Capstone MovieLens

Jose Quesada

Introduccion.

Before starting I want to apologize for my English, one of the challenges of this course for me is that my english is not good.

A recommendation system is a tool that establishes a set of criteria and evaluations on user data to make predictions on recommendations of elements that may be of use or value to the user. These systems select data provided by the user directly or indirectly, and proceed to analyze and process information from the user's history to transform this data into recommendation knowledge.

"GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set."

GroupLens gratefully acknowledges the support of the National Science Foundation under research grants IIS 05-34420, IIS 05-34692, IIS 03-24851, IIS 03-07459, CNS 02-24392, IIS 01-02229, IIS 99-78717, IIS 97-34442, DGE 95-54517, IIS 96-13960, IIS 94-10470, IIS 08-08692, BCS 07-29344, IIS 09-68483, IIS 10-17697, IIS 09-64695 and IIS 08-12148.
https://grouplens.org/datasets/movielens/

Method

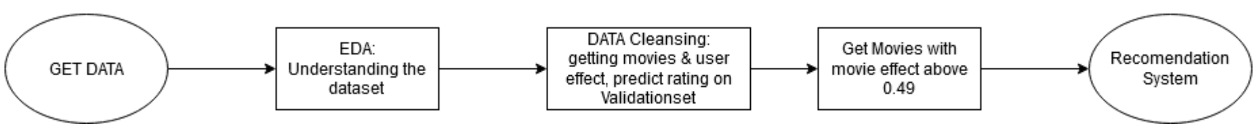


Figure 1: method

tools

1. Jupyter notebook.
2. Apriori algorithm: Arules & ArulesViz

EDA

Understaning the Data SET

```
##
## We are using the moviesId, userId, rating to detect patterns to reduce dimension and reduce the rmse
##
## New columns:
## - date: Then timestamp columns is a int, we need to transform to datetime format.
## - title: remove the relase year from title.
```

```
## - relased_yrear: extracted by using (\W{1}\d+\W{1}) as pattern
## - year: year extracted from date column
##
## Total Distinct Users: 69878
## Total of Distinct Movies: 10677
## Total Observations: 9000055
```

EDX Example	
userId	1
movieId	122
rating	5
timestamp	838985046
title	Boomerang
genres	Comedy|Romance
date	1996-08-02 11:24:06
released_year	(1992)
year	1996

Top 10 Movies by Views

title	avg_rating	n_reviews
Pulp Fiction	4.154789	31362
Forrest Gump	4.012822	31079
Silence of the Lambs, The	4.204101	30382
Jurassic Park	3.663522	29360
Shawshank Redemption, The	4.455131	28015
Braveheart	4.081852	26212
Fugitive, The	4.009155	25998
Terminator 2: Judgment Day	3.927859	25984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)	4.221311	25672
Apollo 13	3.885789	24284

In the table of the most viewed movies I was expecting better ratings.

Top 10 Movies by Avg. Rating

title	avg_rating	n_reviews
Hellhounds on My Trail	5.00	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³)	5.00	2
Shadows of Forgotten Ancestors	5.00	1
Fighting Elegy (Kenka erejii)	5.00	1
Sun Alley (Sonnenallee)	5.00	1
Blue Light, The (Das Blaue Licht)	5.00	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	4.75	4
Human Condition II, The (Ningen no joken II)	4.75	4
Human Condition III, The (Ningen no joken III)	4.75	4
Constantine's Sword	4.75	2

In the table Top ten movies by avg_rating, all these movie has fewers views than top ten movies by views

Movies Views & Rating Summary		
	n_reviews	avg_rating
	Min. : 1.0	Min. :0.500
	1st Qu.: 30.0	1st Qu.:2.844
	Median : 122.0	Median :3.268
	Mean : 842.9	Mean :3.192
	3rd Qu.: 565.0	3rd Qu.:3.609
	Max. :31362.0	Max. :5.000

we can see the overall distribution of all of the ratings. It is screwed to the right. All half stars are less frenquient than full stars. A red dased line of the overall average rating is also plotted here as a reference.

