Study of the predictive power of the different movie

1) Introduction

The following exercise is an attempt to determinate some of the factors influencing the score given by a sample of thousands of movie viewers to a particular group of films, by using the data science techniques acquired during the different modules of the course.

The available data is a subset of a bigger database called *movilens*, which contains millions of movie ratings made by thousands of users through many years. It is partitioned in two groups: a training dataset (called "edx") and a test dataset (called "validation").

The edx data contains a total of 9.000.055 movie ratings, made by 69.878 users to 10.677 different movies, between 1996 and 2016. The validation set, a smaller one, contains a total of 999.999 ratings, made by 68.534 of those users to 9.809 of the movies.

The goal of the exercise is to properly predict the ratings given by the users in the test set, based on the ratings those same users made in the training set. The data includes the following variables: movie ID, movie title, user ID, movie genre, movie release year and the date when every rating was made.

2) Methodology

The first step is to sub - divide the edx dataset into two new partitions, a *train set* and a *test set*. The algorithms developed in the train set will be evaluated in the test set to prove its effectiveness. This new partition is due to the fact that the validation dataset can only be used to evaluate the final code, so a new test data set is still necessary to optimize the model.

A set.seed(1) will be included before the partition, in order to obtain stable result in every run, consistent with the text presented in this report. The packages utils, caret, tidyverse and lubridate should be installed and loaded for the analysis to be made.

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

Based on the variables provided in the data, it is possible to measure the possible impact of a variety of parameters in every movie rating. Several statistical techniques, including some very complex and advanced ones, may be applied. Due to simplicity purposes, will be taking only four statistical parameters:

1. The user historical average rating, or **user effect**: every user has a different criterion to rate movies, some users are harsher or looser than others in their evaluations. The average rating given by a particular user in the past could give us an idea about how hard will evaluate other movies in the future.

2. The movie historical average rating, o **movie effect**: the average rating received by a movie from the collectivity of users could be considered as a fairly objective measure of the movie quality. Therefore, could give a clue about the rating that will receive from a particular user, independently of the user taste.

3. The genre historical average rating of **gender effect** : every movie is identified with a particular genre (Drama, Comedy, Action, etc.) or, like in several cases, with a particular combination of genres (comedy | children or mystery | action). Every particular combination of genres is considered as a separated one. Some genders may be historically better regarded than others by the sample of users.

4. The time when the movie was rated, or **time effect**: this is a more nuanced approach to the problem. A movie could be appreciated differently depending on the year the rating was made. The criterion from which the movies are rated may vary over time.

The first step to be taken is to measure the possible importance of every parameter. The fact that a certain variable could, in theory, affect the ratings does not mean they are doing it in practice. The most efficient way to test the possible real effects is by graphing the relation between every variable and the final scores.

The second step is to quantify the effects of the parameters resulting to be relevant. The way to find out is to compare the average rating from every category (user, movie, genre or time) with the overall rating. By measuring their average divergence from the global average in the training set, is possible to model the approximate the effect they will be having in the test set.

From the quantification of the individual effect of every parameter, it is possible to make predictions about the rating given by every user in the test set to their respective movies. The effectiveness of the predicted rating will be measured through the resulting RMSE (Root Mean Square Error). This is a statistical measure of the average distance between the predicted results and the actual ones.

