MovieLens Report

Sari Inkila

23/02/2021

Contents

1. Executive Summary

1.1 Introduction to the project

This is a report for the Harvard Data Science: Capstone -course Project on MovieLens. The MovieLens movie prediction model assignment is first of two assignments included in the Capstone course. MovieLens data set, or rather a fraction of it, was also used in the previous Harvard Data Science course Machine Learning. Therefore, this project starts from where the previous course left off and is building on the thought work and models already done during that course. For continuity and ease of understanding the parts that are new and reused (some of the variables, data frames and functions) follow the same structure and notations as in the previous course. Also, the parts that are essentially the same as in the previous course have been noted in the R code.

This project report presents the data evaluation, cleansing and analyses work done to determine the prediction model building blocks. Data analyses that was already done in the course Machine Learning or the initial quiz of the Capstone course are not repeated in this project work.

The body of this report does not include the R code used to ensure that reader is able to follow the thought process and the results easily. In other words, you do not need to be proficient in data science or R code to be able to read this report. The appendix of this report includes all of the code and is targeting an audience interested in understanding the analytical models and R code used to conduct the analysis and model building work done.

1.2 Data set

The original data was obtained from Grouplens and it is the MovieLens 10M data set:

- https://grouplens.org/datasets/movielens/10m/
- http://files.grouplens.org/datasets/movielens/ml-10m.zip

The data table generated from the source files and used in this project includes following columns:

- userId (data set included 69878 users),

- movieId (data set included 10 677 movies),

- rating (on the scale of 0-5, with 0.5 increments permitted. However, no rating of zero was in the data set),

- timestamp,

- title (includes also the year that the movie was published),

- genres (should be identical to genres in IMDB, and are a pipe-separated list).

In the source files all the users were selected at random. There were no demographic information included and each user was represented only by an id.

1.3 The goal, model used and the results

The goal of the project was to develop a movie recommendation prediction model to predict the user rating of a movie on the validation set (20% of the entire data set) based on the model developed on the edx data set (80% of the entire data set).

The model was developed using the least squares estimate (LSE) approximation and regularization with data on mean rating, movie bias, user bias, genre bias and movie's year of publication. Other models were experimented with (e.g. caret -package algorithms using the train -function as well as matrix factorization), but due to the size of the data set using any other method was not feasible from performance point of view.

Target of the project was to build a model that achieves a residual mean squared error (RMSE) lower than 0.86490. The final result achieved with the model developed using the edx data set resulted on the validation set in RMSE of: [1] 0.8640543

1.4 Key steps

The key steps that were performed were:

1. Data exploration to understand what the data set contained and resulting data cleansing

- Understand data volumes,
- Analyze consistency of the data and need for data cleansing,
- Determine need for data cleansing,
- Conduct necessary data cleansing, e.g. by using string processing to get the data into tidy data format for easier processing.

2. Detailed data analyzes on

- Genres,
- Time stamp,
- Movie's publication year.

3. Creating the training and test sets from the edx data set for the model development

4. The model development

- Starting with the model developed in the course 8. Machine Learning / Recommendation System - Creating functions and tables needed,
- Building on the least squares estimate (LSE) model with genre bias,
- Building on the least squares estimate (LSE) model with time stamp bias,
- Building on the least squares estimate (LSE) model with year of publishing the movie bias,
- Building on the least squares estimate (LSE) model with regularization.

5.Validation of the developed model on the validation data set and calculating the residual mean squared error (RMSE) of the model on the validation data set.

---

2. The Analysis

This section describes the methods and analysis used in this project, including explanations for the analysis process and the techniques used. These will include: data cleaning, data exploration and visualization, insights gained, and modeling approach used.

MovieLens data set, or rather a fraction of it, was also used in a previous Harvard Data Science course Machine Learning. Therefore, this project starts from where the previous course left off. It is building on the thought process and models already developed during that course. Data analyses that was already completed in the previous course, or the initial quiz of this Capstone course, are not repeated in this project work.

However, the list below sums up the key findings from the previous course for those of the readers of this report that are not familiar with them:

- Each movie in the data set had ratings,

- The more often a movie was rated, the higher its average rating was,

- Two effects were included in the model development: movie bias and user bias as well as the average rating (mu),

- There was hinting that time stamp (the time a rating was given) as well as the genre linked to the movie have an effect on the rating. However, they were not added to the model developed during the course 8. Machine Learning.

The final part of the analysis section (2.3) focuses on the process of developing the model. Including the parts that were reused from the Data Science course 8. Machine Learning. The reused parts are noted in the comments as reused and kept close to the original used in the Machine Learning course. They are included as they create data frames and functions required by the further developed model.

The model was developed using the least squares estimate (LSE) approximation and regularization with data on mean rating, movie bias, user bias, genre bias, time stamp bias, and movie's year of publication bias. Other models were experimented with (e.g. caret -package algorithms using the train -function as well as matrix factorization), but due to the size of the data set using any other method was not feasible from performance point of view.

2.1. Data exploration and cleaning

The original data was obtained from Grouplens and it is the MovieLens 10M dataset:

- https://grouplens.org/datasets/movielens/10m/

- http://files.grouplens.org/datasets/movielens/ml-10m.zip

The required libraries and the data set were loaded, the edx and validation data sets were created as per the instructions given in the Capstone Movielens project assignment submission brief.

In the data exploration and cleaning the columns of interest where following:

- Genres,

- Timestamp,

- Year of publishing the movie extracted from the the title -column.

The reason for choosing these columns was that they where not included in the prediction model build in the Machine Learning -course, but there were hints that these could have an impact on a prediction models preformamnce.

In this part the training and test sets needed in the model development were created. (Technical note: Libraries required by the code and the report generation were loaded at the beginning of the .rmd file.)

2.1.1 Analyzing the genres -column data

The number of unique combinations of genres in the genres -column: [1] 797

The least common unique combinations only have few entries. Where as the most common once have hundreds of thousands of entries.

The first the least common genres combinations:

genres	count
Action|Animation|Comedy|Horror	2
Action|War|Western	2
Adventure|Fantasy|Film-Noir|Mystery|Sci-Fi	2
Adventure|Mystery	2
Crime|Drama|Horror|Sci-Fi	2
Documentary|Romance	2
Drama|Horror|Mystery|Sci-Fi|Thriller	2
Fantasy|Mystery|Sci-Fi|War	2
Action|Adventure|Animation|Comedy|Sci-Fi	3
Horror|War|Western	3

The most common genres combinations:

genres	count
Drama	733296
Comedy	700889
Comedy|Romance	365468
Comedy|Drama	323637
Comedy|Drama|Romance	261425
Drama|Romance	259355
Action|Adventure|Sci-Fi	219938
Action|Adventure|Thriller	149091
Drama|Thriller	145373
Crime|Drama	137387

When analyzing the genre types column, there were two genres that were outside the standard categories namely: IMAX and "no genres listed". You can see them in the bottom of the following list.

	genres
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994
Horror	691485
Mystery	568332
War	511147

	genres
Animation	467168
Musical	433080
Western	189394
Film-Noir	118541
Documentary	93066
IMAX	8181
(no genres listed)	7

