MovieLens Recommendation System

Til Stein

16/05/2020

1. Introduction

In this analysis, I will try to predict the ratings users give to movies, based on the edx database. I will adopt a linear model including the predictors userId, movieId, genres and week. Our Measure of success is the root mean squared error (RMSE), which describes the difference between our prediction and the actual rating.

1.1 Data Source

The data is provided as part of the HarvardX Data Science: Capstone Course. You will find the script at the beginning of the RMarkdown document. It contains selected data from the popular Movielens dataset.

1.2 Data Splitting

For this analysis, I randomly partitioned the data into a training set and a test set. I will use the training set to build my algorith and the test set to improve it throughout the analysis. 90% of the data is in the training set, while 10% are in the test set. This is a very common way of splitting the data, as it gives me a good amount of entries to train my algorithm, as well as sufficient patients to test my algorithm. I have removed all entries from the test_set that have no matching entries in the train_set. Furthermore, we have a validation data set, which we will use only at the end, to test our algorithm.

1.3 Data Insight

The data set holds six variables with 9000055 observations each.

```
## [1] 9000055       7
```

As this is quiet large, I created an even smaller data set called insight_data with only 10% of the train_set data. This makes it much faster to visualise the dataset, without loosing much information.

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

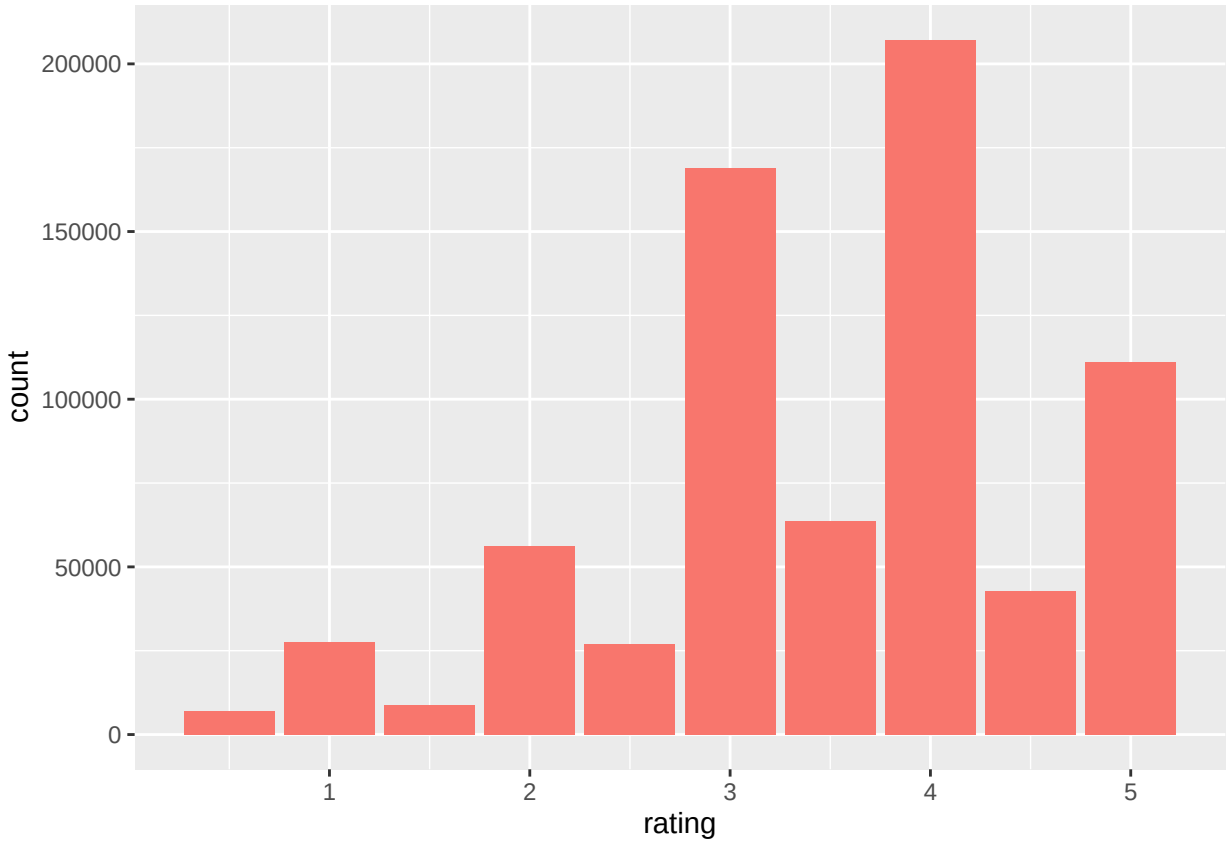
1.3.1 Prediction Variable We will try to predict the variable called "rating" from the remaining variables (predictors). The rating variable can take values between 0.5 and 5, with steps of 0.5. It is given to the movies by the users.

```
## [1] "Unique Ratings:"
```

