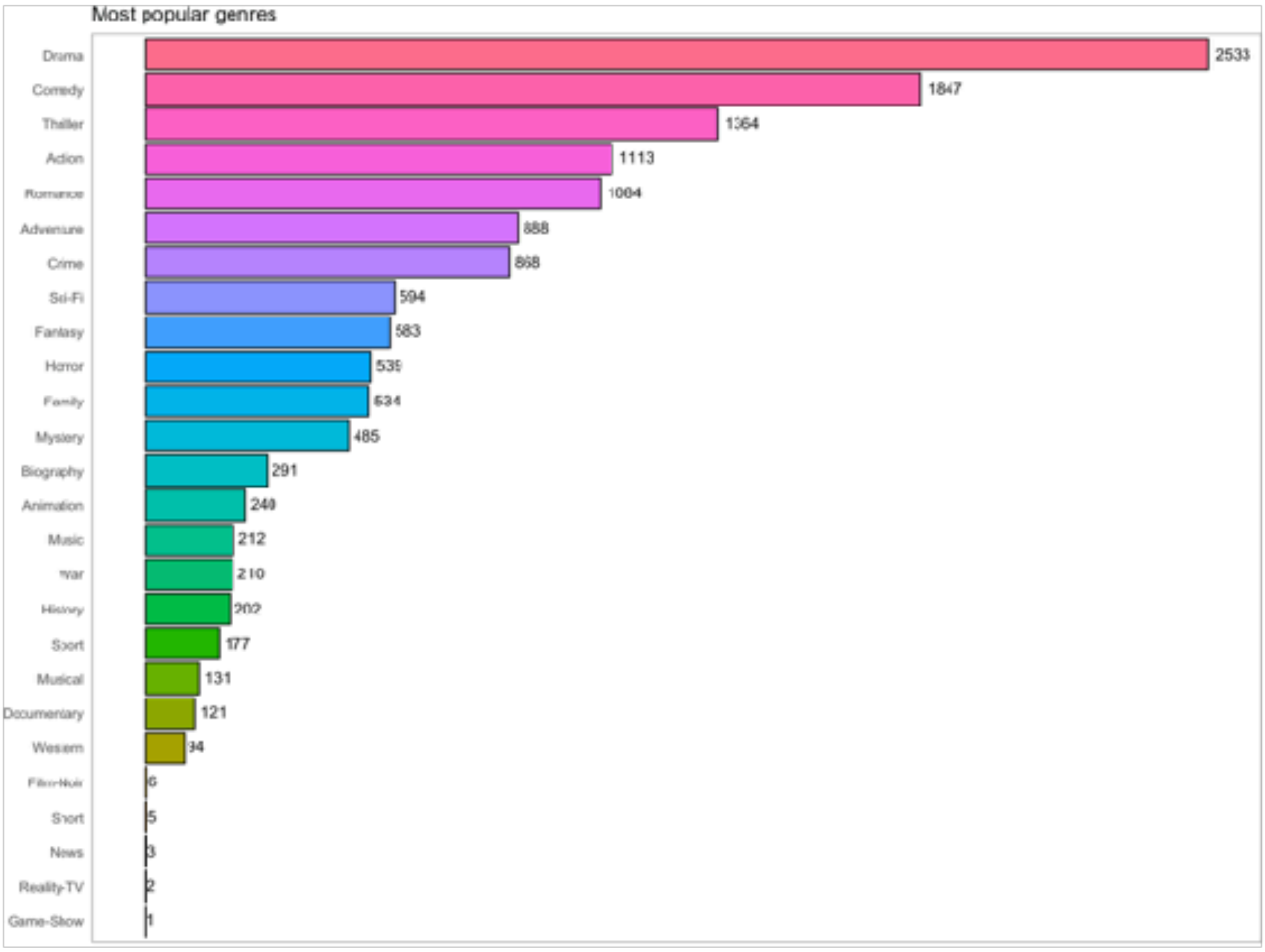
We can see here that "The Dark Knight" has the best score but didn't generated that much profit compared to movies like "Avatar" that thus have a lower score.