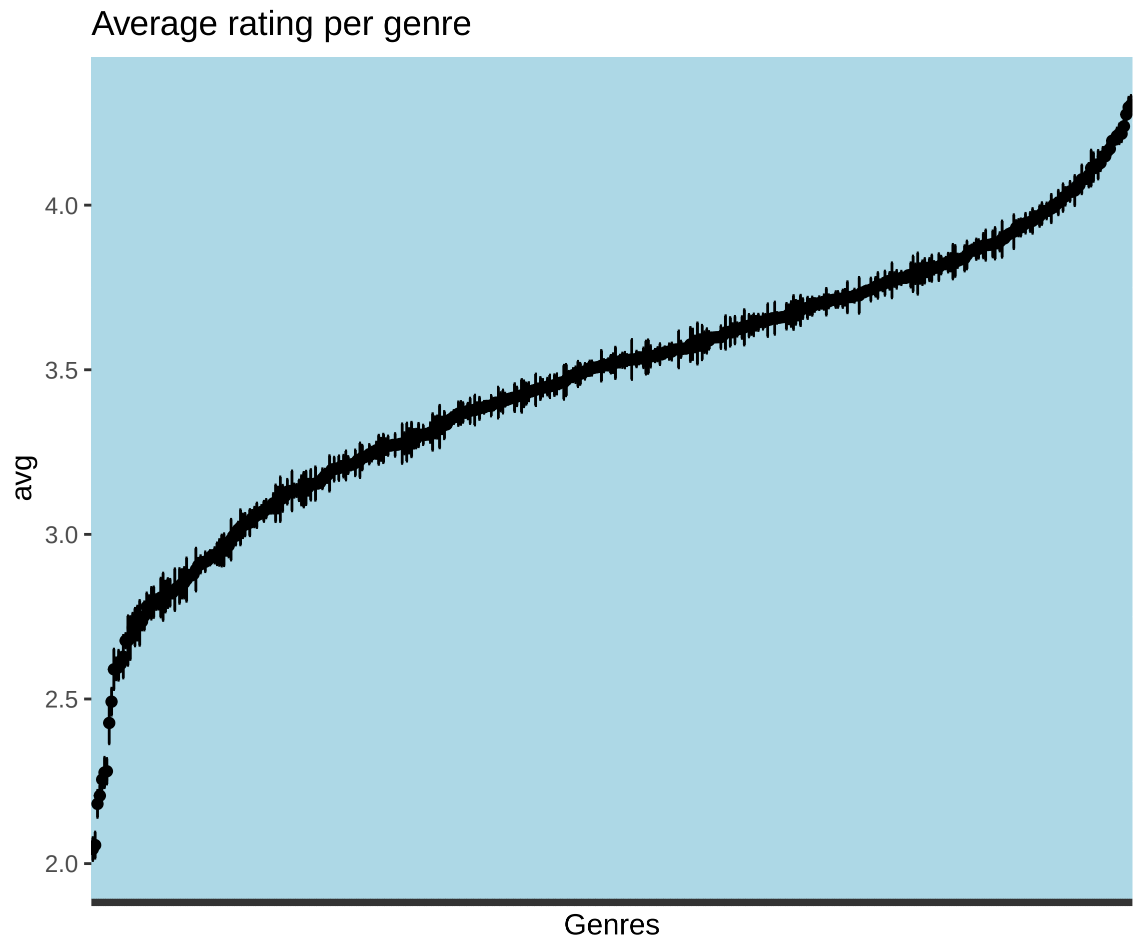
Next we needed to understand which kinds of movies are using the non-standard categories.

movieId	title	genres
8606	Pull My Daisy (1958)	(no genres listed)
8606	Pull My Daisy (1958)	(no genres listed)
8606	Pull My Daisy (1958)	(no genres listed)
8606	Pull My Daisy (1958)	(no genres listed)
8606	Pull My Daisy (1958)	(no genres listed)
8606	Pull My Daisy (1958)	(no genres listed)
8606	Pull My Daisy (1958)	(no genres listed)
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX
4460	Encounter in the Third Dimension (1999)	IMAX

Conclusion of the analysis was that there are very few "IMAX" and "(no genres listed)" and both fall under the category "Short movies".

To improve the data analysis they were both updated to a new category "Short".

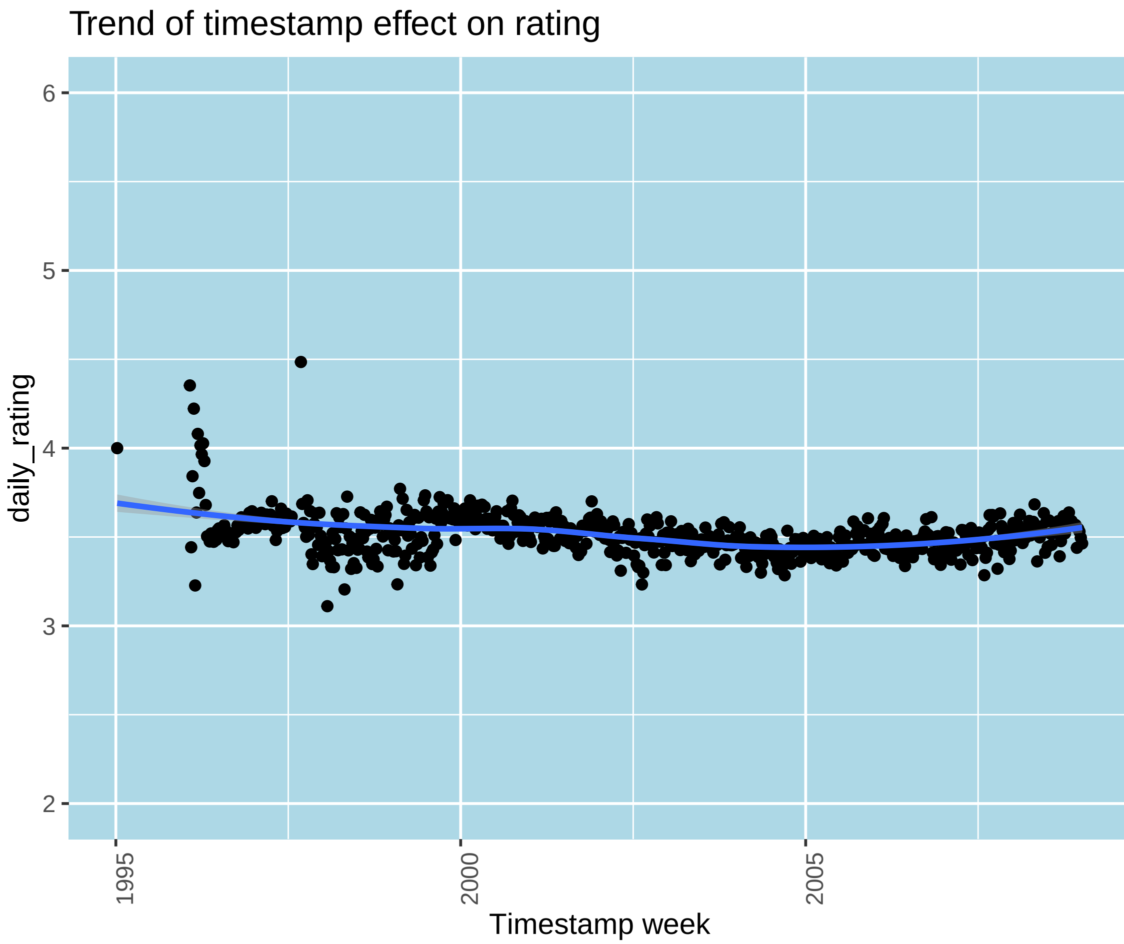
After creating the new version of the genres into a new column (genres_mod), I analyzed the average ratings per genre. Even without seeing the specific genre combination names in the plot below, you can see that there are differences between the genres and the error bars do not overlap.



From this analyses you could already make a conclusion that movie ratings seem to be impacted by the genres (or the combinations of genres) they were tagged with. So, the genres column was included in the model development.

