Data Science: MovieLens Project

Manoj P

01/12/2019

Project OverView

This project is part of my Data Science Professional Course with Harvard University. This is a project, where one is required to create a movie recommendation system using the MovieLens dataset. This project trains a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set.

In October 2006, Netflix, then a service peddling discs of every movie and TV show under the sun, announced “The Netflix Prize,” a competition that lured Mackey and his contemporaries for the computer programmer equivalent of the Cannonball Run. The mission: Make the company’s recommendation engine 10% more accurate (Jackson, 2017)

This project attempts to use machine learning algorithms to derive accuracy in movie rating prediction. We train a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set.

Methodology

We use the code provided in the Edx course work to generate our datasets. We then develop our algorithm using the edx set. For a final test of our algorithm, we predict movie ratings in the validation set as if they were unknown. RMSE is then used to evaluate how close our predictions are to the true values in the validation set.

Data Dependency and preparation

We use the “MovieLens 10M Dataset” by the GroupLens research team. GroupLens is a research lab in the Department of Computer Science and Engineering at the University of Minnesota.

The data links;

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

The below code is provided by Edx as part of the course work. This will be used to create the datasets.

```
################################
# Create edx set, validation set
################################

# Note: this process 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.or
g")

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

Test and Validation Sets

We will split the MovieLens into two sets;
1. “edx” will be the training set.
2. “validation” will be used to test ratings

```
# 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)` instead
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)
```

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

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## Loading required package: RColorBrewer
```

Key Data Statistics

In order to prepare for our project, We familiarize ourself and explore key data statistics.

“edx” dataset review

```
# Review the class of "edx" dataset
class(edx)
```

```
## [1] "data.frame"
```

```
# Get a glimpse "edx" dataset
glimpse(edx)
```

```
## Observations: 9,000,055
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 83898339...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|D...
```

```
# number of movies in "edx" dataset
n_distinct(edx$movieId)
```

```
## [1] 10677
```

1.We note that “edx” is a data frame used to store tabular data.
2.We can see that there are 6 columns and 9,000,055 rows in the “edx” dataset.
3.We get a glimpse of the “edx” dataset. This also provides us detail of each of the vareable and it’s type.
4.There are 10,677 movies in the “edx” dataset.

Validation dataset review

```
# review the class of "validation" dataset
class(validation)
```

```
## [1] "data.frame"
```

```
# get a glimpse "validation" dataset
glimpse(validation)
```

```
## Observations: 999,999
## Variables: 5
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5...
## $ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 4...
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 86824564...
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Hom...
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Child...
```

```
# number of movies in "validation" dataset
n_distinct(validation$movieId)
```

```
## [1] 9809
```

1.We note that “validation” is a data frame used to store tabular data.
2.We can see that there are 5 columns and 999,999 rows in the “validation” dataset.
3.We get a glimpse of the “validation” dataset. This also provides us detail of each of the vareable and it’s type.
4.There are 9,809 movies in the “validation” dataset.

Data pre-processing and exploratory analysis

```
#RMSE (root mean squre error) define a function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
# Modify the year as a column in the edx & validation datasets
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
# Modify the genres variable in the edx & validation dataset (column separated)
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")
```