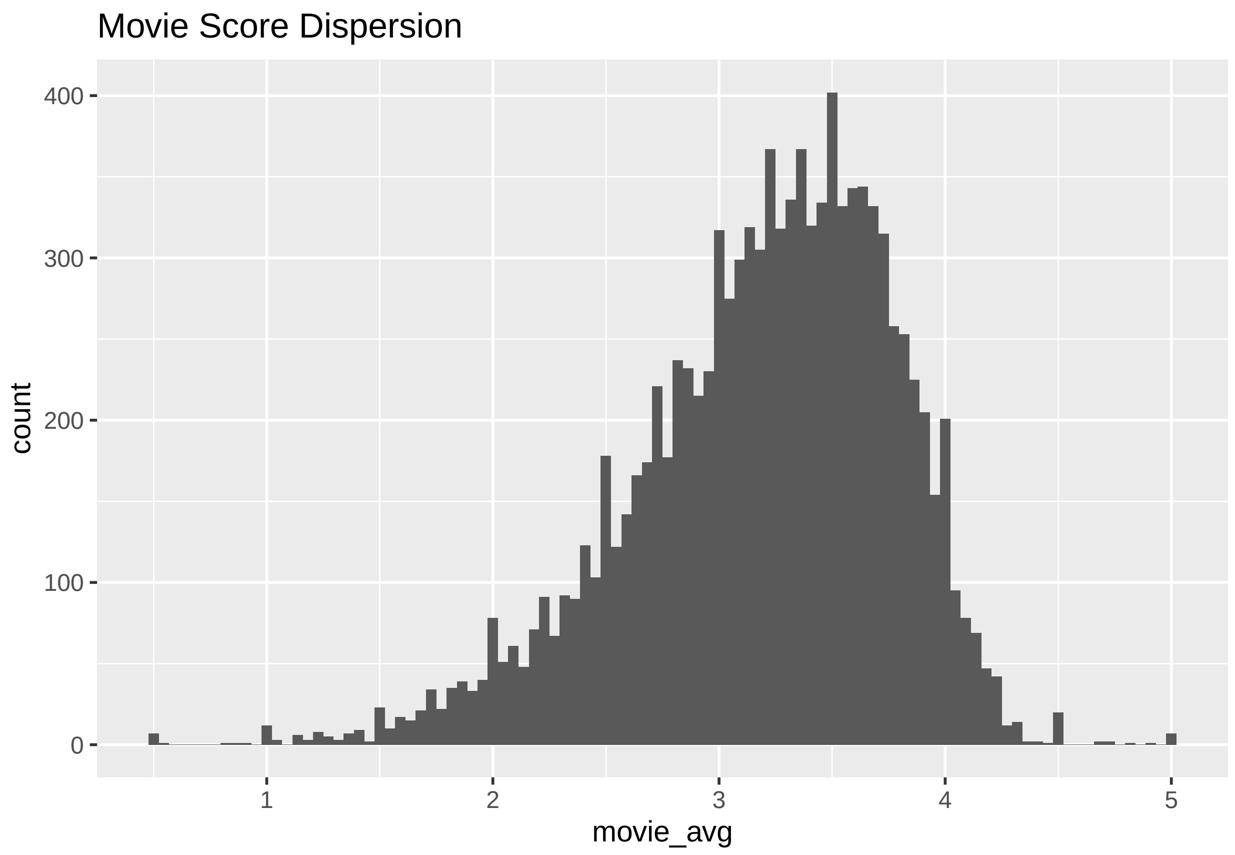
The individual statistical effect of every parameter will be measured by adding it one by one, and measuring their respective influence in the overall effectiveness of the model.

