# MovieLens Project

Ignacio Gurrola

5/30/2021

## 1. Introduction

Recommendation Systems are software agents that extract the interests and preferences of individual consumers and make recommendations accordingly.They enables us to offer products or services to new users. Certainly, this is essential for many online businesses such as Netflix, YouTube and Amazon. This presentation will review what a recommendation system is, which type exist and how to code them from scratch in R.

The document has a structure as follows: Chapter 1 describes the dataset and summarizes the goal of the project and key steps. In chapter 2, It is reviewed the process and techniques used, such as data cleaning, data exploration and visualization, any insights gained, and the modeling approach. Chapter 3 present the modeling results and discuss the model performance. Chapter 4 will conclude with a brief summary of the report, its limitations and future work. At the end of the document you will find the references and bibliography for this work. This project wouldn't be possible without the guidance of Dr. Irizarry during all Data Science sessions.

In general, code and output are not shown in this PDF presentation.

### 1.1 Types of recommendation Systems

There are two main types of recommendation systems:

**Content-based recommendation system**- System based on using the features of the books in order to offer similar products. For example, on Amazon Kindle section the system will recommend a regular follower of Dan Brown, other digital books and lectures of the same author or bibliography within the same category.

**Collaborative Recommendation System**- This case does not use the features of the products, but rather the opinions of the users. There are two main types of collaborative recommendation systems:

- User-based collaborative system- Based on finding similar users and find the items those users have liked but we have not tried yet.
- Item-based collaborative system- in this case, we will find similar products to the one the user has bought and we will recommend those products that are similar to those which has ratted as best.

In addition, two systems can be combined creating hybrid models, as in the case of ensemble models in Machine Learning.

## 2. Methods and Analysis

## 2.1 Data Preparation

For this project, a movie recommendation system will be created using the MovieLens dataset. The entire latest MovieLens dataset is here. It is used the 10M version of the MovieLens dataset to make the computation a little easier.

This section is to download and prepare the dataset used in the analysis. Dataset is split in two parts; the training set called edx and the evaluation set called validation with 90% and 10% of the original dataset respectively.

The edx set is set in two parts, the train set and test set with 90% and 10% of edx set respectively. The model is built and trained in the train set and tested in the test set until the RMSE target is achieved, then finally train the model again in the entire edx set and validate in the validation set. The name of this method is cross-validation.

I will use echo=FALSE to hide the code in the output. The code is evaluated when the Rmd file is knit, however only the output is rendered on the output document.

```
Loading required package: tidyverse

Warning: package 'tidyverse' was built under R version 4.0.5

-- Attaching packages --------------------------------------- tidyverse 1.3.1 --

v ggplot2 3.3.3     v purrr   0.3.4
v tibble  3.1.2     v dplyr   1.0.6
v tidyr   1.1.3     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1

Warning: package 'tidyr' was built under R version 4.0.5

Warning: package 'dplyr' was built under R version 4.0.5

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()

Loading required package: caret

Warning: package 'caret' was built under R version 4.0.5

Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift

Loading required package: data.table

Attaching package: 'data.table'
```

```
The following objects are masked from 'package:dplyr':

    between, first, last

The following object is masked from 'package:purrr':

    transpose

Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
used

Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

The edx set is used for training and testing, and the validation set is used for final validation to simulate the new data.

Here, the edx set is split in 2 parts: the training set and the test set.

The model building is done in the training set, and the test set is used to test the model. When the model is complete, the validation set is used to calculate the final RMSE.

The same procedure is used to create edx and validation sets.

The training set will be 90% of edx data and the test set will be the remaining 10%.

```
Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
used

Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

From this initial exploration, we discover that edx has 6 columns:

userId integer movieId integer rating numeric timestamp numeric title character genres character

How many rows and columns are there in the edx dataset?

```
[1] 9000055       6
```

The next table shows the structure and content of edx dataset

The dataset is in tidy format, i.e. each row has one observation and the column names are the features. The rating column is the desired outcome. The user information is stored in userId; the movie information is both in movieId and title columns. The rating date is available in timestamp measured in seconds since January 1st, 1970. Each movie is tagged with one or more genre in the genres column.

```
   userId movieId rating timestamp                          title
1:      1     122      5 838985046               Boomerang (1992)
2:      1     185      5 838983525                Net, The (1995)
3:      1     292      5 838983421                Outbreak (1995)
4:      1     316      5 838983392                Stargate (1994)
5:      1     329      5 838983392 Star Trek: Generations (1994)
6:      1     355      5 838984474       Flintstones, The (1994)
                        genres
1:                Comedy|Romance
2:          Action|Crime|Thriller
3:  Action|Drama|Sci-Fi|Thriller
4:        Action|Adventure|Sci-Fi
5: Action|Adventure|Drama|Sci-Fi
6:       Children|Comedy|Fantasy
```

The next sections discover more details about each feature and outcome.

### 2.2.1 Genres

Along with the movie title, MovieLens provides the list of genres for each movie. Although this information can be used to make better predictions, this research doesn't use it. However it's worth exploring this information as well. The data set contains 797 different combinations of genres. Here is the list of the first six.

```
# A tibble: 6 x 2
  genres                                                n
  <chr>                                             <int>
1 (no genres listed)                                    7
2 Action                                            24482
3 Action|Adventure                                  68688
4 Action|Adventure|Animation|Children|Comedy         7467
5 Action|Adventure|Animation|Children|Comedy|Fantasy  187
6 Action|Adventure|Animation|Children|Comedy|IMAX      66
```

The table above shows that several movies are classified in more than one genre. The number of genres in each movie is listed in this table, sorted in descend order.

```
`summarise()` has grouped output by 'count'. You can override using the `.groups` argument.
```

```
# A tibble: 6 x 3
# Groups:   count [3]
  count genres                                                        n
  <int> <chr>                                                     <int>
1     7 Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller   256
2     6 Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery    10975
3     6 Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery      355
4     6 Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance    515
5     5 Action|Adventure|Animation|Children|Comedy|Fantasy             187
6     5 Action|Adventure|Animation|Children|Comedy|IMAX                 66
```