2.1.2 Analyzing timestamp -column data

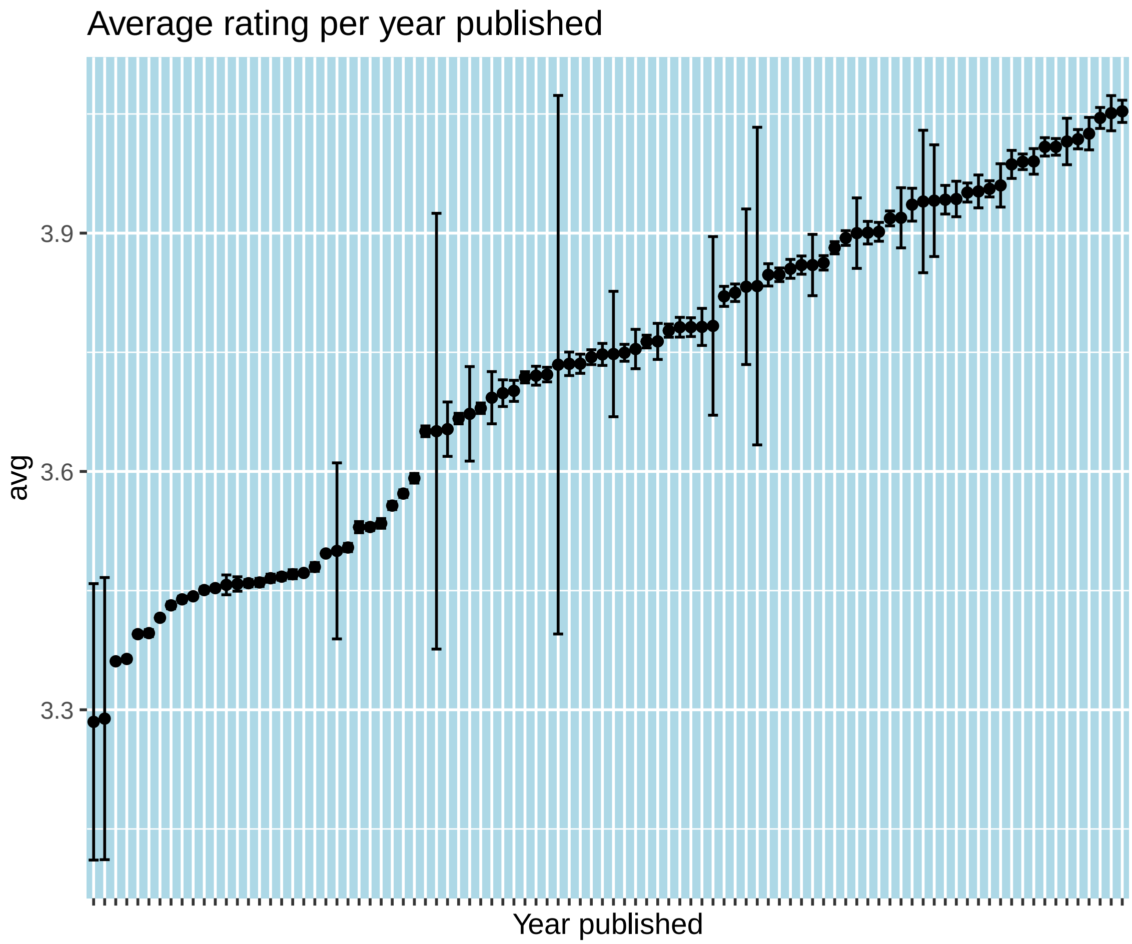
In order to analyze the timestamp impact, it was converted into a rounded week the rating was given. This consolidated all the ratings given within a single week into to a week. The plot below shows the results.



The conclusion from the plot above was that the time stamp data has a weak impact on the rating given. So, the time stamp column was added to the model development and therefore, a new column was added (date_w) to enable easier model development.

2.1.3 Analyzing the year a movie was published

In order to conduct analysis on the movie's year of publishing, some data manipulation was required in form of string processing. A new column (year_pub) was created to store the year a movie was published. The data was pulled from the title field. The plot below shows the average rating per year of publishing.The plot includes also the error bars, and without few exceptions the error bars do not overlap much.



The conclusion from the plot was that there seems to be a trend that the older the movie was, the lower the rating is. This justified the inclusion of the new the column (year_pub) in the model development.

2.2 Creating the training and test sets for model development

Next the edx data set was split into training and test data sets to be used in the model development. I split the edx data set into a 20% test set and 80% training set to ensure enough data volume was available for the training as well as for the testing of the model during development.

2.3 The model development

This section focuses on the process of developing the Movielens recommendation model.

The model was developed using the least squares estimate (LSE) approximation and regularization with data on mean rating, movie bias, user bias, genre bias, time stamp bias and movie's year of publication bias. Other models were experimented with (e.g. caret -package algorithms using the train -function as well as matrix factorization), but due to the size of the data set using any other method was not feasible from performance point of view.

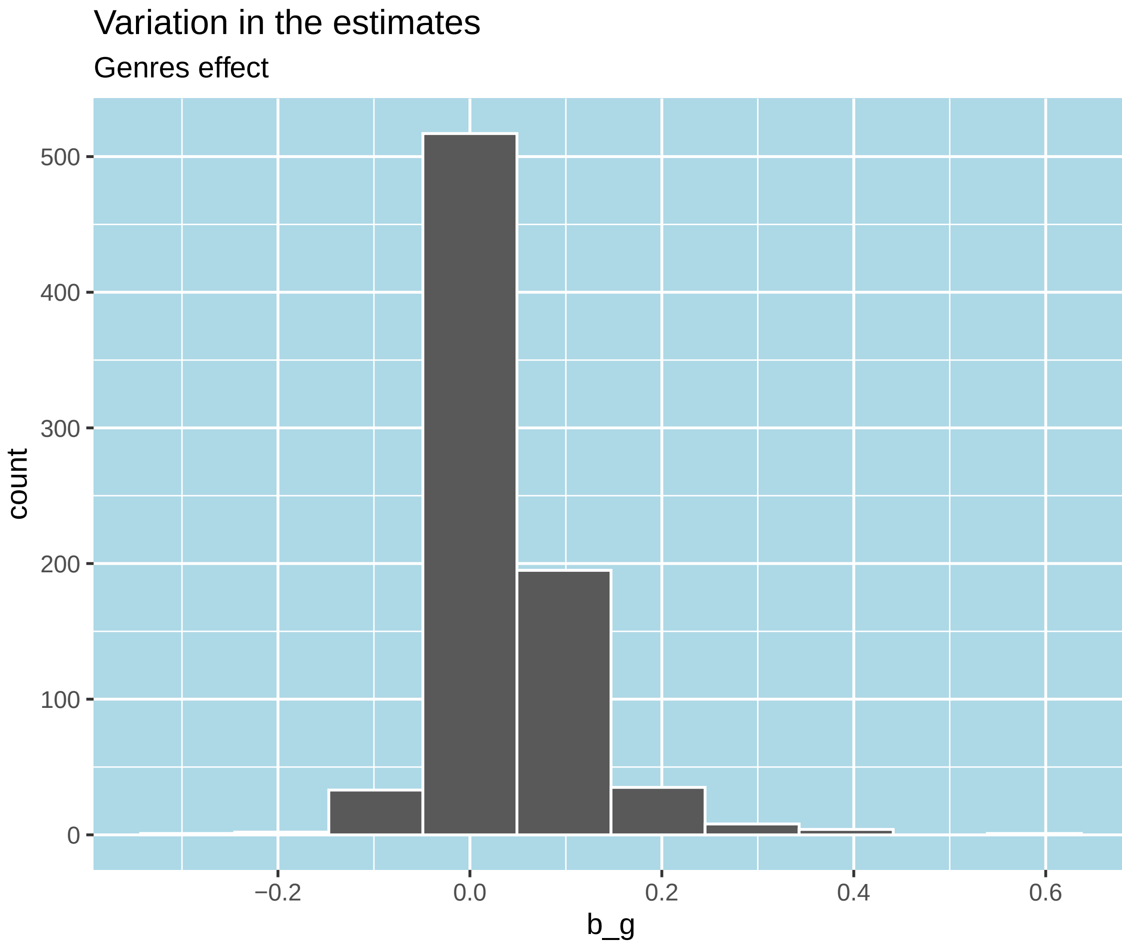
The parts that were reused from the Data Science course 8. Machine Learning are commented in the code comments. The RMSE function and storing of the RMSE results followed the same conventions as used in the previous course.