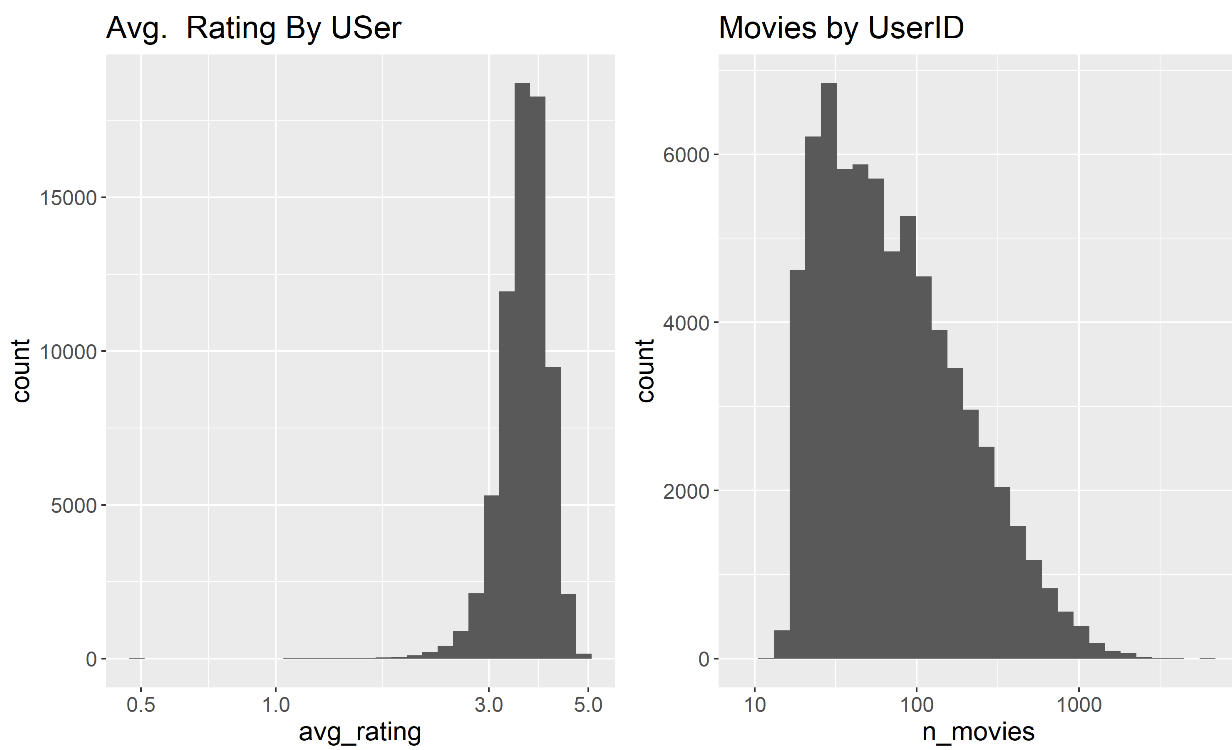


Figure 2: USERDIST

Aggregating Movies by views

Now im using this n_reviews summary to classifie in 4 big groug and get a better views of the top movies: 1. Belowe first Quantile. 2. Between 1th Quantile and Median. 3. Between Median and 3rd Quantile. 4. Blockbuster

```
set_class <-  function(col){
    if (col<=30){
        x = "1. Below 1th quantile"
    }
```

```
    else if( between(col,30,122)){
        x = "2. between 1th Q and median"
    }
    else if (between(col,122,565)){
        x = "3. Median to 3rd Q"
    }
    else{
        x = "4. BlockBuster"}
    x
}

movies_avgs<- movies_avgs %>% mutate(class=sapply(n_reviews,FUN=set_class))
```

