PROJECT I: MovieLens

Fidan H

October 8, 2020

Inroduction

The analysis performed within this project is based on a recommendation system developed by the winners of the *Netflix challenge*[1]. Since Netflix data is not publicly available the data used to develop this recommendation system is a subset of the **MovieLens** data. The original "MovieLens"(20M) data set was generated by the GroupLens[2] research lab and can be found here:

- MovieLens for 20M dataset: https://grouplens.org/datasets/movielens/20m/
- MovieLens for 10M dataset: https://grouplens.org/datasets/movielens/10m/

The recommendations were developed using the **edx** data, which is a subset of the **MovieLens** data. For the evaluation of the recommendation algorithm a **validation** data set was generated. The **validation** data set was only used in the final step to test the final algorithm and contained only 10% of **edx** data. The final model with the smallest RMSE was chosen to be applied to the **edx** data to calculate the parameters of the model. For the final step, this model was evaluated by calculating the RMSE (residual mean squared error) of the **validation** set.

The following libraries were used:

```
library(dslabs)
library(tidyverse)
library(caret)
library(dplyr)
library(lubridate)
library(ggplot2)
library(gridExtra)
library(data.table)
```

```
#  Code to generate edx data set

ratings <- fread(text = gsub("::", "\t", readLines("ml-10M100K/ratings.dat")),
                 col.names = c("userId", "movieId", "rating", "timestamp"))
movies <- str_split_fixed(readLines("ml-10M100K/movies.dat"), "\\::", 3)


colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies, stringsAsFactors=TRUE) %>%
  mutate(movieId = as.numeric(levels(movieId))[movieId],
         title = as.character(title),
         genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

[1] http://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-new-contest/
[2] https://grouplens.org/

```r
#  Code to generate the validation set
set.seed(675)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(ratings, movies, test_index, temp, movielens, removed)
```

Data Exploration and Data Processing

The **edx** data set contains 9,000,055 observations and 6 variables represented in 6 columns. Each row represents one user giving one rating to one specific movie.

```r
#  Number of variables
ncol(edx)
```

```
## [1] 6
```

```r
#  Number of observarions
nrow(edx)
```

```
## [1] 9000055
```

The generated data set **edx** contains no missing data and consists of following variables:

- *movieId* is a numerical variable denotes id's for each movie
- *title* is a string variable describing the title of a movie with a release year
- *genres* is a categorical variable that represent 19 different genres
- *userId* a numerical variable to identify unique users
- *rating* a numerical variable from 0 to 5 in 0.5 increments
- *timestamp* represents time when rating was given in seconds since January 1, 1970

```r
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18114   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35732   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35867   Mean   : 4119   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53601   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Before using the data for visualization and analysis purposes, several steps were taken to transform the data. The *title* variable was split in 2 variables: *title* and *release_year*. The *timestamp* variable was transformed into a year format called *rating_year*. The *age_years* variable was created as the difference between *release_year* and *rating_year*. The original *genres* variable represents a combination of several genres, for the purpose of data exploration this variable was split into 19 distinct genres and was only used to create plots.

There are no missing values present in the **edx** data set.

```
#  Check missing values
sum(is.na(edx))
```

```
## [1] 0
```

```
head(edx,10)
```

```
##     userId movieId rating timestamp                               title
##  1:      1     122      5 838985046                           Boomerang
##  2:      1     185      5 838983525                            Net, The
##  3:      1     231      5 838983392                       Dumb & Dumber
##  4:      1     292      5 838983421                            Outbreak
##  5:      1     316      5 838983392                            Stargate
##  6:      1     329      5 838983392            Star Trek: Generations
##  7:      1     355      5 838984474                    Flintstones, The
##  8:      1     356      5 838983653                        Forrest Gump
##  9:      1     362      5 838984885                    Jungle Book, The
## 10:      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult
##     release_year                        genres rating_year age_years
##  1:         1992                Comedy|Romance        1996         4
##  2:         1995         Action|Crime|Thriller        1996         1
##  3:         1994                        Comedy        1996         2
##  4:         1995  Action|Drama|Sci-Fi|Thriller        1996         1
##  5:         1994       Action|Adventure|Sci-Fi        1996         2
##  6:         1994 Action|Adventure|Drama|Sci-Fi        1996         2
##  7:         1994       Children|Comedy|Fantasy        1996         2
##  8:         1994      Comedy|Drama|Romance|War        1996         2
##  9:         1994    Adventure|Children|Romance        1996         2
## 10:         1994                 Action|Comedy        1996         2
```

Users and Movies

There are 69878 unique users, 10677 movies and 10407 movie titles in the **edx** data set.

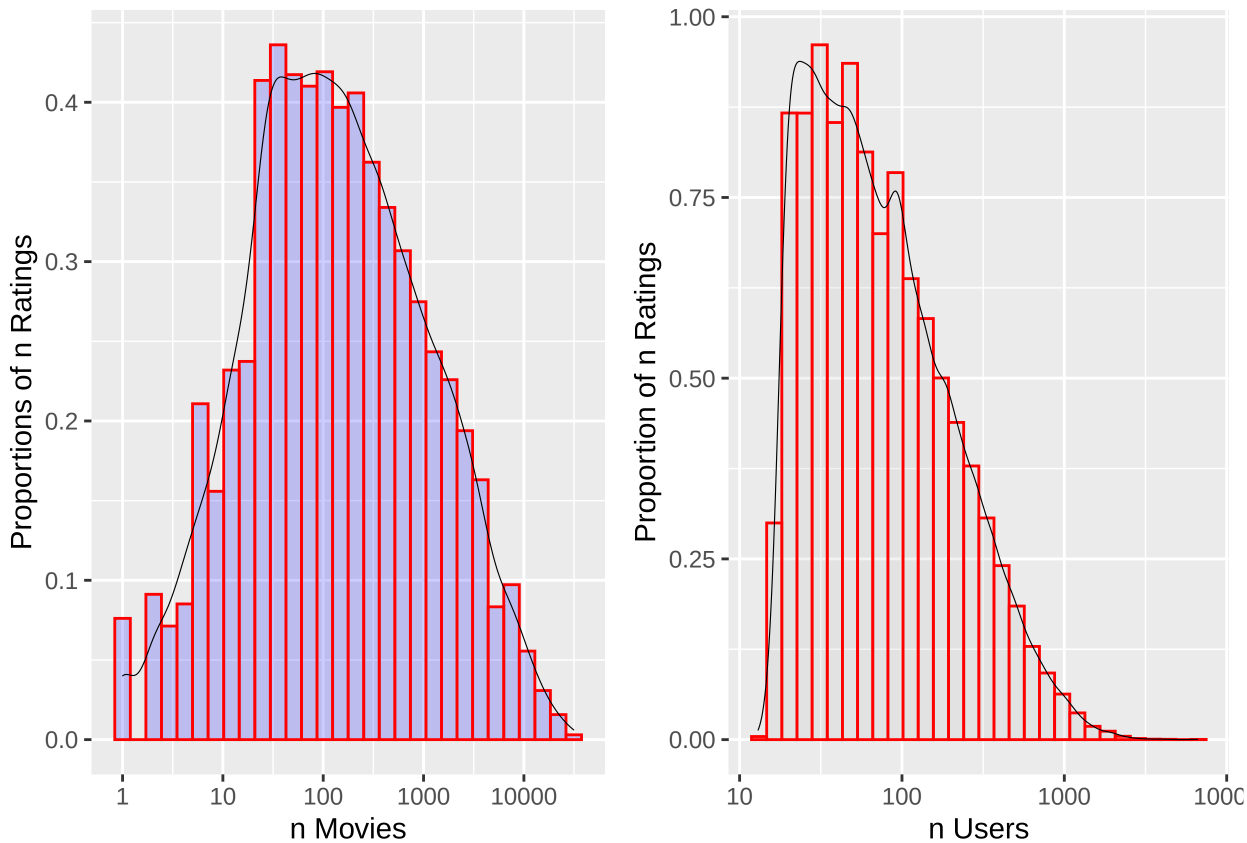
```
#  Number of unique users, movies and movie titles in edx
edx %>%summarize(n_users  = n_distinct(userId),
                 n_movies = n_distinct(movieId),
                 n_title  = n_distinct(title), .groups = 'drop')
```

```
##   n_users n_movies n_title
## 1   69878    10677   10407
```

```
#  Top 10 of movies with greatest number of ratings
edx%>%group_by(movieId, title)%>%
  summarise(count = n(), .groups = 'drop')%>%arrange(desc(count))%>%
  top_n(10, count)
```

```
## # A tibble: 10 x 3
##    movieId title                                                    count
```