### 2.2.2 Date

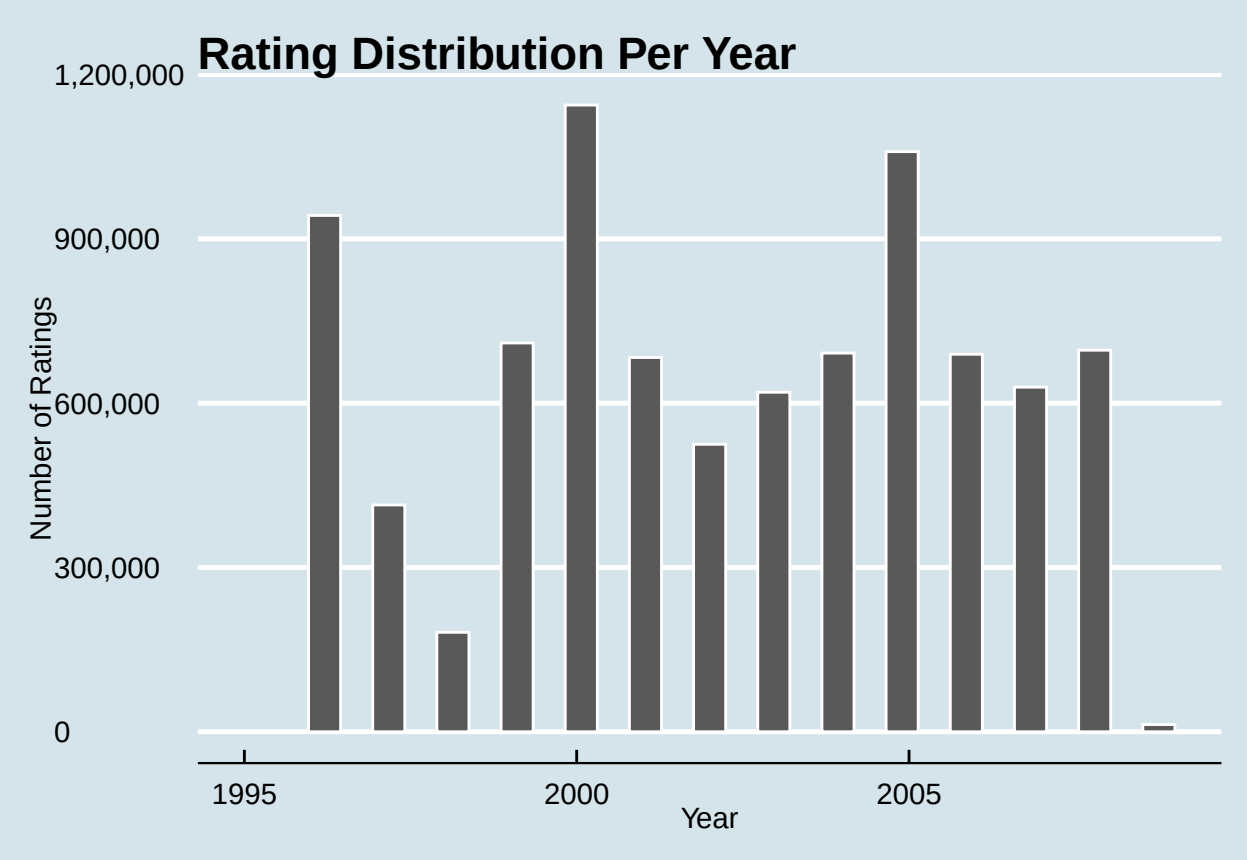
The rating period was collected over almost 14 years.

```
Attaching package: 'lubridate'
```

```
The following objects are masked from 'package:data.table':

    hour, isoweek, mday, minute, month, quarter, second, wday, week,
    yday, year
```

```
The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
# A tibble: 1 x 3
  `Initial Date` `Final Date` Period
  <date>         <date>       <Duration>
1 1995-01-09     2009-01-05   441479727s (~13.99 years)

Loading required package: ggthemes

Warning: package 'ggthemes' was built under R version 4.0.5

Loading required package: scales

Attaching package: 'scales'

The following object is masked from 'package:purrr':

    discard

The following object is masked from 'package:readr':

    col_factor

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




The following table lists the days with more ratings. Not surprisingly, the movies are blockbusters.

```
`summarise()` has grouped output by 'date'. You can override using the `.groups` argument.
```

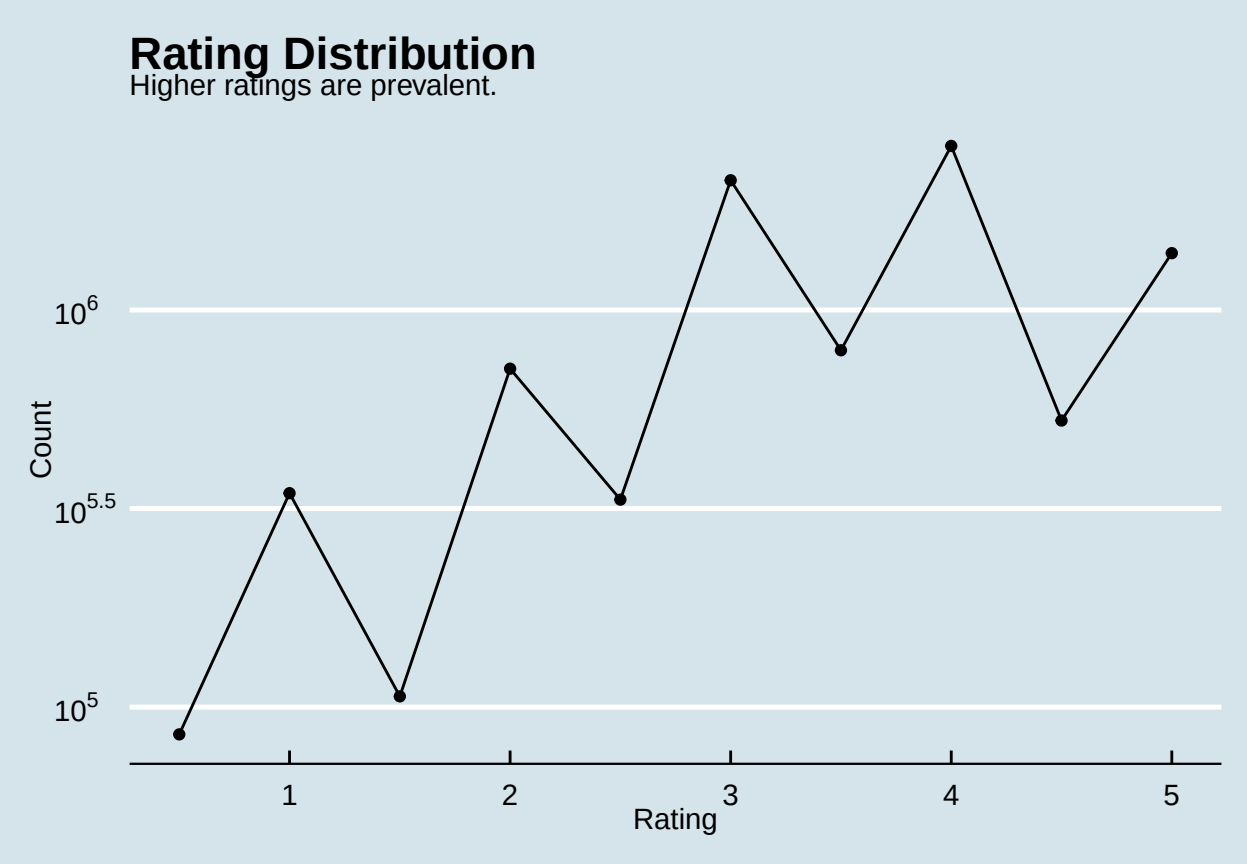
```
# A tibble: 10 x 3
# Groups:   date [4]
   date       title                                                     count
   <date>     <chr>                                                     <int>
 1 1998-05-22 Chasing Amy (1997)                                          322
 2 2000-11-20 American Beauty (1999)                                      277
 3 1999-12-11 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   254
 4 1999-12-11 Star Wars: Episode V - The Empire Strikes Back (1980)       251
 5 1999-12-11 Star Wars: Episode VI - Return of the Jedi (1983)           241
 6 2005-03-22 Lord of the Rings: The Two Towers, The (2002)               239
 7 2005-03-22 Lord of the Rings: The Fellowship of the Ring, The (2001)   227
 8 2000-11-20 Terminator 2: Judgment Day (1991)                           221
 9 1999-12-11 Matrix, The (1999)                                          210
10 2000-11-20 Jurassic Park (1993)                                        201
```

### 2.2.3 Ratings

Users have the option to choose a rating value from 0.5 to 5.0, totaling 10 possible values. This is unusual scale, so most movies get a rounded value rating, as shown in the chart below.