Rating Vs Views by Movies

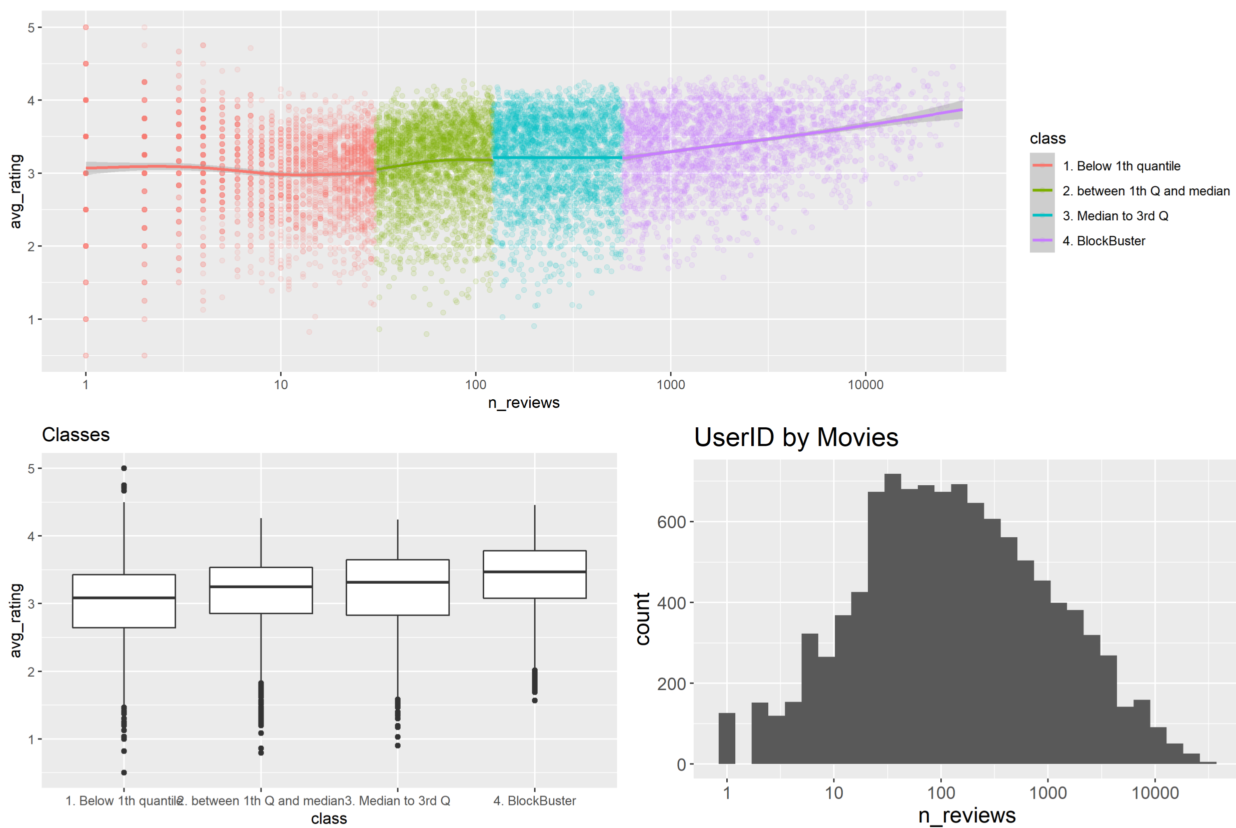


Figure 3: RatingVsViews

With this analysis I can say that determining the popularity of a movie is not as simple as the amount of views or ratings that users give, we have movies with very few views with high ratings and vice versa. There is a correlation between high views and highest ratings.

Users Views & Rating Summary		
	n_movies	avg_rating
	Min. : 10.0	Min. :0.500
	1st Qu.: 32.0	1st Qu.:3.357
	Median : 62.0	Median :3.635
	Mean : 128.8	Mean :3.614
	3rd Qu.: 141.0	3rd Qu.:3.903
	Max. :6616.0	Max. :5.000