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

First, we want to take a look at the distribution of our ratings variable, that we want to predict. From the plot we can see, that the integer ratings are much more common, than the half-star ratings. We can also see

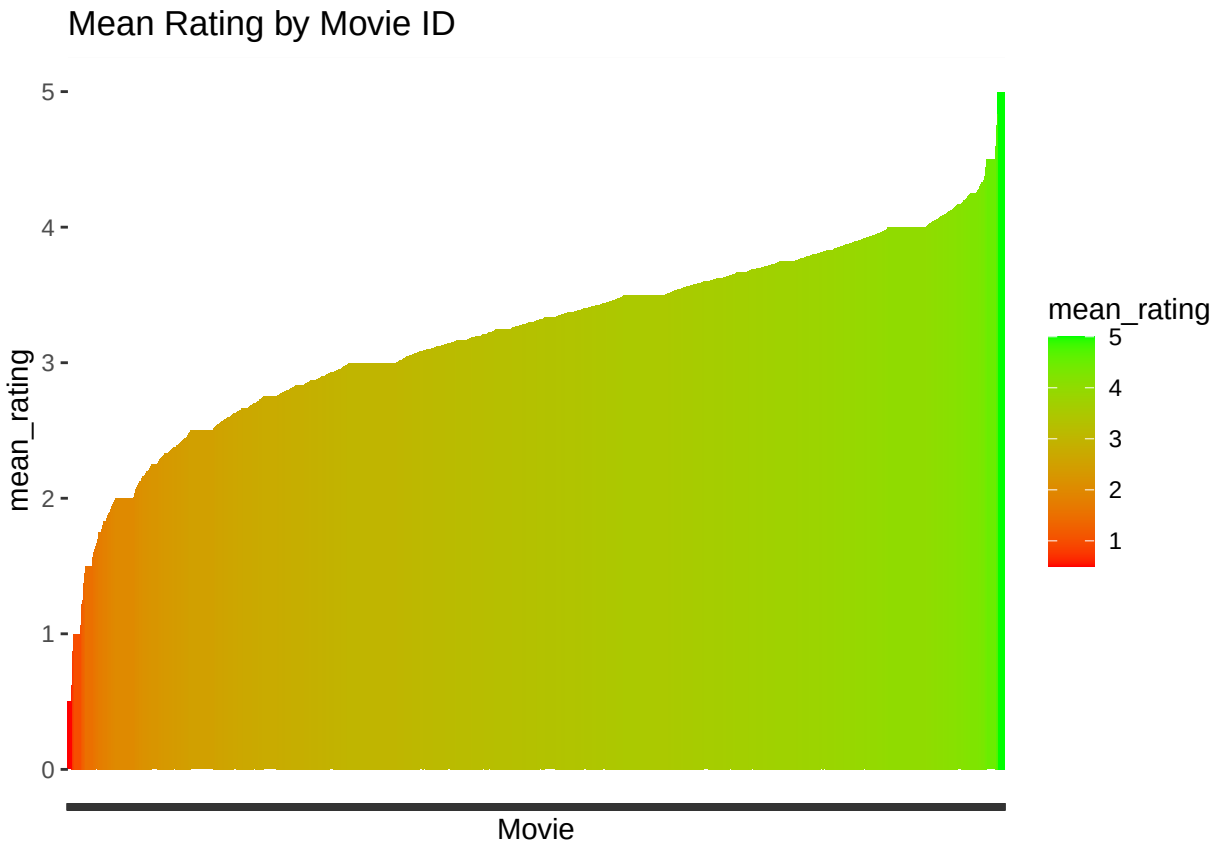
that “3” and “4” are the most common ratings.

