Capstone project

Sofya Rbinovich

16/10/2021

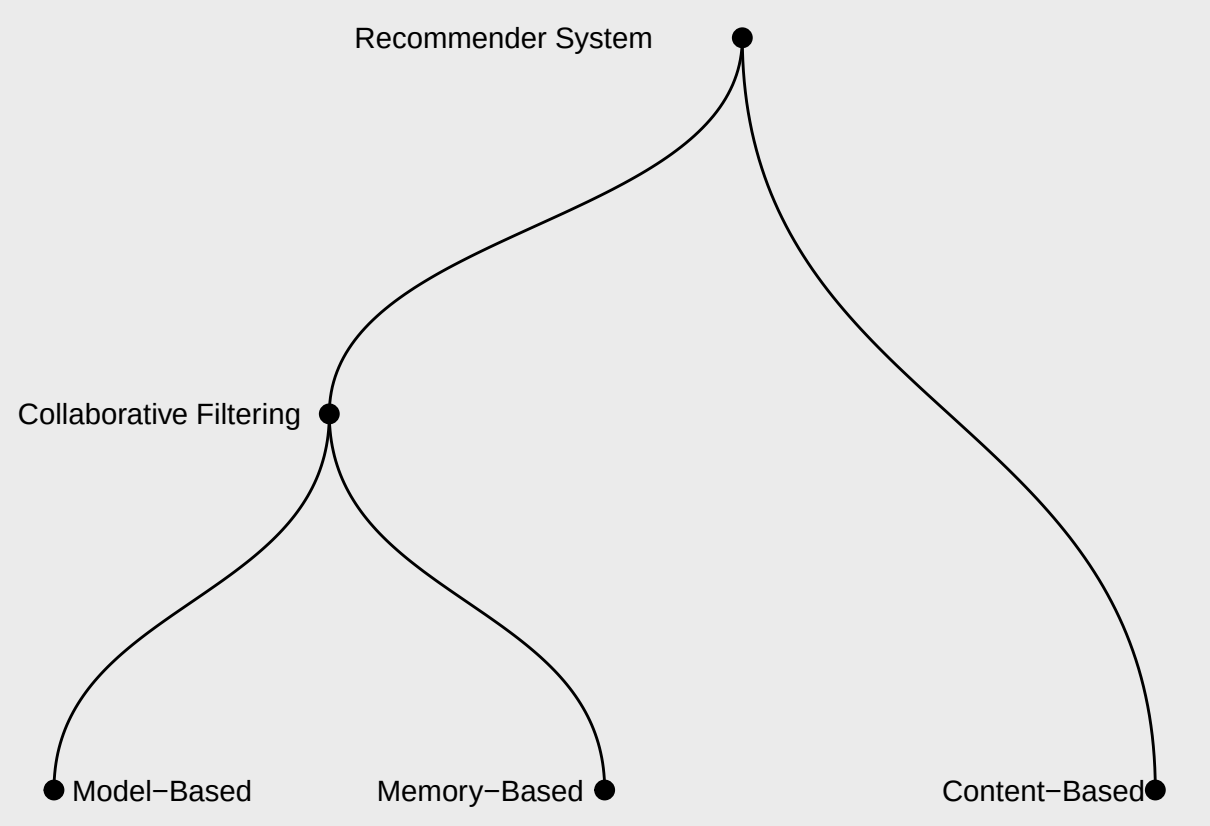
Used Libraries

```
library(broom)
library(dslabs)
library(HistData)
library(tidyverse)
library(tidyr)
library(tidyselect)
library(tidytext)
library(dplyr)
library(gridExtra)
library(ggplot2)
library(ggpubr)
library(ggrepel)
library(ggsci)
library(ggsignif)
library(ggthemes)
library(readxl)
library(readr)
library(reshape2)
library(lpSolve)
library(lubridate)
library(caret)
library(e1071)
library(MASS)
library(purrr)
library(pdftools)
library(matrixStats)
library(rpart)
library(recosystem)
library(Rborist)
library(randomForest)
library(ggraph)
library(igraph)
```

Introduction

The “Movielense” project was made for the capstone project of the HarvardX: PH125.9x module. Recommedation systems are an important part of the machine learning algorithms and help to offer a suggestions

for the users according their preferences and selection of movies/products. Companies such as LinkedIn, Amazon, Netflix and etc. are using recommender systems to satisfy and ease the search of their customers. Recommender systems are generally divided into a collaborative filtering and content-based systems. Collaborative filtering system is based on the previous selections of users, so the input will be organised as a historical data of user interaction with targets. The data is stored in the matrix where the userId is the row and the itemId is in the columns (the same as we will use in our model later in this project). On the other hand, content-based model will use other inputs to make a predictions. Additional inputs might be age, job, sex, location and other personal information information provided by the user while we was registering.



In this project we will look at the Netflix Prize open competition that was introduced on the 2nd of October 2006. The aim of the competition was to provide the best collaborative filtering algorithm to predict the user rating to the film based on the previous ratings using limited information. By the June 2007 over 20 000 teams had registered for the competition and in the September 2009 the team called "BellKor's Pragmatic Chaos" won the prize and achieved RMSE = 0.8567. The grand prize was US$1,000,000.

Aims of the project

The aim of the project is to train different linear models to achieve the most accurate RMSE result using the provided training set (edx) and test set (validation). Objectives: 1. Explore the data; 2. Look for the correlation between different parameters; 3. Preprocess the data by removing any NAs and zero variance parameters; 4. Train different linear models and compare the RMSE results.

Data Download

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1,
                                  list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data Exploration

Overview data

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

userId	movieId	rating	timestamp	title	genres
56915	62245	4.0	1223985447	Music Room, The (Jalsaghar) (1958)	Drama
59269	59680	3.0	1229014701	One Hour with You (1932)	Comedy|Musical|Romance
59269	64325	3.0	1229014646	Long Night, The (1947)	Crime|Drama|Film-Noir|Romance|Thriller
59342	61768	0.5	1230070861	Accused (Anklaget) (2005)	Drama
60713	4820	2.0	1119156754	Won't Anybody Listen? (2000)	Documentary
68986	61950	3.5	1223376391	Boot Camp (2007)	Thriller

Now we will investigate the number of unique movies and the number of unique users

```
table_of_unique_movies_and_users <- data.frame(unique_users = edx %>%
                                                 summarise(n_users = n_distinct(userId)),
                                               unique_movies = edx %>%
                                                 summarise(n_movies = n_distinct(movieId))) %>%

  knitr::kable()
table_of_unique_movies_and_users
```

n_users	n_movies
69878	10677

Therefore, we can conclude that the number of users is 69878 and the number of movies is 10677 in the edx dataset.