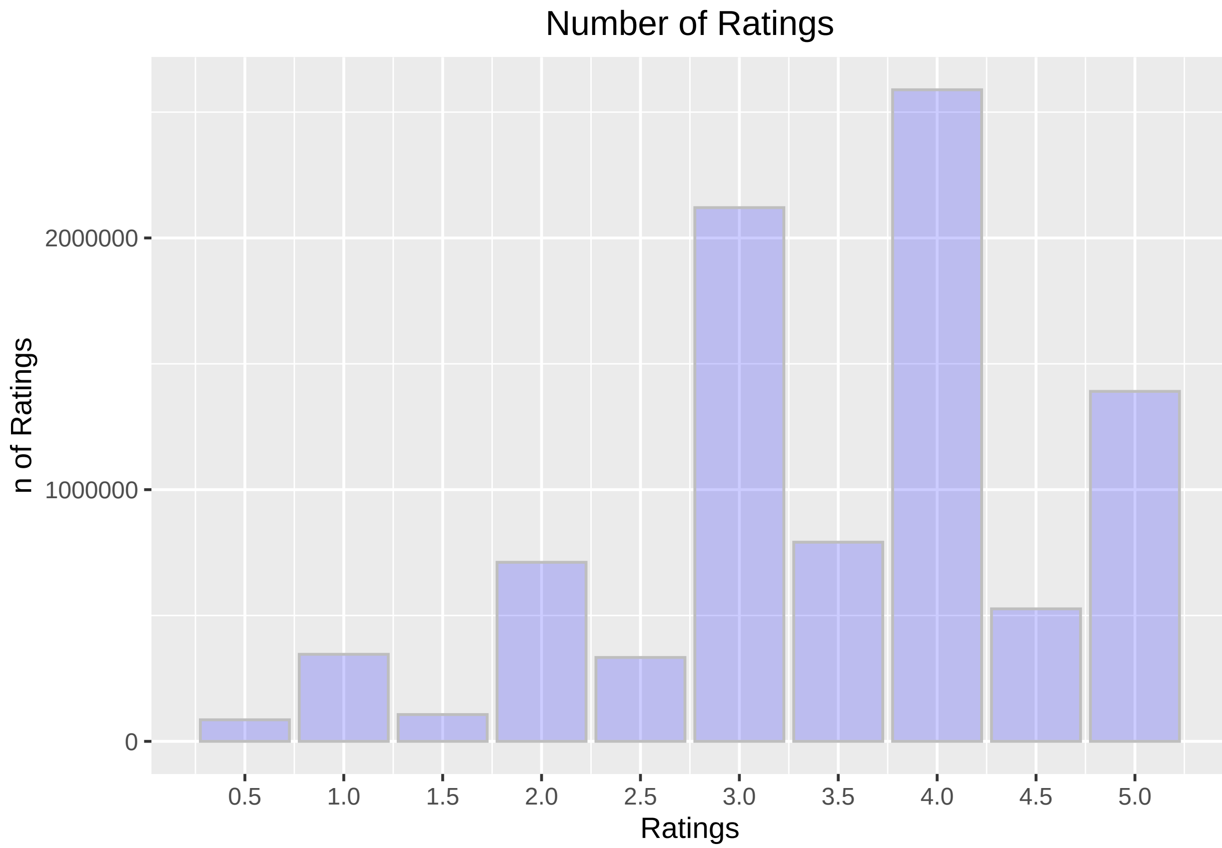
```
##       <dbl> <chr>                                                    <int>
##  1     296 "Pulp Fiction "                                          31408
##  2     356 "Forrest Gump "                                          31095
##  3     593 "Silence of the Lambs, The "                             30265
##  4     480 "Jurassic Park "                                         29428
##  5     318 "Shawshank Redemption, The "                             28003
##  6     110 "Braveheart "                                            26270
##  7     457 "Fugitive, The "                                         26023
##  8     589 "Terminator 2: Judgment Day "                            25955
##  9     260 "Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) " 25705
## 10     592 "Batman "                                                24279
```



As we can see the distribution of rating count among the number of movies and number of users is skewed. Not all users were equally active in giving ratings, and some movies received more ratings than others. For this reason, the movie and user effect were taken into account when modeling the rating prediction.

Distribution of Ratings

The most frequently given ratings were 3.0 and 4.0. This indicates that users were more likely to give a rating when they liked a movie. Full ratings outnumbered the half ratings.



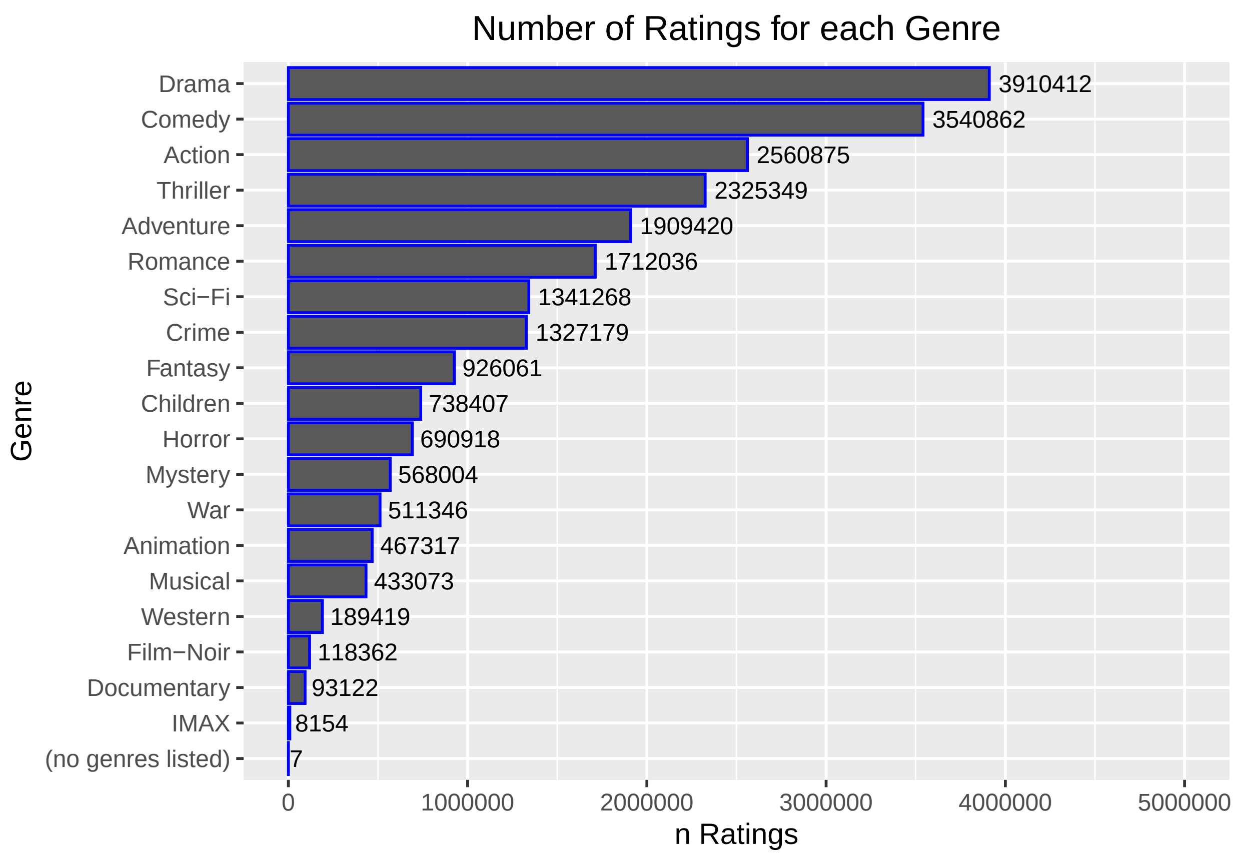
Genre

In order to demonstrate the effect of genre the *genres* variable was transformed to have a single genre per row. There were 19 unique genres, movies without genres were removed.