Review Movie Genres and categories

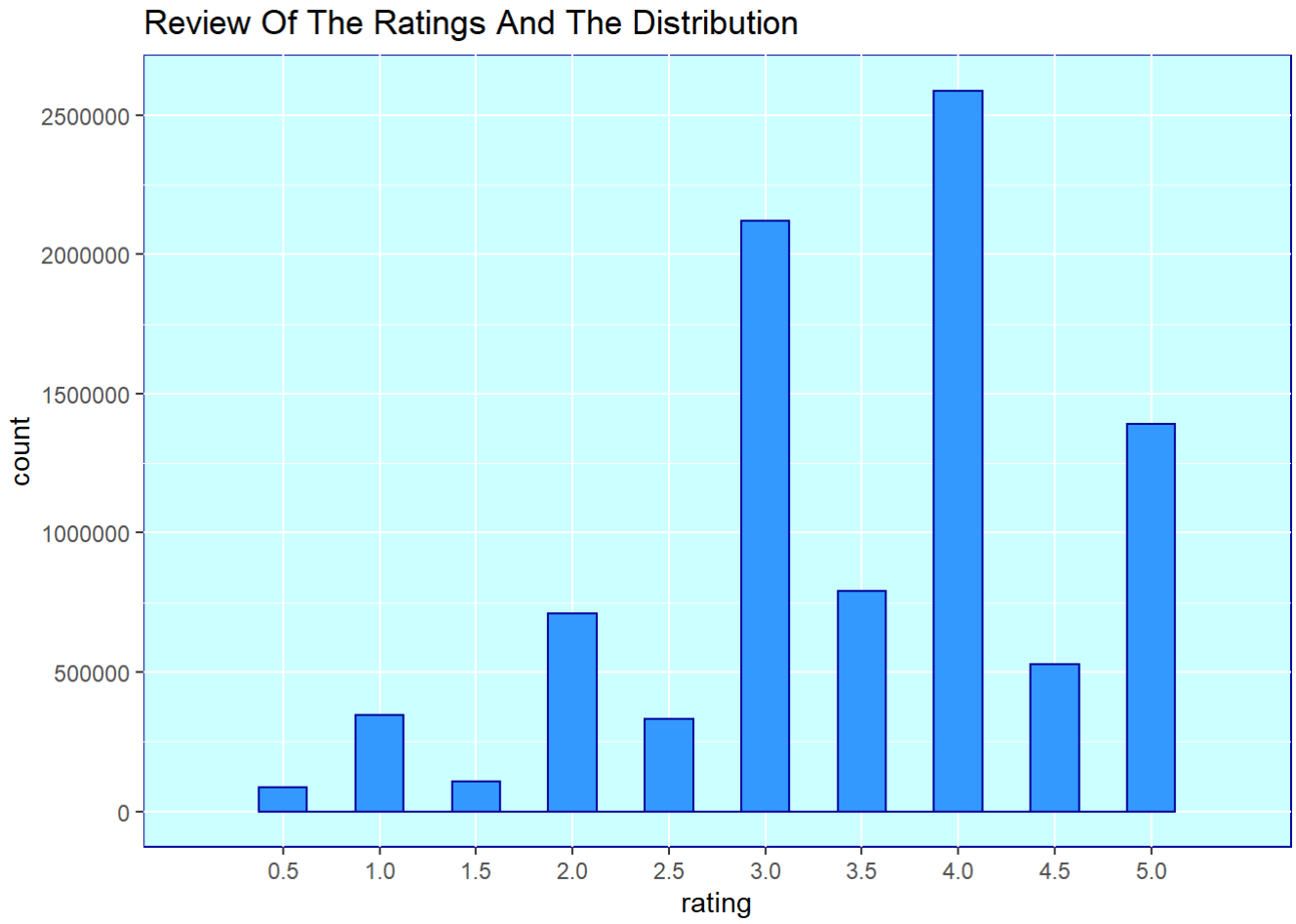
```
#list movie genres ratings in descending order
split_edx%>%
group_by(genres) %>%
summarize(count = n()) %>%
arrange(desc(count))
```

genres <chr>	count <int>
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994

Above is a list of movie genres and categories

Review of ratings and the distribution

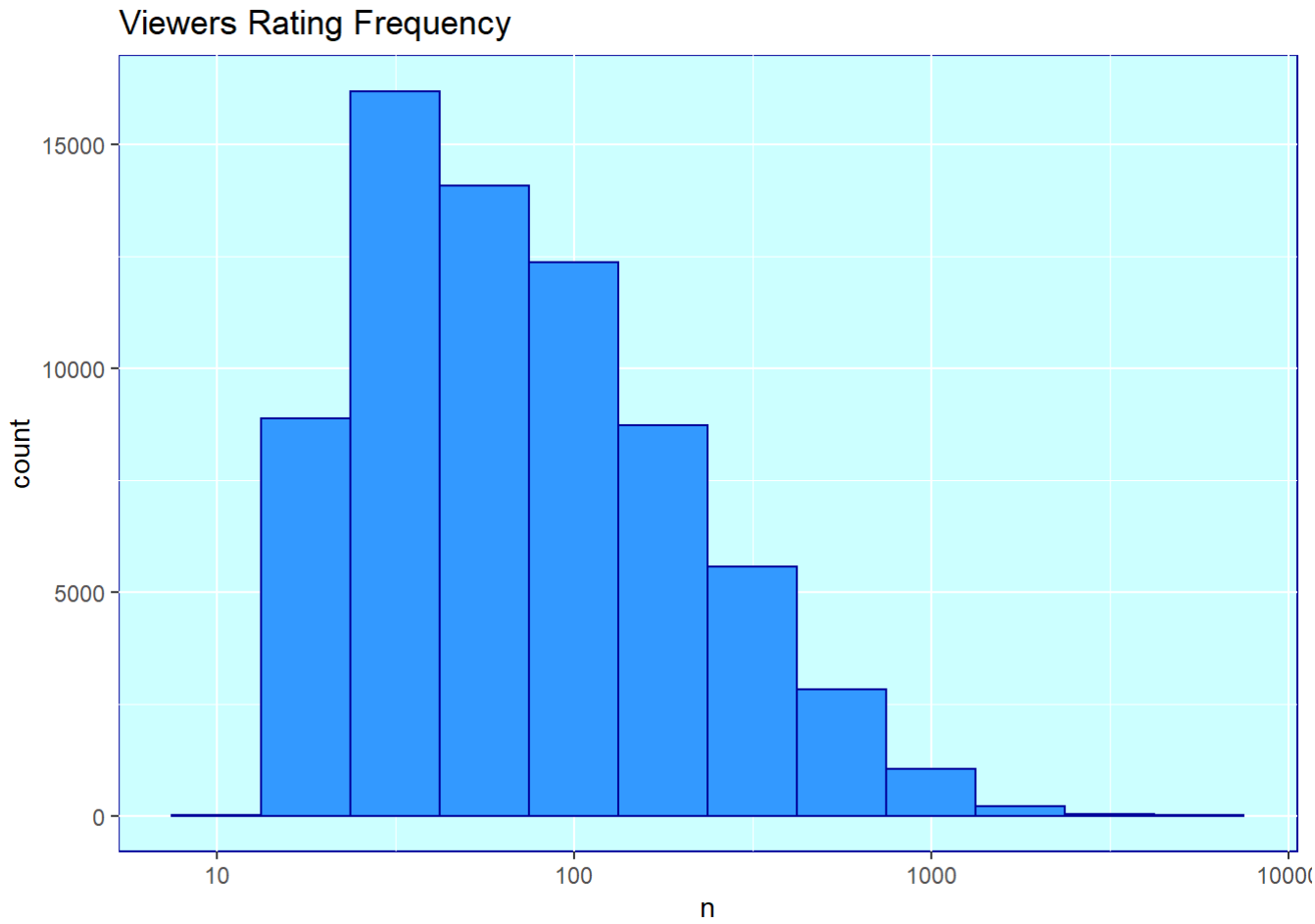
```
edx %>%
  ggplot(aes(rating))+
  (theme(panel.background = element_rect(fill = '#CCFFFF', colour = '#000099')))+
  (geom_histogram(fill="#3399FF", binwidth = 0.25, color = "#000099"))+
  (scale_x_discrete(limits = c(seq(0.5,5,0.5))))+
  (scale_y_continuous(breaks = c(seq(0, 3000000, 500000))))+
  (ggtitle("Review Of The Ratings And The Distribution"))
```