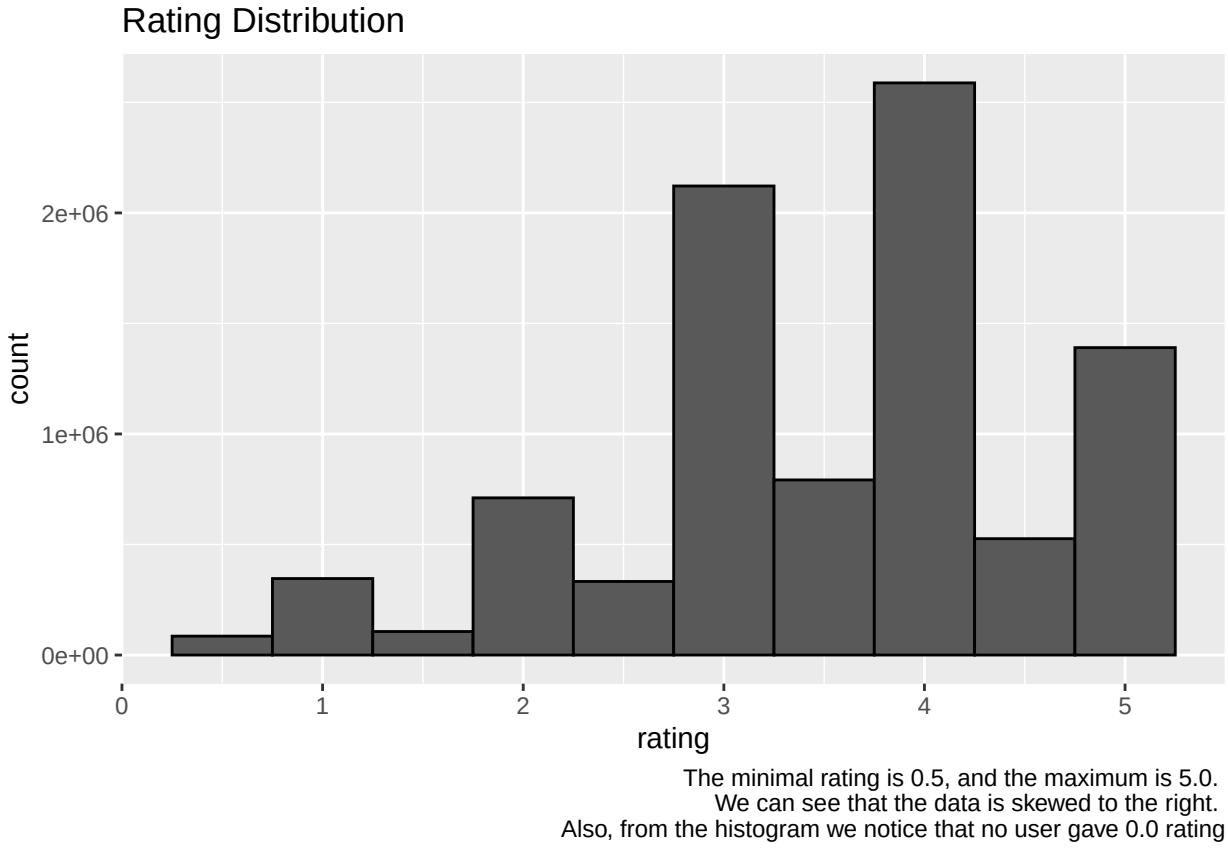
```
knitr::kable(summary(edx))
```

	userId	movieId	rating	timestamp	title	genres
	Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000061	Length:9000061
	1st Qu.:18122	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
	Median :35743	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
	Mean :35869	Mean : 4120	Mean :3.512	Mean :1.033e+09	NA	NA
	3rd Qu.:53602	3rd Qu.: 3624	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA

userId	movieId	rating	timestamp	title	genres
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

From the table above we can see that the minimum rating given to a movie was 0.5, meanwhile, the highest rating given was 5.0.

```
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.5, color = I('black')) +
  ggtitle("Rating Distribution") +
  labs(caption = "The minimal rating is 0.5, and the maximum is 5.0.
       We can see that the data is skewed to the right.
       Also, from the histogram we notice that no user gave 0.0 rating")
```