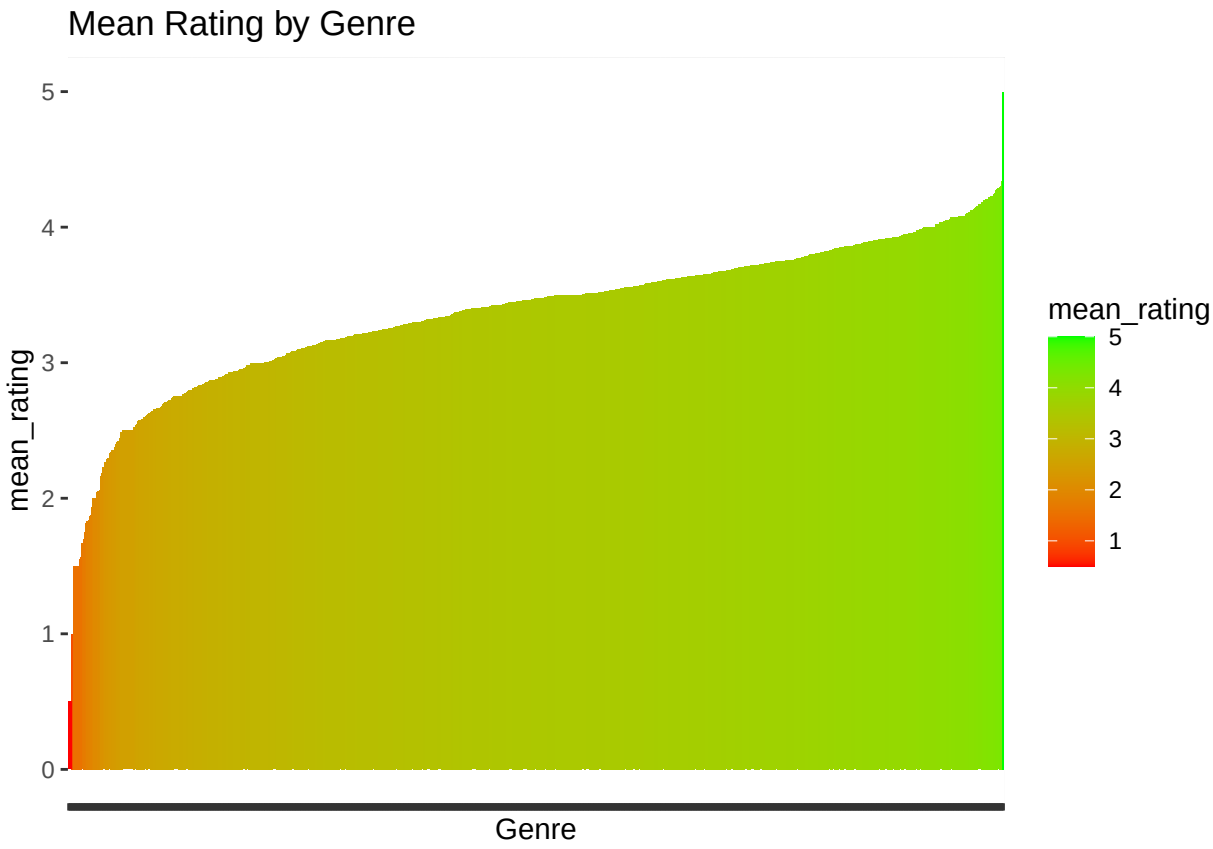


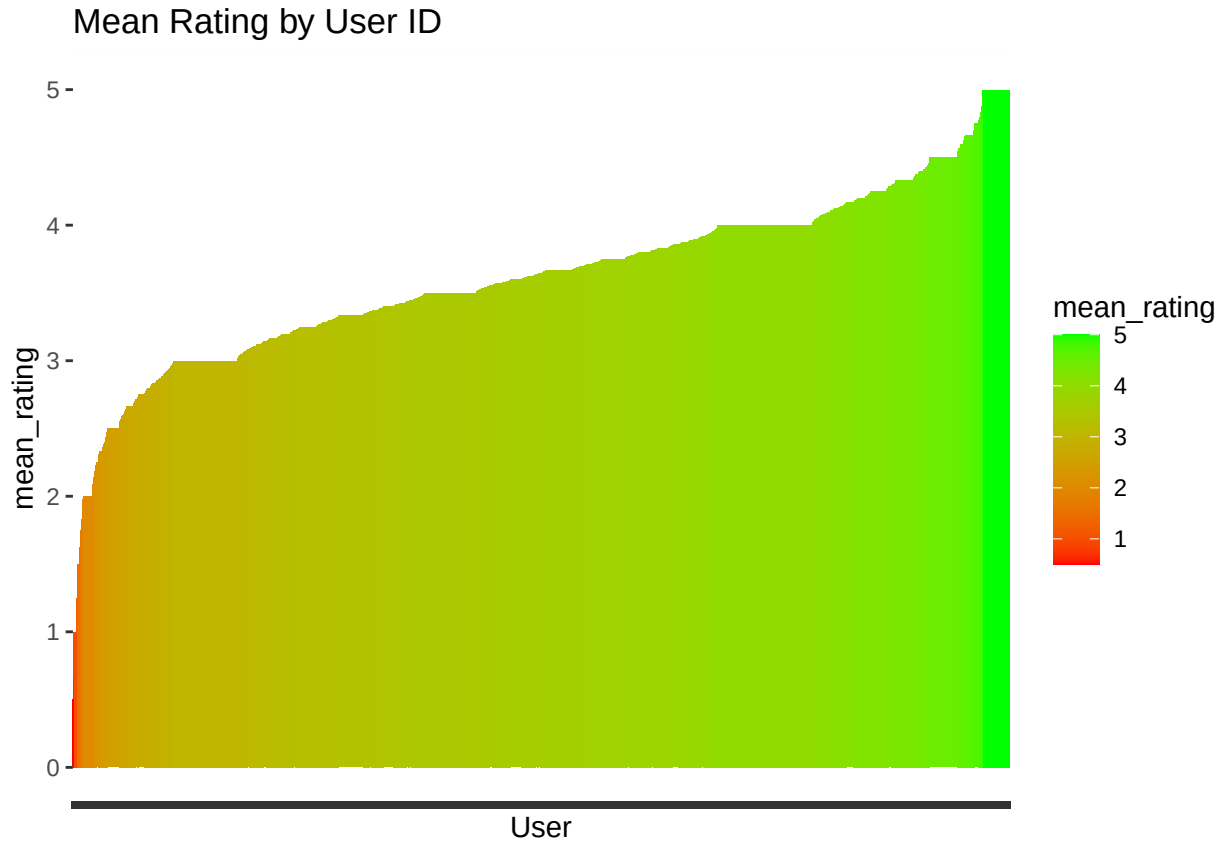
1.3.2 Predictors Let’s briefly talk about the predictors. First, there is the UserID. It is an integer number unique to each user. Then, there are the variables movieID and title. These contain the same information. However, one in movieID, the information is encoded as an integer, while in title it is encoded as a character string. Then, there is the genres variable. It is a string and can hold multiple genres of a movie. For example, the genre can be only “Comedy” or “Romance”, but also “Comedy | Romance”, which stand for “Comedy” and “Romance”. Lastly, we have a timestamp for each rating. It is originally stored as an integer, but I also created the date variable in the format date.