Above shows a distribution summary of how various viewers rate the movies. It shows a strong trend of users giving a rating of 4, 3 and 5. Low ratings are less observed.

Viewers Rating Frequency

```
edx %>% count(userId) %>%
  ggplot(aes(n))+
  (theme(panel.background = element_rect(fill = '#CCFFFF', colour = '#000099')))+
  (geom_histogram(fill="#3399FF", binwidth = 0.25, color = "#000099"))+
  (scale_x_log10()) +
  (ggtitle("Viewers Rating Frequency"))
```



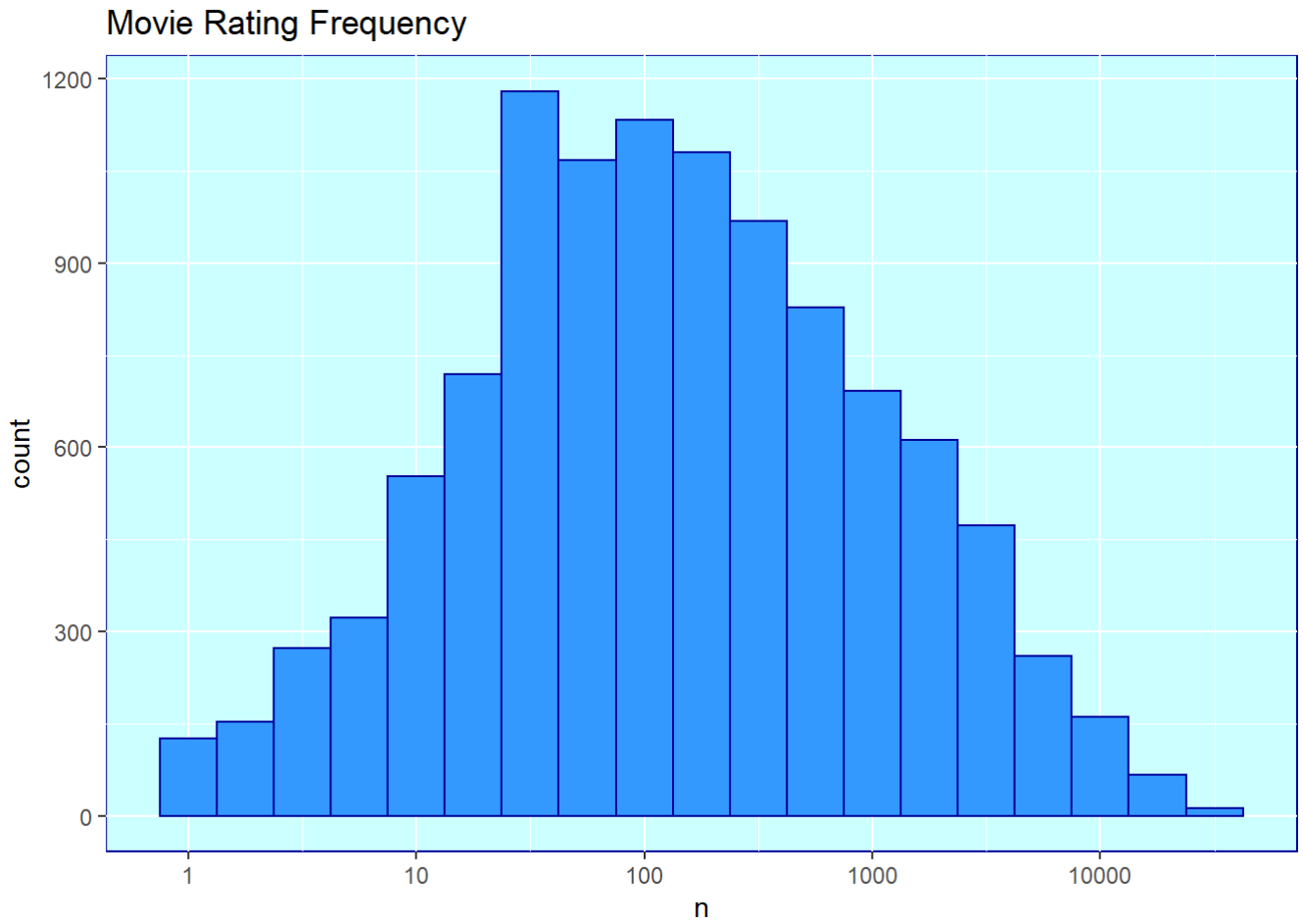
The above shows a breakdown of viewers and their ratings. Some viwers have rated more movies than others and this can have significant impact on the predictions.