```
# A tibble: 10 x 2
   rating       n
    <dbl>   <int>
 1    0.5   85374
 2    1    345679
 3    1.5  106426
 4    2    711422
 5    2.5  333010
 6    3   2121240
 7    3.5  791624
 8    4   2588430
 9    4.5  526736
10    5   1390114
```

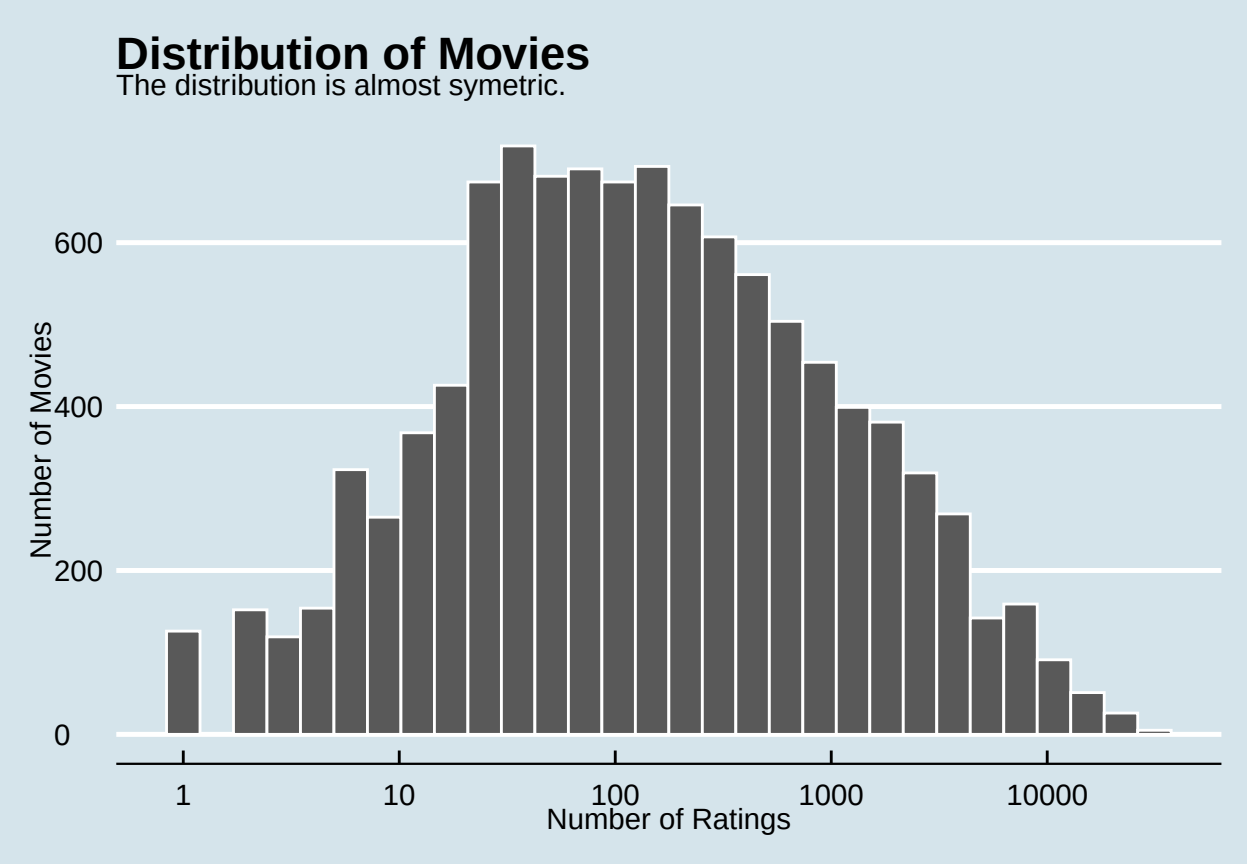
How many ratings are in edx?




### 2.2.4 Movies

There are 10,681 different movies in the edx set. Some of them are rated more than others, since many movies are watched by few users and blockbusters tend to have more ratings.

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



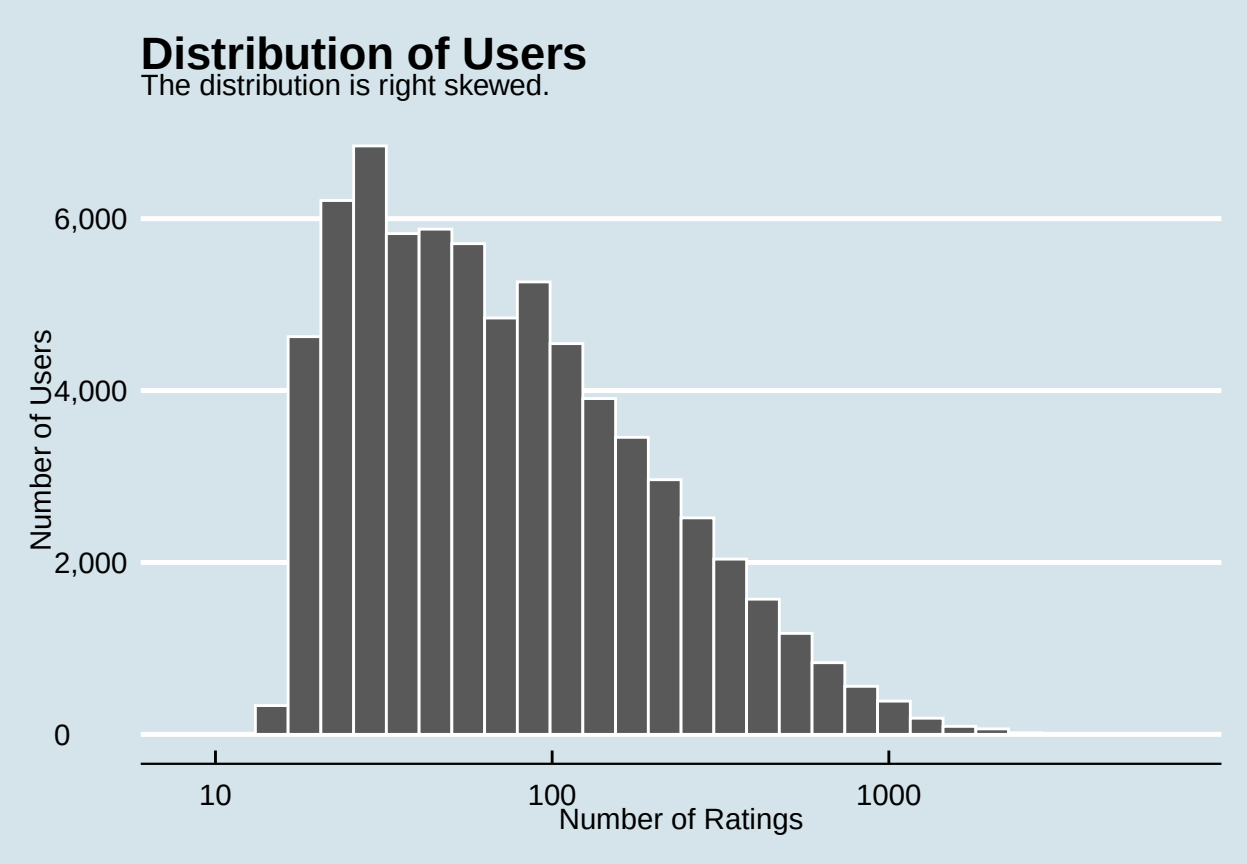

### 2.2.5 Users

There are 71,567 users of the online movie recommender service MovieLens. The majority of users rate few movies, while a few users rate more than a thousand movies. 5% users rated less than 20 movies.

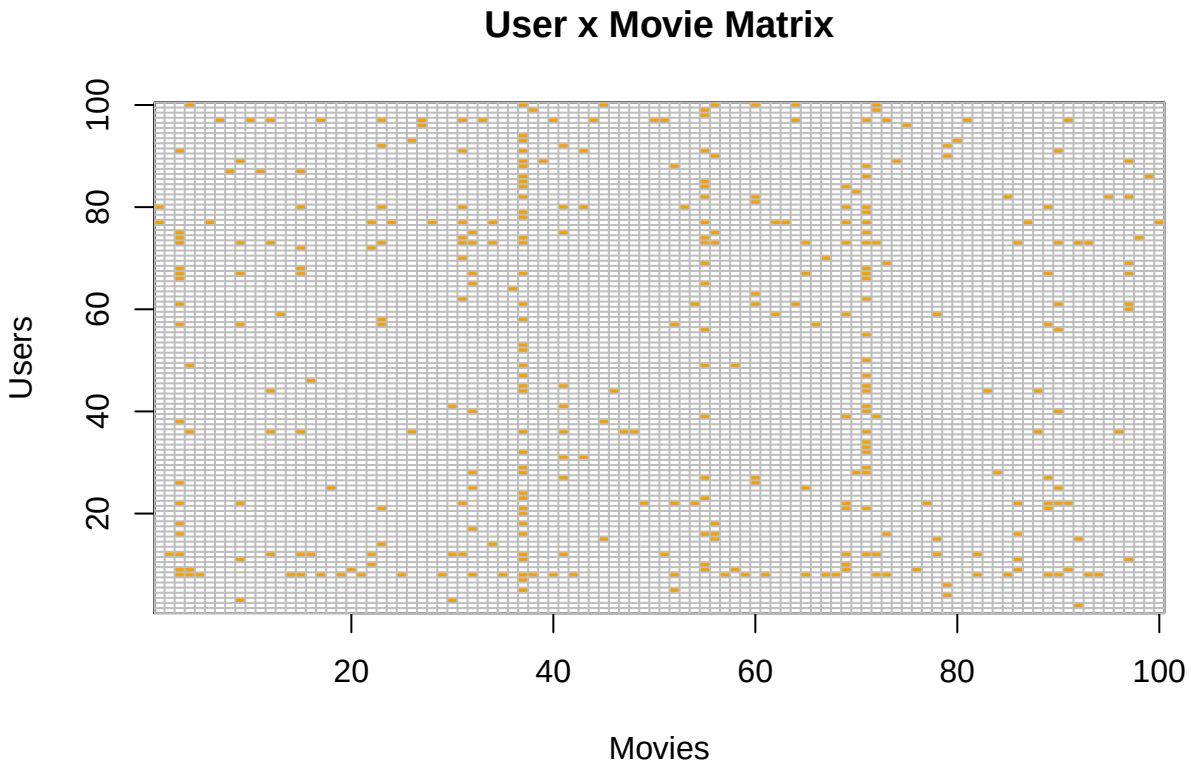
```
# A tibble: 6 x 2
  userId     n
   <int> <int>
1  62516    10
2  22170    12
3  15719    13
4  50608    13
5    901    14
6   1833    14
```

The user distribution is right skewed.

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




Show the heatmap of users x movies This user-movie matrix is sparse, with the vast majority of empty cells. Notice that four movies have more ratings, and one or two users are more active.




### 2.3 Data Cleaning

As previously discussed, several features can be used to predict the rating for a given user. However, many predictors increases the model complexity and requires more computer resources, so in this research the estimated rating uses only movie and user information.

### 2.4 Modeling

#### 2.4.1 Random Prediction

A very simple model is just randomly predict the rating using the probability distribution observed during the data exploration. For example, if we know the probability of all users giving a movie a rating of 3 is 10%, then we may guess that 10% of the ratings will have a rating of 3.

Such prediction sets the worst error we may get, so any other model should provide better result.

#### 2.4.2 Linear Model

The simplest model predicts all users will give the same rating to all movies and assumes the movie to movie variation is the randomly distributed error. Although the predicted rating can be any value, statistics theory says that the average minimizes the RMSE, so the initial prediction is just the average of all observed ratings, as described in this formula:

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

Where $\hat{Y}$ is the predicted rating, $\mu$ is the mean of observed data and $\epsilon_{i,u}$ is the error distribution. Any value other than the mean increases the RMSE, so this is a good initial estimation.

Part of the movie to movie variability can be explained by the fact that different movies have different rating distribution. This is easy to understand, since some movies are more popular than others and the public preference varies. This is called movie effect or movie bias, and is expressed as $b_i$ in this formula:

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

The movie effect can be calculated as the mean of the difference between the observed rating $y$ and the mean $\mu$.

$\hat{b}_i = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{\mu})$