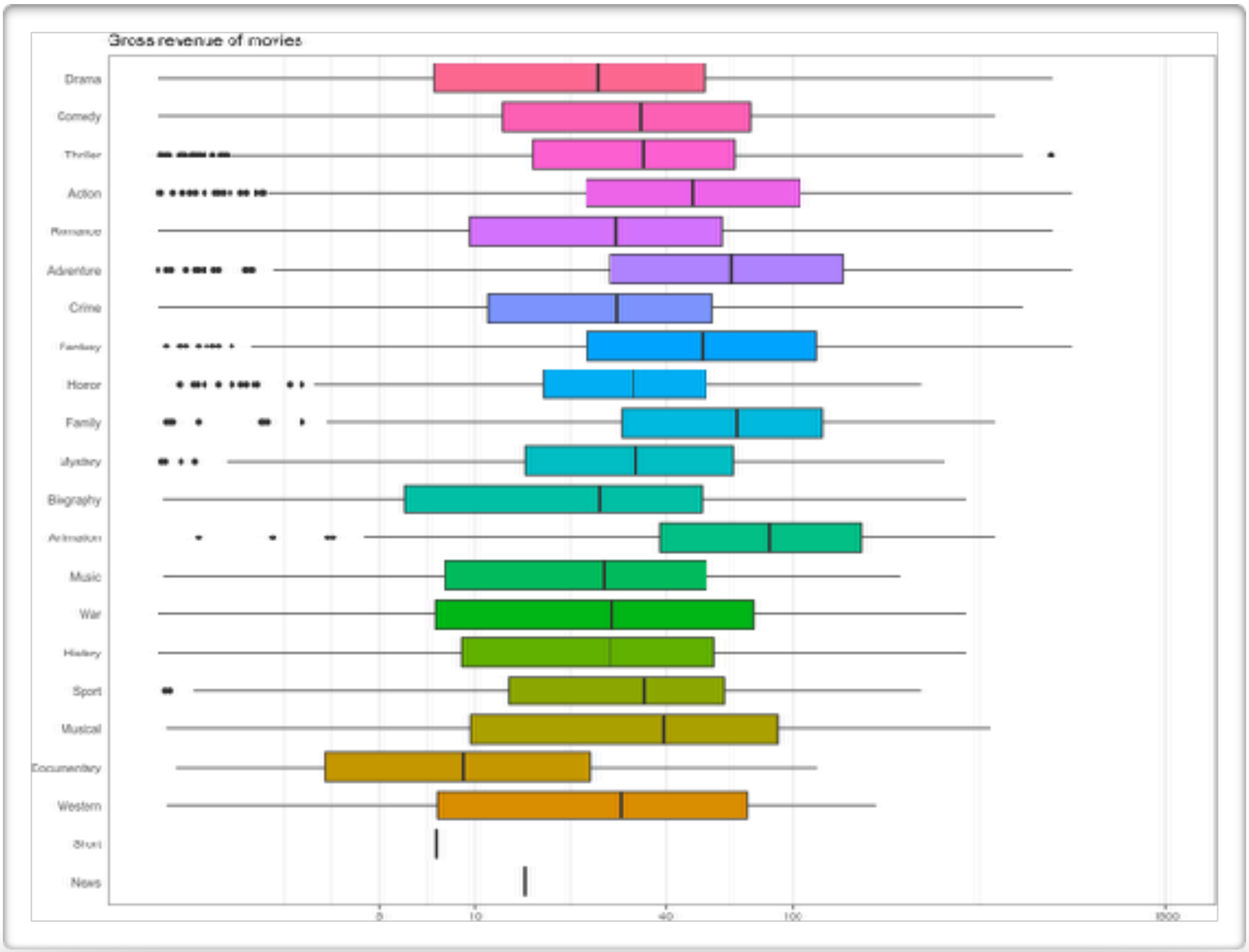
Now regarding the most profitable directors, things are clearer, Steven Spielberg is way higher than the rest. Although he's done more films, he has generated much more profit than everyone else. It appears that his films are not very well rated though.