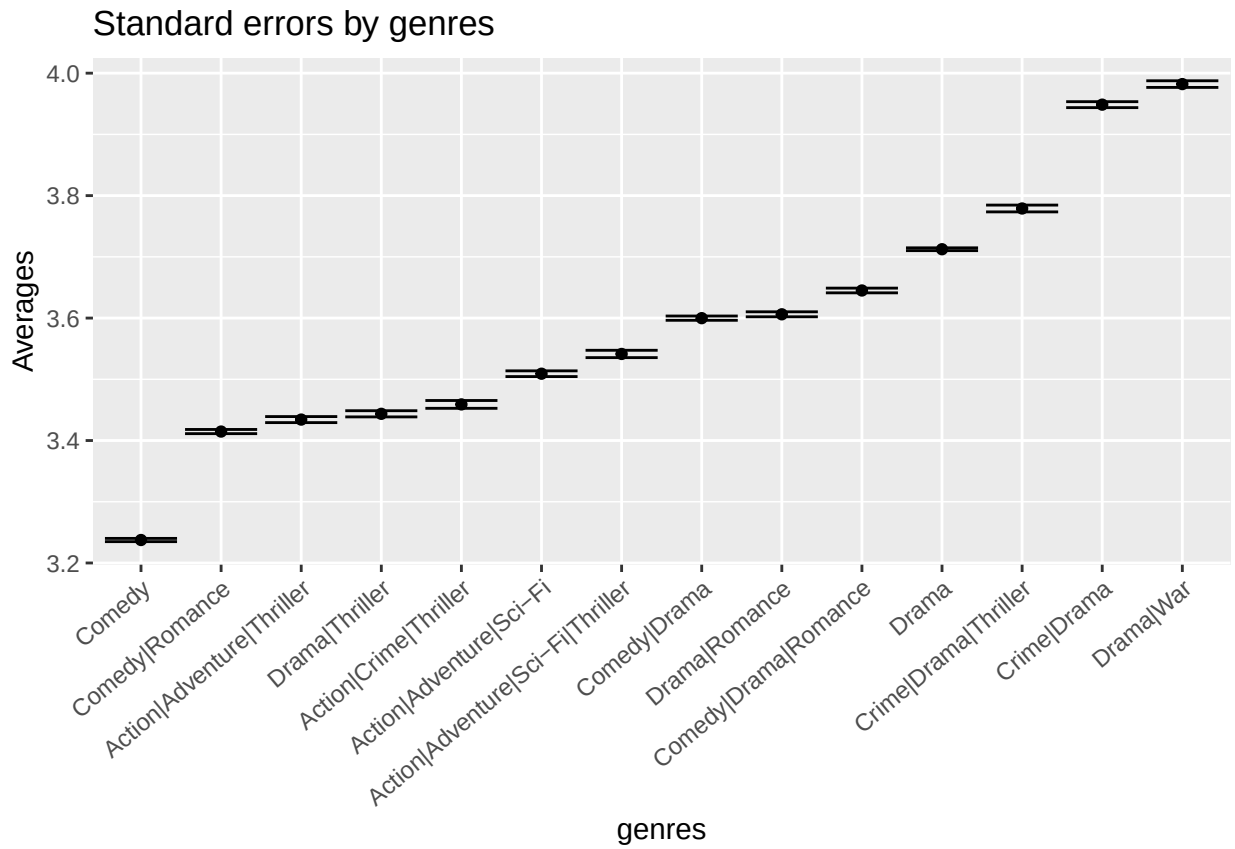


It is obvious that some people preferred one movie genre to another, therefore, it is worth to investigate the effect of the genre on the movie rating. We will look at top 20 genres in the data set.

```
top_genres <- edx %>%
  group_by(genres) %>%
  summarise(count = n()) %>%
  top_n(20, count) %>%
  arrange(desc(count)) %>%
  knitr::kable()
top_genres
```

genres	count
Drama	733353
Comedy	700883
Comedy|Romance	365894
Comedy|Drama	323518
Comedy|Drama|Romance	261098
Drama|Romance	259735
Action|Adventure|Sci-Fi	220363
Action|Adventure|Thriller	148933
Drama|Thriller	145359
Crime|Drama	137424
Drama|War	111122
Crime|Drama|Thriller	105893
Action|Adventure|Sci-Fi|Thriller	104906
Action|Crime|Thriller	102120
Action|Drama|War	99086
Action|Thriller	96235
Action|Sci-Fi|Thriller	95486
Thriller	94651
Horror|Thriller	75106
Comedy|Crime	73343

```
edx %>%
  group_by(genres) %>%
  summarise(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 100000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 40, hjust = 1)) +
  ylab("Averages") +
  ggtitle("Standard errors by genres")
```



We can see that the top genre is Drama followed by Comedy and Action. It is interesting to know, which movies are at the top.

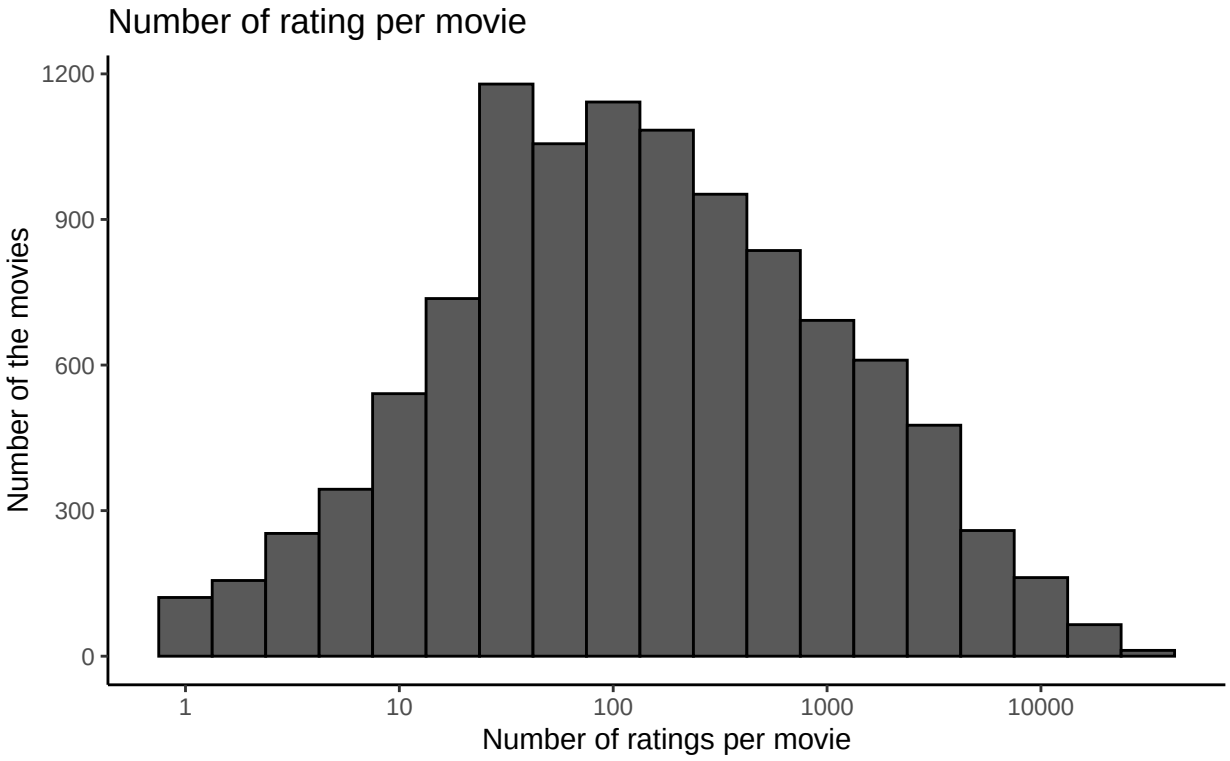
```
top_movies <- edx %>%
  group_by(title) %>%
  summarise(rating_count = n()) %>%
  top_n(15, rating_count) %>%
  arrange(desc(rating_count)) %>%
  knitr::kable()
top_movies
```

title	rating_count
Pulp Fiction (1994)	31336
Forrest Gump (1994)	31076
Silence of the Lambs, The (1991)	30280
Jurassic Park (1993)	29291
Shawshank Redemption, The (1994)	27988
Braveheart (1995)	26258
Terminator 2: Judgment Day (1991)	26115
Fugitive, The (1993)	26050
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25809
Batman (1989)	24343
Apollo 13 (1995)	24277
Toy Story (1995)	23826
Independence Day (a.k.a. ID4) (1996)	23360

title	rating_count
Dances with Wolves (1990)	23312
Schindler's List (1993)	23234

As we established there are 10677 movies in the data set and using the logic we can say that some movies are rated watched more than others and therefore are rated more frequently. Whereas, others can be rated only few times.

```
edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 0.25, bins = 30, color = I('black')) +
  scale_x_log10() +
  theme_classic() +
  xlab("Number of ratings per movie") +
  ylab("Number of the movies") +
  ggtitle("Number of rating per movie") +
  labs(caption = "The distribution is almost symmetric.
  Few rating count can make the model inaccurate.
  Therefore these movies should be excluded from the list in the preprocessiing step.")
```



We can see that some movies were rated only one time, and some were rated for more than 10'000 time. Now we should define which movies were rated only once in order to consider their exclusion from the data set in the preprocessiong step.

```
rated_1_time <- edx %>%
  group_by(movieId) %>%
  summarise(number_of_ratings = n()) %>%
  filter(number_of_ratings == 1) %>%
  left_join(edx, by = "movieId") %>%
  dplyr::select(title, movieId, number_of_ratings) %>%
  knitr::kable()
rated_1_time
```