Similar to the movie effect, different users have different rating pattern or distribution. For example, some users like most movies and consistently rate 4 or 5, while other users dislike most movies rating 1 or 2. This is called user effect or user bias and is expressed in this formula:

$\hat{b}_u = \frac{1}{N} \sum_{i=1}^{N} (y_{u,i} - \hat{b}_i - \hat{\mu})$

The prediction model that includes the user effect becomes:

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

Movies can be grouped into categories or genres, with different distributions. In general, movies in the same genre get similar ratings. In this project we won't evaluate the genre effect.

### 2.4.3 Regularization

The linear model provides a good estimation for the ratings, but doesn't consider that many movies have very few number of ratings, and some users rate very few movies. This means that the sample size is very small for these movies and these users. Statistically, this leads to large estimated error.

The estimated value can be improved adding a factor that penalizes small sample sizes and have have little or no impact otherwise. Thus, estimated movie and user effects can be calculated with these formulas:

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

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

For values of $N$ smaller than or similar to $\lambda$, $\hat{b}_i$ and $\hat{b}_u$ is smaller than the original values, whereas for values of $N$ much larger than $\lambda$, $\hat{b}_i$ and $\hat{b}_u$ change very little.

An effective method to choose $\lambda$ that minimizes the RMSE is running simulations with several values of $\lambda$.

### 2.4.4 Matrix Factorization

Matrix factorization is widely used machine learning tool for predicting ratings in recommendation systems. This method became widely known during the Netflix Prize challenge.

The data can be converted into a matrix such that each user is in a row, each movie is in a column and the rating is in the cell, then the algorithm attempts to fill in the missing values. The table below provides a simple example of a 4×54×5 matrix.

The concept is to approximate a large rating matrix $R_{mxn}$ into the product of two lower dimension matrices $P_{kxm}$ and $Q_{kxn}$, such that

$R \approx PQ$

The $R$ recosystem11 package provides methods to decompose the rating matrix and estimate the user rating, using parallel matrix factorization.

## 3. Results

This section presents the code and results of the models.

### 3.1 Model Evaluation Functions

Definition of the loss functions.

### 3.2 Random Prediction

The first model randomly predicts the ratings using the observed probabilities in the training set. First, the probability is calculated of each rating in the training set, then it is predicted the rating for the test set and compare with actual rating. Any model should be better than this one.

Since the training set is a sample of the entire population and the real distribution of ratings is not known, the Monte Carlo simulation with replacement provides a good approximation of the rating distribution.

A Monte Carlo simulation is a model used to predict the probability of different outcomes when the intervention of random variables is present. Monte Carlo simulations help to explain the impact of risk and uncertainty in prediction and forecasting models.