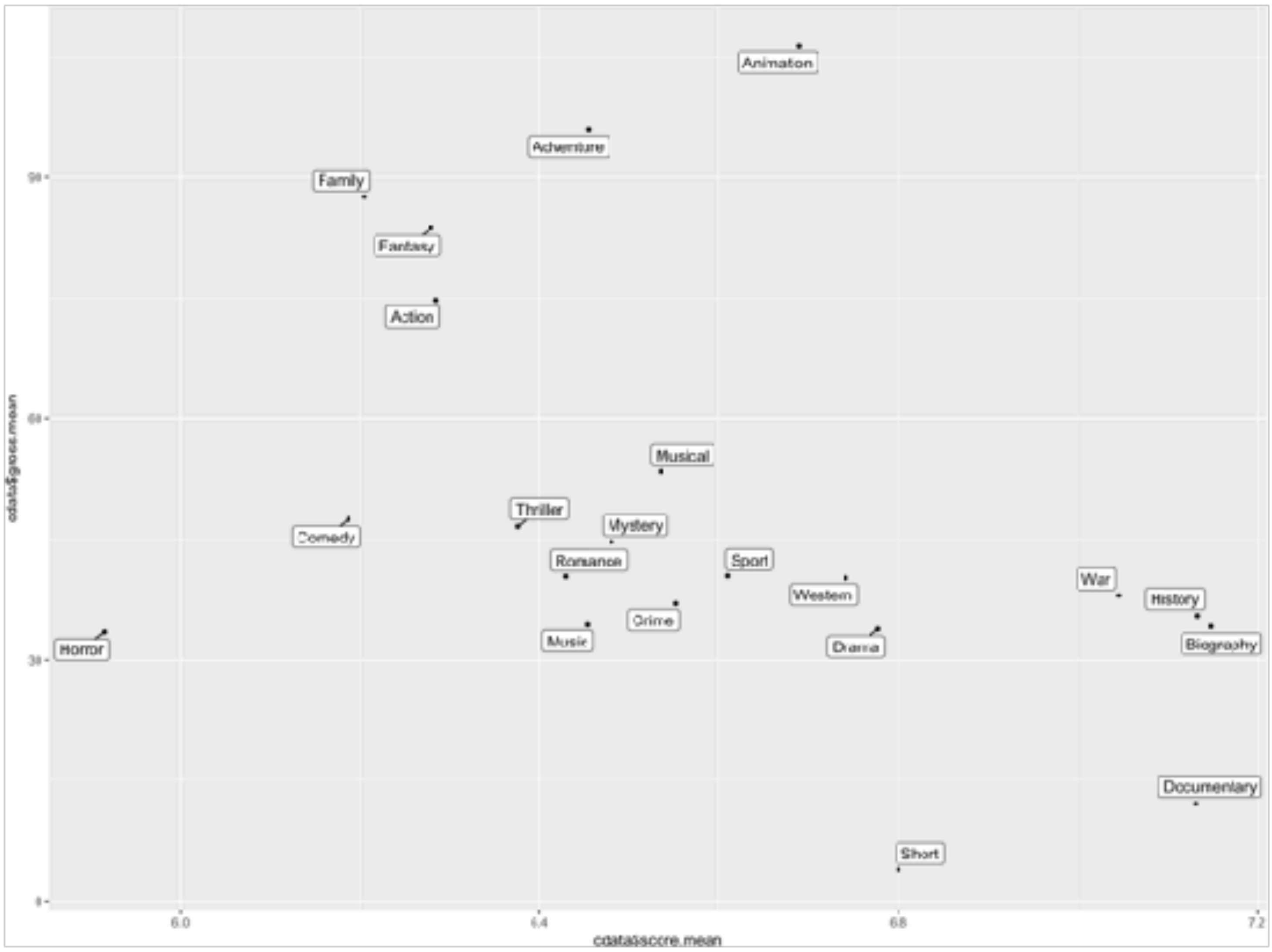
Movie genre study

In this part, I will show the results I got studying the type of film. The following plot show the most popular movie genre. Drama seems to be very popular as it is clearly the most present genre in this dataset.