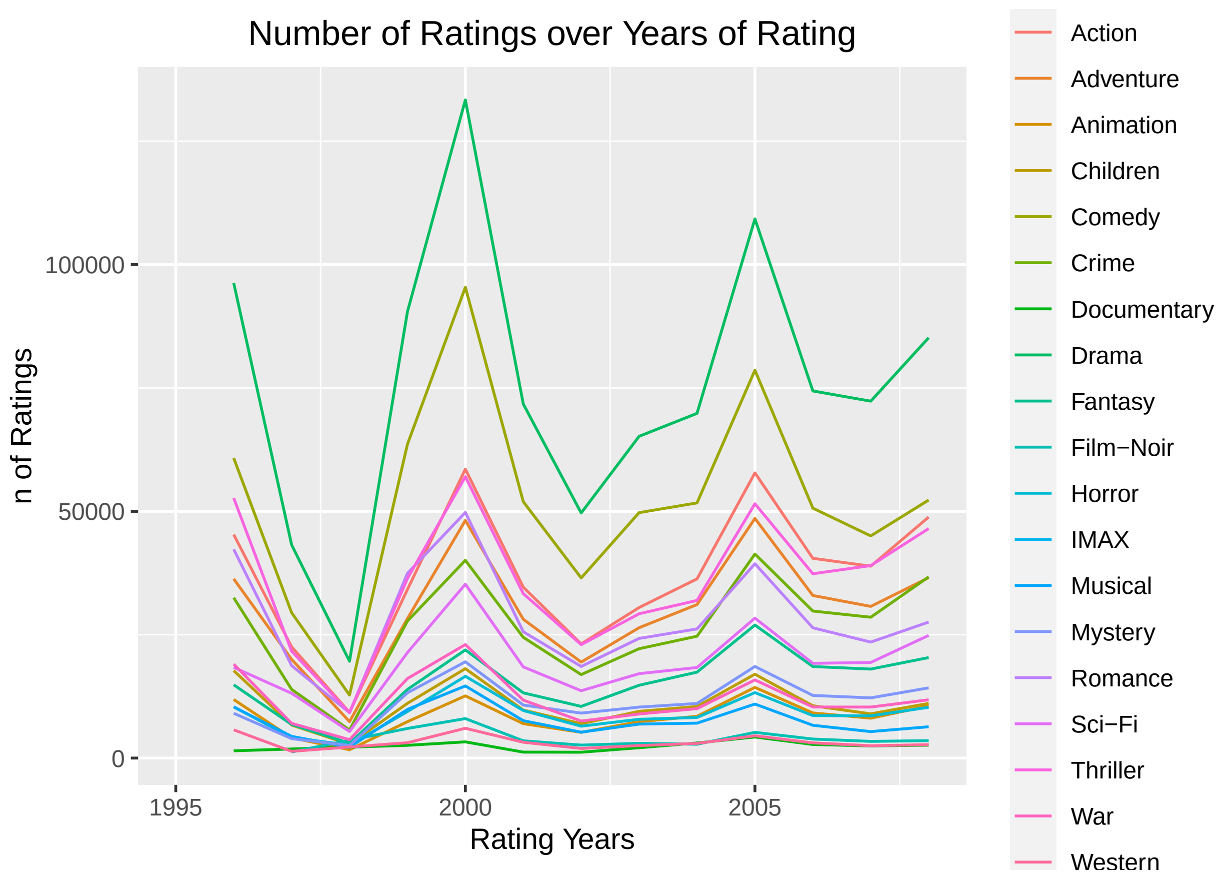
```
#  Number of different genres
edx_genre%>%summarize(genre = n_distinct(genres), .groups = 'drop')
```

```
## # A tibble: 1 x 1
##   genre
##   <int>
## 1    20
```

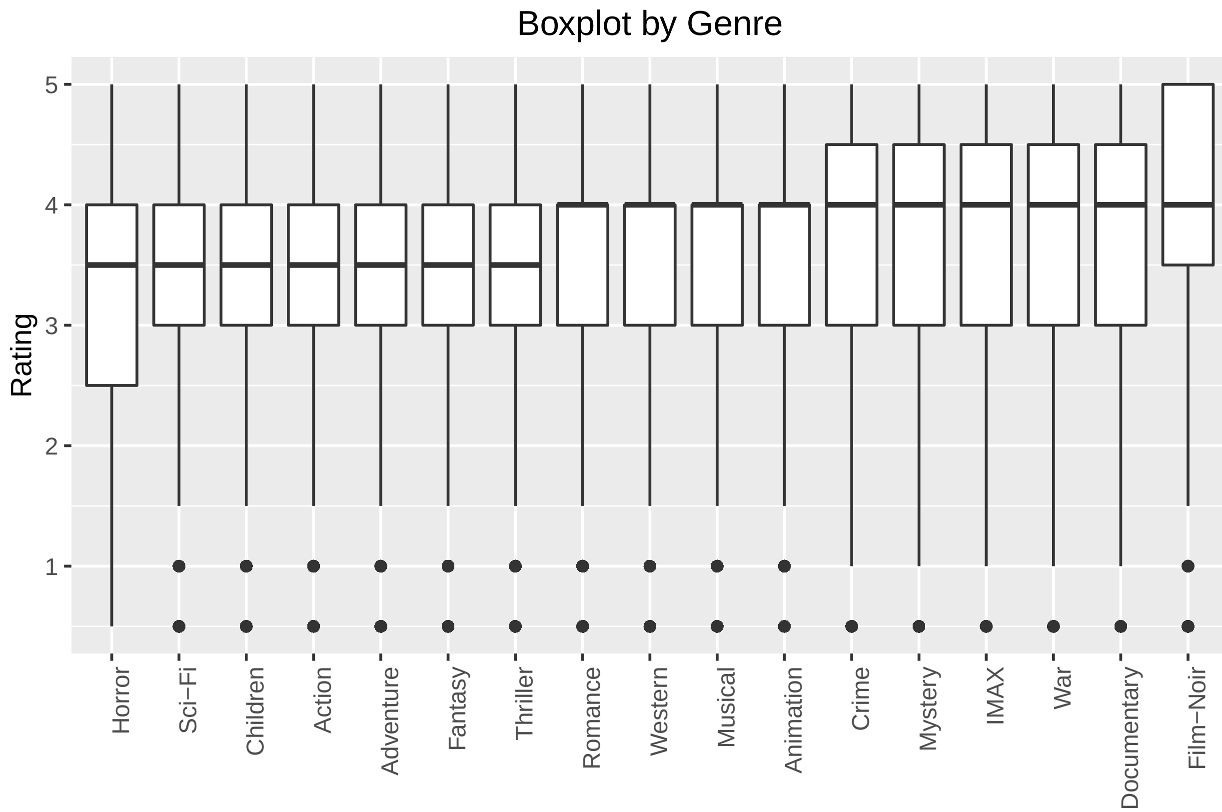
The number of ratings for each genre shows that "Drama", "Comedy" and "Action" were top 3 genres that received the most ratings. "Film-Noir", "Documentaries" and "IMAX" received the least number of ratings.



"Drama" remained a popular choice over the years by receiving the most number of positive ratings of 4.0 and above. As shown in the plot below, "Drama" and "Comedy" received the highest number of rating 4.0 and above in year 2000, while "Documentary" remained flat over time.