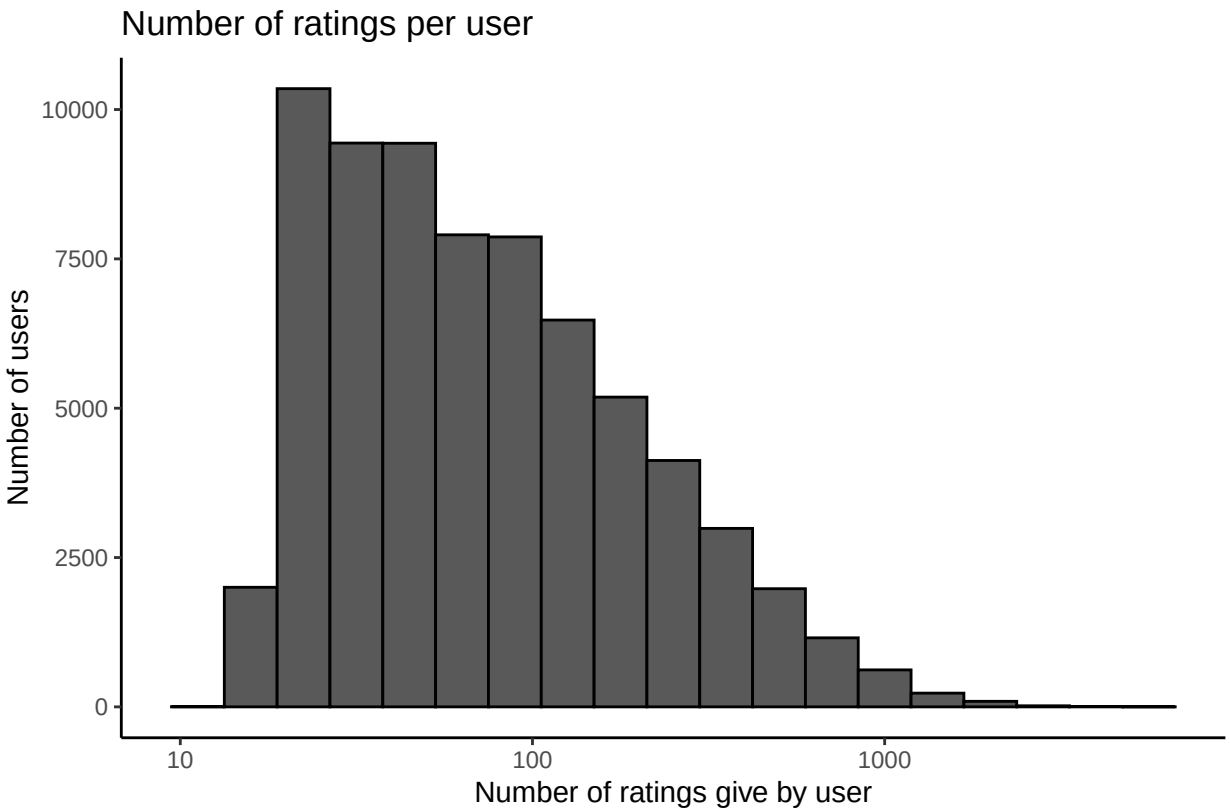
title	movieId	number_of_ratings
Hellhounds on My Trail (1999)	3226	1
Train Ride to Hollywood (1978)	3234	1
Condo Painting (2000)	3356	1
Stacy's Knights (1982)	3561	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3583	1
Dog Run (1996)	4071	1
Monkey's Tale, A (Les Château des singes) (1999)	4075	1
Won't Anybody Listen? (2000)	4820	1
Spooky House (2000)	5320	1
Dogwalker, The (2002)	5565	1
Neil Young: Human Highway (1982)	6085	1
Dischord (2001)	6189	1
Boys Life 4: Four Play (2003)	6589	1
Emerald Cowboy (2002)	6758	1
Once in the Life (2000)	6838	1
Just an American Boy (2003)	6941	1
Love Forbidden (Défense d'aimer) (2002)	6943	1
Prisoner of Paradise (2002)	7145	1
Mickey (2003)	7452	1
Du côté de la côte (1958)	7537	1
Love Life (2001)	7568	1
Hundred and One Nights, A (Cent et une nuits de Simon Cinéma, Les) (1995)	7750	1
Demon Lover Diary (1980)	7823	1
Hi-Line, The (1999)	8394	1
Grief (1993)	8707	1
Thérèse (2004)	8945	1
Prelude to War (Why We Fight, 1) (1943)	25885	1
Deadly Companions, The (1961)	26067	1
Blind Shaft (Mang jing) (2003)	27740	1
Lessons of Darkness (Lektionen in Finsternis) (1992)	31547	1
Down and Derby (2005)	33140	1
Death of a Dynasty (2003)	33160	1
Revenge of the Ninja (1983)	36752	1
4 (2005)	37957	1
Forty Shades of Blue (2005)	38435	1
Living 'til the End (2005)	39412	1
Confess (2005)	39429	1
God's Sandbox (Tahara) (2002)	39439	1
Fists in the Pocket (I Pugni in tasca) (1965)	40833	1
Shadows of Forgotten Ancestors (1964)	42783	1
Ace of Hearts, The (1921)	45707	1

title	movieId	number_of_ratings
Brothers of the Head (2005)	46581	1
Father Sergius (Otets Sergiy) (1917)	48374	1
Chapayev (1934)	48649	1
Man of Straw (Untertan, Der) (1951)	48899	1
Testament of Orpheus, The (Testament d'Orphée) (1960)	50477	1
Fighting Elegy (Kenka erejii) (1966)	51209	1
Diggers (2006)	52865	1
Flandres (2006)	53342	1
Sun Alley (Sonnenallee) (1999)	53355	1
David Holzman's Diary (1967)	53833	1
Last Time, The (2006)	54144	1
Cruel Story of Youth (Seishun zankoku monogatari) (1960)	54318	1
Relative Strangers (2006)	55324	1
Symbiopsychotaxiplasm: Take One (1968)	56253	1
Please Vote for Me (2007)	58185	1
Mala Noche (1985)	58520	1
Quiet City (2007)	58595	1
Hey Hey It's Esther Blueburger (2008)	59625	1
One Hour with You (1932)	59680	1
Cold Sweat (De la part des copains) (1970)	59840	1
Wings of Eagles, The (1957)	59854	1
Archangel (1990)	60189	1
Dog Day (Canicule) (1984)	60237	1
Bad Blood (Mauvais sang) (1986)	60336	1
Borderline (1950)	60494	1
Jimmy Carter Man from Plains (2007)	60666	1
Family Game, The (Kazoku gêmu) (1983)	60880	1
Variety Lights (Luci del varietà) (1950)	61279	1
Half Life of Timofey Berezin, The (PU-239) (2006)	61586	1
Flu Bird Horror (2008)	61638	1
Ladrones (2007)	61695	1
Maradona by Kusturica (2008)	61742	1
Accused (Anklaget) (2005)	61768	1
In Bed (En la cama) (2005)	61862	1
Africa addio (1966)	61913	1
100 Feet (2008)	61948	1
Moonbase (1998)	61970	1
Dead Man's Letters (Pisma myortvogo cheloveka) (1986)	62063	1
Part of the Weekend Never Dies (2008)	62237	1
Music Room, The (Jalsaghar) (1958)	62245	1
Magicians (2007)	62378	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...) (1971)	62474	1
Deux mondes, Les (2007)	62660	1
Fallout (1998)	62671	1
Rockin' in the Rockies (1945)	63141	1
Tokyo! (2008)	63179	1
Caótica Ana (2007)	63194	1
Gold Raiders (1951)	63271	1
Säg att du älskar mig (2006)	63327	1
Guard Post, The (G.P. 506) (2008)	63332	1
Where A Good Man Goes (Joi gin a long) (1999)	63662	1
Gaucho, The (1927)	63688	1