The reused data frames and variables created in the course 8 version of the Movielens recommendation method that are required to further develop the model, also followed the same coding conventions.

2.3.1 Building on the model with genre bias

The first new bias added to the model developed in the course 8 was genre bias, or in other words the genre effect, on the ratings of the movies. As we analyzed in the section 2.1.1 the average ratings given to a movie differ based on the genre (or combination of genres) that have been linked to the movie.

So, I analyzed how the LSE approximation changed, when genre bias was added to the mix in addition to the original mu (average rating), movie bias and user bias. The diagram below highlights the variations in the estimates.

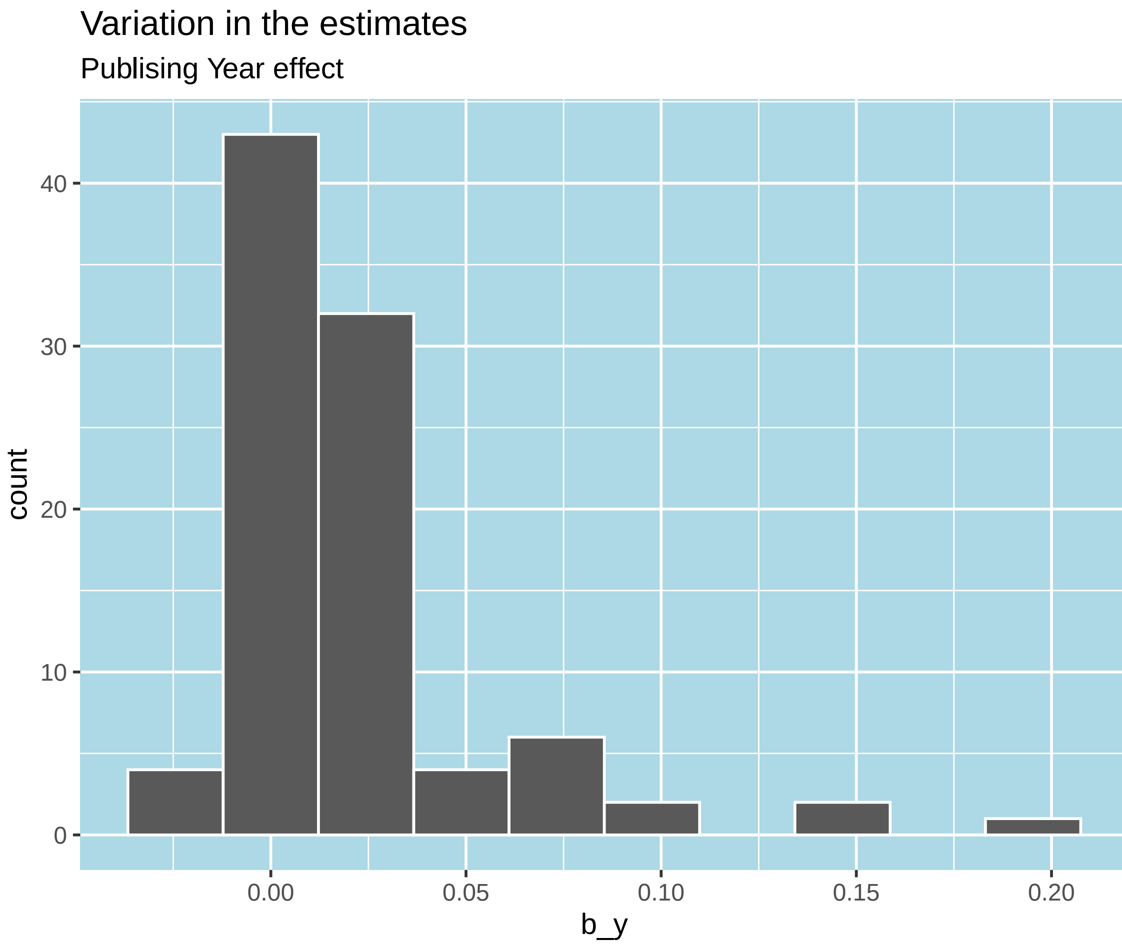


Computing the combination of the genre, movie and user effects prediction and its RMSE resulted in an improvement in the residual mean squared error (RMSE) compared to the one achieved by the models developed in the course 8. Machine Learning. You can see the RMSE results in the table below. However, the improvement is not quite where it should be to achieve the RMSE target set for this project.

method	RMSE
Target below	0.8649000
Genres, Movie and User effect	0.8655943

2.3.2 Building on the model with year of publishing the movie bias

Next the publishing year's bias (or effect) on the movie rating was added to the model. The results of the variation in the estimates is plotted below.



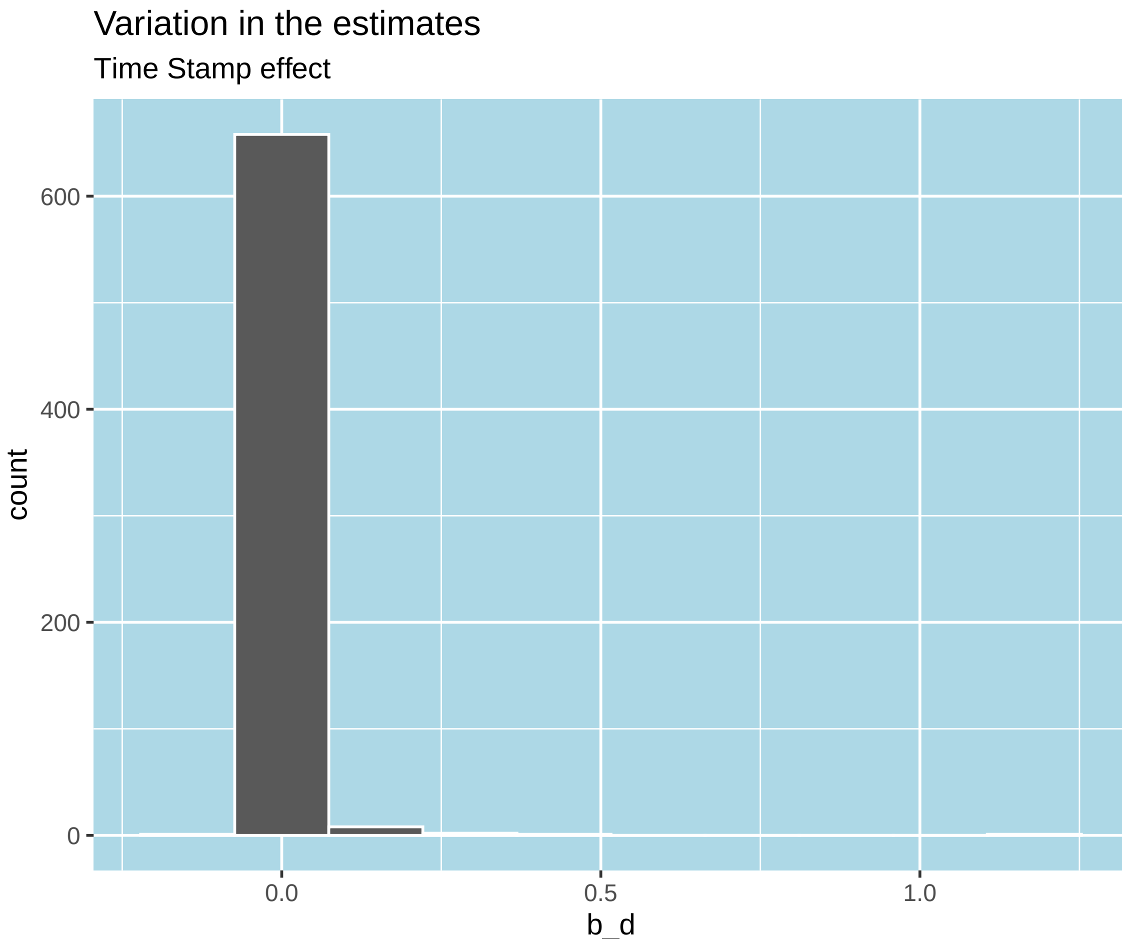
Based on the plot above variation in the average ratings can be observed. So, next the RMSE for the prediction model was calculated.

method	RMSE
Target below	0.8649000
Genres, Movie and User effect	0.8655943
Publishing Year, Genres, Movie and User effect	0.8654192

The previous and improved models RMSE results can be observed in the table above. We can see a little bit of improvement in the RMSE, but still not enough to reach the targeted RMSE level.