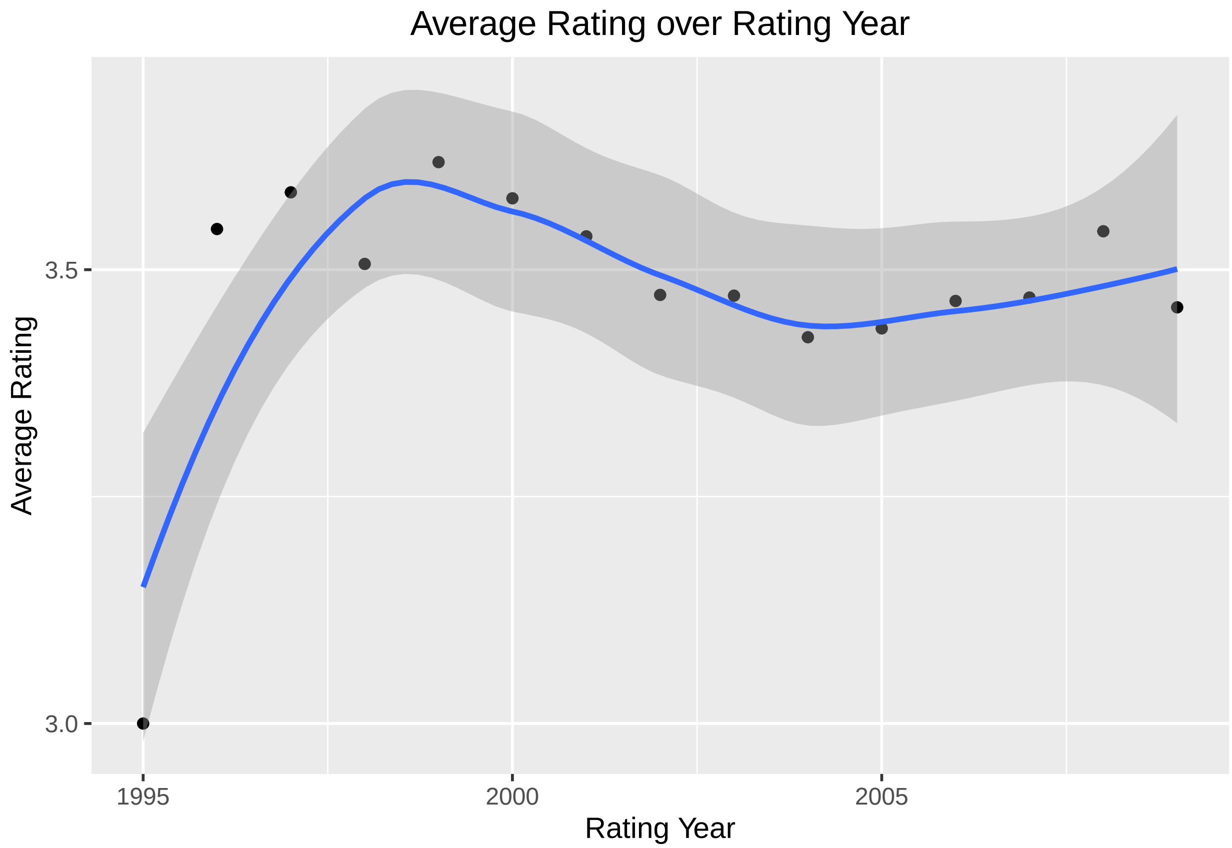
The following box plots show how the ratings for each genre are spread in the **edx** data. For example, "Film-Noir" shows the highest percentage of positive ratings even though this genre received fewer ratings than other genres. Similarly, "Documentary" is a highly rated genre with not many ratings.



Time

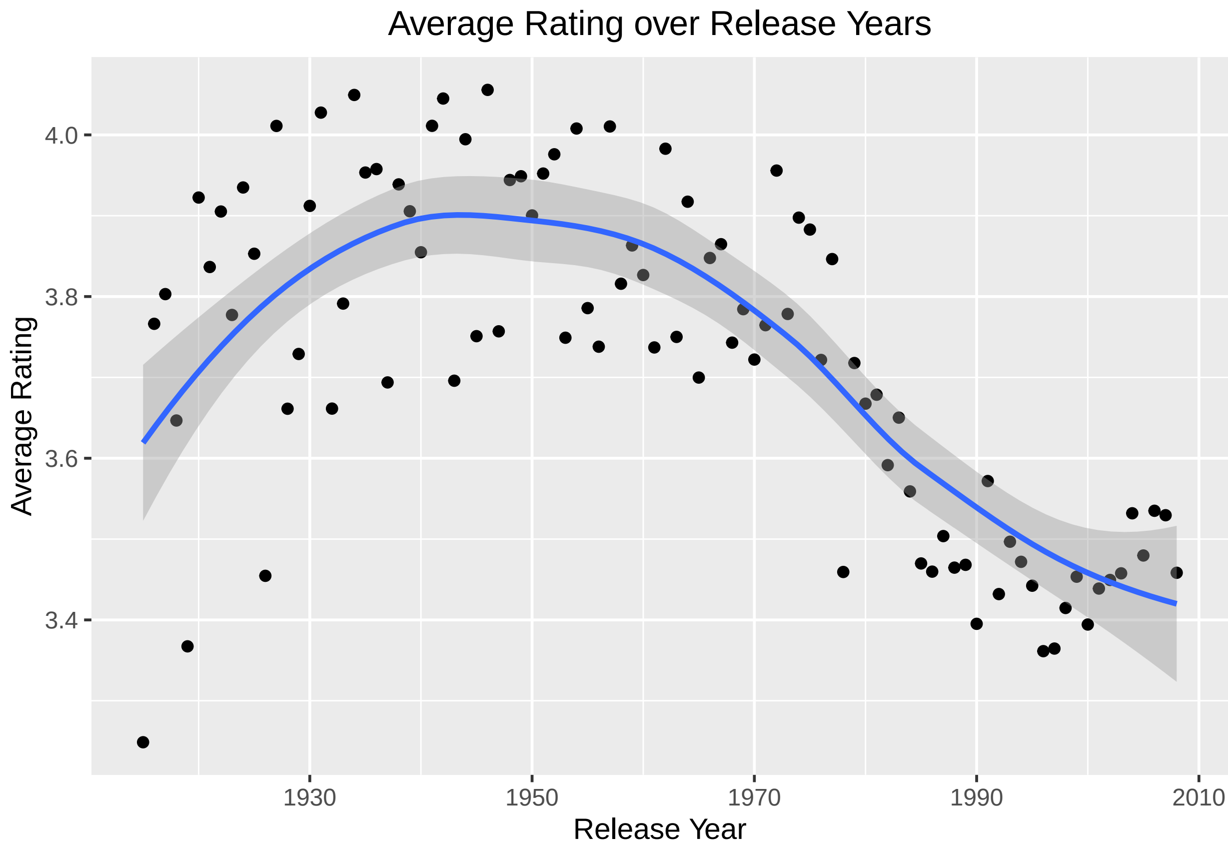
The following plot shows the average rating of movies over time. The average rating has remained close to 3.5 over the data sets history.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