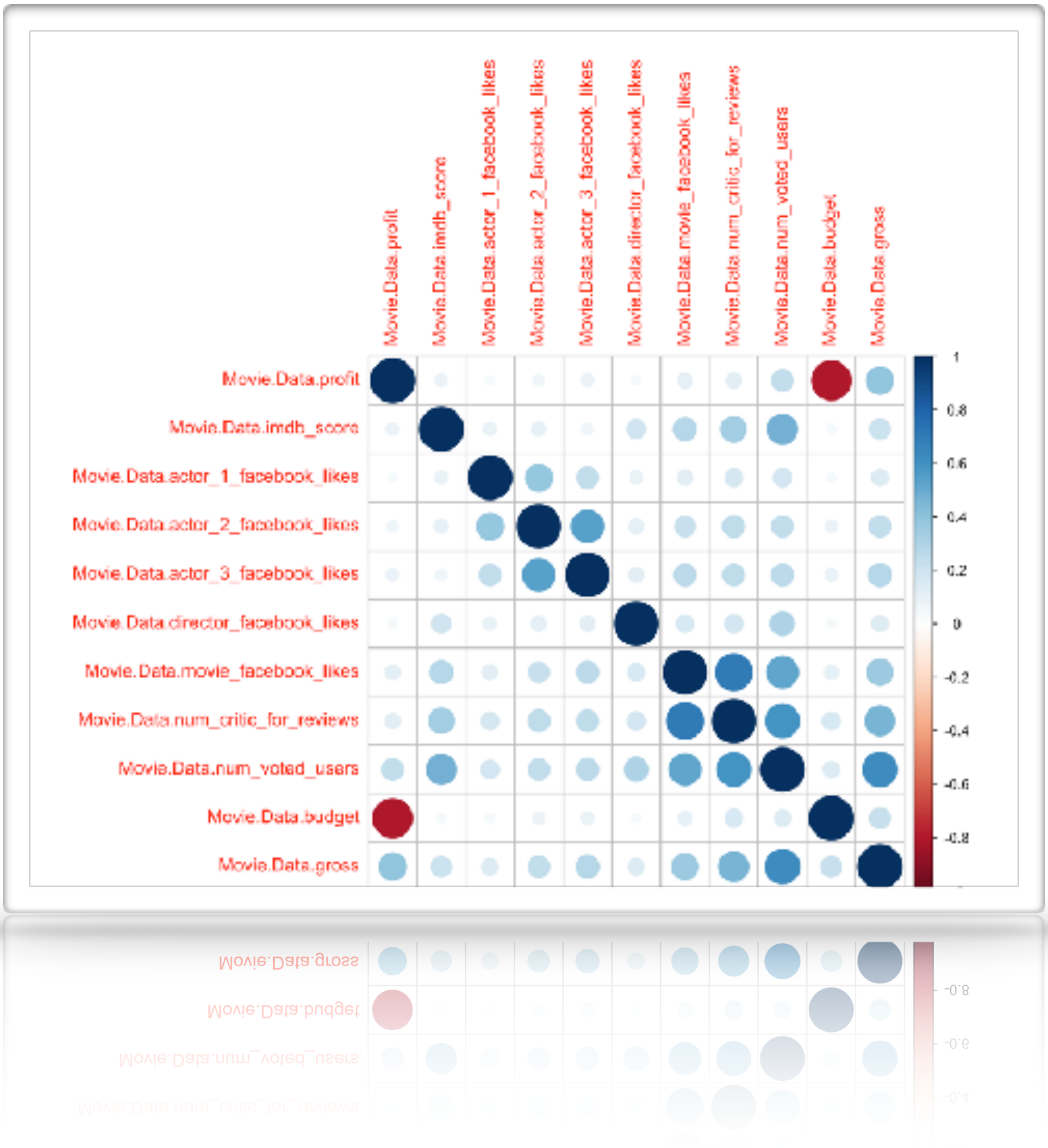


Here we can see that the popular genres are not necessarily those which are the most profitable.



This plot showing mean gross with respect to mean score is interesting in the way that it provides some nice informations that may be unexpected. Indeed, we can see that the gross is not correlated with the score. The perfect illustration of this phenomena is the adventure genre, where the mean gross is above 90 millions whereas the mean score barely exceeds 6.4 which is not very high comparing to others genres. Inversely, Documentaries have the second best mean score whereas they hardly generate 15 millions.