2.3.3 Building on the model with the timestamp of the rating give to a movie -bias

Next the he timestamp of the rating give to a movie -bias was added to the model. The results of the variation in the estimates is plotted below.



Based on the plot above variation in the average ratings just barely exists, which is why I initially left it out of the final model. But since there is tiny impact on improving the RMSE, I ended up adding it to the model. You can see the RMSE for the prediction model below.

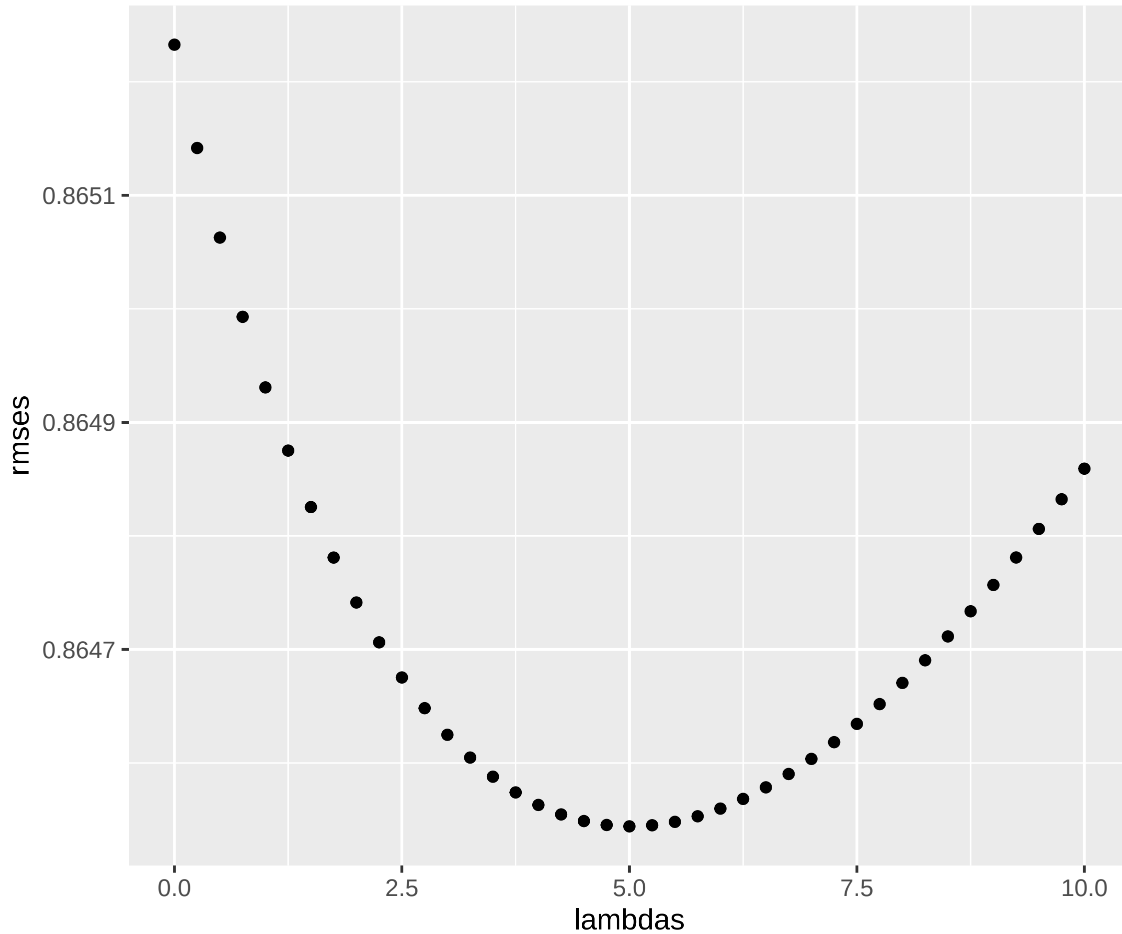
method	RMSE
Target below	0.8649000
Genres, Movie and User effect	0.8655943
Publishing Year, Genres, Movie and User effect	0.8654192
Timestamp, Publishing Year, Genres, Movie and User effect	0.8652327

2.3.4 Using Regularization

Next I tried regularization on the the least squares estimate (LSE) model to see if it had an impact on rating prediction RMSE. Regularization was used to improve the model performance in the course 8 also, but with fewer biases. Regularization was used on the model that had the additional biases from genres, publication year and timestamp.

As we know regularization permits us to penalize large estimates that are formed using small sample sizes. I needed to test out how penalized least squares behaves in the project's data set (edx) with the least squared estimate approximation model including movie, user, genre and publishing year biases.

I used (sapply) cross validation in choosing the penalty term. The plot below visualizes the lambda values, i.e. the penalty terms using cross validation, over the RMSEs.



The minimum lambda value is: [1] 5

The table below shows the improvement in the prediction's RMSE with earlier predictions' RMSE as comparison points:

method	RMSE
Target below	0.8649000
Genres, Movie and User effect	0.8655943
Publishing Year, Genres, Movie and User effect	0.8654192
Timestamp, Publishing Year, Genres, Movie and User effect	0.8652327
Regularized combined effects Model	0.8645442

3. The Results

This results section presents the prediction of the regularized least squares estimate (LSE) approximation model results and discusses the model performance.

The following table displays the final results of the model's performance on the validation data set. The performance is measures by calculating the residual mean squared error (RMSE) of the model developed with the edx data set and evaluating the movie rating prediction it gives on the validation data set. As the validation set has not been used to train the model, this should give a fair representation of the model's performance.

method	RMSE
Target below	0.8649000
Genres, Movie and User effect	0.8655943
Publishing Year, Genres, Movie and User effect	0.8654192
Timestamp, Publishing Year, Genres, Movie and User effect	0.8652327
Regularized combined effects Model	0.8645442
Final model on validation	0.8640543

Now we have achieved the projects target of developing a model that performs better on the validation data set than a residual mean squared error (RMSE) lower than 0.86490 as the achieved RMSE was: [1] 0.8640543

---

4. The Conclusions

In conclusion the target set for the project was achievable roughly within the time guidelines given by the Capstone course assignment brief and with the available computer memory and CPU capacity. The prediction of the regularized least squares estimate (LSE) approximation model achieved a residual mean squared error (RMSE) lower than the target.

In order to perform more elaborate analysis e.g. using the caret -package train function and the multitude of algorithms that it includes, would have required more processing capacity e.g. using cloud computing. However, as that is outside the scope of the Capstone course that was not done. If it were not for the performance issues, it would have been interesting to see how much a model based ensemble of several algorithms would have improved the predictions RMSE.

Appendix: All code for this report

For the readers of the RScript

I have to admit that producing a readable report using RMarkdown proved more time consuming than I had anticipated. Also, there were challenges in keeping the code bases same in the plain RScript and the RMarkdown file once I started to add the parts required in the report development. However, the ability to update and modify the report using Rmarkdown file are unmatched with other more manual means of creating the final version of the report.

Unfortunately, as the code bases in the RMarkdown file and the RScript started to change, they are now not identical anymore. Due to my lack of ability to get the autodependency option / parameters to work as I wanted, I had to resort to using RDS save and read -functions in the code chunks. This may confuse the reader of the code. The unintended benefit of doing so resulted in an improved ability to debug any deviations between the two versions of the RScript after splitting to RScript file and RMarkdown file. So, keep these in mind as you read the code below.