```
Warning in set.seed(4321, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler used
```

The RMSE of random prediction is very high.

Root Mean Square Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data.

Heuristically RMSE can be thought of as some kind of (normalized) distance between the vector of predicted values and the vector of observed values.

In data science, RMSE has a double purpose: - To serve as a heuristic for training models - To evaluate trained models for usefulness / accuracy

```
# A tibble: 2 x 4
  Method             RMSE   MSE   MAE
  <chr>             <dbl> <dbl> <dbl>
1 Project Goal      0.865 NA    NA
2 Random prediction 1.50   2.25  1.17
```

### 3.3 Linear Model

Linear regression is used to predict the value of a continuous variable Y based on one or more input predictor variables X. The aim is to establish a mathematical formula between the the response variable (Y) and the predictor variables (Xs).

Linear model is built based on the formula:

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

### 3.3.1 Initial Prediction

The initial prediction is just the mean of the ratings, $\mu$

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

```
# A tibble: 3 x 4
  Method             RMSE   MSE    MAE
  <chr>             <dbl> <dbl>  <dbl>
1 Project Goal      0.865 NA    NA
2 Random prediction 1.50   2.25  1.17
3 Mean              1.06   1.12  0.855
```

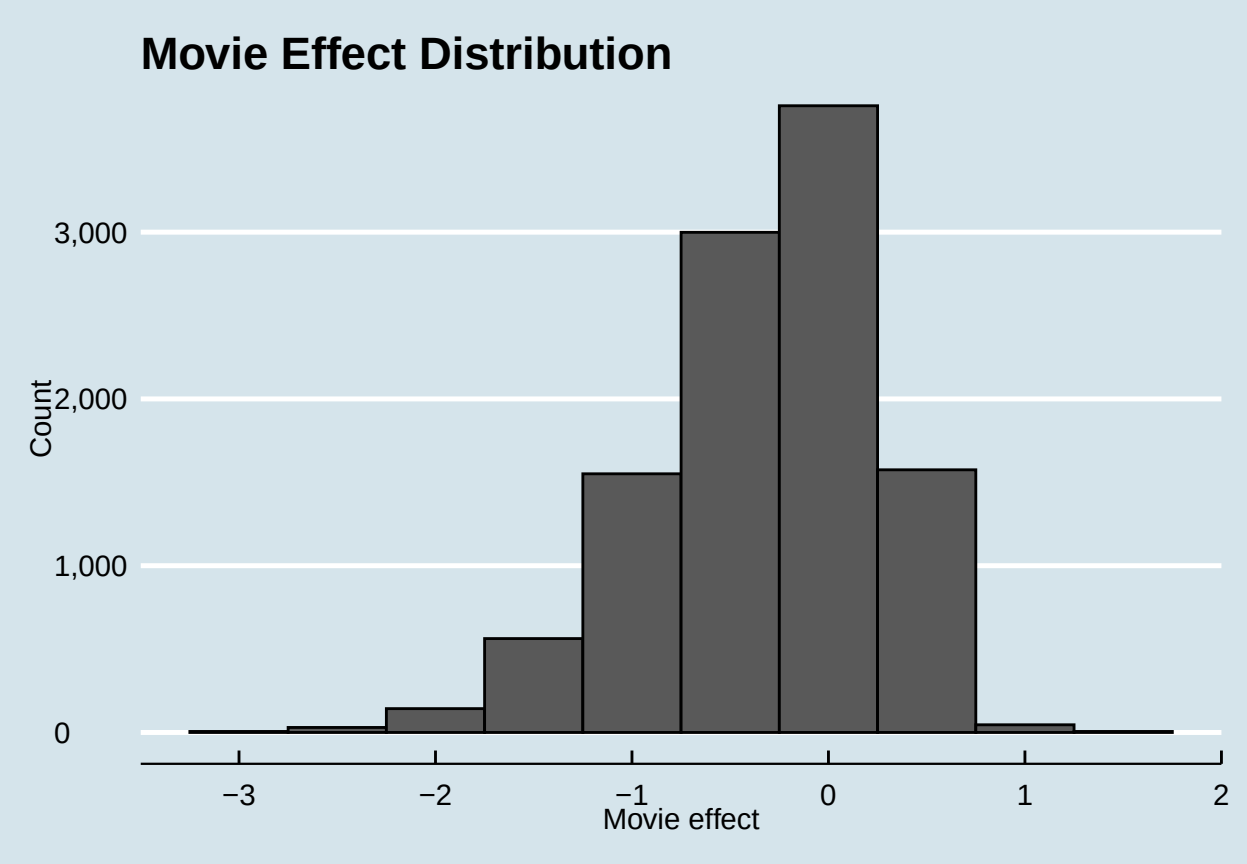
### 3.3.2 Include Movie Effect (bi)

$b_i$ is the movie effect (bias) for movie $i$.

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

```
# A tibble: 6 x 2
  movieId    b_i
    <dbl>  <dbl>
1       1  0.415
2       2 -0.306
3       3 -0.361
4       4 -0.637
5       5 -0.442
6       6  0.302
```

The movie effect is normally left skewed distributed.