Note:

1. We note that all the movies are not equally rated. Some have been rated many times and others very few.
2. Some viewers will have a more favorable reviews than others. This is a matter of opinion.
3. Need to consider viewers opinions over time and their ratings

Movie Rating Frequency

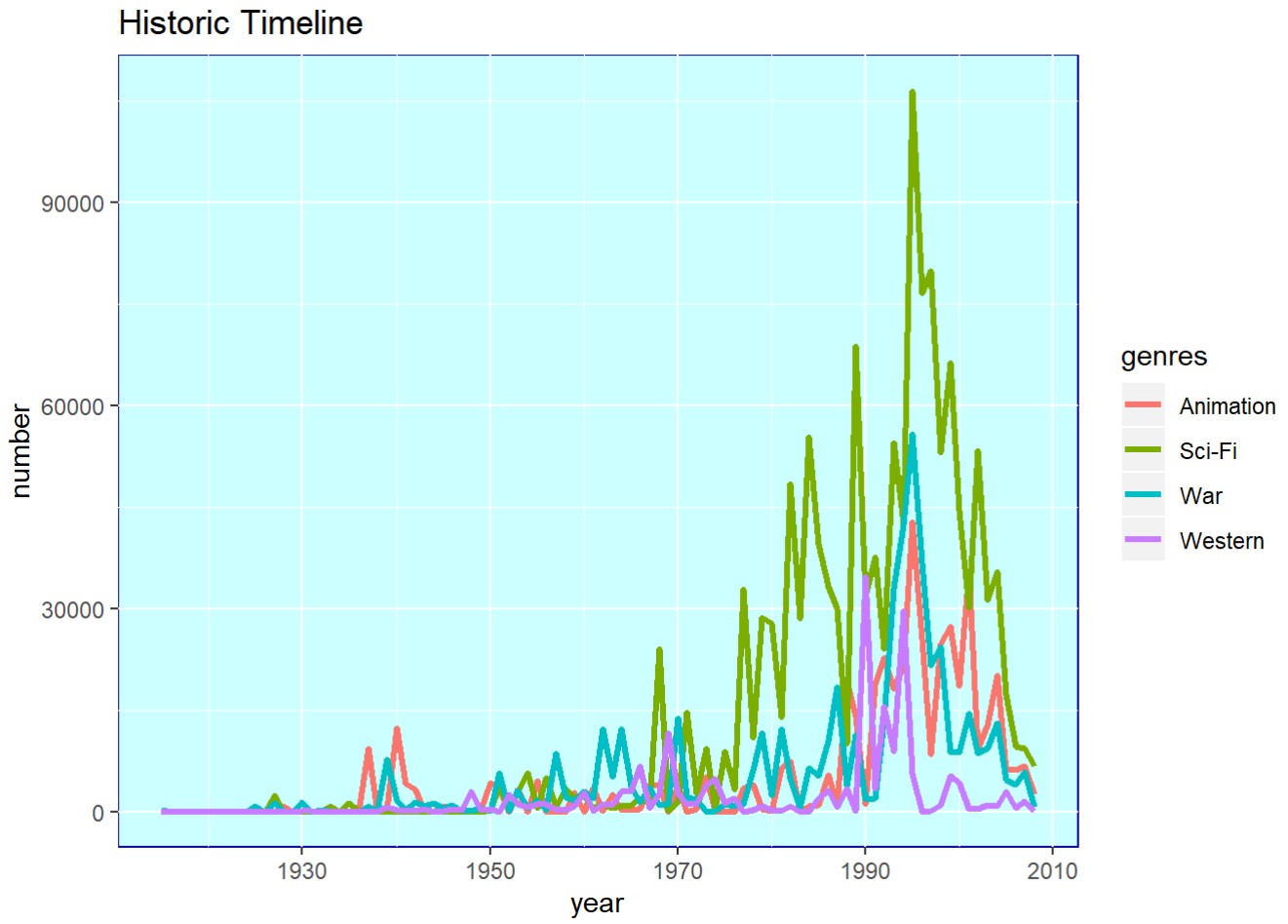
```
edx %>% count(movieId) %>%
  ggplot(aes(n))+
  (theme(panel.background = element_rect(fill = '#CCFFFF', colour = '#000099')))+
  (geom_histogram(fill="#3399FF", binwidth = 0.25, color = "#000099"))+
  (scale_x_log10()) +
  (ggtitle("Movie Rating Frequency"))
```



The above shows the spread of how many times they have been reviewed. Some movies have only been rated once. Regularization and penalty term will need to be considered in this project.