Also note that the RMarkdown file needs to be run twice to ensure that the final results are displayed in the executive summary.

Below you will find all of the code related to this report.

```r
# set global chunk options:
library(knitr)
opts_chunk$set(cache=TRUE, autodep = FALSE)

#Note to reader: SaveRDS and readRDS are used in the RMarkdown file,
#but not needed when you run the plain Rscript,
#unless you want to debug set by step

# Note: The process of loading the libraries could take a couple of minutes
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(dslabs)) install.packages("dslabs", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(tinytex)) install.packages("tinytex", repos = "http://cran.us.r-project.org")
if(!require(xfun)) install.packages("xfun", repos = "http://cran.us.r-project.org")
if(!require(latexpdf)) install.packages("latexpdf", repos = "http://cran.us.r-project.org")
if(!require(rmarkdown)) install.packages("rmarkdown", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(formatR)) install.packages("formatR", repos = "http://cran.us.r-project.org")

#Loading libraries
library(dplyr)
library(dslabs)
library(tidyverse)
library(stringr)
library(lubridate)
library(tinytex)
library(xfun)
library(latexpdf)
library(rmarkdown)
library(ggplot2)
library(formatR)
```

```r
options(digits = 7)
options(pillar.sigfig = 7)
#Note! When you knit this report for the first time comment next line out.
#After you have the pdf file created for the first time, close it and knit
#it again now without next line being commented out
readRDS("MovieLens_validation_RMSE2_2021-02-23.RDS")
# Note: Loading of the libraries could take a couple of minutes

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)

#This code is directly from the assignment instructions
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")


# when 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, sample.kind="Rounding") # if using R 3.5 or earlier, use`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)
```

```
#New code - Save to file
saveRDS(edx, file = "MovieLens_edx_2021-02-23.RDS")
saveRDS(validation, file = "MovieLens_validation_2021-02-23.RDS")

#read from file
edx<- readRDS("MovieLens_edx_2021-02-23.RDS")
# Number of unique genres
n_distinct(edx$genres)
#read from file
edx<- readRDS("MovieLens_edx_2021-02-23.RDS")

#Analyzing the genres -column data
#Analyzing volumes and if there are any anomalies

temp1 <- edx %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(count)

# The least common
slice_head(temp1, n= 10)
#read from file
edx<- readRDS("MovieLens_edx_2021-02-23.RDS")

#Analyzing the genres -column data
#Analyzing volumes and if there are any anomalies

temp1 <- edx %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
# The most common
slice_head(temp1, n= 10)

#read from file
edx<- readRDS("MovieLens_edx_2021-02-23.RDS")

#Analyzing the genre types there were two genres from outside the standard
#(IMAX and "no genres listed)
genre_types = c("Action", "Adventure", "Animation", "Children", "Comedy", "Crime",
                "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "Musical",
                "Mystery", "Romance", "Sci-Fi", "Thriller", "War", "Western",
                "IMAX", "(no genres listed)")

genres <- sapply(genre_types, function(g) {
  sum(str_detect(edx$genres, g))
})

genres <- as.data.frame(genres) %>%
  arrange(desc(genres))

slice_tail(genres , n= 20)
```

```
#read from file
edx<- readRDS("MovieLens_edx_2021-02-23.RDS")

#Analyzing which movies have no genre listed or IMAX as genre in both data sets
temp2 <- edx %>%
  filter(genres == "(no genres listed)" | genres == "IMAX") %>%
  select(movieId, title, genres) %>%
  arrange(desc(movieId))

slice_tail(temp2 , n= 21)

#=> Conclusion: As there are very few "IMAX" and "(no genres listed)" and
#both fall under the category "Short movies", they will be  updated to new
#category "Short"
#read from file
edx<- readRDS("MovieLens_edx_2021-02-23.RDS")

#creating the new fields for genres_mod
edx <- edx %>%
  mutate(genres_mod = if_else(genres == "IMAX" | genres =="(no genres listed)", "Short", genres))

#Save to file
saveRDS(edx, file = "MovieLens_edx1_2021-02-23.RDS")

#Read from file
edx<- readRDS("MovieLens_edx1_2021-02-23.RDS")

#Analyzing the average rating per genre:
temp_genres <- edx %>%
  group_by(genres_mod) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 1000) %>%
  mutate(genres_mod = reorder(genres_mod, avg)) %>%
  ggplot(aes(x = genres_mod, y = avg, ymin = avg - 2*se, ymax = avg + 2*se))  +
  geom_point() +
  geom_smooth()+
  geom_errorbar() +
  ggtitle("Average rating per genre")+
  xlab("Genres") +
  theme(
    axis.text.x = element_blank(),
    panel.background = element_rect(fill = "lightblue",
                                    colour = "lightblue",
                                    size = 0.5, linetype = "solid"),
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "lightblue"))

temp_genres
#Read from file
edx<- readRDS("MovieLens_edx1_2021-02-23.RDS")

# Analyzing the timestamp impact
```

```r
temp_d <- edx %>%
  mutate(date = as_datetime(timestamp)) %>%
  mutate(date = round_date(date, unit = "week")) %>%
  group_by(date) %>%
  summarize(daily_rating = mean(rating), n = n(), se = sd(rating)/sqrt(n())) %>%
  ggplot(aes(x = date, y = daily_rating, ymin = daily_rating - 2*se, ymax = daily_rating + 2*se)) +
  geom_point() +
  geom_smooth() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  ggtitle("Trend of timestamp effect on rating")+
  xlab("Timestamp week") +
  theme(
  panel.background = element_rect(fill = "lightblue",
                                  colour = "lightblue",
                                  size = 0.5, linetype = "solid"),
  panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                  colour = "white"),
  panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                  colour = "white")
  )

temp_d

##Conclusion: There was only a weak impact on the timestamp of when the
# rating was given. But we'll include it in to model still

##Creating a new date column (date_w) on the edx data set
edx<- edx %>%
  mutate(date_w = as_datetime(timestamp)) %>%
  mutate(date_w = round_date(date_w, unit = "week"))

#Save to file
saveRDS(edx, file = "MovieLens_edx1_2021-02-23.RDS")

#Read from file
edx<- readRDS("MovieLens_edx1_2021-02-23.RDS")

##Creating a new column for year a movie was published called year_pub

#Patterns for extracting the year of publishing
pattern1 <- "\\(\\d{4}\\)$"
pattern2 <- "\\d{4}"

#creating the new fields for year_pub
edx <- edx %>%
  mutate(year_pub = str_match(title, pattern1)) %>%
  mutate(year_pub = str_extract(year_pub, pattern2))

#Analyzing the average rating per year of publishing:
temp_year_pub <- edx %>%
  group_by(year_pub) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
```

```r
  mutate(year_pub = reorder(year_pub, avg)) %>%
  ggplot(aes(x = year_pub, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_smooth() +
  geom_errorbar() +
  theme(axis.text.x = element_blank())+
  ggtitle("Average rating per year published")+
  xlab("Year published") +
  theme(
  panel.background = element_rect(fill = "lightblue",
                                  colour = "lightblue",
                                  size = 0.5, linetype = "solid"),
  panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                  colour = "white")
  )

temp_year_pub

##Conclusion: There seems to be a trend that the older the movie the
#lower the rating. We'll include the year_pub in the model development.

#Save to file
saveRDS(edx, file = "MovieLens_edx2_2021-02-23.RDS")


#Read from file
edx<- readRDS("MovieLens_edx2_2021-02-23.RDS")

##Building the training and test data sets from the edx data set

#Set the seed to 1
# set.seed(1) if using R 3.5 or earlier
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later

#Movielens test and training sets
#Creating a 20% test set and 80% training set out of the edx data set
test_index2 <- createDataPartition(y = edx$rating, times = 1, p = 0.2,
                                   list = FALSE)
train_set <- edx[-test_index2,]
test_set <- edx[test_index2,]

#To make sure we don't include users and movies in the test set that
#do not appear in the training set, we remove these entries using
#the semi_join function
test_set  <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test set back into train set
removed <- anti_join(test_set, test_set)
train_set <- rbind(train_set, removed)
```

```r
#Save to file
saveRDS(train_set, file = "MovieLens_train_set_2021-02-23.RDS")
saveRDS(test_set, file = "MovieLens_test_set_2021-02-23.RDS")

#We use the same function as used in the course 8 to compute
#the RMSE for vectors of ratings and their corresponding predictors:
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")

## 1st model used in the course 8
#Predict the same rating for all movies regardless of user:
mu <- mean(train_set$rating)

##2nd model used in the course 8
#Modeling movie effects or movie bias by using least squares to estimate:
# I'll be using approximation of mu and biases as for performance reasons,
#I cannot use the lm function:
#fit <- lm(rating ~ as.factor(movieId), data = train_set)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

##3rd model used in the course 8:
#Computing an approximation of user effect or user bias
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#Save to file
saveRDS(mu, file = "MovieLens_mu_2021-02-23.RDS")
saveRDS(movie_avgs, file = "MovieLens_movie_avgs_2021-02-23.RDS")