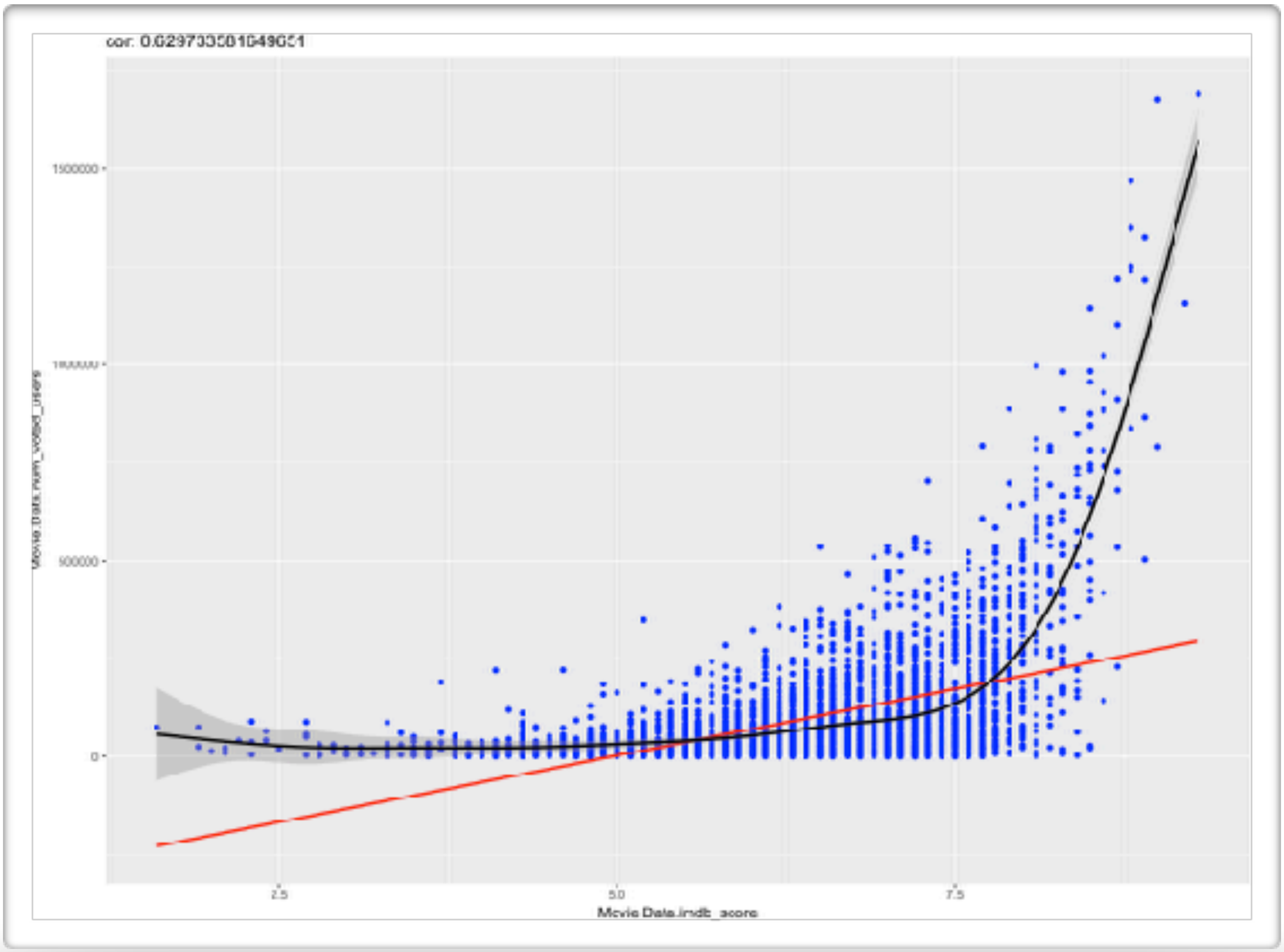
Correlation Study



This correlation matrix shows us some very unexpected correlations like gross~num_voted_users or num_voted_users~score.

We propose to go further on this last surprising correlation. We plot num_voted_users w.r.t. the score, and print in red the linear model and in black the curve generated by geom_smooth that uses predictdf. The correlation is 0.6 which is quite high for two features I didn't expected to be correlated at all. The interpretation is :

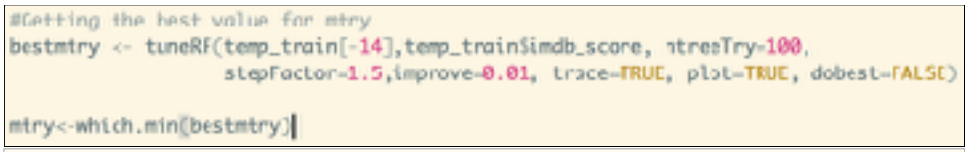
"The more votes a movie gets, the better its score".