title	movieId	number_of_ratings
Ring of Darkness (2004)	63806	1
Splinter (2008)	63826	1
Kanak Attack (2000)	63840	1
Fireproof (2008)	64114	1
Devil's Chair, The (2006)	64153	1
Bell Boy, The (1918)	64245	1
Blue Light, The (Das Blaue Licht) (1932)	64275	1
Women of the Night (Yoru no onnatachi) (1948)	64283	1
Ace of Hearts (2008)	64288	1
Long Night, The (1947)	64325	1
Fools' Parade (1971)	64327	1
Malaya (1949)	64365	1
Man Named Pearl, A (2006)	64418	1
Much Ado About Something (2001)	64427	1
Delgo (2008)	64652	1
Love (2005)	64754	1
Mr. Wu (1927)	64897	1
Nazis Strike, The (Why We Fight, 2) (1943)	64903	1
Battle of Russia, The (Why We Fight, 5) (1943)	64926	1
Face of a Fugitive (1959)	64944	1
Dirty Dozen, The: The Fatal Mission (1988)	64953	1
Hexed (1993)	64976	1
5 Centimeters per Second (Byôsoku 5 senchimêtoru) (2007)	64993	1
Constantine's Sword (2007)	65001	1
Impulse (2008)	65006	1
Zona Zamfirova (2002)	65011	1
Double Dynamite (1951)	65025	1
Death Kiss, The (1933)	65027	1

There are 126 movies that are rated only once and it is 1.18% only from all the movies. As we defined previously there is 69878 and we can check the number of ratings given by them.

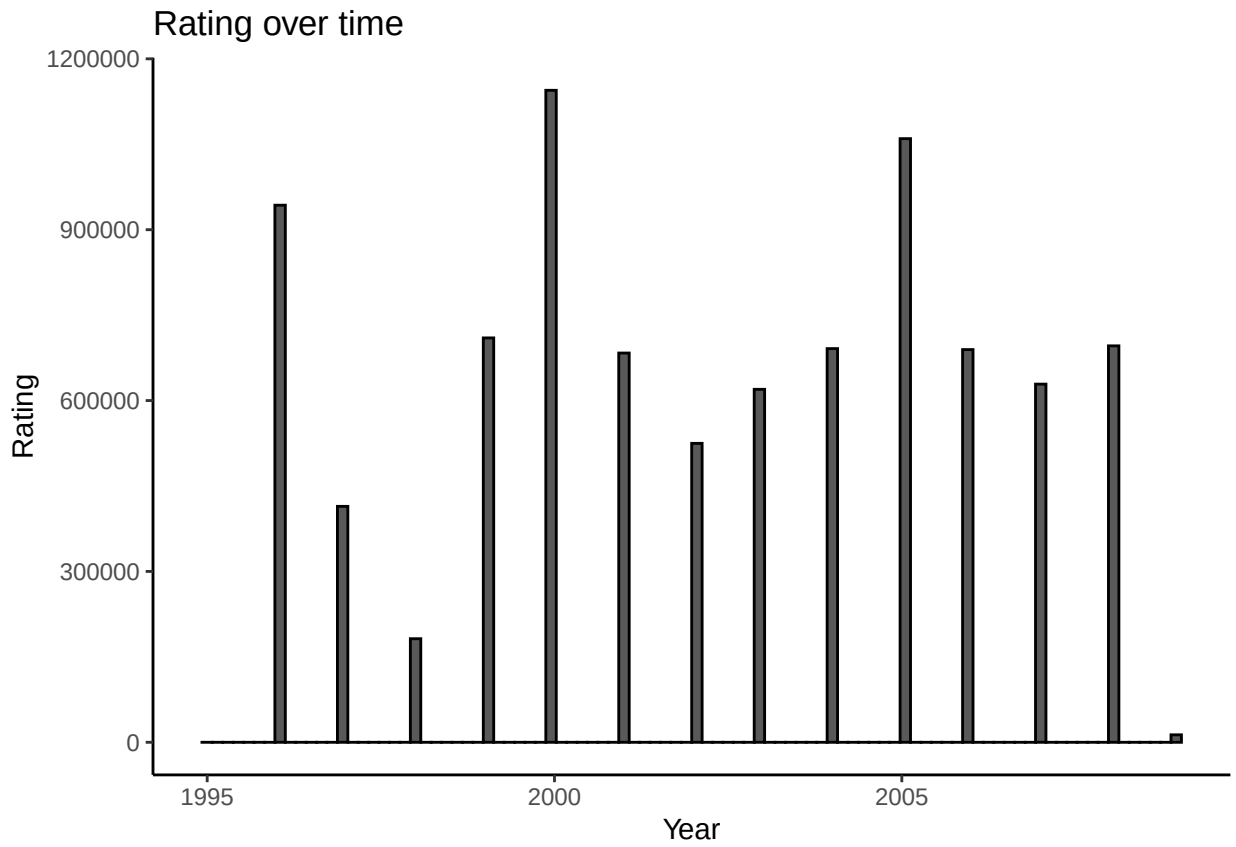
```
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 0.15, bins = 30, color = I('black')) +
  scale_x_log10() +
  theme_classic() +
  xlab("Number of ratings give by user") +
  ylab("Number of users") +
  ggtitle("Number of ratings per user") +
  labs(caption = "The graph is skewed to the right (positive).")
```



The graph is skewed to the right (positive).