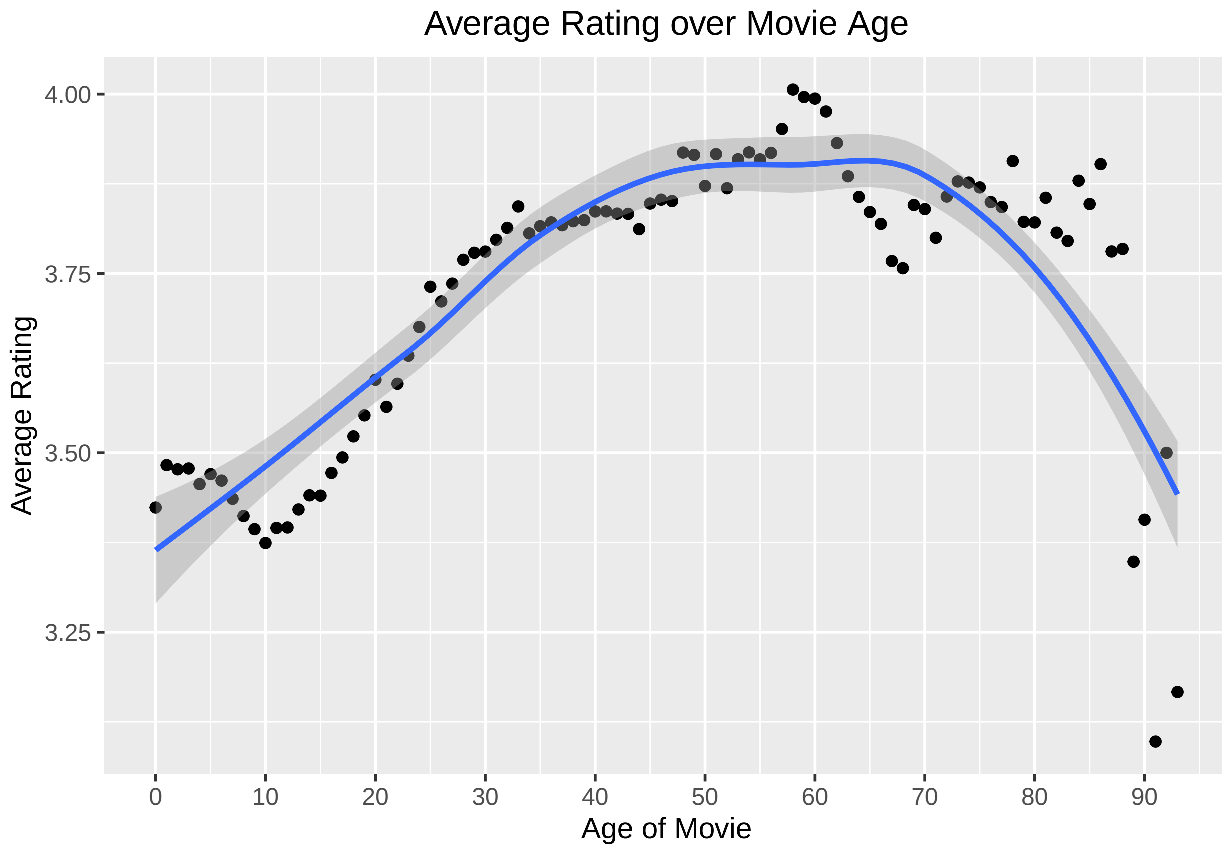
Movies released longer ago don't contain enough data points and show higher variability.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The average rating of a movie increases over time until around 70 years.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Method

In this section all the variables described above were gradually integrated in to models in order to predict ratings with the decreasing value of RMSE.

It was important NOT to use the **validation** set to train the algorithm. The **edx** data was split into a **train__set** and a **test__set**, where the test_set was set to 20% of the **edx** data. The **train__set** was used to train the entire model, whereas the **test__set** was used to estimate predictions of ratings and to calculate the RMSE in order to evaluate the model.

```
set.seed(110)
test_index <- createDataPartition(y=edx$rating, times = 1, p = 0.2, list = FALSE)
test_set <- edx[test_index, ]
train_set <- edx[-test_index, ]
```

The `semi_join` function removes entries for users and movies in **test__set** that don't appear in **train__set**.

```
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

There were 7 possible predictors of rating:

1. *genres*
2. *release__year*
3. *rating__year*
4. *movieId*

5. *userId*
6. *title*
7. *age_years*

Models based on a combination of several predictors were tested on the **train_set**. The **test_set** was used to calculate the RMSE of each model as it was done in the *Netflix challenge*. The RMSE of each model was compared with each other. The model that yielded the smallest RMSE was used for the final evaluation on the **validation** set to test the final algorithm. **RMSE < 0.86490** was considered acceptable.

The RMSE (residual mean squared error) is defined as: $RMSE = \sqrt{\frac{1}{N}\sum_{u,m}(\hat{y}_{u,m} - y_{u,m})^2}$

Where $y_{u,m}$ is observed rating by user u for movie m and $\hat{y}_{u,m}$ is the prediction of the rating. N is a number of movies and users.

The following code for RMSE was used:

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2, na.rm = T))
}
```

The methods described here are from the book by Rafael A. Irizarry[3]. First, the simplest model with movie effects was applied:

$$y_{u,m} = \mu + b_m + \epsilon_{u,m}$$

Where b_m is a movie effect, $y_{u,m}$ observed rating and $\epsilon_{u,m}$ independent errors. This model was extended with additional effects such as b_u - user effect, b_g - genre effect and b_a - age of movie effect.

- (M1) Model with movie effects: $\hat{b_m}$ estimated as an average of $y_{u,m} - \hat{\mu}$, predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m}$
- (M2) Model with movie and user effects: $\hat{b_u}$ estimated as an average of $y_{u,m} - \hat{\mu} - \hat{b_m}$, predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m} + \hat{b_u}$
- (M3) Model with movie, user and genre effects: $\hat{b_g}$ estimated as an average of $y_{u,m} - \hat{\mu} - \hat{b_m} - \hat{b_u}$, predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m} + \hat{b_u} + \hat{b_g}$
- (M4) Model with movie, user, genre and age effects: $\hat{b_a}$ estimated as an average of $y_{u,m} - \hat{\mu} - \hat{b_m} - \hat{b_u} - \hat{b_g}$, predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m} + \hat{b_u} + \hat{b_g} + \hat{b_a}$

In order to limit the variability of this effect a penalty term λ which minimizes effects for the small sample sizes and stabilizes big samples.

- (M5) Model with the regularized movie effects: $\hat{b_m} = \frac{1}{\lambda + n_m}\sum_{n=1}^{n_m}(y_{u,m} - \hat{\mu})$ and $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m}$
- (M6) Model with the regularized regularized movie and user effects: $\hat{b_u} = \frac{1}{\lambda + n_u}\sum_{n=1}^{n_u}(y_{u,m} - \hat{\mu} - \hat{b_m})$ and predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m} + \hat{b_u}$
- (M7) Model with the regularized movie, user and genre effects: $\hat{b_g} = \frac{1}{\lambda + n_g}\sum_{n=1}^{n_g}(y_{u,m} - \hat{\mu} - \hat{b_m} - \hat{b_u})$ and predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m} + \hat{b_u} + \hat{b_g}$
- (M8) Model with the regularized movie, user, genre and age effects: $\hat{b_a} = \frac{1}{\lambda + n_a}\sum_{n=1}^{n_a}(y_{u,m} - \hat{\mu} - \hat{b_m} - \hat{b_u} - \hat{b_g})$ and predicted ratings $\hat{y}_{u,m} = \hat{\mu} + \hat{b_m} + \hat{b_u} + \hat{b_g} + \hat{b_a}$

[3] https://rafalab.github.io/dsbook/

Results

The results of the models (M1) - (M4) are presented:

```
# # # # # # # # # # # # # # # # # # # # #
#  (M1) Model with movie effects
# # # # # # # # # # # # # # # # # # # # #

# Average rating
mu <- mean(train_set$rating)

# Estimate movie effect b_m
movie_ef <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu), .groups = 'drop')

#  Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef, by='movieId') %>%
  mutate(pred = mu + b_m) %>%
  pull(pred)

#  Calculate RMSE of Model 1
rmes_1<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- data_frame(Model = "(M1) Movie Effects", RMSE = rmes_1)
rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295

```
# # # # # # # # # # # # # # # # # # # # # # #
#  (M2) Model with movie and user effects
# # # # # # # # # # # # # # # # # # # # # # #

# Average rating
mu <- mean(train_set$rating)

# Estimate movie effect b_m
movie_ef <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu), .groups = 'drop')
# Estimate user effect b_u
user_ef <- train_set %>%
  left_join(movie_ef, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m), .groups = 'drop')
# Estimate predicted rating
predicted_ratings <- test_set %>%
  left_join(movie_ef, by='movieId') %>%
  left_join(user_ef,  by='userId') %>%
  mutate(pred = mu + b_m + b_u) %>%
  pull(pred)
```