saveRDS(user_avgs, file = "MovieLens_user_avgs_2021-02-23.RDS")

#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")
mu<- readRDS("MovieLens_mu_2021-02-23.RDS")
movie_avgs <- readRDS("MovieLens_movie_avgs_2021-02-23.RDS")
user_avgs <- readRDS("MovieLens_user_avgs_2021-02-23.RDS")

##Adding genre effect or genre bias to the model used in the course 8:
#Computing an approximation of movie + user + genre effects using the modified genres_mod -column
genre_avgs <-train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres_mod) %>%
  summarize(b_g = mean(rating - mu - b_i-b_u))
```

```r
#Visualization of the results
genre_avgs %>%
  ggplot(aes(b_g)) +
  geom_histogram(bins = 10, color ="white")+
  ggtitle("Variation in the estimates", subtitle = "Genres effect") +
  theme(
    panel.background = element_rect(fill = "lightblue",
                                    colour = "lightblue",
                                    size = 0.5, linetype = "solid"),
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "white")
  )
#We can see that these estimates vary.

#Save to file
saveRDS(genre_avgs, file = "MovieLens_genre_avgs_2021-02-23.RDS")

#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")
mu<- readRDS("MovieLens_mu_2021-02-23.RDS")
movie_avgs <- readRDS("MovieLens_movie_avgs_2021-02-23.RDS")
user_avgs <- readRDS("MovieLens_user_avgs_2021-02-23.RDS")
genre_avgs <- readRDS("MovieLens_genre_avgs_2021-02-23.RDS")

#Computing genre, movie and user effects prediction and RMSE:
predicted_ratings_ge <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres_mod') %>%
  mutate(pred = mu + b_i + b_u +b_g) %>%
  pull(pred)
genres_effect_RMSE <- RMSE(predicted_ratings_ge, test_set$rating)

#Create a results table (following the convention used in the course 8) with target RMSE:
rmse_results <- tibble(method = "Target below", RMSE = 0.86490)

#Adding the results as a new row to the RMSE tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(
                            method="Genres, Movie and User effect",
                            RMSE = genres_effect_RMSE))

#The results
rmse_results
##=> Conclusion: Just using the Genres -column (modified) data
#with LSE did not improve the results significantly

#Saving to file
saveRDS(rmse_results, file = "MovieLens_rmse_results_2021-02-23.RDS")

#Read from file
```

```r
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")
mu<- readRDS("MovieLens_mu_2021-02-23.RDS")
movie_avgs <- readRDS("MovieLens_movie_avgs_2021-02-23.RDS")
user_avgs <- readRDS("MovieLens_user_avgs_2021-02-23.RDS")
genre_avgs <- readRDS("MovieLens_genre_avgs_2021-02-23.RDS")
rmse_results <- readRDS("MovieLens_rmse_results_2021-02-23.RDS")

## Adding year of publishing the movie bias or effect
#Computing an approximation of year of publishing
#the movie + movie + user +
#genre effects using the modified year_pub -column
pub_year_avgs <-train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres_mod') %>%
  group_by(year_pub) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u - b_g))

#Visualization of the results
pub_year_avgs %>%
  ggplot(aes(b_y)) +
  geom_histogram(bins = 10, color ="white")+
  ggtitle("Variation in the estimates", subtitle = "Publising Year effect") +
  theme(
    panel.background = element_rect(fill = "lightblue",
                                    colour = "lightblue",
                                    size = 0.5, linetype = "solid"),
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "white")
  )

#Conclusion: Again there seems to be variation in the average ratings

#Saving to file
saveRDS(pub_year_avgs , file = "MovieLens_pub_year_avgs_2021-02-23.RDS")

#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")
mu<- readRDS("MovieLens_mu_2021-02-23.RDS")
movie_avgs <- readRDS("MovieLens_movie_avgs_2021-02-23.RDS")
user_avgs <- readRDS("MovieLens_user_avgs_2021-02-23.RDS")
genre_avgs <- readRDS("MovieLens_genre_avgs_2021-02-23.RDS")
pub_year_avgs <- readRDS("MovieLens_pub_year_avgs_2021-02-23.RDS")
rmse_results <- readRDS("MovieLens_rmse_results_2021-02-23.RDS")

#Computing year of publishing the movie, genre, movie and user effects
#prediction and RMSE:
predicted_ratings_ye <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
```

```r
  left_join(genre_avgs, by='genres_mod') %>%
  left_join(pub_year_avgs, by='year_pub') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
  pull(pred)

pub_year_effect_RMSE <- RMSE(predicted_ratings_ye, test_set$rating)

#Adding the results as a new row to the RMSE tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(
                            method="Publishing Year, Genres, Movie and User effect",
                            RMSE = pub_year_effect_RMSE))

rmse_results

#Saving to file
saveRDS(rmse_results, file = "MovieLens_rmse_results_2021-02-23.RDS")

#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")
mu<- readRDS("MovieLens_mu_2021-02-23.RDS")
movie_avgs <- readRDS("MovieLens_movie_avgs_2021-02-23.RDS")
user_avgs <- readRDS("MovieLens_user_avgs_2021-02-23.RDS")
genre_avgs <- readRDS("MovieLens_genre_avgs_2021-02-23.RDS")
pub_year_avgs <- readRDS("MovieLens_pub_year_avgs_2021-02-23.RDS")

## Adding timestamp rounded into weeks bias or effect
#Computing an approximation of year of publishing
#the movie + movie + user + genre effects +
# timestamp effects using the modified date_w -column
#Note! The timestamp rounded into weeks
timestamp_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres_mod') %>%
  left_join(pub_year_avgs, by='year_pub') %>%
  group_by(date_w) %>%
  summarize(b_d = mean(rating - mu - b_i - b_u - b_g - b_y))

#Visualization of the results
timestamp_avgs  %>%
  ggplot(aes(b_d)) +
  geom_histogram(bins = 10, color ="white")+
  ggtitle("Variation in the estimates", subtitle = "Time Stamp effect") +
  theme(
    panel.background = element_rect(fill = "lightblue",
                                    colour = "lightblue",
                                    size = 0.5, linetype = "solid"),
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "white")
  )
```

```
#Conclusion: Again there seems to be variation in the average ratings

#Saving to file
saveRDS(timestamp_avgs , file = "MovieLens_timestamp_avgs_2021-02-23.RDS")



#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")
mu<- readRDS("MovieLens_mu_2021-02-23.RDS")
movie_avgs <- readRDS("MovieLens_movie_avgs_2021-02-23.RDS")
user_avgs <- readRDS("MovieLens_user_avgs_2021-02-23.RDS")
genre_avgs <- readRDS("MovieLens_genre_avgs_2021-02-23.RDS")
pub_year_avgs <- readRDS("MovieLens_pub_year_avgs_2021-02-23.RDS")
timestamp_avgs <- readRDS("MovieLens_timestamp_avgs_2021-02-23.RDS")
rmse_results <- readRDS("MovieLens_rmse_results_2021-02-23.RDS")

#Computing year of publishing the movie, genre, movie and user effects
#prediction and RMSE:
predicted_ratings_ts <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres_mod') %>%
  left_join(pub_year_avgs, by='year_pub') %>%
  left_join(timestamp_avgs, by = 'date_w') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y + b_d) %>%
  pull(pred)

pub_timestamp_effect_RMSE <- RMSE(predicted_ratings_ts, test_set$rating)

#Adding the results as a new row to the RMSE tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(
                            method=
                              "Timestamp, Publishing Year, Genres, Movie and User effect",
                            RMSE = pub_timestamp_effect_RMSE))

rmse_results

#Saving to file
saveRDS(rmse_results, file = "MovieLens_rmse_results_2021-02-23.RDS")


#Read from file
train_set<- readRDS("MovieLens_train_set_2021-02-23.RDS")
test_set<- readRDS("MovieLens_test_set_2021-02-23.RDS")


##Regularization was used to improve the model performance in the course 8.
#Regularization permits us to penalize large estimates that are formed
# using small sample sizes
```

```r
#Testing out how penalized least squares behaves in the projects data set
#with the model including timestamp, publishing year, genre biases,
# movie bias, user bias and average rating

#Choosing the penalty terms using cross validation
lambdas <- seq(0, 10, 0.25)

#Using the same structure for the sapply as used in the course 8 to help
#follow the code
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i_tab <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u_tab <- train_set %>%
    left_join(b_i_tab, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n()+l))
  b_g_tab<- train_set %>%
    left_join(b_i_tab, by='movieId') %>%
    left_join(b_u_tab, by='userId') %>%
    group_by(genres_mod) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+l))
  b_y_tab <-train_set %>%
    left_join(b_i_tab, by='movieId') %>%
    left_join(b_u_tab, by='userId') %>%
    left_join(b_g_tab, by='genres_mod') %>%
    group_by(year_pub) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u - b_g)/(n()+l))
  b_d_tab <- train_set %>%
    left_join(b_i_tab, by='movieId') %>%
    left_join(b_u_tab, by='userId') %>%
    left_join(b_g_tab, by='genres_mod') %>%
    left_join(b_y_tab, by='year_pub') %>%
    group_by(date_w) %>%
    summarize(b_d = sum(rating - mu - b_i - b_u - b_g - b_y)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i_tab, by = 'movieId') %>%
    left_join(b_u_tab, by = 'userId') %>%
    left_join(b_g_tab, by='genres_mod') %>%
    left_join(b_y_tab, by='year_pub') %>%
    left_join(b_d_tab, by='date_w') %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y + b_d) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

#Visualizing lamdas
qplot(lambdas, rmses)

#Extracting the minimum value for the lambda
```