Genres Analysis

Correlation between Genres

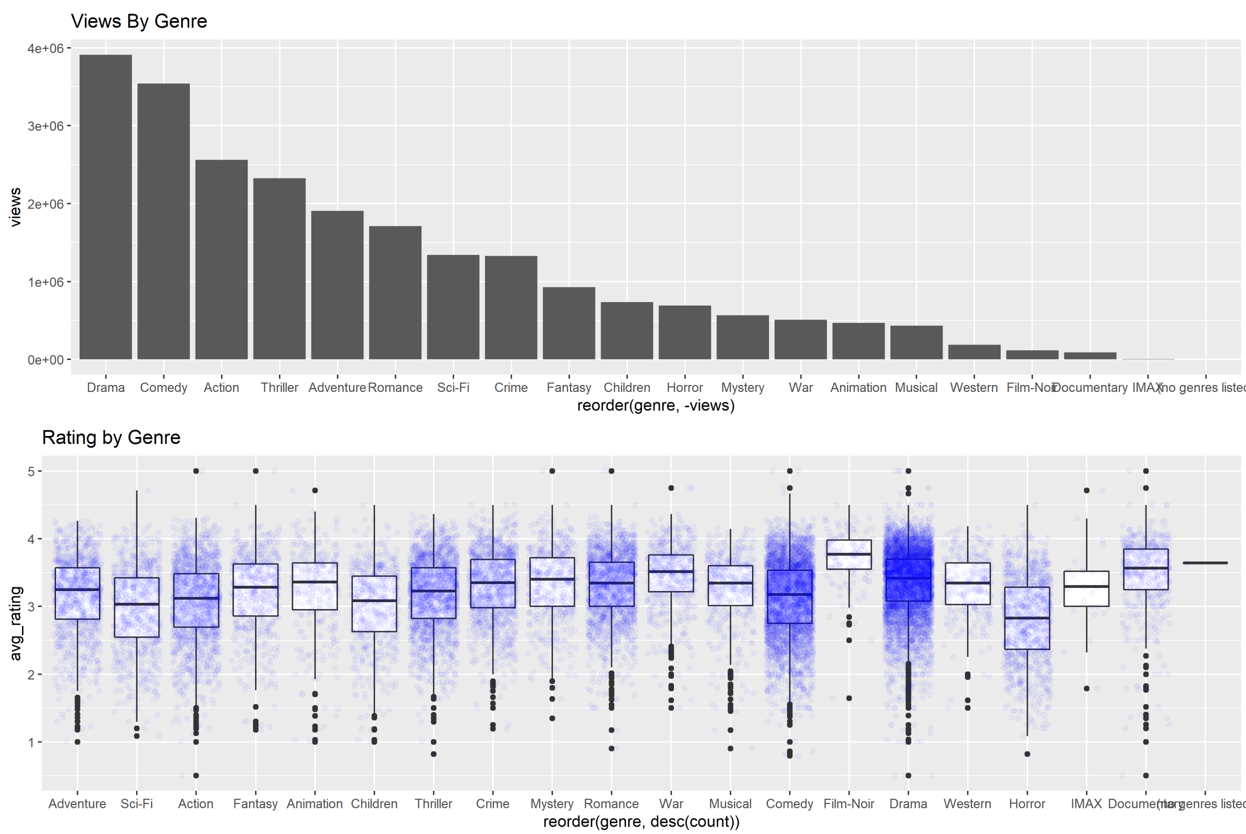


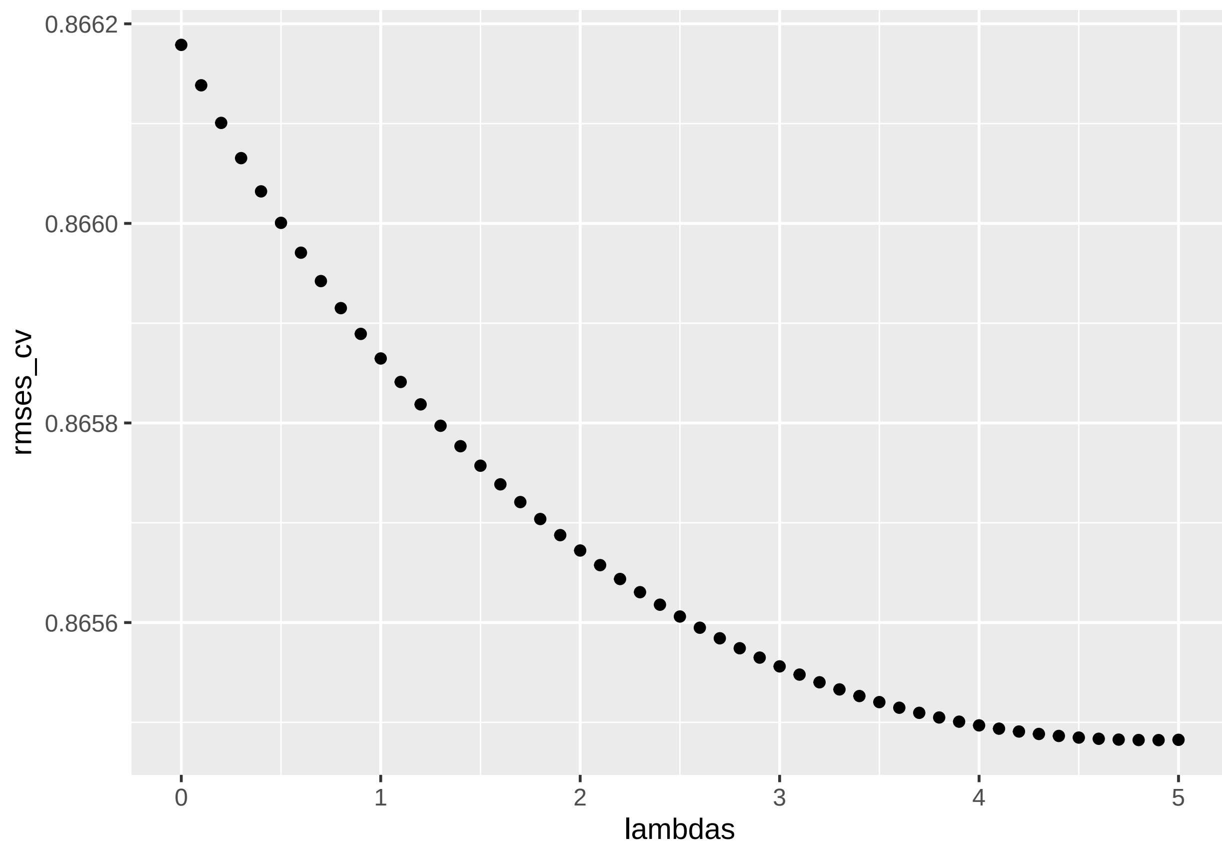
Figure 4: Genres

Get Movies and User Effect

Im using Kfolds and movies and user effects to predict ratings.

```
# define a empty matrix, k * length(lambda)
rmses <- matrix(nrow=k,ncol=length(lambdas))


# perform 5-fold cross validation to determine the optimal lambda
```

```r
for(n in 1:k) {
  train_set <- edx[cv[[n]],]
  test_set <- edx[-cv[[n]],]

  # Make sure userId and movieId in test set are also in the train set
  test_final <- test_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId")

  # Add rows removed from validation set back into edx set
  removed <- anti_join(test_set, test_final)
  train_final <- rbind(train_set, removed)

  mu <- mean(train_final$rating)

  rmses[n,] <- sapply(lambdas, function(l){
    #print(l,n)
    b_i <- train_final %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu)/(n()+l))
    b_u <- train_final %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu)/(n()+l))
    predicted_ratings <-
      test_final %>%
      left_join(b_i, by = "movieId") %>%
      left_join(b_u, by = "userId") %>%
      mutate(pred = mu + b_i + b_u) %>%
      pull(pred)
    rmse <- RMSE(predicted_ratings, test_final$rating)
    #printing results
    #print(rmse)
    return(rmse)
  })
}

rmses_cv <- colMeans(rmses)
lambda <- lambdas[which.min(rmses_cv)]
qplot(lambdas,rmses_cv)
```



Conclusions & Insight

Predicting ratings on Validation Set (RMSE RESULT)

```
#Model Validation and Result
mu <- mean(edx$rating)
reg_movies <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))
reg_users  <- edx %>%
    left_join(reg_movies, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
predicted_ratings <-
    validation %>%
    left_join(reg_movies, by = "movieId") %>% #Movie Effect
    left_join(reg_users, by = "userId") %>% #User Effect
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
model<- RMSE(predicted_ratings, validation$rating)   # 0.8648185
cat(sprintf("Lambda after %s-Kfold: %s\n RMSE: %s",k, lambda,model))
```

```
## Lambda after 5-Kfold: 4.9
##  RMSE: 0.864818497538427
```

There are others factors that we could evaluate:

1. studio that makes the film.
2. budget for the film. 3. it's a sequel?

On hand data the best variable for this model is the movie effect, by the movie effect we got a more realistic top ten movies, than working with the averages.