```
# A tibble: 4 x 4
  Method             RMSE    MSE    MAE
  <chr>             <dbl>  <dbl>  <dbl>
1 Project Goal      0.865 NA     NA
2 Random prediction 1.50   2.25   1.17
3 Mean              1.06   1.12   0.855
4 Mean + bi         0.943  0.889  0.737
```

### 3.3.3 Include User Effect (bu)

$b_u$ is the user effect (bias) for user $u$.

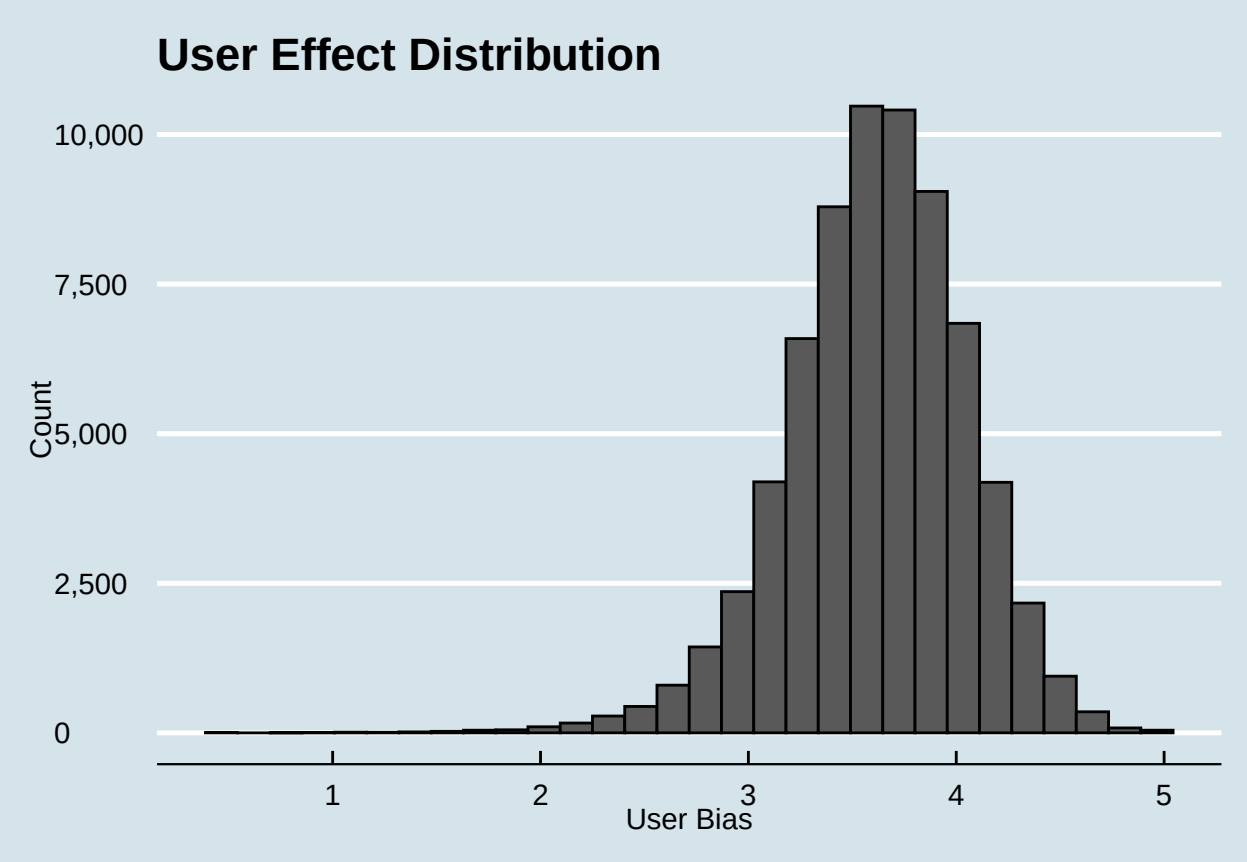
$\hat{y}_{u,i} = \mu + b_i + +b_u + \epsilon_{u,i}$

Predict the rating with $\mu + b_i + +b_u$

```
# A tibble: 5 x 4
  Method             RMSE    MSE    MAE
  <chr>             <dbl>  <dbl>  <dbl>
1 Project Goal      0.865 NA     NA
2 Random prediction 1.50   2.25   1.17
3 Mean              1.06   1.12   0.855
4 Mean + bi         0.943  0.889  0.737
5 Mean + bi + bu    0.865  0.748  0.668
```

The user effect is normally distributed.

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




### 3.3.4 Evaluating the model result

The RMSE improved from the initial estimation based on the mean. However, it is necessary to check if the model makes good ratings predictions.

Check the 10 largest residual differences

```
    userId movieId rating                              title       b_i  residual
 1:  26423    6483    5.0       From Justin to Kelly (2003) -2.638139  4.125683
 2:   5279    6371    5.0            PokÃ©mon Heroes (2003) -2.472133  3.959677
 3:  57863    6371    5.0            PokÃ©mon Heroes (2003) -2.472133  3.959677
 4:   2507     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
 5:   7708     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
 6:   9214     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
 7:   9568     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
 8:   9975     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
 9:  10749     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
10:  13496     318    0.5 Shawshank Redemption, The (1994)  0.944111 -3.956567
```

```
# A tibble: 6 x 1
  title
  <chr>
1 Hellhounds on My Trail (1999)
```

```
2 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)
3 Shadows of Forgotten Ancestors (1964)
4 Fighting Elegy (Kenka erejii) (1966)
5 Sun Alley (Sonnenallee) (1999)
6 Blue Light, The (Das Blaue Licht) (1932)
```