We can see that some users have rated less than 30 movies, and this will underestimate our models. We also know that more recent movies are tend to be rated more frequently than older ones. Therefore we can convert the timestamp column of the edx data set into a date of the rating was given. Afterwards, we will be able to explore the relationship of the date and rating.

```
edx %>%
  mutate(year = year(as_datetime(timestamp))) %>%
  ggplot(aes(year)) +
  geom_histogram(binwidth = 0.15, bins = 30, color = I('black')) +
  ggtitle("Rating over time") +
  xlab("Year") +
  ylab("Rating") +
  theme_classic()
```



Looking at the graph can suggest some relationship between the time and the rating, however, there is no strong correlation.

Data Preprocessing

First of all we should say that the data set is tidy, however, it is better to check if there is any NAs in the data.

```
anyNA(edx)
```

Next step would be the removal of the variance that is close to zero.

```
nearZeroVar(edx$rating)
```

Methods and Analysis

Linear Model

The start model assumes the same prediction for all users and movies, explaining difference by random variation. Here μ represents the "true" rating for all movies. ϵ is an independent errors sampled from the same distribution, that is centered at zero. The first equation to use will be:

$$\hat{Y}_{u,i} = \mu + \epsilon_{u,i}$$

The $\hat{Y}$ is the predicted rating. Any additional value will increase the root mean squared error (RMSE).

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Therefore, we will add an additional variable b_i that represents the average ranking for the movie i, improving our RMSE model:

$$b_i = mean(\hat{Y}_{u,i} - \mu)$$

$$\hat{Y}_{u,i} = \mu + b_i + \epsilon_{u,i}$$

Knowing that some users ranked more movies than other users brings in place another bias, user-specific effect. We will call this variable b_u, and therefore, increase the RMSE:

$$\hat{Y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

From the data exploration we defined that there is some effect of the time on the rating, therefore we can imply time-specific effect to the RMSE model. However, the correlation was not significant and may cause in decrease of the RMSE, so we would not use time-specific effect in our analysis.

Regularisation

The linear model $\hat{Y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$ will provide a good estimation of ratings, however it will not penalize large estimates that come from small samples. For example, movies that were rated few times or users that gave ratings for very few movies. Not accounting this will lead to the large estimated errors. Therefore, the estimated value will be improved by applying the penalty term. The b's estimate now will be:

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda\sum_{i} b_i^2$$

The $\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2$ is the mean square error, and the $\lambda\sum_i b_i^2$ is a penalty term. Note that when b is getting large the penalty term increases. Now we can calculate the movie-specific and the user-specific effects using regularization:

$$\hat{b_i} = \frac{1}{\lambda + n_i}\sum_{u=1}^{n_i}(y_{u,i} - \hat{\mu})$$

$$\hat{b_u} = \frac{1}{\lambda + n_u}\sum_{i=1}^{n_u}(y_{u,i} - \hat{b_i} - \hat{\mu})$$

Here λ is a tuning parameter and we can use cross-validation to choose the minimum one that gives the most accurate RMSE.

Matrix Factorisation

Data can be converted into matrix to study the same rating patterns in the movies and users groups. Each user gets a row and each movie gets a column. The aim is to approximate the large matrix $R_{m\times n}$ into two smaller vectors $P_{k\times m}$ and $Q_{k\times m}$, such that : $R \approx P'Q$. p_u is the u-th row of P, and q_i is the v-th row of Q. Therefore, the the rating given by the user u for the movie i is $p_u q_i'$.This allows us to apply more variance in the original RMSE model:

$$\hat{Y}_{u,i} = \mu + b_i + b_u + p_u q_i + \epsilon_{i,j}$$

. We can use Singular value composition (SVD) that finds the vectors p and q that permit us to write the matrix of residuals r with m rows and n columns. By using principal components analysis (PCA), matrix factorization can capture structure in the data determined by user opinions about movies. For Matrix Factorisation we will user the Recosystem package.

Results

First of all we will define a dataframe that will store all our RMSE values as we go along with our analysis. And we will include the desired RMSE value in it.

Secondly, we should define the RMSE function to calculate it using different models. In the Machine Learning module we were using test sets and training sets. However, here we have an edx data table as the training set and a validation data table for predicted ratings.

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Now we will try to predict the RMSE value using Monte Carlo simulation. Since we are not using any linear models or regulisations, the RMSE result would be the highest among all other models.

```
## Monte Carlo prediction
set.seed(4321)
# Create the probability of each rating
p <- function(x, y) mean(y == x)
rating <- seq(0.5,5,0.5)
#Estimate the probability of each rating
B <- 10000
N <- 100
MS <- replicate(B, {
  X <- sample(edx$rating,N, replace = TRUE)
  sapply(rating, p, y = X)
})
probability <- sapply(1:nrow(MS), function(x) mean(MS[x,]))
#Random ratings prediction
y_hat <- sample(rating, size = nrow(edx),
                replace = TRUE, prob = probability)
#Store the prediction in the table
model_results <- bind_rows(model_results,
                           data.frame(method = "Monte Carlo simulation",
                                      RMSE = RMSE(edx$rating, y_hat)))
```

Naive RMSE

Now we start to apply a simple linear model, $\hat{Y}_{u,i} = \mu + \epsilon_{u,i}$.

```
#Find the "true" ratings for all movies, mu
mu_hat <- mean(edx$rating)
mu_hat
#Add this model to the table
model_results <- bind_rows(model_results,
                           data.frame(method = "Just the average",
                           RMSE = RMSE(edx$rating, mu_hat)))
```

Movie effect model

Now we will include movie effect (b_i). Firstly, we should define b_i using the equation:

$$b_i = mean(\hat{Y}_{u,i} - \mu)$$

```
b_i <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu_hat))
head(b_i)
```

```
## # A tibble: 6 x 2
##   movieId    b_i
##     <dbl>  <dbl>
## 1       1  0.414
## 2       2 -0.310
## 3       3 -0.362
## 4       4 -0.632
## 5       5 -0.434
## 6       6  0.299
```

Secondly, we will calculate the $\hat{Y}_{u,i}$ using the following equation:

$$\hat{Y}_{u,i} = \mu + b_i + \epsilon_{u,i}$$

```
predicted_ratings <- mu_hat + validation %>%
  left_join(b_i, by = 'movieId') %>%
  .$b_i
```

And lastly we will calculate the RMSE and add this model to the table for comparasion

```
movie_specific_RMSE <- RMSE(validation$rating, predicted_ratings)
movie_specific_RMSE
```

```
## [1] 0.9437046
```

```
#Add to the model table
model_results <- bind_rows(model_results,
                           data.frame(method = "Movie Effect Model",
                                      RMSE = movie_specific_RMSE))
```

User effect model

Here we will calculate the RMSE using the user effect, b_u. And then using the formula, $\hat{Y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$, we will compute the prediction ratings.

```
b_u <- edx %>%
  left_join(b_i, by = 'movieId') %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu_hat - b_i))
predicted_ratings <- validation %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  mutate(prediction = mu_hat + b_i + b_u) %>%
  .$prediction
user_specific_RMSE <- RMSE(validation$rating, predicted_ratings)
```

```
# Add the RMSE to the model table
model_results <- bind_rows(model_results,
                           data.frame(method = "Movie + User Effect Model",
                                      RMSE = user_specific_RMSE))
```