```r
# Calculate RMSE of Model 2
rmse_2<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M2) Movie and User Effects Model",
                                     RMSE = rmse_2))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245

```r
# # # # # # # # # # # # # # # # # # # # # # # # # #
#  (M3) Model with movie, user and genre effects
# # # # # # # # # # # # # # # # # # # # # # # # # #

#  Average rating
mu <- mean(train_set$rating)
#  Estimate movie effect b_m
movie_ef <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu), .groups = 'drop')
#  Estimate user effect b_u
user_ef <- train_set %>%
  left_join(movie_ef, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m), .groups = 'drop')
#  Estimate genre effect b_g
genre_ef<-train_set%>%
  left_join(movie_ef, by="movieId")%>%
  left_join(user_ef, by="userId")%>%
  group_by(genres)%>%
  summarize(b_g = mean(rating - mu - b_m- b_u), .groups = 'drop')

#  Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef, by='movieId') %>%
  left_join(user_ef,  by='userId') %>%
  left_join(genre_ef, by='genres') %>%
  mutate(pred = mu + b_m + b_u + b_g) %>%
  pull(pred)

#  Calculate RMSE of Model 3
rmse_3<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M3) Movie, User and Genre Effects Model",
                                     RMSE = rmse_3))
rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970

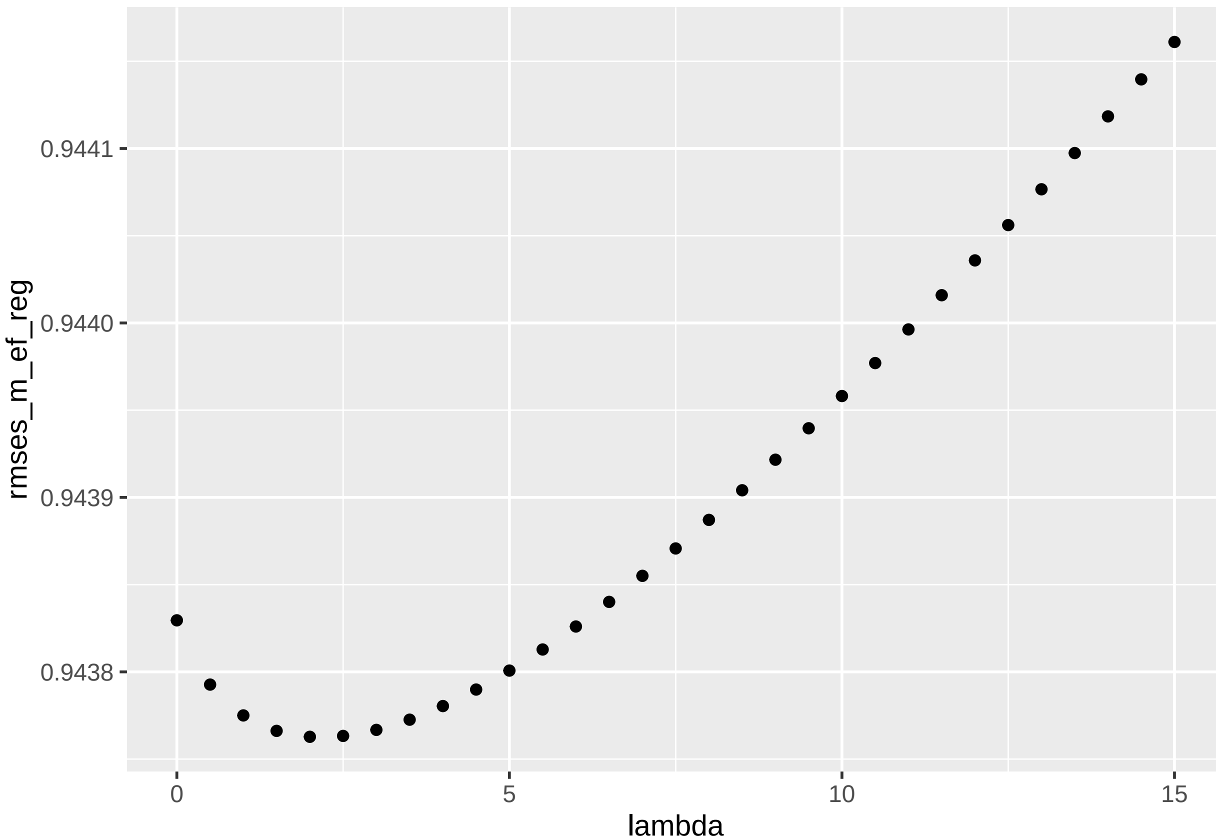
```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#  (M4) Model with movie, user, genre and age effects
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

#  Average rating
mu <- mean(train_set$rating)
# Estimate movie effect b_m
movie_ef <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu), .groups = 'drop')
#  Estimate user effect b_u
user_ef <- train_set %>%
  left_join(movie_ef, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m), .groups = 'drop')
#  Estimate genre effect b_g
genre_ef<-train_set%>%
  left_join(movie_ef, by="movieId")%>%
  left_join(user_ef, by="userId")%>%
  group_by(genres)%>%
  summarize(b_g = mean(rating - mu - b_m- b_u), .groups = 'drop')
#  Estimate age effect b_a
age_ef<-train_set%>%
  left_join(movie_ef, by="movieId")%>%
  left_join(user_ef,  by="userId")%>%
  left_join(genre_ef, by="genres") %>%
  group_by(age_years)%>%
  summarize(b_a = mean(rating - mu - b_m - b_u - b_g), .groups = 'drop')
# Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef, by='movieId') %>%
  left_join(user_ef,  by='userId') %>%
  left_join(genre_ef, by='genres') %>%
  left_join(age_ef,   by ="age_years")%>%
  mutate(pred = mu + b_m + b_u + b_g  + b_a) %>%
  pull(pred)

rmse_4<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M4) Movie, User, Genre and Age Effects Model",
                                     RMSE = rmse_4))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295

Model	RMSE
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970
(M4) Movie, User, Genre and Age Effects Model	0.8655156

The results showed that by adding additional effect the value of RMSE went down. (M1) Model with movie effects with only one effect resulted in a greatest RMSE among all models.

In this part, results of the models (M5) - (M8) with the regularized effects are presented. First the optimal (with the smallest RMSE) tuning parameter λ was chosen.