3) Results

3.1) Visualization study

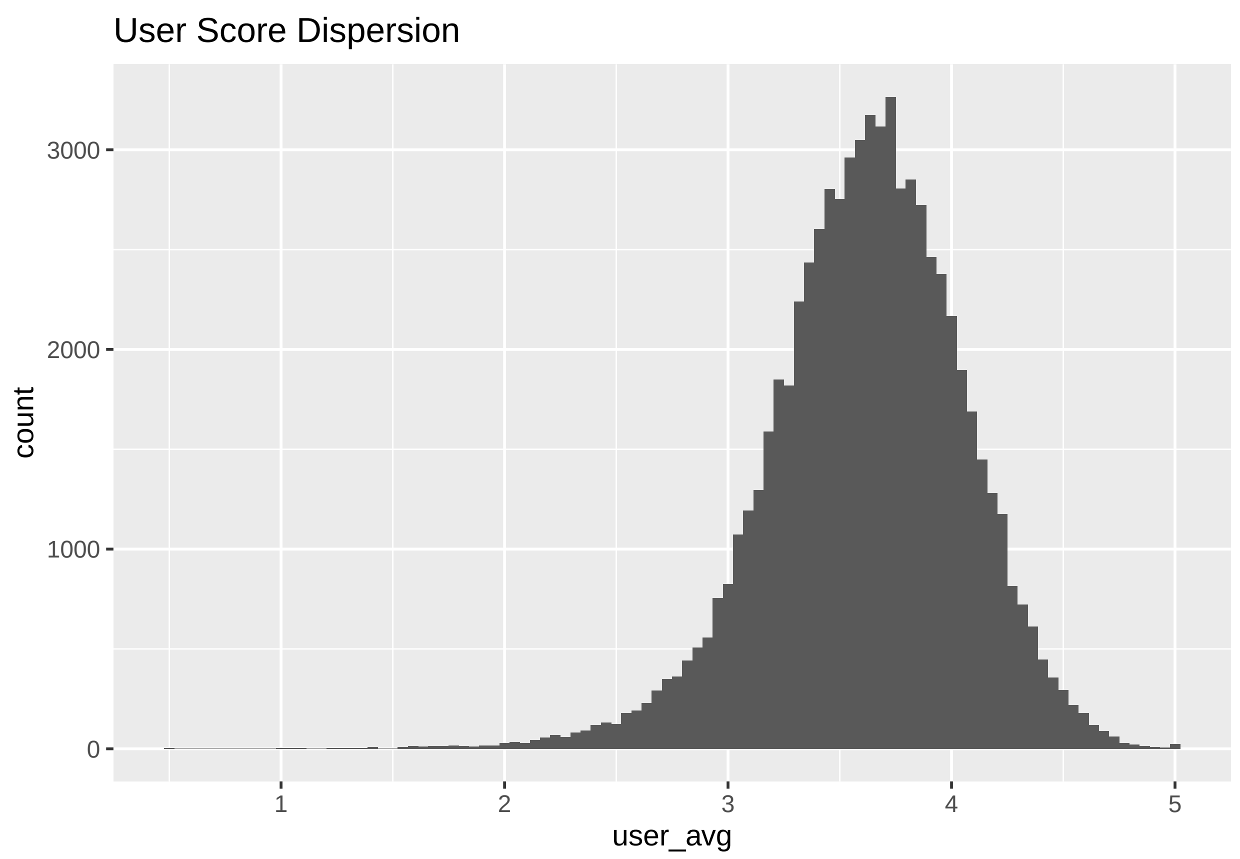
The visualization study tries to uncover relationships between the variables that could remain hidden otherwise . The human brain has a limited ability to detect patterns by only looking at the raw data, but is naturally capable of detecting patterns through images. In this case, we will try to see how much the average score from the different individuals that make up the variables (movie, user, gender, time) differs from the global average.

```r
train_set %>% group_by(movieId) %>% summarize(movie_avg = mean(rating)) %>%
  ggplot() + geom_histogram(aes(movie_avg), bins = 100) +
  labs(title = "Movie Score Dispersion")
```