Cleaning EDX data Set

Top Best Movies After Reg

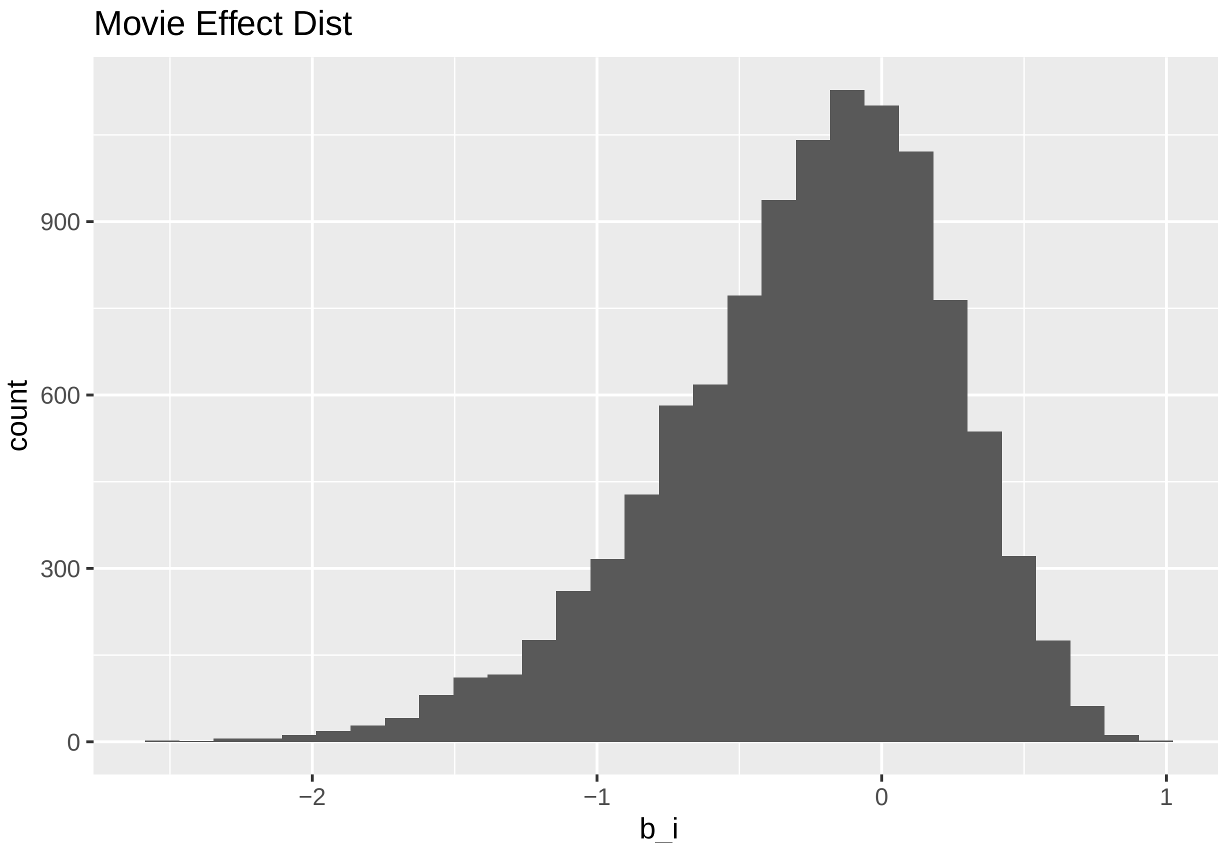
title	avg_rating	n_reviews	b_i
Shawshank Redemption, The	4.455131	28015	0.9425011
Godfather, The	4.415366	17747	0.9026516
Usual Suspects, The	4.365854	21648	0.8531953
Schindler's List	4.363493	23193	0.8508483
Casablanca	4.320424	11232	0.8076063
Rear Window	4.318651	7935	0.8056888
Sunset Blvd. (a.k.a. Sunset Boulevard)	4.315880	2922	0.8020693
Third Man, The	4.311426	2967	0.7976432
Double Indemnity	4.310817	2154	0.7965399
Paths of Glory	4.308721	1571	0.7937795

