Report – MovieLens

Lucie Schaynová

2021-11-01

Contents

1 Introduction

In this project, we will work with MovieLens data set coming from the code:

```
if (!require(lubridate)) install.packages("lubridate")
library(lubridate)
if (!require(tidyverse)) install.packages("tidyverse")
library(tidyverse)
if (!require(caret)) install.packages("caret")
library(caret)
if (!require(data.table)) install.packages("data.table")
library(data.table)

dl <- tempfile()
movies <- str_split_fixed(readLines("ml-10M100K/movies.dat"), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))
ratings <- str_split_fixed(readLines("ml-10M100K/ratings.dat"), "\\::", 4)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- as.data.frame(ratings) %>% mutate(userId = as.numeric(userId),
                                           movieId = as.numeric(movieId),
                                           rating = as.numeric(rating),
                                           timestamp = as.numeric(timestamp))
movielens <- left_join(ratings, movies, by = "movieId")
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, temp, test_index, movielens, removed)
```

The code above is provided in the edX capstone project and gives us `edx` data set. We will analyze data in the following section.

2 Analysis

Let us have a look on data and basic summary statistics.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 3                        Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

Each row represents a rating given by one user to one movie. Data in the `title` column includes two information – name of a movie and a year of the movie's premiere. So we can create `year` column. There is also the `timestamp` column from which we can get the day of the week the rating was done, in scale 1–7 where 1 is Monday. We will create `rateday` column which will be useful for our further analysis.

```
## 'data.frame':    9000061 obs. of  8 variables:
##  $ userId   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: num  8.39e+08 8.39e+08 8.39e+08 8.39e+08 8.39e+08 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-Fi|Thriller" ...
##  $ year     : num  1992 1995 1994 1995 1994 ...
##  $ rateday  : num  5 5 5 5 5 5 5 5 5 5 ...
```

In total, there are 9000061 observations and 8 variables.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   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              year         rateday
##  Length:9000061     Length:9000061     Min.   :1915   Min.   :1.00
##  Class :character   Class :character   1st Qu.:1987   1st Qu.:2.00
##  Mode  :character   Mode  :character   Median :1994   Median :4.00
##                                        Mean   :1990   Mean   :3.86
##                                        3rd Qu.:1998   3rd Qu.:6.00
##                                        Max.   :2008   Max.   :7.00
```

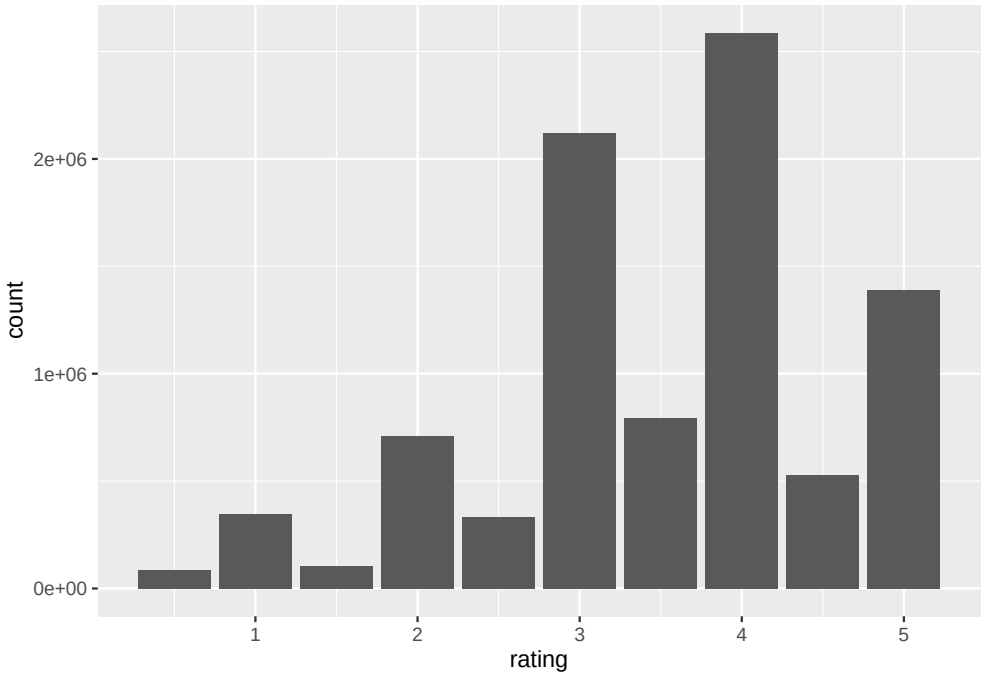
We can see that the data set is in the tidy format. The oldest movie is 106 years old.