Variable	Class
UserId	integer
MovieId	numeric
Title	character
Genres	character
Timestamp	integer
Date	POSIXct

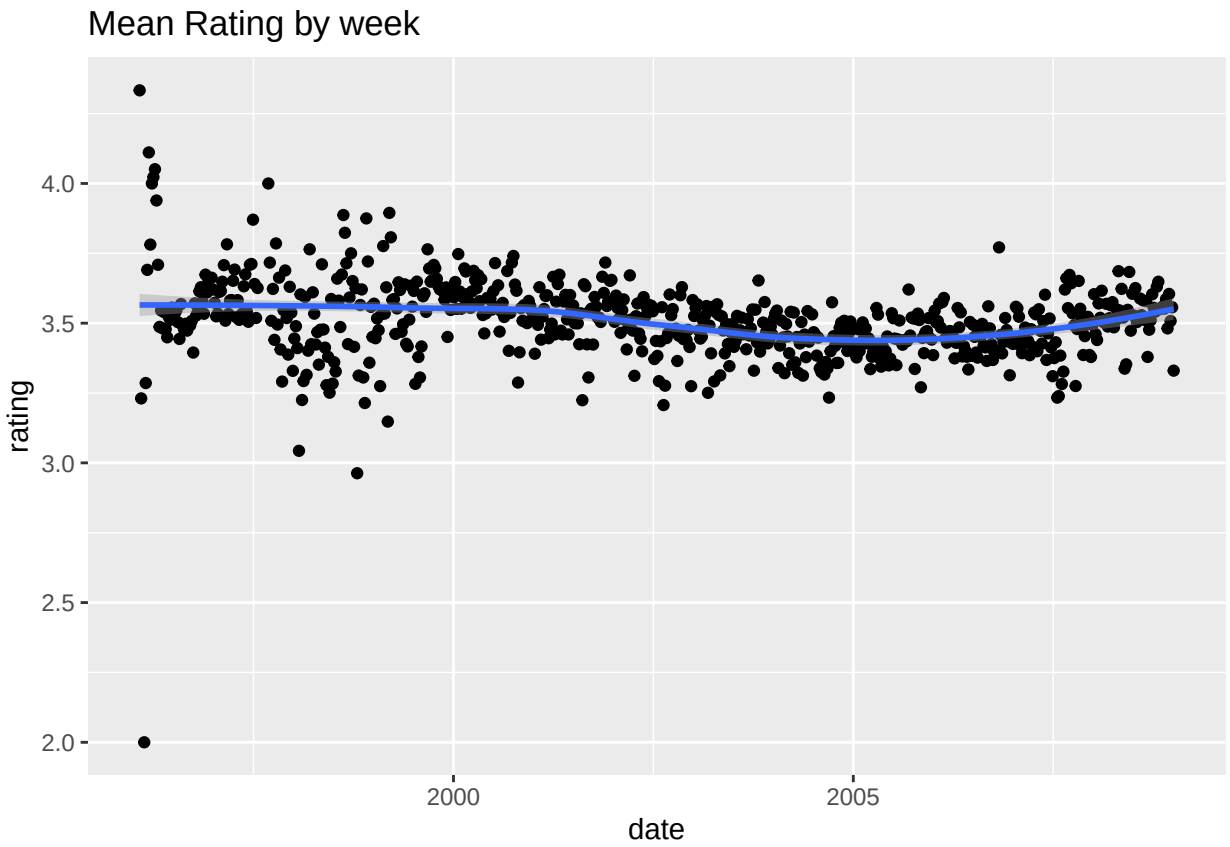
In the following, I have created graphs that show correlations between the rating a movie is given and a single predictor.