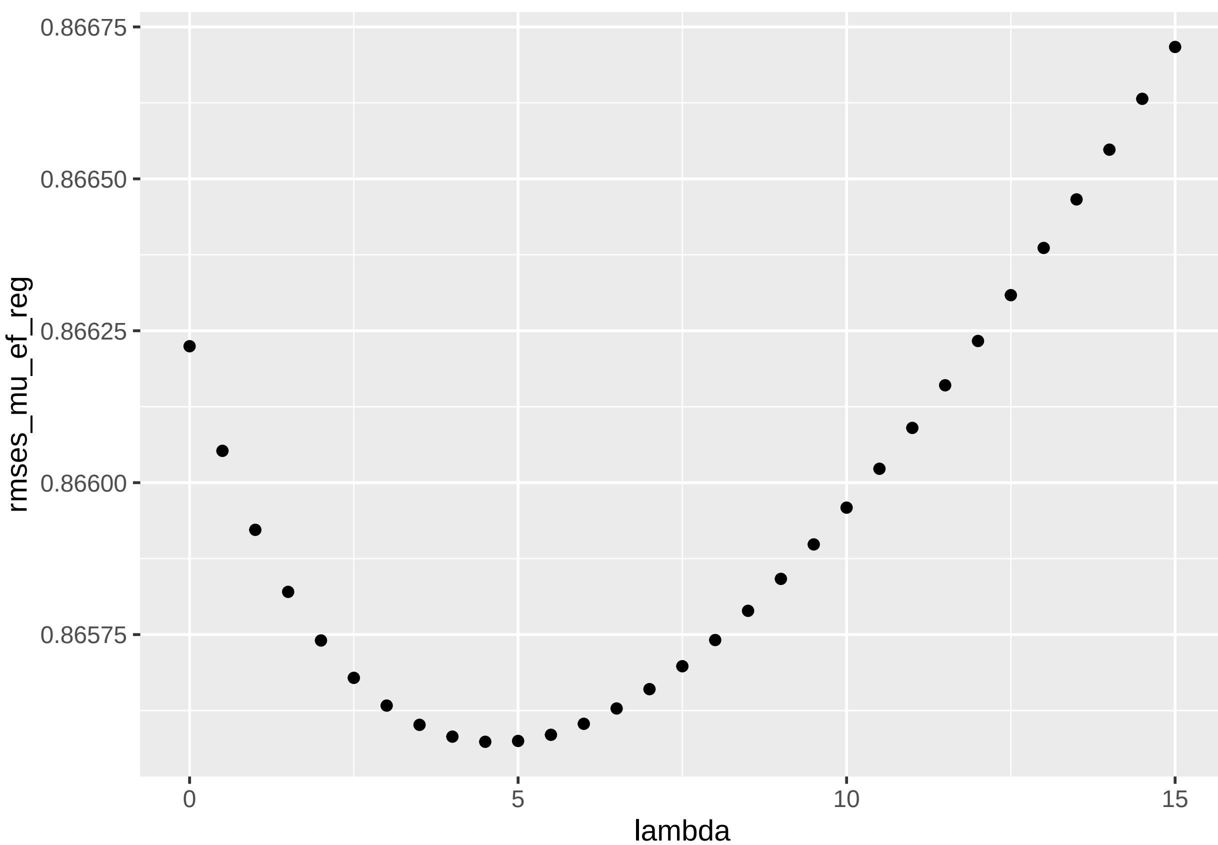

```
#  Choose tuning parameter l_1 with minimal RMSE
l_1<-lambda[which.min(rmses_m_ef_reg)]
l_1
```

```
## [1] 2
```

```
# Average rating
mu <- mean(train_set$rating)
# Estimate movie effect b_m
movie_ef_reg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+l_1), n_m = n(), .groups = 'drop')
# Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef_reg, by = "movieId") %>%
```

```r
  mutate(pred = mu + b_m) %>%
  pull(pred)
# Calculate RMSE of Model 5
rmse_5<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M5) Regularized Movie Effects Model",
                                     RMSE = rmse_5))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970
(M4) Movie, User, Genre and Age Effects Model	0.8655156
(M5) Regularized Movie Effects Model	0.9437628

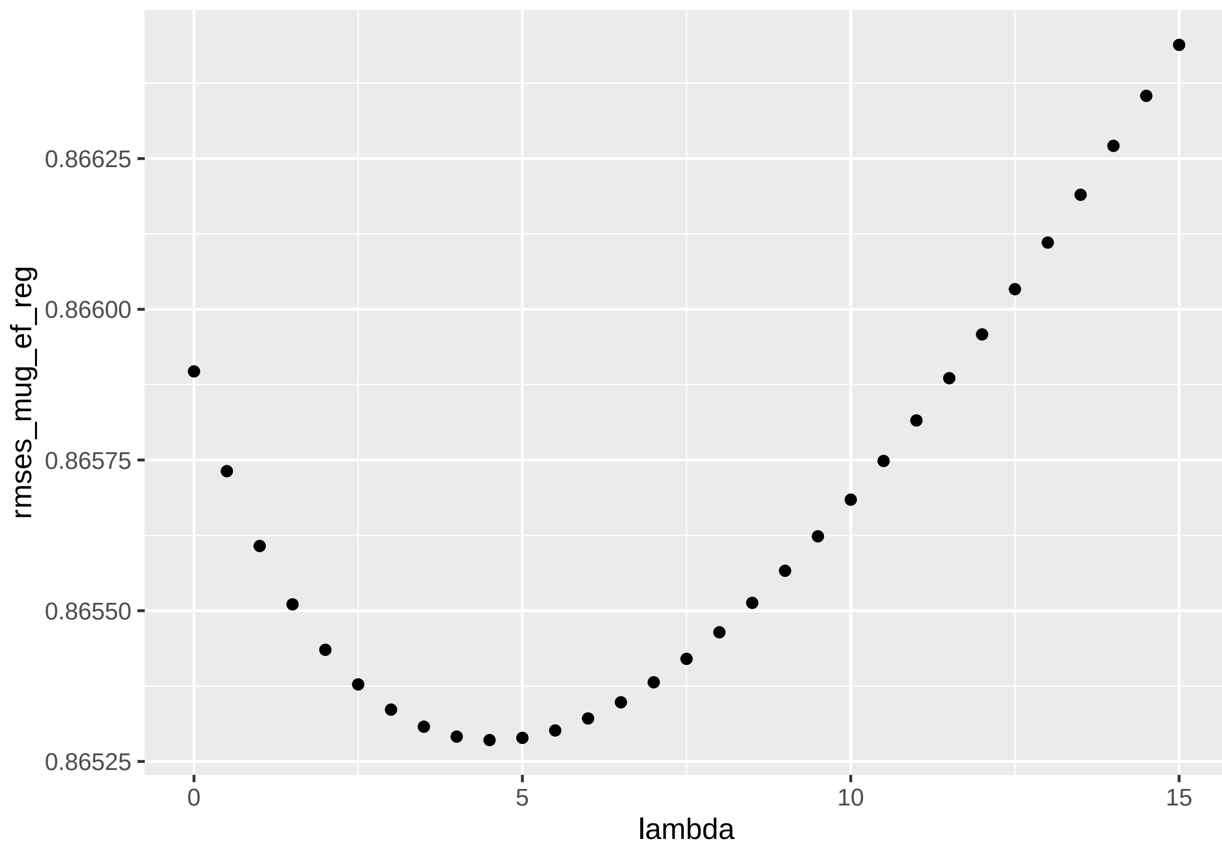



```r
#  tuning parameter l_2 with minimal RMSE
l_2<-lambda[which.min(rmses_mu_ef_reg)]
l_2
```

```
## [1] 4.5
```

```r
# Average rating
mu <- mean(train_set$rating)
# Estimate movie effect b_m
movie_ef_reg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+l_2), n_m= n(), .groups = 'drop')
# Estimate user effect b_u
user_ef_reg <- train_set %>%
  left_join(movie_ef_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - mu)/(n() +l_2), n_u = n(), .groups = 'drop')
#  Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg, by = "userId") %>%
  mutate(pred = mu + b_m + b_u) %>%
  pull(pred)

# Calculate RMSE of Model 6
rmse_6<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M6) Regularized Movie and User Effects Model",
                                     RMSE = rmse_6))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970
(M4) Movie, User, Genre and Age Effects Model	0.8655156
(M5) Regularized Movie Effects Model	0.9437628
(M6) Regularized Movie and User Effects Model	0.8655736