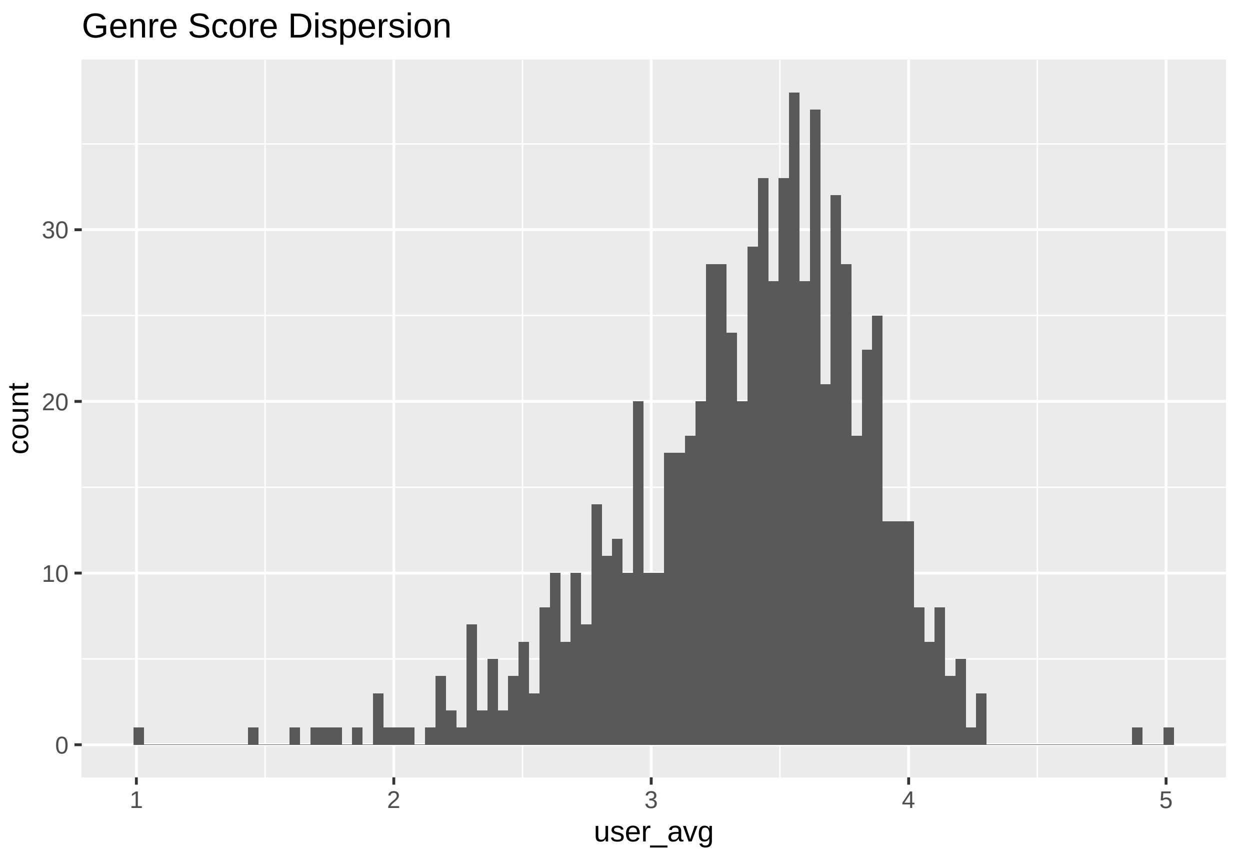
In the first graph, we can analyze the movie effect. It shows a high concentration of scores between 3 a 4 points. As expected, the distribution shows some resemblance with a normal bell curve, with a high concentration of observation around the average, which decreases as it approaches the tails. It is evident that some movies are better rated than others, independently of the rest of the variables.

```
train_set %>% group_by(userId) %>% summarize(user_avg = mean(rating)) %>%
  ggplot() + geom_histogram(aes(user_avg), bins = 100) +
  labs(title = "User Score Dispersion")
```