We can observe that with that the applying b_i and b_u to the model improved our RMSE. However, further steps are required to reach the goal.

```
knitr::kable(model_results)
```

method	RMSE
Goal RMSE	0.8649000
Monte Carlo simulation	1.5002198
Just the average	1.0603926
Movie Effect Model	0.9437046
Movie + User Effect Model	0.8655329

Regulization

Now we should regularize movie and user effect using tuning parameter λ. We will define lambdas as following:

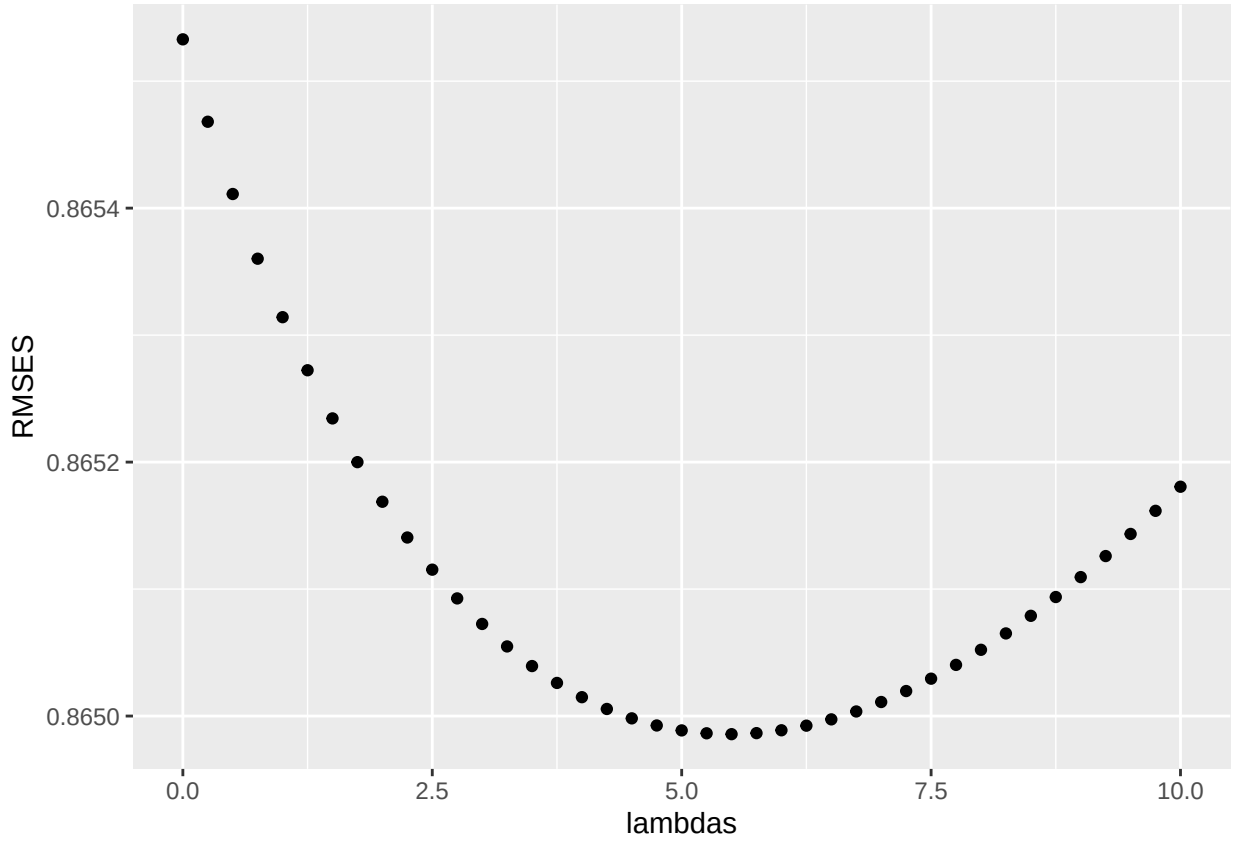
```
lambdas <- seq(0, 10, 0.25)
```

We can use cross-validation to estimate the best value of λ for the most improved RMSE. For this we should define the "RMSES" function. It will take few minutes to be completed.

```
RMSES <- sapply(lambdas, function(l){
  mu_hat <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu_hat)/(n() + l))
  b_u <- edx %>%
    left_join(b_i, by = 'movieId') %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - b_i - mu_hat)/(n() + l))
  predicted_user_movie_rating <- validation %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    mutate(prediction = mu_hat + b_i + b_u) %>%
    .$prediction
  return(RMSE(validation$rating, predicted_user_movie_rating))
})
```

Now we can plot lambdas over RMSES results

```
qplot(lambdas, RMSES)
```



Now we can find the value of lambda that referrers to the minimal RMSE

```
lambda <- lambdas[which.min(RMSES)]
lambda
```

```
## [1] 5.5
```

Therefore, the final model RMSE is:

```
## [1] 0.8649857
```

The last step is to add our final RMSE value to the model results table

```
model_results <- bind_rows(model_results,
                           data.frame(method = "Regularised Movie + User Effect Model",
                                      RMSE = regularised_user_movie_specific_rmse))
knitr::kable(model_results)
```

method	RMSE
Goal RMSE	0.8649000
Monte Carlo simulation	1.5002198
Just the average	1.0603926
Movie Effect Model	0.9437046
Movie + User Effect Model	0.8655329

method	RMSE
Regularised Movie + User Effect Model	0.8649857

We can see that the final RMSE satisfies the goal of the project and is < 0.8649000. However, it is not perfect until we do our Matrix Factorisation.

Matrix Factorisation

We will use Recosystem package to do the matrix factorisation. Firstly, we will make a train_set and test_set from edx and validation data, respectively. Also, we will clean an unused memory to make the process more quickly.

```
set.seed(123)
invisible(gc())
train_set <- with(edx, data_memory(user_index = userId,
                                   item_index = movieId,
                                   rating = rating))
test_set <- with(validation, data_memory(user_index = userId,
                                         item_index = movieId,
                                         rating = rating))
```