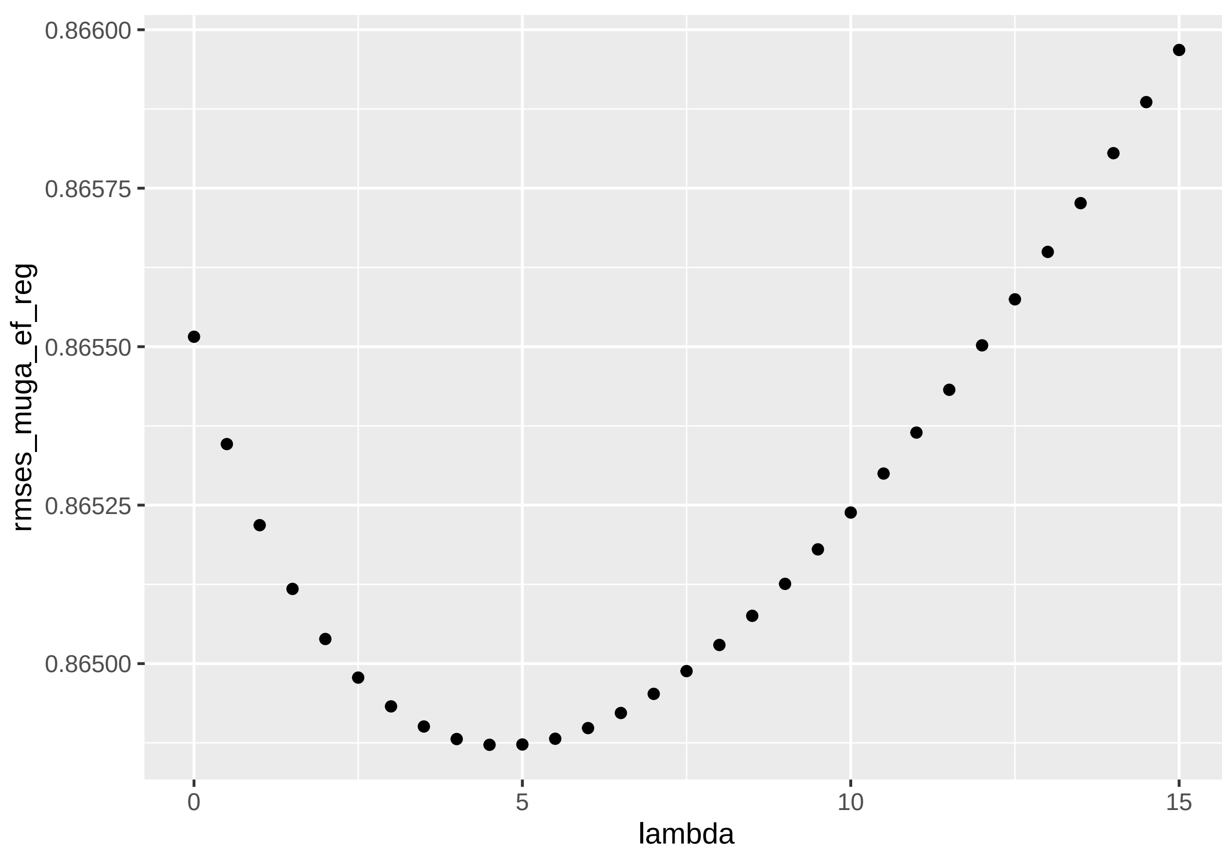

```
#  tuning parameter l_3 with minimal RMSE
l_3<-lambda[which.min(rmses_mug_ef_reg)]
l_3
```

```
## [1] 4.5
```

```
#  Average rating
mu <- mean(train_set$rating)
# Estimate movie effect b_m
movie_ef_reg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+l_3), n_m= n(), .groups = 'drop')
# Estimate user effect b_u
user_ef_reg <- train_set %>%
  left_join(movie_ef_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - mu)/(n() +l_3), n_u = n(), .groups = 'drop')
#  Estimate genre effect b_g
genre_ef_reg <- train_set %>%
  left_join(movie_ef_reg, by="movieId") %>%
  left_join(user_ef_reg, by="userId") %>%
  group_by(genres)%>%
  summarize(b_g = sum(rating - b_m - b_u - mu)/(n()+l_3), n_g = n(), .groups = 'drop')
#  Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg, by = "userId") %>%
```

```r
  left_join(genre_ef_reg, by="genres")%>%
  mutate(pred = mu + b_m + b_u + b_g) %>%
  pull(pred)

# Calculate RMSE of Model 7
rmse_7<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M7) Regularized Movie, User and Genre Effects Model",
                                     RMSE = rmse_7))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970
(M4) Movie, User, Genre and Age Effects Model	0.8655156
(M5) Regularized Movie Effects Model	0.9437628
(M6) Regularized Movie and User Effects Model	0.8655736
(M7) Regularized Movie, User and Genre Effects Model	0.8652854




```r
#  tuning parameter l_4 with minimal RMSE
l_4<-lambda[which.min(rmses_muga_ef_reg)]
```

```r
l_4
```

```
## [1] 4.5
```

```r
#  Average rating
mu <- mean(train_set$rating)
# Estimate movie effect b_m
movie_ef_reg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+l_4), n_m= n(), .groups = 'drop')
# Estimate user effect b_u
user_ef_reg <- train_set %>%
  left_join(movie_ef_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - mu)/(n() +l_4), n_u = n(), .groups = 'drop')
#  Estimate genre effect b_g
genre_ef_reg <- train_set %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg,  by = "userId") %>%
  group_by(genres)%>%
  summarize(b_g = sum(rating - b_m - b_u - mu)/(n()+l_4), n_g = n(), .groups = 'drop')
# Estimate age effect b_a
age_ef_reg <- train_set %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg,  by = "userId") %>%
  left_join(genre_ef_reg, by = "genres")%>%
  group_by(age_years)%>%
  summarize(b_a = sum(rating - b_g - b_m - b_u - mu)/(n()+l_4), n_a = n(), .groups = 'drop')
#  Estimate predicted ratings
predicted_ratings <- test_set %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg,  by = "userId") %>%
  left_join(genre_ef_reg, by = "genres")%>%
  left_join(age_ef_reg,   by = "age_years")%>%
  mutate(pred = mu + b_m + b_u + b_g + b_a) %>%
  pull(pred)

# Calculate RMSE of Model 8
rmse_8<-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="(M8) Regularized Movie, User, Genre and Age Effects Model",
                                     RMSE = rmse_8))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970
(M4) Movie, User, Genre and Age Effects Model	0.8655156
(M5) Regularized Movie Effects Model	0.9437628
(M6) Regularized Movie and User Effects Model	0.8655736
(M7) Regularized Movie, User and Genre Effects Model	0.8652854

Model	RMSE
(M8) Regularized Movie, User, Genre and Age Effects Model	0.8648720

Since the smallest **RMSE = 0.8648** was produced by (M8) Regularized Movie, User, Genre and Age Effects Model, this model was used on the **edx** data to train the algorithm, and on the **validation** data to test and to calculate the final RMSE.

```
# # # # # # # # # # #
#  Final Model
# # # # # # # # # # #

# Validation Data processing

#  Split title and year into separate columns by using regex
validation<-extract(validation, title, c("title", "release_year"), "(.*)\\((\\d{4})\\)$")
#  Convert year character into to an integer
validation<-transform(validation, release_year = as.numeric(release_year))
#  Transform the rating timestamp to datetime year
validation<-transform(validation, rating_year = year(as_datetime(timestamp)))
#  Create an age_years variable that represents time in years
#  between release time and time when rating was given
validation<-validation%>%mutate(age_years=as.numeric(rating_year)-as.numeric(release_year))%>%filter(ag
```

```
#  tuning parameter l_4 with minimal RMSE
l_4<-lambda[which.min(rmses_muga_ef_reg)]
l_4
```

```
## [1] 4.5
```

```
#  Average rating
mu <- mean(edx$rating)
# Estimate movie effect b_m
movie_ef_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+l_4), n_m= n(), .groups = 'drop')
# Estimate user effect b_u
user_ef_reg <- edx %>%
  left_join(movie_ef_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - mu)/(n() +l_4), n_u = n(), .groups = 'drop')
#  Estimate genre effect b_g
genre_ef_reg <- edx %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg,  by = "userId") %>%
  group_by(genres)%>%
  summarize(b_g = sum(rating - b_m - b_u - mu)/(n()+l_4), n_g = n(), .groups = 'drop')
# Estimate age effect b_a
age_ef_reg <- edx %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg,  by = "userId") %>%
  left_join(genre_ef_reg, by = "genres")%>%
  group_by(age_years)%>%
  summarize(b_a = sum(rating - b_g - b_m - b_u - mu)/(n()+l_4), n_a = n(), .groups = 'drop')
#  Estimate predicted ratings
```

```r
predicted_ratings <- validation %>%
  left_join(movie_ef_reg, by = "movieId") %>%
  left_join(user_ef_reg,  by = "userId") %>%
  left_join(genre_ef_reg, by = "genres")%>%
  left_join(age_ef_reg,   by = "age_years")%>%
  mutate(pred = mu + b_m + b_u + b_g + b_a) %>%
  pull(pred)

# Calculate RMSE of Final Model
rmse_final<-RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          tibble(Model="Final Regularized Movie, User, Genre and Age Effects Model",
                                 RMSE = rmse_final))

rmse_results%>% knitr::kable()
```

Model	RMSE
(M1) Movie Effects	0.9438295
(M2) Movie and User Effects Model	0.8662245
(M3) Movie, User and Genre Effects Model	0.8658970
(M4) Movie, User, Genre and Age Effects Model	0.8655156
(M5) Regularized Movie Effects Model	0.9437628
(M6) Regularized Movie and User Effects Model	0.8655736
(M7) Regularized Movie, User and Genre Effects Model	0.8652854
(M8) Regularized Movie, User, Genre and Age Effects Model	0.8648720
Final Regularized Movie, User, Genre and Age Effects Model	0.8628988

The final calculated **RMSE= 0.8629** was in acceptable range (RMSE<0.86490).

Conclusion

The algorithm was developed by splitting the **edx** data into **test__ set** and **train__set**. In the last step movie ratings were predicted with the **validation** set as if they were unknown. The acceptable RMSE was reached by applying "Regularized Movie, User, Genre and Age Effects Model" with $\lambda = 4.5$. While working on this project the main difficulty was the the memory capacity which led to the reduction of the visualization and analysis techniques. The calculations were run on AWS.