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

#Save to file
saveRDS(min(rmses), file = "MovieLens_reg_rmses_2021-02-23.RDS")
saveRDS(lambda, file = "MovieLens_lambda_2021-02-23.RDS")
readRDS("MovieLens_lambda_2021-02-23.RDS")

#Read from file
minrmses <- readRDS("MovieLens_reg_rmses_2021-02-23.RDS")

#Adding the results as a new row to the RMSE tibble
rmse_results <- readRDS("MovieLens_rmse_results_2021-02-23.RDS")
rmse_results <- bind_rows(rmse_results,
                          tibble(
                            method= "Regularized combined effects Model",
                            RMSE = minrmses))
rmse_results

#Saving to file
saveRDS(rmse_results, file = "MovieLens_rmse_results_2021-02-23.RDS")


##########################################################
## Verifying the model with validation data set
##########################################################

#Read from file
edx<- readRDS("MovieLens_edx2_2021-02-23.RDS")
validation<- readRDS("MovieLens_validation_2021-02-23.RDS")
l <- readRDS("MovieLens_lambda_2021-02-23.RDS")
rmse_results <- readRDS("MovieLens_rmse_results_2021-02-23.RDS")

#creating the new field for genres_modin validation data set
validation <- validation %>%
  mutate(genres_mod = if_else(genres == "IMAX" |
                                genres =="(no genres listed)",
                              "Short", genres))
##Creating a new date column on the validation data set
validation <- validation %>%
  mutate(date_w = as_datetime(timestamp)) %>%
  mutate(date_w = round_date(date_w, unit = "week"))

#Creating a new column for year movie was published
#Patterns for extracting the year of publishing
pattern1 <- "\\(\\d{4}\\)$"
pattern2 <- "\\d{4}"

#creating the new field for year_pub
validation1 <- validation %>%
  mutate(year_pub = str_match(title, pattern1)) %>%
  mutate(year_pub = str_extract(year_pub, pattern2))
```

```r
#save to file
saveRDS(validation1, file = "MovieLens_validation1_results_2021-02-23.RDS")

#Calculating mu
mu <- mean(edx$rating)

#Regularized averages for Movie effect
b_i_tab <- edx %>%
  group_by(movieId) %>%
  summarize(b_i_reg = sum(rating - mu)/(n()+l))

#Regularized averages for User effect
b_u_tab <- edx %>%
  left_join(b_i_tab, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u_reg = sum(rating - b_i_reg - mu)/(n()+l))

#Regularized averages for the movie, user and genres effects
b_g_tab <- edx %>%
  left_join(b_i_tab, by='movieId') %>%
  left_join(b_u_tab, by='userId') %>%
  group_by(genres_mod) %>%
  summarize(b_g_reg = sum(rating - mu- b_i_reg - b_u_reg)/(n()+l))

#Regularized averages for the publishing year, movie, user and genres effects
b_y_tab <-edx %>%
  left_join(b_i_tab, by='movieId') %>%
  left_join(b_u_tab, by='userId') %>%
  left_join(b_g_tab, by='genres_mod') %>%
  group_by(year_pub) %>%
  summarize(b_y_reg = sum(rating - mu - b_i_reg - b_u_reg - b_g_reg)/(n()+l))

#Regularized averages for the timestamp rounded into weeks, publishing year, movie, user and genres e
b_d_tab <- edx %>%
  left_join(b_i_tab, by='movieId') %>%
  left_join(b_u_tab, by='userId') %>%
  left_join(b_g_tab, by='genres_mod') %>%
  left_join(b_y_tab, by='year_pub') %>%
  group_by(date_w) %>%
  summarize(b_d_reg =
              sum(rating - mu - b_i_reg - b_u_reg - b_g_reg - b_y_reg)/
              (n()+l))


## Verifying the model with validation data sets
#Computing timestamp, year of publishing the movie, genre, movie
#and user effects and average rating
#prediction and RMSE:
predicted_ratings_valid <- validation1 %>%
  left_join(b_i_tab, by = 'movieId') %>%
  left_join(b_u_tab, by = 'userId') %>%
  left_join(b_g_tab, by='genres_mod') %>%
```

```
  left_join(b_y_tab, by='year_pub') %>%
  left_join(b_d_tab, by='date_w') %>%
  mutate(pred =
           mu + b_i_reg  + b_u_reg  + b_g_reg  + b_y_reg + b_d_reg) %>%
  pull(pred)


validation_RMSE <- RMSE(predicted_ratings_valid, validation1$rating)

#Adding the results as a new row to the RMSE tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(
                            method = "Final model on validation",
                            RMSE = validation_RMSE))

rmse_results

#Saving to file
saveRDS(rmse_results, file = "MovieLens_rmse_results2_2021-02-23.RDS")
saveRDS(validation_RMSE, file = "MovieLens_validation_RMSE2_2021-02-23.RDS")

readRDS("MovieLens_validation_RMSE2_2021-02-23.RDS")
```