Historic Timeline view of genres

```
genres_timeline <- split_edx %>%
# remove missing data
  na.omit() %>%
# choose the columns
  select(movieId, year, genres) %>%
# represents genres as factors
  mutate(genres = as.factor(genres)) %>%
# group by year & genre
  group_by(year, genres) %>%
# number count
  summarise(number = n()) %>%
# insert any missing years/genres
  complete(year = full_seq(year, 1), genres, fill = list(number = 0))

# Genres vs year; 4 genres are chosen for readability: animation, sci-fi, war and western mov
ies.
genres_timeline %>%
  filter(year > 1910) %>%
  filter(genres %in% c("War", "Sci-Fi", "Animation", "Western")) %>%
  ggplot(aes(x = year, y = number)) +
  (theme(panel.background = element_rect(fill = '#CCFFFF', colour = '#000099')))+
  geom_line(size=1.2,aes(color=genres)) +
  scale_fill_brewer(palette = "Dark2")+
  (ggtitle("Historic Timeline"))
```

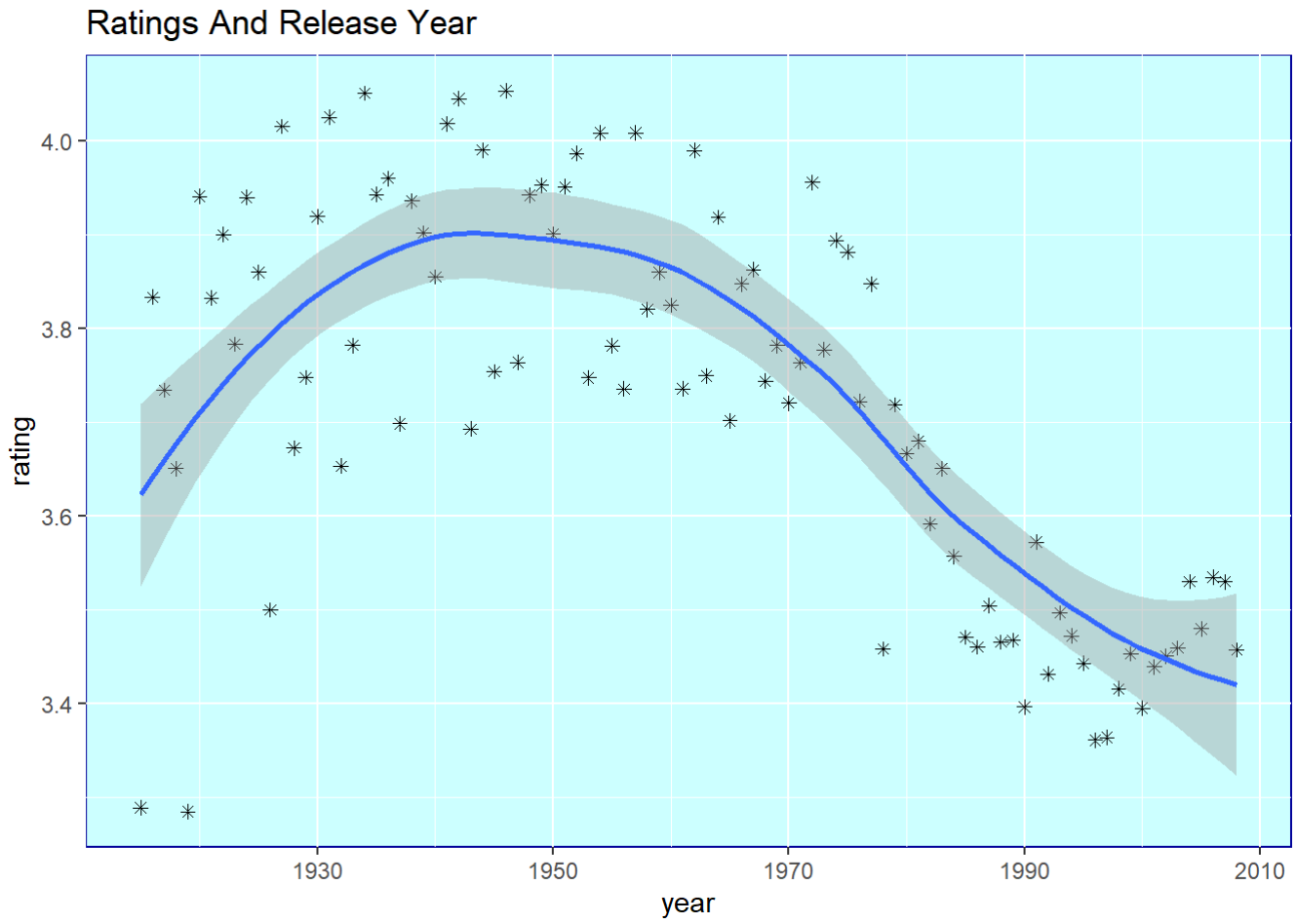


The above shows the evolution and tastes and moview preferences overtime.