Mean Rating by Movie ID
mean_rating
5
4
3
2
1
0
Movie
mean_rating
5
4
3
2
1


Mean Rating by Genre
mean_rating
5
4
3
2
1
0
Genre
mean_rating
5
4
3
2
1




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



As we can see, all variables have an influence on the mean rating. The effect is especially strong for movieId and UserId. The week a user rated a movie is much less of an influence. Our findings are very intuitive. For example, some movies are generally seen as better than others and therefore rated better. Furthermore, some people generally seem to be more generous when rating movies than others.

2. Analysis

Now we will start our prediction. I will use a linear model that predicts y_hat by looking at mu (mean rating) and the variance from mu inserted by the movie, user, genre and date effect. As mentioned in the introduction, we will use the RMSE as our measure of predction success. The function is as follows:

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

2.1 Guessing

As a first comparison, we will try to simply guess the movie ratings. This will give us a reference RMSE value. I have included the code, so you can see how our guesses were generated.

```
#Guessing
set.seed(1, sample.kind= "Rounding")
```

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

```
#Creating a vector of all possible outcomes
rating_outcomes <- c(0.5,1,1.5,2,2.5,3,3.5,4,4.5,5)
```

```r
#The number of times we want to replicate the random "drawing" of ratings
n=10

#Setting the seed
set.seed(1, sample.kind= "Rounding")
```

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

```r
#With this function, we replicate a random drawing of values from our vector random_drawing vector n ti
results_guessing_rmse <- replicate(n,{
y_hat <- sample(rating_outcomes, length(test_set$rating), replace=TRUE)
RMSE(test_set$rating,y_hat)
})
#Getting the average RMSE of our n drawings
guessing_rmse <- mean(results_guessing_rmse)
```

method	RMSE
Guessing	1.940721

We can see that our RMSE lies at 1.9408. This means than on average, we are about 2 ratings of the true rating. Let's see if we can improve on that.

2.2 Using the Mean

```r
#getting mu
mu <- mean(train_set$rating)
#calculating our naive_rmse by comparing true rating with our mean
naive_rmse <- RMSE(test_set$rating,mu)
#adding our naive_rmse to the rmse_results dataframe
rmse_results <- bind_rows(rmse_results,
                          tibble(method ="Mean",
                                     RMSE = naive_rmse))
#Printing our RMSE
rmse_results[2,] %>% knitr::kable()
```

method	RMSE
Mean	1.059904

Now we start with the real analysis. We will try predicting the ratings by always guessing the mean rating. The code explains how it is done. We can see that our RMSE has significantly improved from just guessing, but is still quiet high at about 1.0599.

2.3 Including MovieId

From now on, I only include the code that shows the development of our code and not the calculation and printing of RMSE.

```r
#Creating our b_i variable by looking at the difference between rating and mu
movie_avgs <- train_set %>%
     group_by(movieId) %>%
```

```
     summarize(b_i = mean(rating - mu))
#calculating our y_hat(predicted ratings) with our new model
predicted_ratings <- mu + test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     .$b_i
```

method	RMSE
Movie Effect Model	0.9437429

This time, we add the variation that is beeing explained by the movie. By including this, we achieve a significant improvement of our RMSE, which is now at 0.9437.