Score Prediction

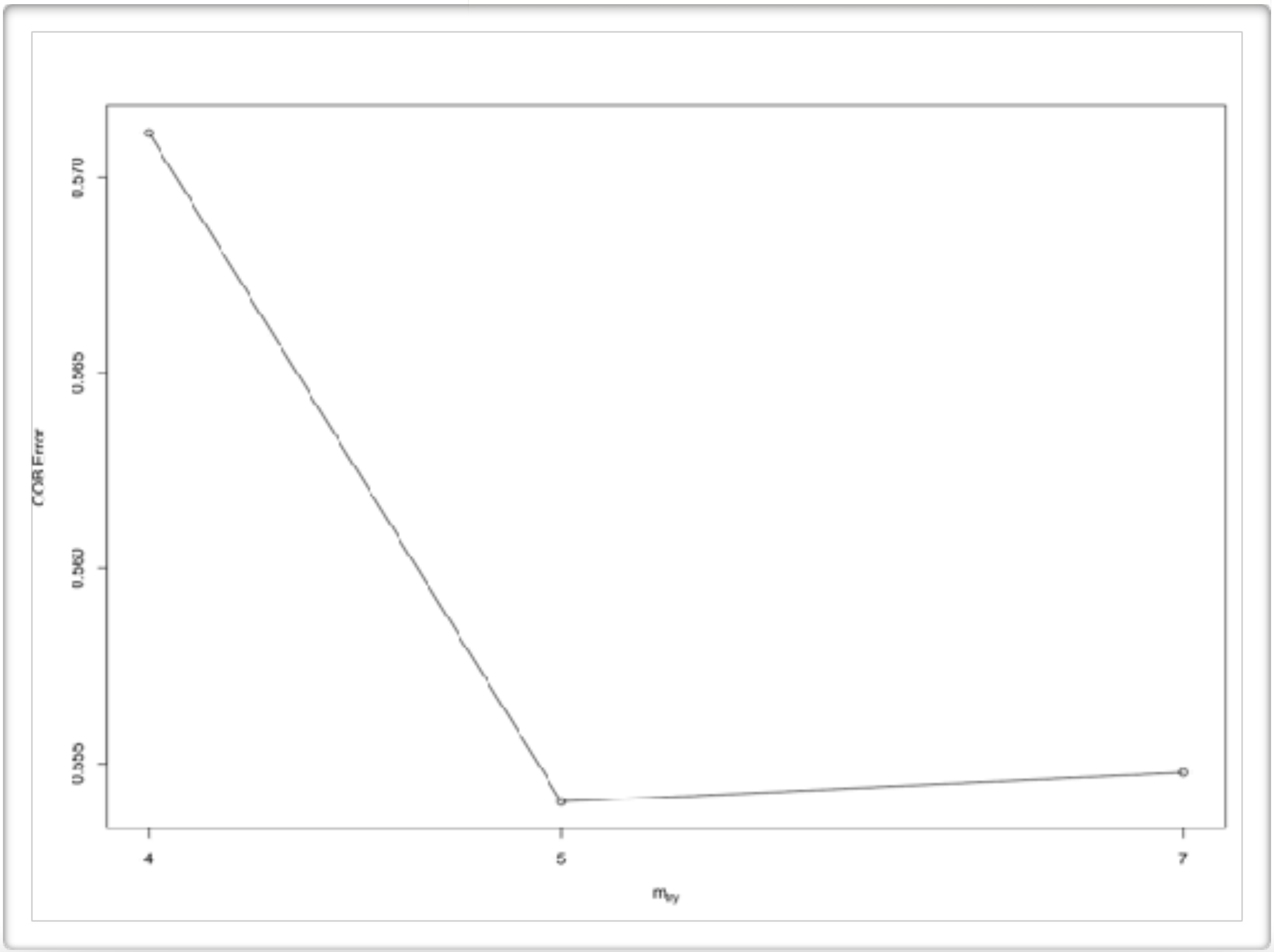
In this last part, I wanted to implement a predictive model for the score.

I chose to use the randomForest R library because it was the most adapted one in my opinion.After splitting the data set into training and test set, I have found on the internet a function that could compute the best mtry parameter : tuneRF.

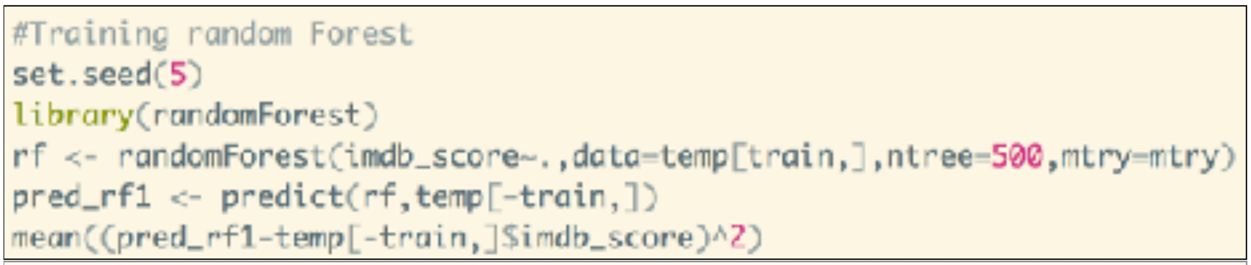


```
#Getting the best value for mtry
bestmtry <- tuneRF(temp_train[-14],temp_train$imdb_score, ntreeTry=100,
                   stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE, dobest=FALSE)

mtry<-which.min(bestmtry)
```