Comparison Of Ratings Over The Release Year

```
edx %>% group_by(year) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year, rating)) +
  theme(panel.background = element_rect(fill = '#CCFFFF', colour = '#000099'))+
  geom_point(shape=8)+
  geom_smooth()+
  ggtitle("Ratings And Release Year")
```

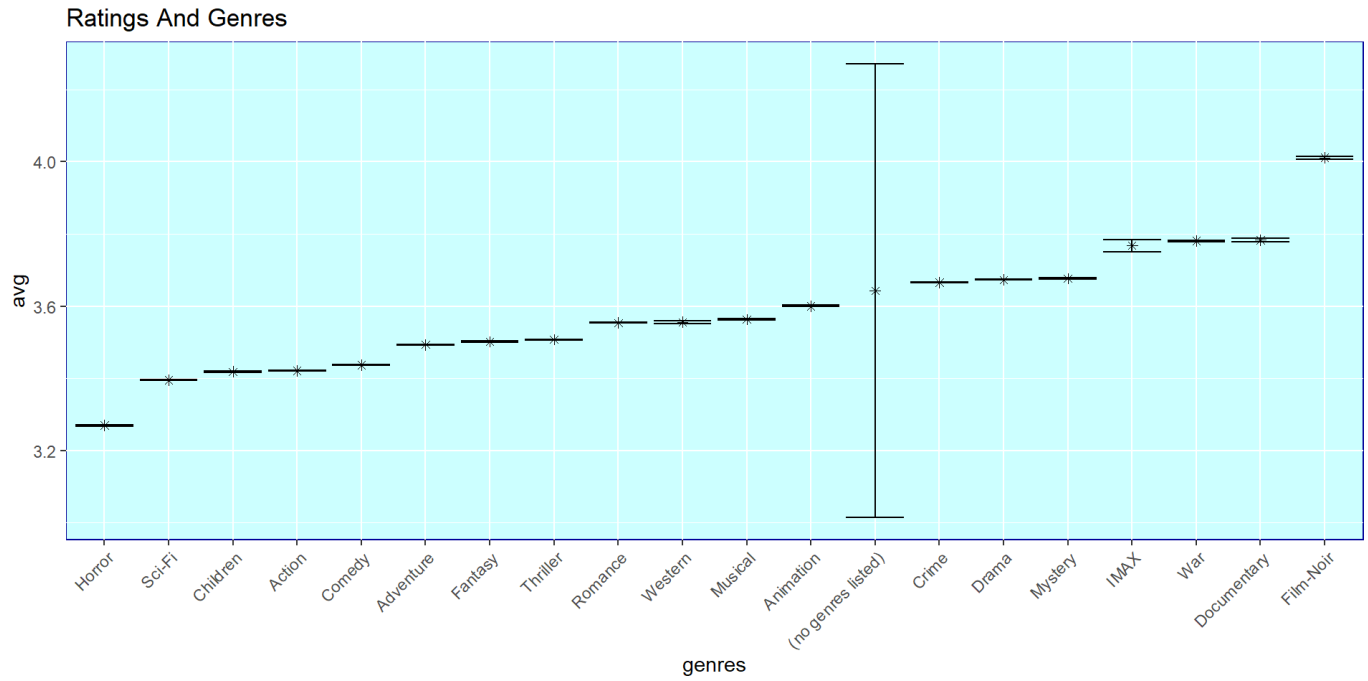
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The above shows the review of ratings against the release year. It shows that in the earlier years the ratings were higher than recent times. This could be due to several reasons e.g There were lesser number of movies in earlier years and more satisfaction derived by viewers compared to recent years where there is a larger range of movies.

Comparison Of Ratings With Genres

```
split_edx %>% group_by(genres) %>%
summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
mutate(genres = reorder(genres, avg)) %>%
ggplot(aes(x = genres, y = avg, ymin = avg - 1.5*se, ymax = avg + 1.5*se)) +
theme(panel.background = element_rect(fill = '#CCFFFF', colour = '#000099'))+
geom_point(shape=8) +
geom_errorbar() +
theme(axis.text.x = element_text(angle = 45, hjust = 1))+
ggtitle("Ratings And Genres")
```



The above shows viewers choices of the categories of the movies they like. Highly preferred are on the right-hand side of the plot; film noir, documentary, etc Less preferred are on left-hand side of the plot; horror, sci-fi, etc

Model Design

We calculate the Root Mean Square Error (RMSE) on several options. Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results (www.statisticshowto.datasciencecentral.com/rmse/).