```
##   n_users n_movies
## 1   69878    10677
```

We can see the number of unique users and movies which were rated. This gives us larger number ($69878 \times 10677 = 746087406$) than our 9 million rows. This means that not every user rated every movie.

```
##  [1] 5.0 3.0 2.0 4.5 3.5 4.0 1.0 1.5 2.5 0.5
```

We can see that there are not zeros given as ratings. The highest rate is 5.0, the lowest is 0.5.



The histogram shows us that half star ratings are less common than whole star ratings. The top 5 ratings from most to least are: 4, 3, 5, 3.5 and 2.

```
## # A tibble: 1 x 3
##   movieId numRatings movieTitle
##     <dbl>      <int> <chr>
## 1     296      31336 Pulp Fiction (1994)
```

Pulp Fiction movie has the greatest number of ratings. So there are some movies that are rated more often than others.

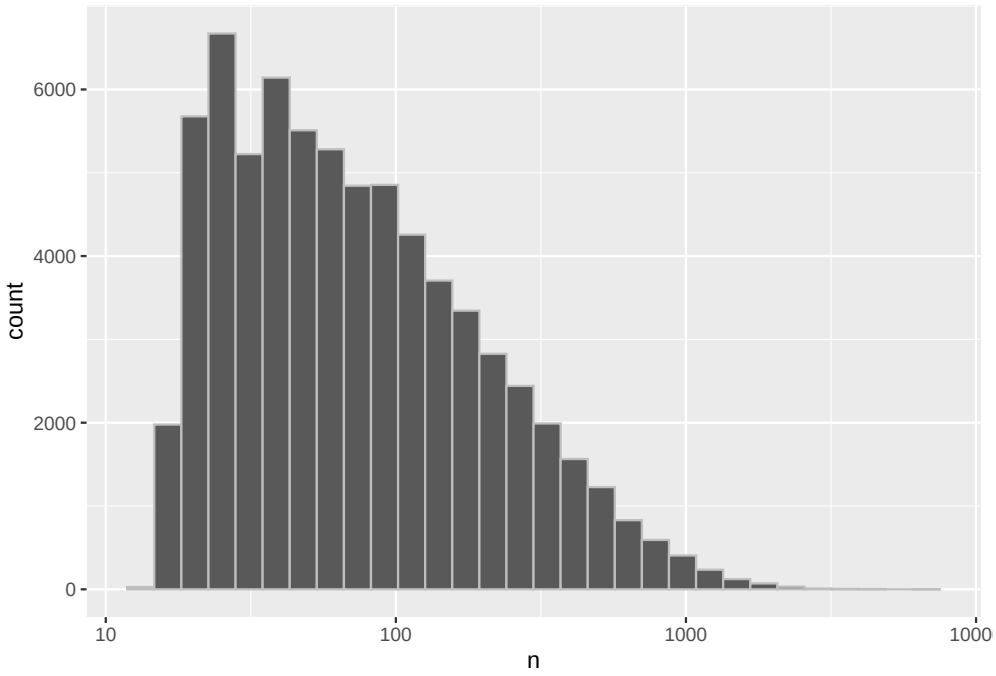
```
## # A tibble: 3 x 3
##   movieId numRatings movieTitle
##     <dbl>      <int> <chr>
## 1     318      14788 Shawshank Redemption, The (1994)
## 2     296      13533 Pulp Fiction (1994)
## 3     593      11754 Silence of the Lambs, The (1991)
```

The Shawshank Redemption, Pulp Fiction, and The Silence of the Lambs are top three movies with the greatest number of ratings equal to 5.0.