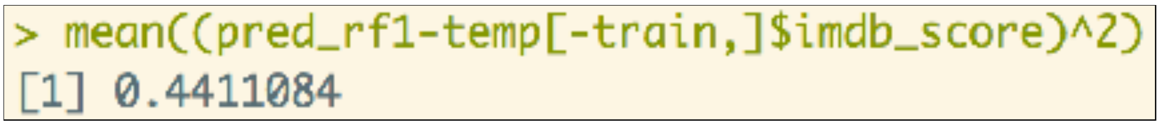


So according to this function, the best mtry is 5. Now we are ready to train our random forest.



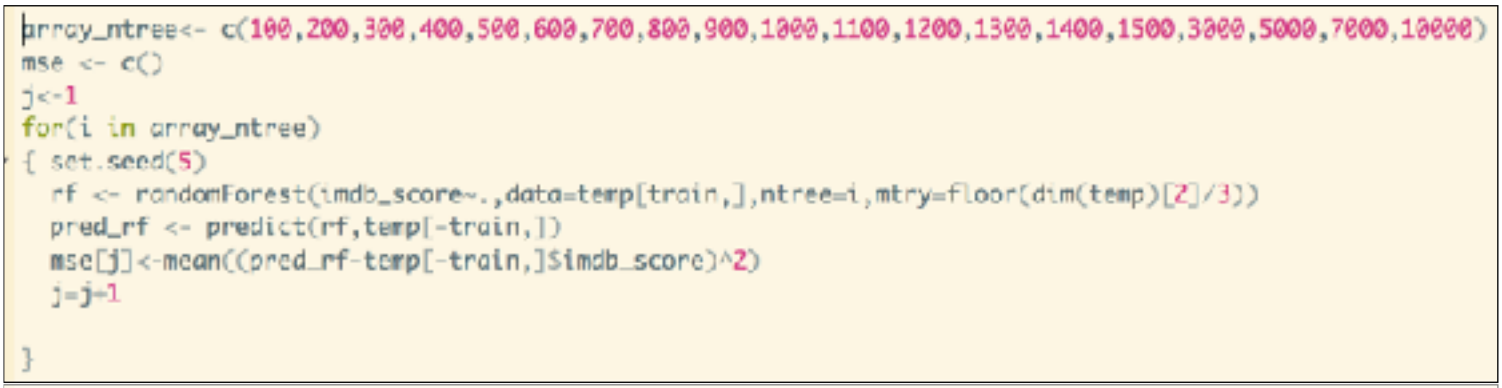
```
#Training random Forest
set.seed(5)
library(randomForest)
rf <- randomForest(imdb_score~.,data=temp[train,],ntree=500,mtry=mtry)
pred_rf1 <- predict(rf,temp[-train,])
mean((pred_rf1-temp[-train,]$imdb_score)^2)
```

Firstly, I trained a random forest using mtry=5 and ntree arbitrarily set to 500. I obtained a mean squared error of 0.44, which is quite ok but far from perfect.



```
> mean((pred_rf1-temp[-train,]$imdb_score)^2)
[1] 0.4411084
```

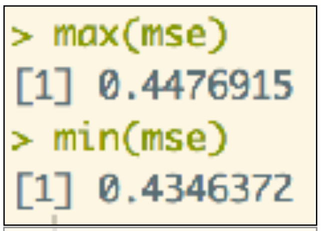
To improve my prediction model, I tried tuning ntree parameter. For each values, I tested the model on the test set and computed the mean squared error.