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

This becomes our basis of calculation. The lower the RMSE the more accurate is out model.

For this project we will calculate RMSE using the below;

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

Model 1: Simple Mean

By using the average/mean ratings of the datasets, this models predicts same ratings for all movies. We will use the argument “mu” which is a number indicating the true value of the mean.

```
    mu <- mean(edx$rating)
    mu
```

```
## [1] 3.512465
```

We now use this “mu” to predict using the simple mean.

```
model1_rmse <- RMSE(validation_CM$rating, mu)
model1_rmse
```

```
## [1] 1.061202
```

Below is a summary using the simple mean model 1

```
    rmse_summary <- data_frame(method = "Model 1: Simple Mean", RMSE = model1_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
    rmse_summary %>% knitr::kable()
```

method	RMSE
Model 1: Simple Mean	1.061202

Our next challenge is to try and improve on the above RMSE to a lower result.

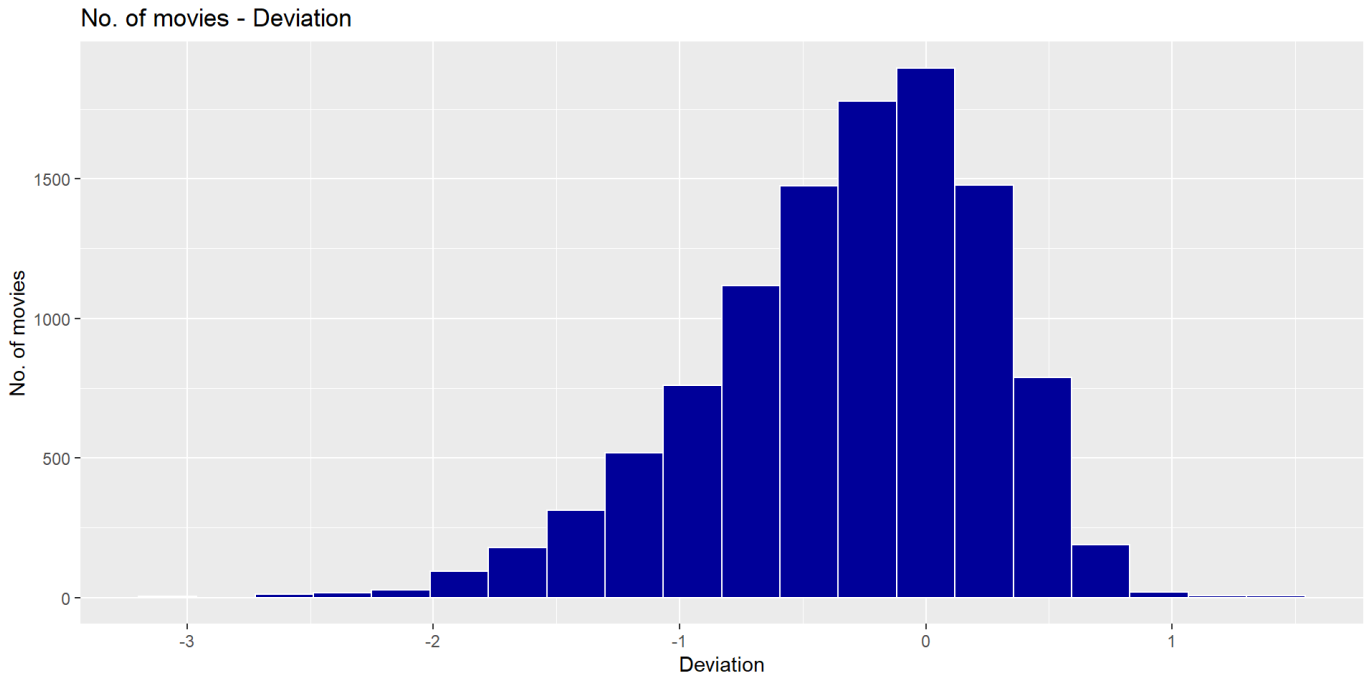
Model 2: Movie Ratings Impact

By using the average/mean ratings of the datasets, this models predicts same ratings for all movies. We will use the argument “mu” which is a number indicating the true value of the mean.

As we have seen earlier that some movies are rated higher than the others. This can be due to popularity reasons. Some movies are highly popular than the others. Popular movies will get a higher ratings where as the unpopular ones will get lower ratings. We calculate the deviation of movies mean ratings from the total mean.