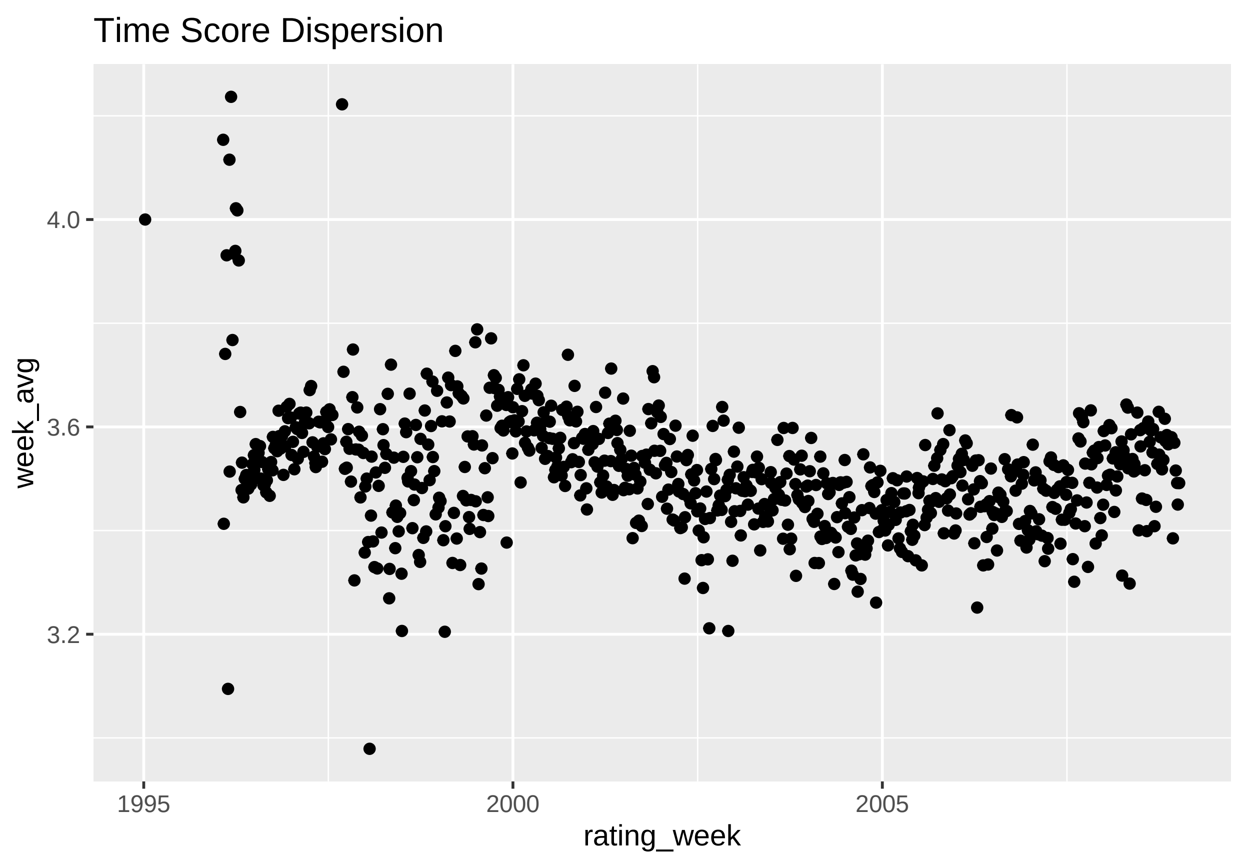
In a similar fashion, the user average score shows a bell curve, although with a significantly higher kurtosis. The rating criterion of the users is less dispersed than the quality of the movies. In any case, it still looks like a relevant factor to take into account.

```
train_set %>% group_by(genres) %>% summarize(user_avg = mean(rating)) %>%
  ggplot() + geom_histogram(aes(user_avg), bins = 100) +
  labs(title = "Genre Score Dispersion")
```



On the other hand, the genre effects show a more dispersed distribution than the first two. A bigger portion of individuals (genres) fall in the threshold between 2 and 3 points. Nevertheless, the modal segment continues to be the one between 3 and 4 points.

```r
train_set %>% mutate(rating_week = round_date(as_datetime(timestamp), unit = weeks())) %>%
  group_by(rating_week) %>% summarize(week_avg = mean(rating)) %>%
  ggplot(aes(rating_week,week_avg)) + geom_point() +
labs(title = "Time Score Dispersion")
```



Finally, the time effect graphic differs from the others in the fact that it is not an histogram. It is a dot plot that shows the changes for the behaviour of the rankings through times, and effectible shows an underlying time effect. Close to 1996, there was a handful of higher than average ratings, followed by a downward trend between 2000 and 2005, partially reverted between 2006 and 2016.

In summary, judging by the resulting graphics, the four variables have significant effects on the rating scores, even though not to the same degree.

3.2) Paremetres Estimation

According to the results presented in the graphical analysis, we now proceed to estimate the parameters from the four previously described effects. The standard procedure to estimate the effect of a variable is the Least Square. However, given the number of different individuals involved in any case, this solution could not be feasible in practice.

As was mentioned before, a more manageable approach is to quantify the difference between the average rating made by every individual (user, movie, genre or lapse of time), and the average global rating.

```
global_avg <- mean(train_set$rating)
global_avg
```

```
## [1] 3.512463
```

```
predicted_ratings_0 <- test_set %>%
mutate(pred = global_avg) %>% pull(pred)
```

```
Test_0 <- RMSE(predicted_ratings_0, test_set$rating)
Test_0
```

```
## [1] 1.060568
```

The average rating given to the movies in the dataset is 3.512, and the RMSE obtained by only using the global average as a predictor is 1.06013, a quite poor result, but not a surprising one. This is the benchmark over which further procedures will be evaluated.