```
# A tibble: 6 x 1
  title
  <chr>
1 Besotted (2001)
2 Hi-Line, The (1999)
3 Accused (Anklaget) (2005)
4 Confessions of a Superhero (2007)
5 War of the Worlds 2: The Next Wave (2008)
6 SuperBabies: Baby Geniuses 2 (2004)
```

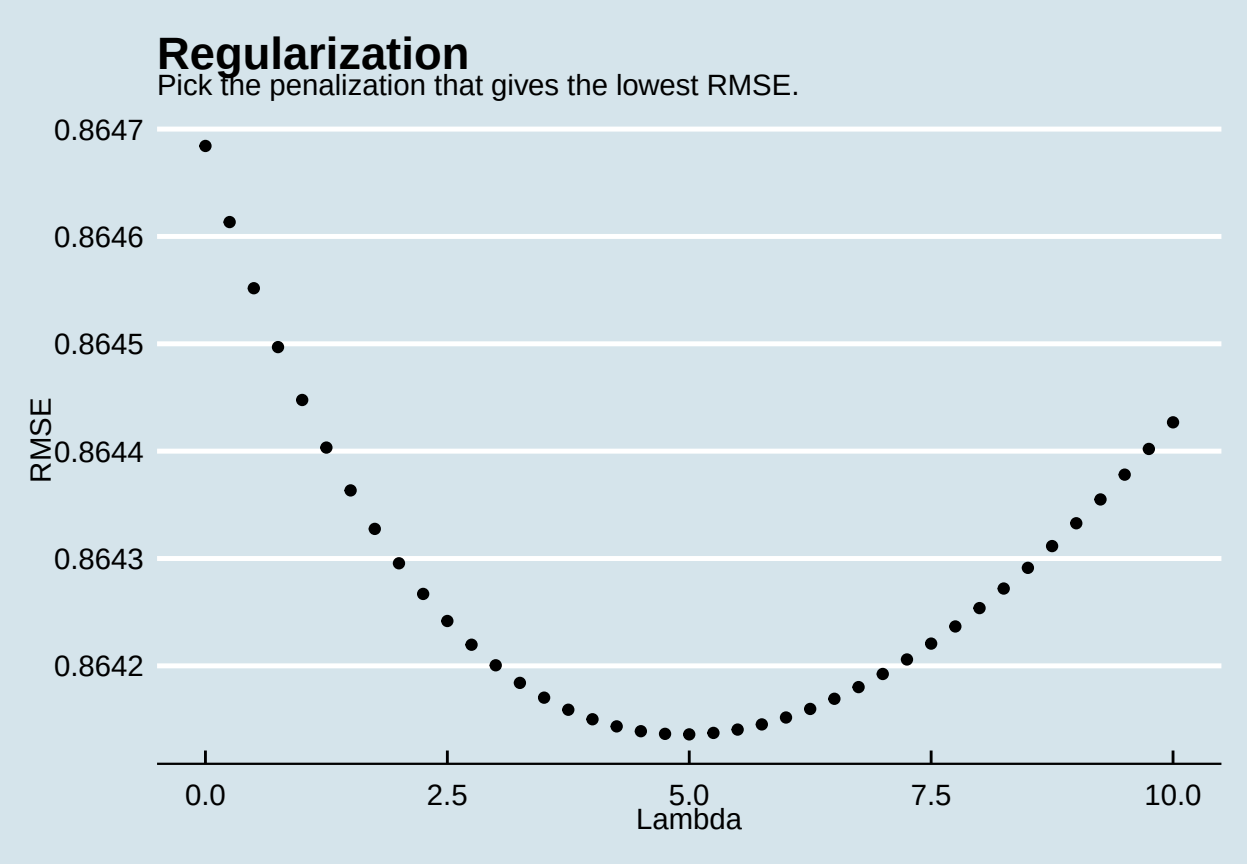
```
# A tibble: 10 x 2
   title                                                        n
   <chr>                                                    <int>
 1 "'burbs, The (1989)"                                      1201
 2 "'night Mother (1986)"                                     178
 3 "'Round Midnight (1986)"                                    40
 4 "'Til There Was You (1997)"                                242
 5 "\"Great Performances\" Cats (1998)"                         4
 6 "*batteries not included (1987)"                           389
 7 "...All the Marbles (a.k.a. The California Dolls) (1981)"   17
 8 "...And God Created Woman (Et Dieu... crÃ©a la femme) (1956)"  68
 9 "...And God Spoke (1993)"                                   19
10 "...And Justice for All (1979)"                            500
```

```
 [1] 1 1 1 1 1 1 4 2 4 4
```

## 3.4 Regularization

Now, regularize the user and movie effects adding a penalty factor $\lambda$, which is a tuning parameter. defined a number of values for and use the regularization function to pick the best value that minimizes the RMSE.

Regularization permits us to penalize large estimates that are formed using small sample sizes. It has commonalities with the Bayesian approach that shrunk predictions.




Next, I apply the best to the linear model.

```
# A tibble: 6 x 4
  Method                 RMSE    MSE    MAE
  <chr>                 <dbl>  <dbl>  <dbl>
1 Project Goal          0.865 NA     NA
2 Random prediction     1.50   2.25   1.17
3 Mean                  1.06   1.12   0.855
4 Mean + bi             0.943  0.889  0.737
5 Mean + bi + bu        0.865  0.748  0.668
6 Regularized bi and bu 0.864  0.747  0.669
```

### 3.5 Matrix Factorization

Matrix factorization approximates a large user-movie matrix into the product of two smaller dimension matrices. Information in the train set is stored in tidy format, with one observation per row, so it needs to be converted to the user-movie matrix before using matrix factorization. This code executes this transformation.

```
# Matrix factorization
train_data <- train_set %>%
  select(userId, movieId, rating) %>%
  spread(movieId, rating) %>%
  as.matrix()
```

The code above uses more memory than a commodity laptop is able to process, so I use an alternative

method: the recosystem package, which provides the complete solution for a recommendation system using matrix factorization.

When using *eval = FALSE*, I do not evaluate (or run) this code chunk when knitting the RMD document. The code in this chunk will still render in my knitted output, however it will not be evaluated or run by R . The package vignette describes how to use recosystem:

Usage of recosystem

The usage of recosystem is quite simple, mainly consisting of the following steps:

1. Create a model object (a Reference Class object in R) by calling Reco().
2. (Optionally) call the $tune() method to select best tuning parameters along a set of candidate values.
3. Train the model by calling the train() method. A number of parameters can be set inside the function, possibly coming from the result of tune().
4. (Optionally) export the model via output(), i.e. write the factorization matrices $P$ and $Q$ into files or return them as $R$ objects.
5. Use the $predict() method to compute predicted values.

```
Loading required package: recosystem
```

```
Warning: package 'recosystem' was built under R version 4.0.5
```

```
Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler used
```

```
iter      tr_rmse          obj
   0       0.9825   1.1046e+07
   1       0.8747   8.9725e+06
   2       0.8432   8.3387e+06
   3       0.8226   7.9718e+06
   4       0.8063   7.7053e+06
   5       0.7938   7.5177e+06
   6       0.7835   7.3700e+06
   7       0.7746   7.2511e+06
   8       0.7671   7.1567e+06
   9       0.7606   7.0761e+06
  10       0.7548   7.0080e+06
  11       0.7495   6.9494e+06
  12       0.7450   6.8997e+06
  13       0.7409   6.8552e+06
  14       0.7369   6.8137e+06
  15       0.7335   6.7802e+06
  16       0.7303   6.7481e+06
  17       0.7273   6.7181e+06
  18       0.7247   6.6955e+06
  19       0.7222   6.6720e+06
```

```
 [1] 4.950446 3.781814 3.239837 2.945597 3.586349 3.872310 3.578363 3.226444
 [9] 3.972112 3.048939
```

```
# A tibble: 7 x 4
  Method                              RMSE    MSE    MAE
  <chr>                              <dbl>  <dbl>  <dbl>
```

```
1 Project Goal                       0.865 NA     NA
2 Random prediction                  1.50   2.25   1.17
3 Mean                               1.06   1.12   0.855
4 Mean + bi                          0.943  0.889  0.737
5 Mean + bi + bu                     0.865  0.748  0.668
6 Regularized bi and bu              0.864  0.747  0.669
7 Matrix Factorization - recosystem 0.786  0.618  0.605
```

### 3.6 Final Validation

As seen from the result table, regularization and matrix factorization achieved the target RMSE. So, finally train the complete edx set with both models and calculate the RMSE in the validation set. The project goal is achieved if the RMSE stays below the target.

#### 3.6.1 Linear Model with Regularization

During the training and testing phases, the linear model with regularization achieved the target RMSE with a small margin. Here I do the final validation with the validation set.