Next step is to build up a recommender system.

```
rec <- Reco()
```

Now we can tune the train set. Note that this will take time (approximately 20 minutes) to be completed.

```
opts <- rec$tune(train_set, opts = list(dim = c(10,20,30),
                                        lrate = c(0.1,0.2),
                                        costp_l1 = 0,
                                        costq_l1 = 0,
                                        nthread = 1,
                                        niter = 10))
opts
```

```
## $min
## $min$dim
## [1] 30
##
## $min$costp_l1
## [1] 0
##
## $min$costp_l2
## [1] 0.01
##
## $min$costq_l1
## [1] 0
##
## $min$costq_l2
## [1] 0.1
```

```
##
## $min$lrate
## [1] 0.1
##
## $min$loss_fun
## [1] 0.7969827
##
##
## $res
##    dim costp_l1 costp_l2 costq_l1 costq_l2 lrate  loss_fun
## 1   10        0     0.01        0     0.01   0.1 0.8245793
## 2   20        0     0.01        0     0.01   0.1 0.8074963
## 3   30        0     0.01        0     0.01   0.1 0.8137367
## 4   10        0     0.10        0     0.01   0.1 0.8257848
## 5   20        0     0.10        0     0.01   0.1 0.8016588
## 6   30        0     0.10        0     0.01   0.1 0.8006405
## 7   10        0     0.01        0     0.10   0.1 0.8292131
## 8   20        0     0.01        0     0.10   0.1 0.8036105
## 9   30        0     0.01        0     0.10   0.1 0.7969827
## 10  10        0     0.10        0     0.10   0.1 0.8378680
## 11  20        0     0.10        0     0.10   0.1 0.8292643
## 12  30        0     0.10        0     0.10   0.1 0.8274482
## 13  10        0     0.01        0     0.01   0.2 0.8146384
## 14  20        0     0.01        0     0.01   0.2 0.8674960
## 15  30        0     0.01        0     0.01   0.2 0.9701898
## 16  10        0     0.10        0     0.01   0.2 0.8166378
## 17  20        0     0.10        0     0.01   0.2 0.8588463
## 18  30        0     0.10        0     0.01   0.2 0.8596419
## 19  10        0     0.01        0     0.10   0.2 0.8212675
## 20  20        0     0.01        0     0.10   0.2 0.7999755
## 21  30        0     0.01        0     0.10   0.2 0.7989653
## 22  10        0     0.10        0     0.10   0.2 0.8406393
## 23  20        0     0.10        0     0.10   0.2 0.8265322
## 24  30        0     0.10        0     0.10   0.2 0.8268478
```

We are then training the system.

```
rec$train(train_set, opts = c(opts$min, nthread = 1, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9720   1.2026e+07
##    1       0.8714   9.8676e+06
##    2       0.8382   9.1650e+06
##    3       0.8165   8.7475e+06
##    4       0.8014   8.4743e+06
##    5       0.7899   8.2741e+06
##    6       0.7802   8.1272e+06
##    7       0.7721   8.0064e+06
##    8       0.7653   7.9120e+06
##    9       0.7594   7.8311e+06
##   10       0.7542   7.7629e+06
##   11       0.7496   7.7046e+06
##   12       0.7453   7.6533e+06
```

```
##    13       0.7415   7.6095e+06
##    14       0.7380   7.5718e+06
##    15       0.7348   7.5360e+06
##    16       0.7318   7.5044e+06
##    17       0.7290   7.4763e+06
##    18       0.7265   7.4491e+06
##    19       0.7241   7.4265e+06
```

And finally we can calculate our prediction, $\hat{Y}$, and calculate the final RMSE.

```
#Calculate the prediction
y_hat <- rec$predict(test_set, out_memory())
#Add the result into the model nresults table
model_results <- bind_rows(model_results,
                           data.frame(method = "Matrix Factorisation",
                                      RMSE = RMSE(validation$rating, y_hat)))
knitr::kable(model_results)
```

method	RMSE
Goal RMSE	0.8649000
Monte Carlo simulation	1.5002198
Just the average	1.0603926
Movie Effect Model	0.9437046
Movie + User Effect Model	0.8655329
Regularised Movie + User Effect Model	0.8649857
Matrix Factorisation	0.7832657

Now we can look at 10 best movies and 10 movies using matrix factorisation.

```
tibble(title = validation$title, rating = y_hat) %>%
  arrange(desc(rating)) %>%
  group_by(title) %>%
  head(10) %>%
  knitr::kable()
```

title	rating
Cats Don't Dance (1997)	6.174707
Shawshank Redemption, The (1994)	5.842876
Even Dwarfs Started Small (Auch Zwerge haben klein angefangen) (1971)	5.819474
Star Wars: Episode V - The Empire Strikes Back (1980)	5.819019
Year of the Horse (1997)	5.809022
Stalker (1979)	5.783485
Die Hard: With a Vengeance (1995)	5.780095
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	5.735769
Becket (1964)	5.734247
Shawshank Redemption, The (1994)	5.729154

If we return back at the start of the report we can notice that the top movie was "Pulp Fiction", however, using a matrix factorisation we defined that the top movie is "Schindler's List". Let's look which movies are considered as the worst movies.

```
tibble(title = validation$title, rating = y_hat) %>%
  arrange(rating) %>%
  group_by(title) %>%
  head(10) %>%
  knitr::kable()
```

title	rating
Beast of Yucca Flats, The (1961)	-1.2762158
Turbo: A Power Rangers Movie (1997)	-0.8602614
Pokémon the Movie 2000 (2000)	-0.8545352
Pearl Harbor (2001)	-0.8278118
Murder on a Sunday Morning (Un coupable idéal) (2001)	-0.7883087
Eternity and a Day (Mia aoniotita kai mia mera) (1998)	-0.6853818
Switchblade Sisters (1975)	-0.6194889
Santa Clause 3: The Escape Clause, The (2006)	-0.5734605
Disaster Movie (2008)	-0.5481386
It Takes Two (1995)	-0.5235486

Conclusion

Using the training set and validation set we have successfully trained several linear regression models, which we studied in the previous courses of the HarvardX Data Science programme. We identified that the linear regression model using regularised user and movie effect model and matrix factorisation gave the desired result of the RMSE < 0.8649000. The matrix factorisation produced the RMSE $= 0.7823751$, and we achieved this using the recosystem package and model from the (website).

Limitations

Due to the big data the matrix factorisation and some machine learning algorithms are running long on the ordinary laptop and require the usage of decent amount of memory. This can cause issues for some people to test and try the code. Moreover, we have used only two parameters, however, in real-life recommendation systems use more parameters, which I have listed in the introduction section. Also, the algorithm uses the dataset for the existing users and films, so the addition of new users and movies will result in the alteration of the RMSE. For the model improvement in the future we should consider to implement and overcome this issue.

References

1. Irizarry, R. (2021). Introduction to Data Science. Retrieved 25 October 2021, from https://rafalab.github.io/dsbook/
2. Netflix Prize - Wikipedia. (2021). Retrieved 25 October 2021, from https://en.wikipedia.org/wiki/Netflix_Prize
3. Qiu, Y. (2021). recosystem: Recommender System Using Parallel Matrix Factorization. Retrieved 25 October 2021, from https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html
4. Seif, G. (2021). An Easy Introduction to Machine Learning Recommender Systems - KDnuggets. Retrieved 25 October 2021, from https://www.kdnuggets.com/2019/09/machine-learning-recommender-systems.html