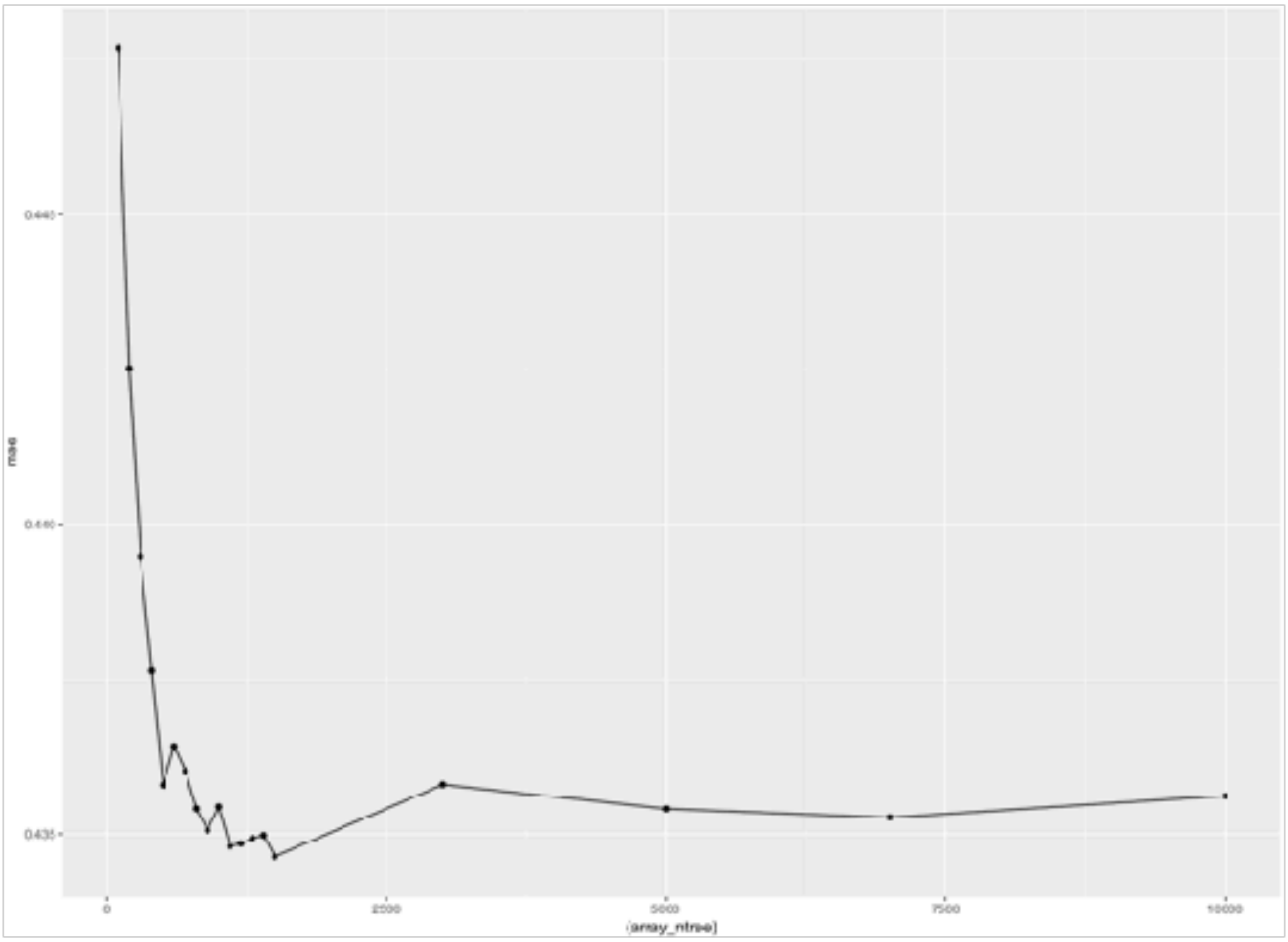
```
array_ntree<- c(100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,1500,3000,5000,7000,10000)
mse <- c()
j<-1
for(i in array_ntree)
{ set.seed(5)
  rf <- randomForest(imdb_score~.,data=temp[train,],ntree=i,mtry=floor(dim(temp)[2]/3))
  pred_rf <- predict(rf,temp[-train,])
  mse[j]<-mean((pred_rf-temp[-train,]$imdb_score)^2)
  j=j+1

}
```

The results are showed in the following plot. We observe a relative drop of the MSE from 0.447 to 0.434.



```
> max(mse)
[1] 0.4476915
> min(mse)
[1] 0.4346372
```


0.445
0.440
0.435
mae
0
2500
5000
7500
10000
(array_ntree)