```
# A tibble: 8 x 4
  Method                                RMSE    MSE    MAE
  <chr>                                <dbl>  <dbl>  <dbl>
1 Project Goal                         0.865 NA     NA
2 Random prediction                    1.50   2.25   1.17
3 Mean                                 1.06   1.12   0.855
4 Mean + bi                            0.943  0.889  0.737
5 Mean + bi + bu                       0.865  0.748  0.668
6 Regularized bi and bu                0.864  0.747  0.669
7 Matrix Factorization - recosystem    0.786  0.618  0.605
8 Final Regularization (edx vs validation) 0.865  0.748  0.669
```

As expected, the RMSE calculated on the validation set (0.8648) is lower than the target of 0.8649 and slightly higher than the RMSE of the test set (0.8641).

```
# A tibble: 10 x 1
# Groups:   title [7]
   title
   <chr>
 1 Usual Suspects, The (1995)
 2 Shawshank Redemption, The (1994)
 3 Shawshank Redemption, The (1994)
 4 Shawshank Redemption, The (1994)
 5 Eternal Sunshine of the Spotless Mind (2004)
 6 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
 7 Schindler's List (1993)
 8 Donnie Darko (2001)
 9 Star Wars: Episode VI - Return of the Jedi (1983)
10 Schindler's List (1993)
```

```
# A tibble: 10 x 1
# Groups:   title [9]
```

```
   title
   <chr>
 1 Battlefield Earth (2000)
 2 Police Academy 4: Citizens on Patrol (1987)
 3 Karate Kid Part III, The (1989)
 4 PokÃ©mon Heroes (2003)
 5 Turbo: A Power Rangers Movie (1997)
 6 Kazaam (1996)
 7 PokÃ©mon Heroes (2003)
 8 Free Willy 3: The Rescue (1997)
 9 Shanghai Surprise (1986)
10 Steel (1997)
```

### 3.6.2 Matrix Factorization

The initial test shows that matrix factorization gives the best RMSE. Now it's time to validate with the entire edx and validation sets.

```
Warning in set.seed(1234, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler used
```

```
iter      tr_rmse          obj
   0       0.9730   1.2010e+07
   1       0.8728   9.8927e+06
   2       0.8385   9.1718e+06
   3       0.8159   8.7419e+06
   4       0.8004   8.4635e+06
   5       0.7886   8.2711e+06
   6       0.7790   8.1154e+06
   7       0.7711   7.9976e+06
   8       0.7645   7.9023e+06
   9       0.7587   7.8217e+06
  10       0.7536   7.7584e+06
  11       0.7491   7.6975e+06
  12       0.7450   7.6517e+06
  13       0.7412   7.6071e+06
  14       0.7377   7.5682e+06
  15       0.7346   7.5320e+06
  16       0.7316   7.5027e+06
  17       0.7289   7.4724e+06
  18       0.7264   7.4477e+06
  19       0.7242   7.4256e+06
```

The final RMSE with matrix factorization is 0.7826, 9.5% better than the linear model with regularization (0.8648).

```
# A tibble: 9 x 4
  Method                                RMSE    MSE    MAE
  <chr>                                <dbl>  <dbl>  <dbl>
1 Project Goal                         0.865 NA     NA
2 Random prediction                    1.50   2.25   1.17
3 Mean                                 1.06   1.12   0.855
```

```
4 Mean + bi                                0.943  0.889  0.737
5 Mean + bi + bu                           0.865  0.748  0.668
6 Regularized bi and bu                    0.864  0.747  0.669
7 Matrix Factorization - recosystem        0.786  0.618  0.605
8 Final Regularization (edx vs validation) 0.865  0.748  0.669
9 Final Matrix Factorization - recosystem  0.783  0.612  0.603
```

Now, let's check the best and worst movies predicted with matrix factorization.

```
# A tibble: 10 x 1
# Groups:   title [8]
   title
   <chr>
 1 Lord of the Rings: The Return of the King, The (2003)
 2 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
 3 Schindler's List (1993)
 4 Shawshank Redemption, The (1994)
 5 Blackbeard's Ghost (1968)
 6 Shawshank Redemption, The (1994)
 7 Cats Don't Dance (1997)
 8 Pulp Fiction (1994)
 9 Boys Life 2 (1997)
10 Shawshank Redemption, The (1994)
```

```
# A tibble: 10 x 1
# Groups:   title [10]
   title
   <chr>
 1 Dead Girl, The (2006)
 2 Time Walker (a.k.a. Being From Another Planet) (1982)
 3 Beast of Yucca Flats, The (1961)
 4 Sorority House Massacre II (1990)
 5 Giant Gila Monster, The (1959)
 6 Alien from L.A. (1988)
 7 Free Willy 2: The Adventure Home (1995)
 8 Phish: Bittersweet Motel (2000)
 9 Material Girls (2006)
10 Madhouse (1990)
```

## 4. Conclusion

The initial challenge was to collect and prepare the dataset for analysis, later the necessity to explore the information looking for insights that could help during the model building.

Next, the creation of a random model that predicts the rating based on the probability distribution of each rating. This model gives the worst result.

The creation of a linear model with a very simple model that is the mean of the observed ratings. Continuing, a movie and user effects were added, that models the user behavior and movie distribution. With regularization a penalty value was added for the movies and users with few number of ratings. The linear model achieved the RMSE of 0.8648, successfully passing the target of 0.8649.

Finally, the recosystem package was evaluated, that implements the LIBMF algorithm, and achieved the RMSE of 0.7826.

## 4.1 Limitations

Some machine learning algorithms are computationally expensive to run in a commodity laptop and therefore were unable to test. The required amount of memory far exceeded the available in a commodity laptop, even with increased virtual memory.

Only two predictors are used, the movie and user information, not considering other features. Modern recommendation system models use many predictors, such as genres, bookmarks, playlists, etc.

The model works only for existing users, movies and rating values, so the algorithm must run every time a new user or movie is included, or when the rating changes. This is not an issue for small client base and a few movies, but may become a concern for large data sets. The model should consider these changes and update the predictions as information changes.

There is no initial recommendation for a new user or for users that usually don't rate movies. Algorithms that uses several features as predictors can overcome this issue.

## 4.2 Future Work

This report briefly describes simple models that predicts ratings. There are two other widely adopted approaches not discussed here: content-based and collaborative filtering. The recommenderlab package implements these methods and provides an environment to build and test recommendation systems.

Besides recommenderlab, there are other packages for building recommendation systems available in The Comprehensive R Archive Network (CRAN) website.

## References


1. Rafael A. Irizarry (2019). Introduction to Data Science
2. Ander Fernandez Jauregui (2021). How to Code a recommendation System in R
3. Leah Wasser, NEON Data Skills. How to use R Markdown Code Chunks
4. R Markdown Syntax: Hyperlinks, Images & Tables
5. Yihui Xie (2005-2020).Chunk options and package options