Top Worst Movies After Reg

title	avg_rating	n_reviews	b_i
From Justin to Kelly	0.9020101	199	-2.547722
SuperBabies: Baby Geniuses 2	0.7946429	56	-2.499147
PokÃ©mon Heroes	1.0291971	137	-2.397518
Glitter	1.1755162	339	-2.303651
Disaster Movie	0.8593750	32	-2.300783
Gigli	1.1932907	313	-2.283428
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie)	1.1782178	202	-2.278965
Barney's Great Adventure	1.1875000	208	-2.271455
Carnosaur 3: Primal Species	1.0882353	68	-2.261284
Yu-Gi-Oh!	1.2312500	80	-2.149555

```
reg_movies %>% ggplot(aes(x=b_i)) + geom_histogram() + labs(title="Movie Effect Dist")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

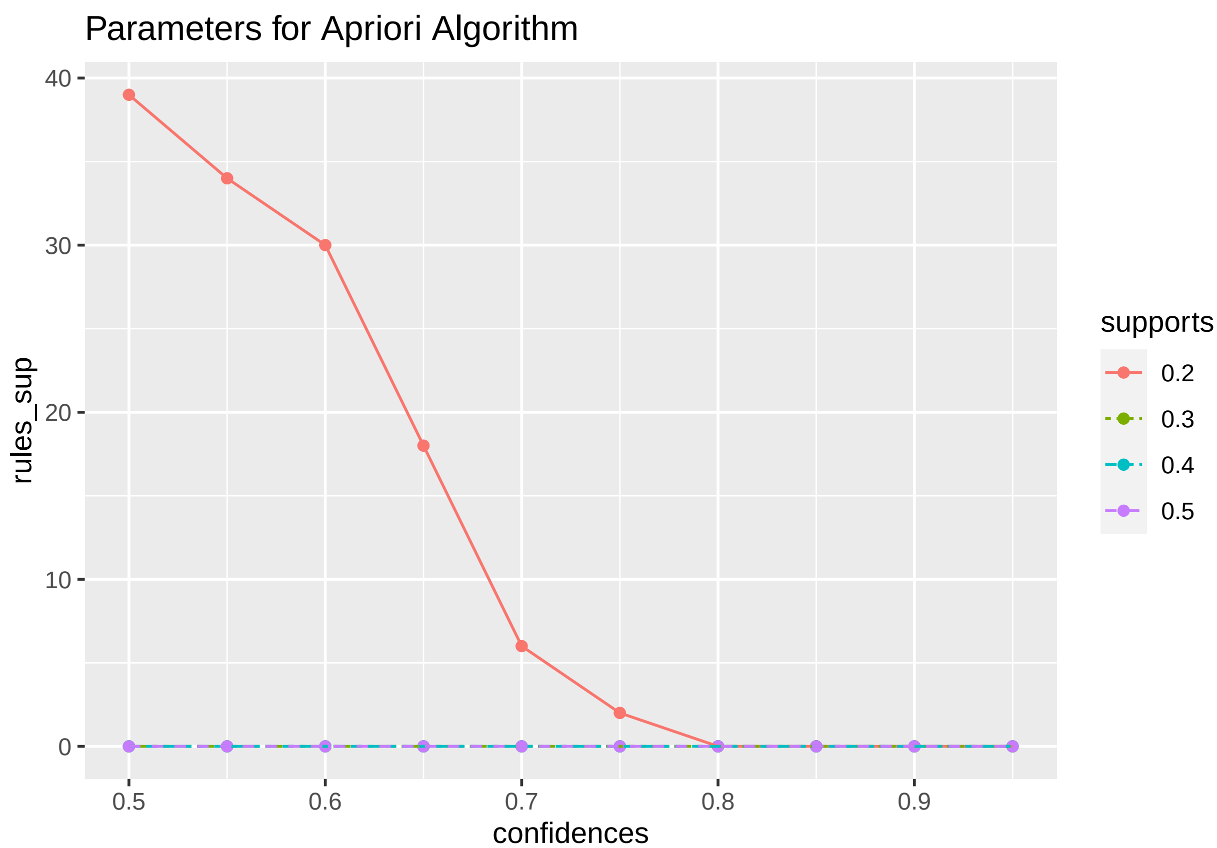


Movie Asociation after data regularization

We filter the dataset to obtain an association model on our new top of movies by b_i >= 0.5 and make a transacional table by userId and movie title.

In this step we going to evaluate different set of parameter for our recommendation model:
1. Confidence from 50% to 95%=, how many times set of movies appears together.
2. Support: how popular an movie is.(20% to 50%).
3. number of rules: amount of m movies in set.

```
parameters_aprio <- data.table(supports,confidences,rules_sup) %>% mutate(supports=as.factor(supports))
parameters_aprio%>%
  ggplot(aes(x=confidences,y=rules_sup))+
  geom_point(aes(color=supports,linetype=supports))+
  geom_line(aes(color=supports,linetype=supports)) +
  labs(title="Parameters for Apriori Algorithm")
```



Parameters from loop

```
##    supports confidences rules_sup
## 1:      0.2        0.75         2
## 2:      0.2        0.70         6
## 3:      0.2        0.65        18
## 4:      0.2        0.60        30
## 5:      0.2        0.55        34
## 6:      0.2        0.50        39
```

confidence to use: 60% support to use: 20% number of rules: 18

html table from inspectDT This table show pair of movies sorted by lift. Lift = Lift is simply the ratio of these values: target response divided by average response.