The first predictor to consider is the movie effect. The distance between the average score obtained by every movie and the global average will be defined and added to the algorithm.It is always possible that some movies included in testset were not included in the training dataset, in which case the algorithm would not be able to make a prediction. In this case, the global average would be used as a substitute.

```
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - global_avg))

predicted_ratings_1 <- test_set %>%
left_join(movie_avgs, by='movieId') %>%
mutate(pred = global_avg + b_i ) %>% pull(pred)
predicted_ratings_1[is.na(predicted_ratings_1)] <- global_avg

Test_1 <- RMSE(predicted_ratings_1, test_set$rating)
Test_1
```

```
## [1] 0.9440174
```

As a first approach, the movie effect was quite successful. The RMSE dropped to around 0.9439. Nevertheless, there is still plenty of space for improvement. The next predictor to be evaluated is the user effect, by applying the same procedure used on the movie effects.

```
user_avgs <- train_set %>%
left_join(movie_avgs, by = 'movieId') %>%
group_by(userId) %>%
summarize(b_u = mean(rating - global_avg - b_i))

predicted_ratings_2 <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = global_avg + b_i + b_u) %>% pull(pred)
predicted_ratings_2[is.na(predicted_ratings_2)] <- global_avg

Test_2 <- RMSE(predicted_ratings_2, test_set$rating)
Test_2
```

```
## [1] 0.8666101
```

TThe resulting RMSE after the inclusion of the user effect is 0.8662. The more variables are added the more nuanced is the improvement of the model. However, every reduction in the RMSE is an advance in the process.

```
genre_avgs <- train_set %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - global_avg - b_i - b_u))

predicted_ratings_3 <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by = 'genres') %>%
  mutate(pred = global_avg + b_i + b_u + b_g) %>% pull(pred)
predicted_ratings_3[is.na(predicted_ratings_3)] <- global_avg

Test_3 <- RMSE(predicted_ratings_3, test_set$rating)

Test_3
```

```
## [1] 0.8662849
```

The third predictor, the genre effect, still manages to improve the performance of the model by some thousandth of points, by further dropping the RMSE to 0.8658.

```
time_avgs <- train_set %>%
  mutate(rating_week = round_date(as_datetime(timestamp), unit = weeks())) %>% group_by(rating_week) %>
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  left_join(genre_avgs, by = 'genres') %>%
  summarize(b_t = mean(rating - global_avg - b_i - b_u - b_g ))

predicted_ratings_4  <- test_set %>%
  mutate(rating_week = round_date(as_datetime(timestamp), unit = weeks())) %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by = 'genres') %>%
  left_join(time_avgs, by='rating_week') %>%
  mutate(pred = global_avg + b_i + b_u + b_g + b_t) %>% pull(pred)
predicted_ratings_4[is.na(predicted_ratings_4)] <- global_avg

Test_4 <- RMSE(predicted_ratings_4, test_set$rating)

Test_4
```

```
## [1] 0.8661616
```

Finnaly, the time effect was able to reduce the RMSE to 0.8657, an almost imperceptible difference, but still present.

As can be seen every effect considered improved the final result to some extent. Nevertheless, it's possible to get an even better result appling some simple techniques that are still in the toolbox.

3.3) Regularization

It's important to take into consideration that the effect of every movie and every user should not be considered equally. Some movies were rated by only a very small percentage of the users, and some users only made a handful of ratings. As a consequence, the uncertainty concerning those ratings is quite higher than in the case of more mainstream films or more experienced movie viewers. This is when the regularization comes into play.

This process tries to dilute the weight of the least frequent movies and individuals, through dividing by a factor. The magnitude of the factor (lambda) should be determined by an iteration process. A vector of 40 different values, from 0 to 10, tested to determine which one will be evaluated to determine which one gets the best RMSE.