```
## # A tibble: 3 x 2
##   genres          count
##   <chr>           <int>
## 1 Drama          733353
## 2 Comedy         700883
## 3 Comedy|Romance 365894
```

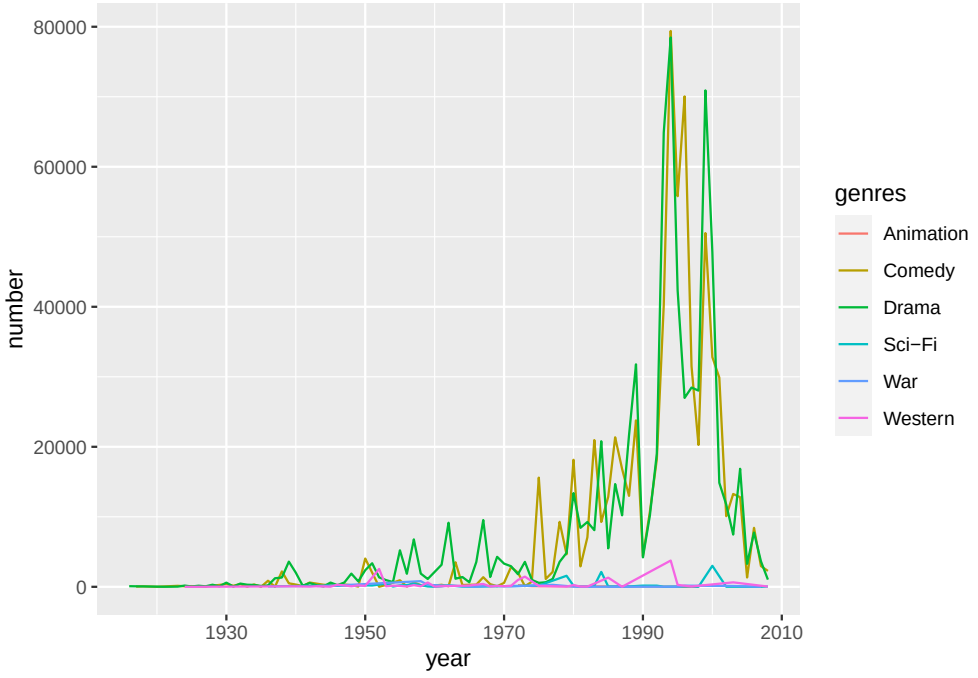
Drama is the most rated genre.

Some movies are rated more often than others. This corresponds to users activity as follows:

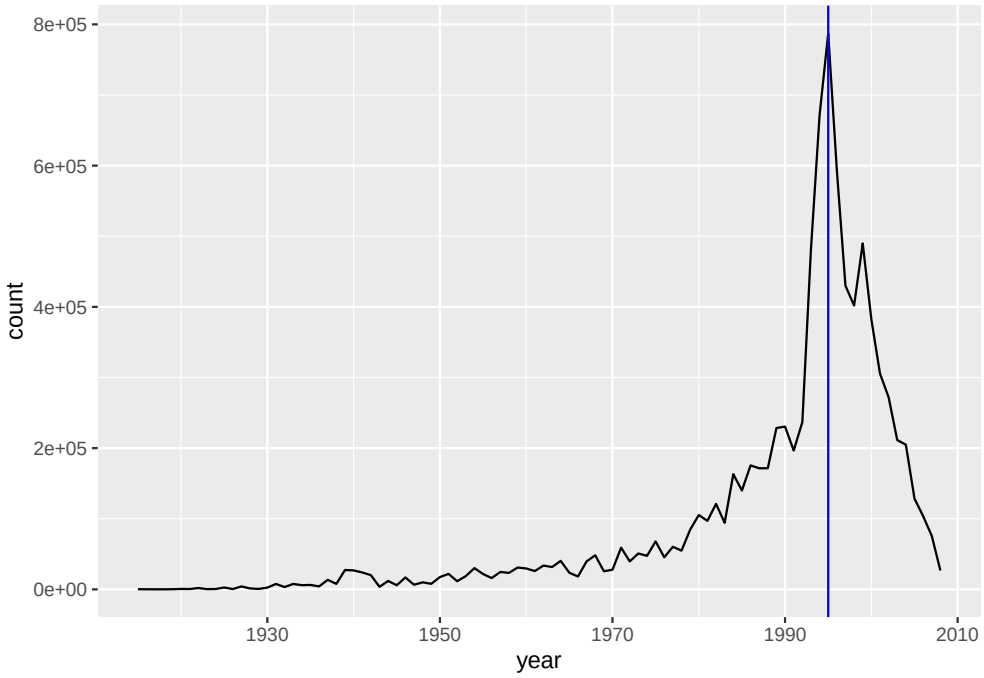


The plot above shows us that number of ratings are different for each user. Some users are more active than others when rating movies.

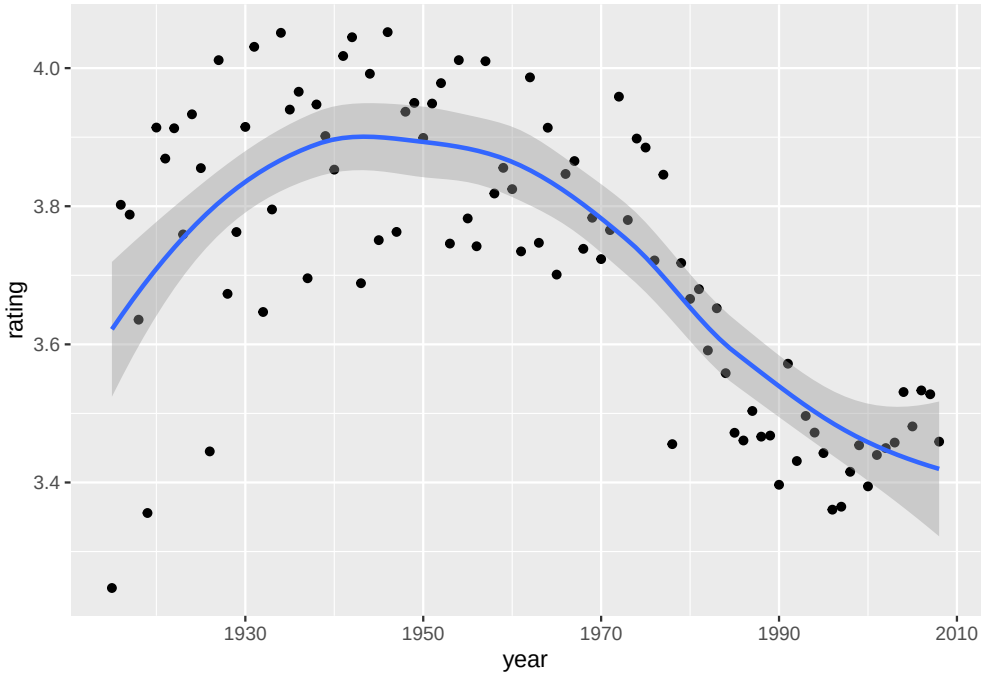
Now we can look at popularity of some genres per year.



Drama and comedy became more popular than others.

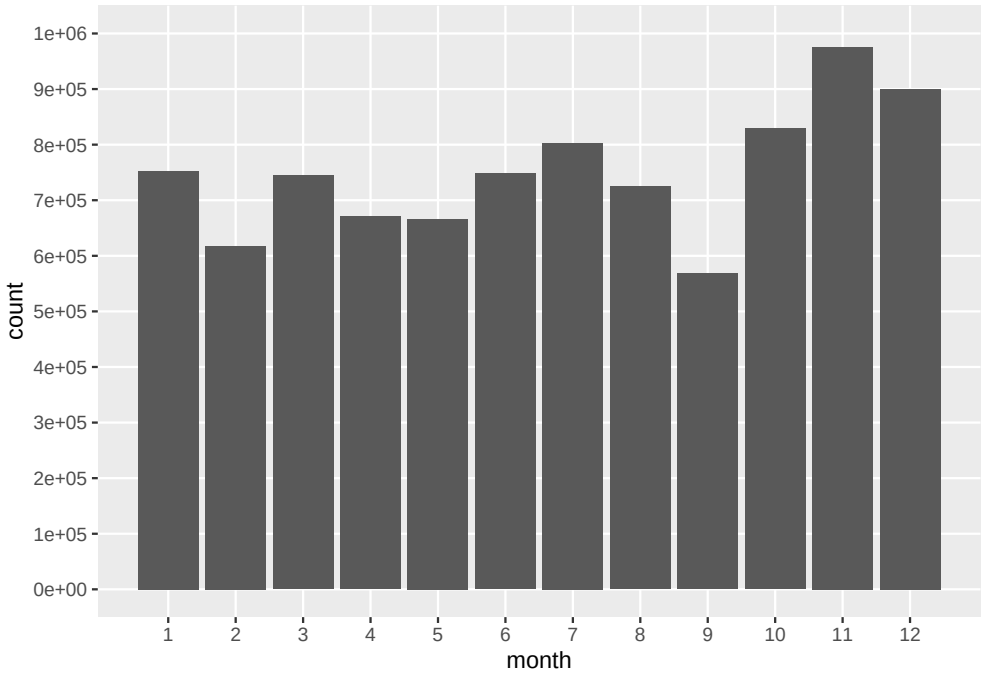