```
inspectDT(top_lift[1:10])
```

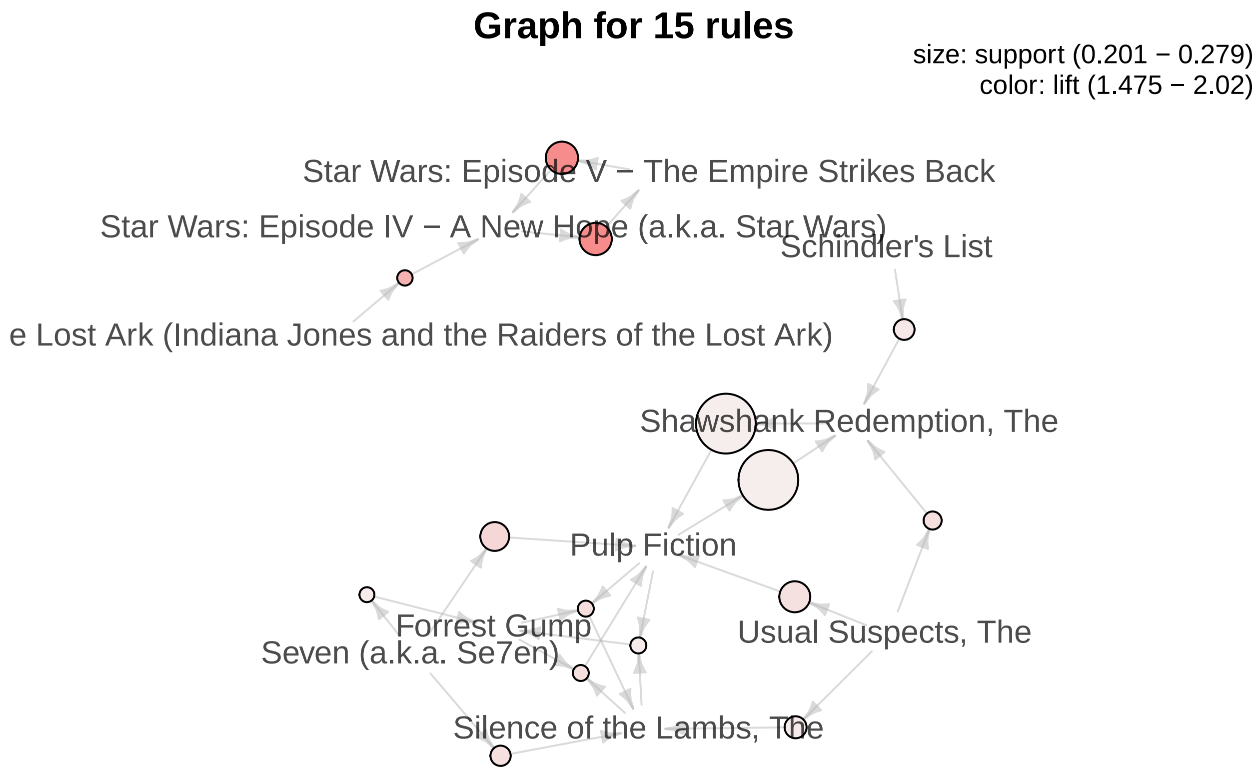
Show 10 entries Search:

	LHS	RHS	support	confidence	coverage	lift	count
[1]	{Star Wars: Episode V - The Empire Strikes Back }	{Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) }	0.231	0.761	0.304	2.020	15,769.000
[2]	{Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) }	{Star Wars: Episode V - The Empire Strikes Back }	0.231	0.614	0.377	2.020	15,769.000
[3]	{Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) }	{Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) }	0.202	0.700	0.289	1.857	13,766.000
[4]	{Seven (a.k.a. Se7en) }	{Pulp Fiction }	0.225	0.756	0.298	1.643	15,356.000
[5]	{Forrest Gump ,Pulp Fiction }	{Silence of the Lambs, The }	0.203	0.708	0.287	1.589	13,840.000
[6]	{Seven (a.k.a. Se7en) }	{Silence of the Lambs, The }	0.211	0.706	0.298	1.585	14,348.000
[7]	{Usual Suspects, The }	{Shawshank Redemption, The }	0.207	0.651	0.318	1.583	14,088.000
[8]	{Forrest Gump ,Silence of the Lambs, The }	{Pulp Fiction }	0.203	0.727	0.279	1.579	13,840.000
[9]	{Usual Suspects, The }	{Pulp Fiction }	0.229	0.721	0.318	1.568	15,614.000
[10]	{Schindler's List }	{Shawshank Redemption, The }	0.211	0.621	0.340	1.511	14,404.000

Showing 1 to 10 of 10 entries Previous 1 Next

Example of high lift value(15 rules from apriori)

```
plot(top_lift,method="Graph")
```


Graph for 15 rules
size: support (0.201 − 0.279)
color: lift (1.475 − 2.02)
Star Wars: Episode V − The Empire Strikes Back
Star Wars: Episode IV − A New Hope (a.k.a. Star Wars)
Schindler's List
e Lost Ark (Indiana Jones and the Raiders of the Lost Ark)
Shawshank Redemption, The
Pulp Fiction
Forrest Gump
Seven (a.k.a. Se7en)
Usual Suspects, The
Silence of the Lambs, The