2.4 Including UserId

```
#Creating our b_u variable by looking at the variance not explained by b_i
user_avgs <- train_set %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
#calculating our y_hat(predicted ratings) with our new model
predicted_ratings <- test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     left_join(user_avgs, by='userId') %>%
     mutate(pred = mu + b_i + b_u) %>%
     .$pred
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

method	RMSE
Movie + User Effects Model	0.865932

In our introduction we also saw that the user had a significant effect on the rating. With the code above, we take this into consideration. Our RMSE has improved again and now stands at 0.8659.

2.5 Including Genres

```
#Creating our b_g variable by looking at the variance not explained by b_i or b_u
genre_avgs <- train_set %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
#calculating our y_hat(predicted ratings) with our new model
predicted_ratings <- test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     left_join(user_avgs, by='userId') %>%
     left_join(genre_avgs, by='genres') %>%
```

```
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred
```

method	RMSE
Movie + User + Genre Effects Model	0.8655941

Now, we have also added the genre as a variable. Our RMSE is quiet low at 0.8655.

2.6 Including Weekly Effect

```
#Creating our b_d variable by looking at the variance not explained by b_i, b_u or b_g
date_avgs <- train_set %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(genre_avgs, by="genres") %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - mu - b_i - b_u - b_g))
#calculating our y_hat(predicted ratings) with our new model
predicted_ratings <- test_set %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(genre_avgs, by='genres') %>%
    left_join(date_avgs, by="date") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
    .$pred
```

method	RMSE
Movie + User + Genre +Date Effects Model	0.8654875

To achieve the final improvement of our RMSE, we include the weekly variation in RMSE. Our RMSE has improved ever so slightly to 0.8654.

3. Results

In this section, we will look at the performance of our algorithm with an unknown data set. I will train our variables mu, b_i, b_u, b_g and b_d from the entire edx data set. Furthermore, I will replace NA Values with 0. Lastly, I predict the validation$ratings.

3.1 Final Model

In the following, you can see the code for our final model.

```
#Creating Mu from entire edx data set
mu <- mean(edx$rating)
#Creating b_i from entire edx data set
movie_avgs <- edx %>% group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
#Creating b_u from entire edx data set
user_avgs <- edx %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>%
```

```
  summarize(b_u = mean(rating- b_i -mu))
#Creating b_g from entire edx data set
genre_avgs <- edx %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
#Creating b_d from entire edx data set
date_avgs <- edx %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(genre_avgs, by="genres") %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - mu - b_i - b_u - b_g))

#Adding the b_i, b_u and b_g variables to the validation data frame
validation <- validation %>%
  mutate(date=as_datetime(timestamp)) %>%
  mutate(date = round_date(date, unit = "week")) %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(genre_avgs, by="genres") %>%
  left_join(date_avgs, by="date")

#Replacing NA Values in validation data frame
validation$b_i[is.na(validation$b_i)] <- 0
validation$b_u[is.na(validation$b_u)] <- 0
validation$b_g[is.na(validation$b_g)] <- 0
validation$b_d[is.na(validation$b_d)] <- 0

#Doing our Prediction
prediction <- validation %>%
  mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
  .$pred
```

3.2 Discussing our results

method	RMSE
Final Result	0.8648392

As we can see, our final RMSE is at 0.8648392. If we compare it with the table below, which shows the development of our RMSE throughout the models, we can see that it is performing slightly better with the unknown validation set, than it did with the training set. This can be down to many factors. It could show that we did not overtrain our algorithm, but it could also be down to luck. Nevertheless, I am satisfied with the result.

method	RMSE
Guessing	1.9407209
Mean	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659320

method	RMSE
Movie + User + Genre Effects Model	0.8655941
Movie + User + Genre +Date Effects Model	0.8654875
Final Result	0.8648392

4 Conclusion

By slowly building a linear regression analysis, we managed to obtain an RMSE of 0.8648392. This is a results I am quiet satisfied with. However, it could propably be improved, if we would look closer at the genres, e.g. counting "Comedy | Romance" as two genres. We could also try to comvine this linear regression model with other machine learning algorithms. Unfortunately, my computer was not strong enough to do such calculations. It was a great project and a great introduction of the field of data analytics.