The plot above depicts number of ratings over the years. The greatest number of ratings is in 1995. We can also visualize average ratings per year:

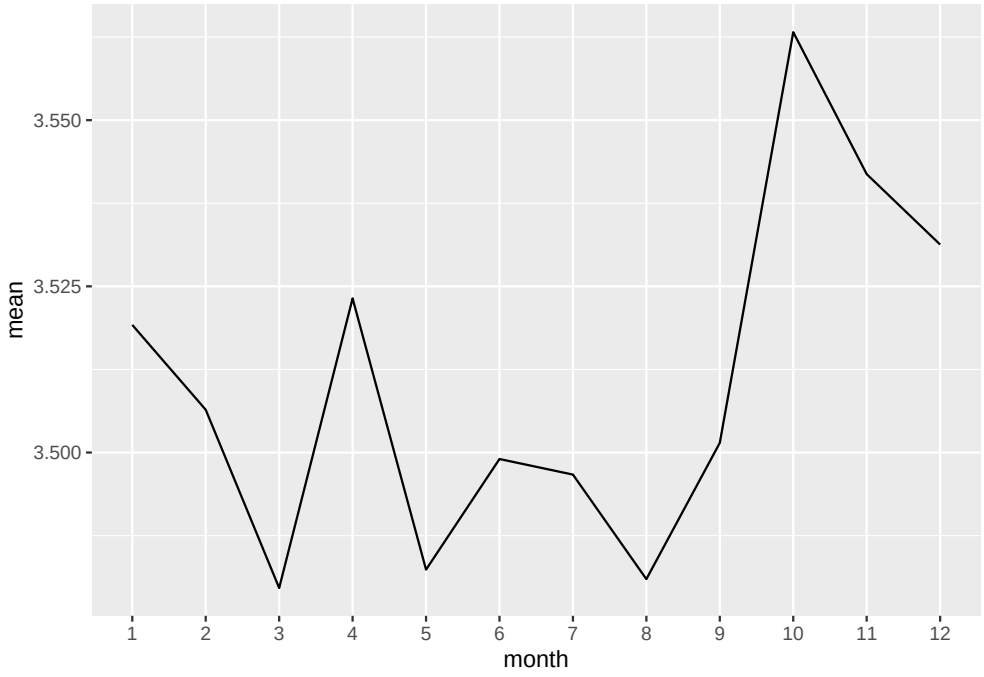


Until 1980, movies have very high rate, but modern movies are rated relatively low. What happened to the movies or users?

We need to arrange additional columns. The new column `datetime` converts `timestamp` to more friendly date format and the column `month` gives us month of the rating. We can use this to visualize a trend of ratings through months over the years:

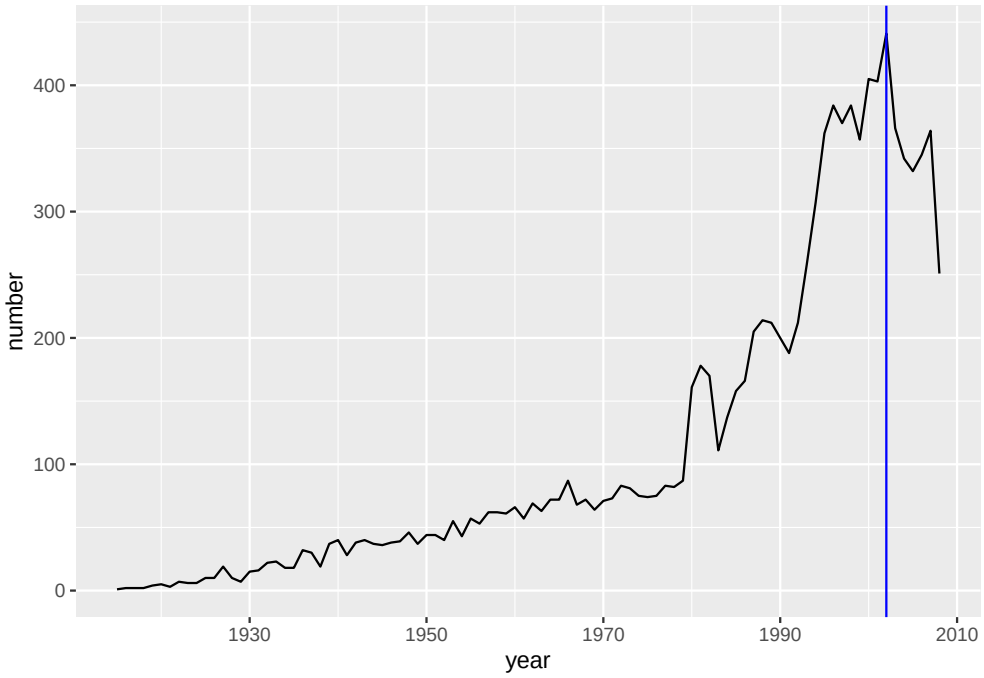


The highest activity of all users is in November, the lowest in September, but the activity is almost balanced. Similarly, we can visualize average ratings over the months:



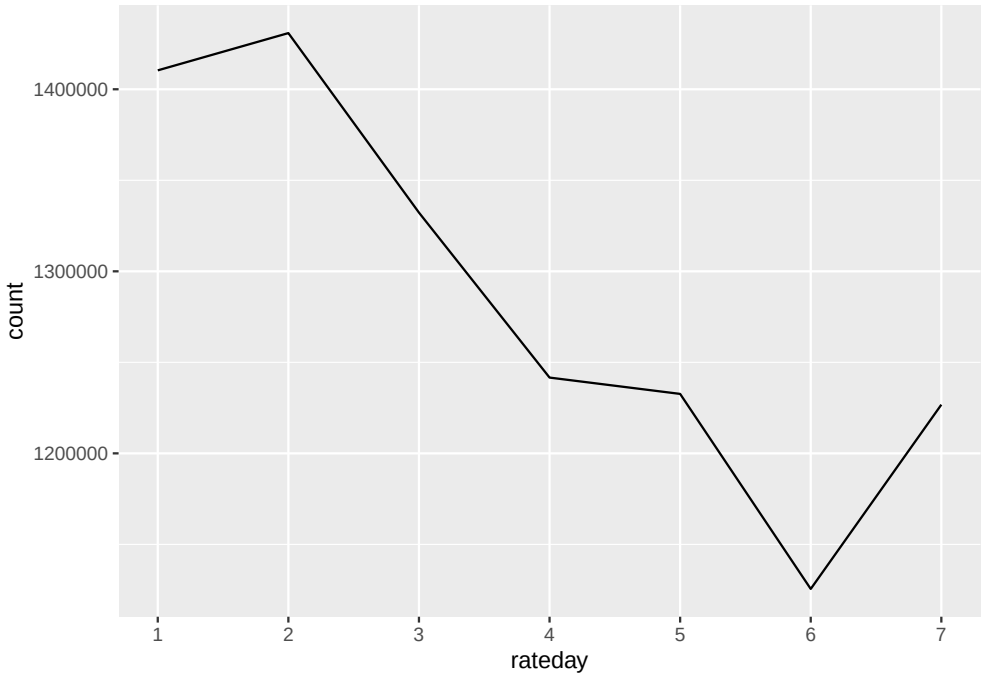
Users give the highest rates in October, the lowest in August.

There might be interesting a number of premieres for every year:



The number of premieres increases until 2002, then it decreases.

Finally, we will look at activity of users during each day of the weeks.



The highest activity is on Tuesday, the lowest on Saturday.

2.1 Model preparation

Our aim is to predict user ratings for a movie, based on chosen set of features (predictors) from our data set. We will use machine learning techniques. Our data set consists of 9 million ratings on 10 thousand

movies, made by 70 thousand users. The goal is to find a prediction method that will generate Residual Mean Squared Error (RMSE) lower than 0.86490.

Until now, we worked with the whole `edx` data set. For our machine learning purposes, we need to split the data set into training set `edx_train` and test set `edx_test` to assess the accuracy of the models we implement.

We cannot include users and movies in the test set that do not appear in the training set, so we need to remove these entries.

We will use the `RMSE` function which is typical error we make when predicting a movie rating. If this number is larger than 1, it means that our typical error is larger than one star, which is not good.

2.2 Naive model

The simplest model which ignores all the features is simply to calculate average of `rating`.