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

rmses <- sapply(lambdas, function(l){

  b_i_r <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i_r = sum(rating - global_avg )/(n()+l))

  b_u_r <- train_set %>%
    group_by(userId) %>%
    left_join(b_i_r, by="movieId") %>%
    summarize(b_u_r = sum(rating -  global_avg - b_i_r)/(n()+l))

  predicted_ratings <- test_set %>%
  left_join(b_i_r, by = "movieId") %>%
  left_join(b_u_r, by = "userId") %>%
  mutate(pred = global_avg + b_i_r + b_u_r) %>% .$pred
  predicted_ratings[is.na(predicted_ratings)] <- global_avg

return(RMSE(predicted_ratings, test_set$rating))
})

Best_Lambda <- lambdas[which.min(rmses)]
Best_Lambda
```

```
## [1] 4.75
```

```r
plot(rmses)
```

```
## Error in plot(rmses): could not find function "plot"
```

The exercises dropped an ideal lambda of 5.0. The

3.4) Final Model

3.4.1) Run on the whole EDX dataset

Once all the possible effects and techniques have been evaluated, it is now time to apply them over the totality of the edx dataset, before test them in the validation set. The whole model, including regularization, have to be runned from the beginning in order to obtain movies, user, genre and time average that better resembles the original data.

```
  movie_avgs_r <- edx %>%
  group_by(movieId) %>%
  summarize(b_i_r = sum(rating - global_avg)/(n() + lambdas[which.min(rmses)]))

user_avgs_r <- edx %>%
  left_join(movie_avgs_r, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u_r = sum(rating - global_avg - b_i_r)/(n() + lambdas[which.min(rmses)]))

genre_avgs <- edx %>%
  left_join(movie_avgs_r, by = "movieId") %>%
  left_join(user_avgs_r, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - global_avg - b_u_r - b_i_r))

time_avgs <- edx %>%
  mutate(rating_week = round_date(as_datetime(timestamp), unit = weeks())) %>%
  left_join(movie_avgs_r, by = "movieId") %>%
  left_join(user_avgs_r, by = "userId") %>%
  left_join(genre_avgs, by = "genres") %>%
  group_by(rating_week) %>%
  summarize(b_t = mean(rating - global_avg - b_u_r - b_i_r - b_g))

predicted_ratings_edx <- edx %>%
  mutate(rating_week = round_date(as_datetime(timestamp), unit = weeks())) %>%
  left_join(movie_avgs_r, by='movieId') %>%
  left_join(user_avgs_r, by='userId') %>%
  left_join(genre_avgs, by = "genres") %>%
  left_join(time_avgs, by='rating_week') %>%
  mutate(pred = global_avg + b_i_r + b_u_r + b_g + b_t) %>% pull(pred)
predicted_ratings_edx[is.na(predicted_ratings_edx)] <- global_avg

RMSE_edx <- RMSE(predicted_ratings_edx, edx$rating)
RMSE_edx
```

```
## [1] 0.856513
```

The RMSE of 0.85654 is the best result obtained so far. It is not surprising, given that it could use the whole totality of the data available in the edx dataset. However, the real quality of the model could only be tested against the validation set.

3.4.2) Validation of the final model

```
Final_Model <- validation %>%
  mutate(rating_week = round_date(as_datetime(timestamp), unit = weeks())) %>%
  left_join(movie_avgs_r, by='movieId') %>%
  left_join(user_avgs_r, by='userId') %>%
  left_join(genre_avgs, by = 'genres') %>%
  left_join(time_avgs, by='rating_week') %>%
  mutate(pred = global_avg + b_i_r + b_u_r + b_g + b_t) %>% pull(pred)
Final_Model[is.na(Final_Model)] <- global_avg

Final_RMSE <- RMSE(Final_Model, validation$rating)
Final_RMSE
```

```
## [1] 0.864286
```

The final RMSE of the model is 0.86428, a fairly good result compared with the previoulsy obtained, and below the marked threshold of 0.86490. The recommendation system algorithm have successfully fulfilled its stated goal.

4) Conclusion

The presented study is an introductory attempt to grapple with the problem. It does not consider other variables, like the individual taste of every user to certain genders, only the overall score of every gender. It neither considers the so-called clustering effect, or the tendency of some groups of users to have alike tastes in movies.

Nevertheless, the marked RMSE threshold was achieved. The four considered effects (movie, user, genre and time) and the regularization process ended up being all relevant and useful.