```
## average:

## [1] 3.512492

## RMSE:

## [1] 1.060006
```

We can see that the mean is not enough for our prediction. We need to improve our model adding some of our features.

2.3 Modeling movie effects

Let us take the movie effect into account as follows:

```
## RMSE:

## [1] 0.9427515
```

Our result is 0.9427515 which is not enough, but we can see an improvement. Let us make it better.

2.4 Modeling movie and user effect

Let us also add `userId` feature, so we will work with movie and user now.

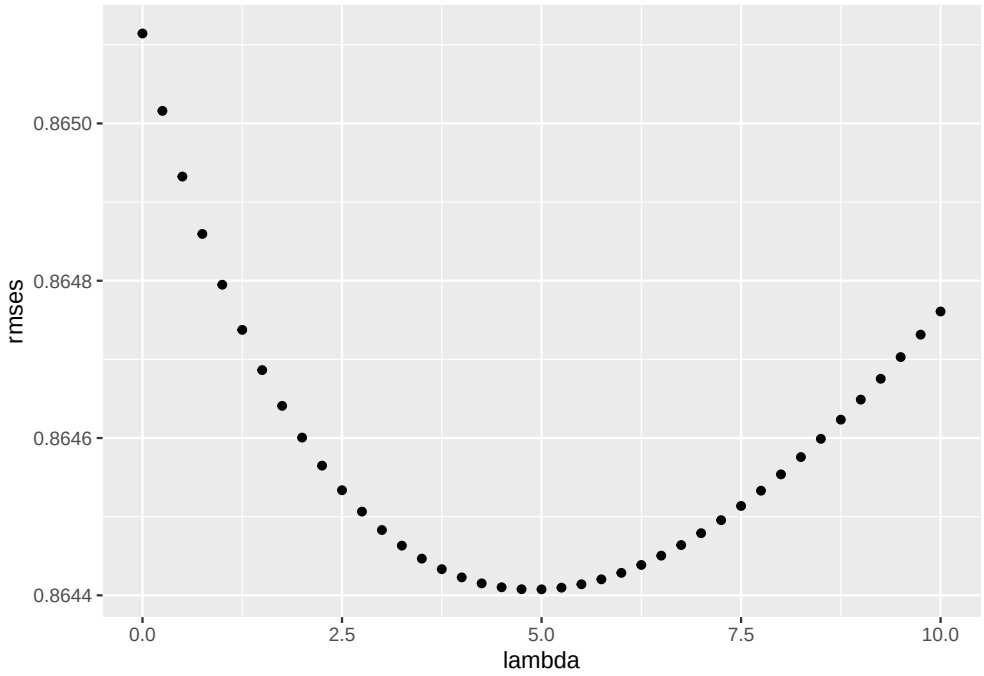
```
## RMSE:

## [1] 0.8651144
```

2.5 Regularization

We know that some users are more active in rating than others, so some movies are rated very few times. RMSE is sensitive to large errors. Large errors can increase our RMSE so we must put a penalty term to give less importance to such effect.

Denote a tuning parameter as `lambda`. We can use cross-validation to choose it. For each lambda we will find movie averages (`b_i`) and user averages (`b_u`) followed by rating.

Let us plot `rmses` vs `lambda` to select the optimal lambda.



The minimum of `lambda` is like this:

```
## minimum of lambda:
```

```
## [1] 5
```

We need to compute regularized estimates of `b_i` using `lambda_min` and regularized estimates of `b_u` using `lambda_min`.

```
## RMSE:
```

```
## [1] 0.8644076
```

The RMSE calculated on train and test data sets is equal to 0.8644076, which is slightly less than 0.86490.

3 Results

The RMSE values of our models are as follows:

```
## # A tibble: 4 x 2
##   method                                 RMSE
##   <chr>                                 <dbl>
## 1 Using mean only                       1.06
## 2 Movie Effect Model                    0.943
## 3 Movie and User Effect Model           0.865
## 4 Regularized Movie and User Effect Model 0.864
```

This indicates an improvement of the model over different assumptions. The final RMSE is 0.8644076 with an improvement. This implies that we can trust our prediction for movie rating given by a movie and a user.

4 Conclusion

In section Analysis, We analyzed original data from many different angles. We offered many statistics with plots. We also had to manipulate with original data to get additional columns. This also helped to us provide more extensive statistics.

Then we used machine learning modeling to predict ratings. The simplest Naive model gave us RMSE grater than 1 which means we may miss the rating by one star, which is not good prediction. Then Movie effect and User effect on model provided an improvement. A deeper insight into the data revealed some data point in the features have large effect on errors. So a regularization model was used to penalize such data points. The final RMSE is 0.8644076 with an improvement with respect to the baseline model. This implies that we can trust our prediction for movie rating given by a users.