As below, left side shows that more movies have negative effects. This can be considered as penalty mobie impact.

```
movie_mean <- edx %>%
  group_by(movieId) %>%
  summarize(Deviation = mean(rating - mu))
  movie_mean %>%
  qplot(Deviation, geom ="histogram", bins = 20, data = ., fill = I("#000099"), color = I("wh
ite"),
ylab = "No. of movies", main = "No. of movies - Deviation")
```



Our predictions should improve once we consider this model.

```
predicted_ratings <- mu +  validation %>%
  left_join(movie_mean, by='movieId') %>%
  pull(Deviation)
model2_rmse <- RMSE(predicted_ratings, validation_CM$rating)
rmse_summary <- bind_rows(rmse_summary,
            data_frame(method="Model 2: Movie Raings Impact",
                                    RMSE = model2_rmse ))%>% distinct
rmse_summary %>% knitr::kable()
```

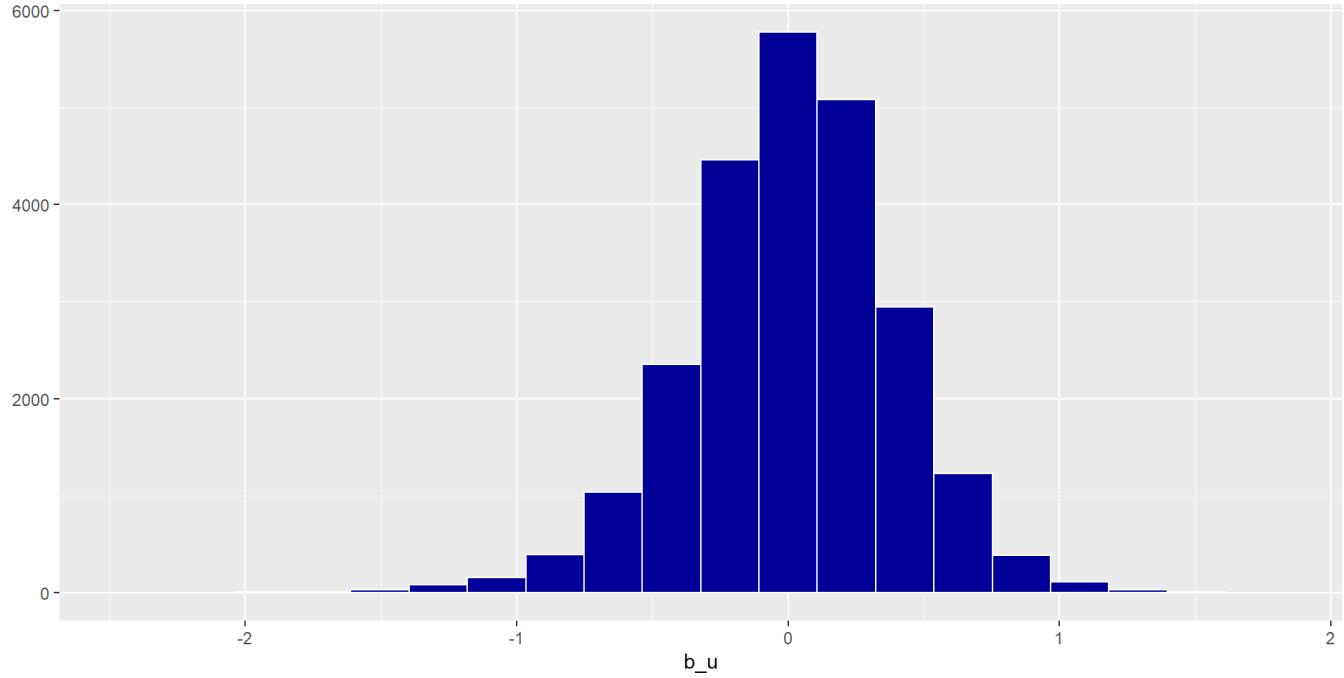
method	RMSE
Model 1: Simple Mean	1.0612018
Model 2: Movie Raings Impact	0.9439087

In the above we observe an improvement in RMSE from 1.0612018 to 0.9439087.

Model 3: Movie with user effect

We calculate the average rating for“mu”, for those that have rated over 100 movies, said penalty term user effect. Users affect the ratings favourably or unfavourably.

```
user_avgs<- edx %>%
  left_join(movie_mean, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu -Deviation))
user_avgs%>% qplot(b_u, geom ="histogram", bins = 20, data = ., fill = I("#000099"), color =
 I("white"))
```



We can see above, there is a huge variance as viewers are very disappointed or very happy with all movies. We need to further improve our modelling.

```
#Calculate approx mu & b_i, estimate b_u
viewer_means <- edx %>%
  left_join(movie_mean, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - Deviation))
```

```
predicted_ratings <- validation%>%
  left_join(movie_mean, by='movieId') %>%
  left_join(viewer_means, by='userId') %>%
  mutate(pred = (mu + Deviation + b_u)) %>%
  pull(pred)
model3_rmse <- RMSE(predicted_ratings, validation_CM$rating)
rmse_summary <- bind_rows(rmse_summary,
                          data_frame(method="Model 3: Movie with user effect",
                                     RMSE = model3_rmse))%>% distinct
rmse_summary %>% knitr::kable()
```

method	RMSE
Model 1: Simple Mean	1.0612018
Model 2: Movie Raings Impact	0.9439087
Model 3: Movie with user effect	0.8653488

Model 4: Regularized Movie with User Impact

We can further try to improve RMSE by applying used of regulariation. We calculate the lamba to minimise the RMSE.

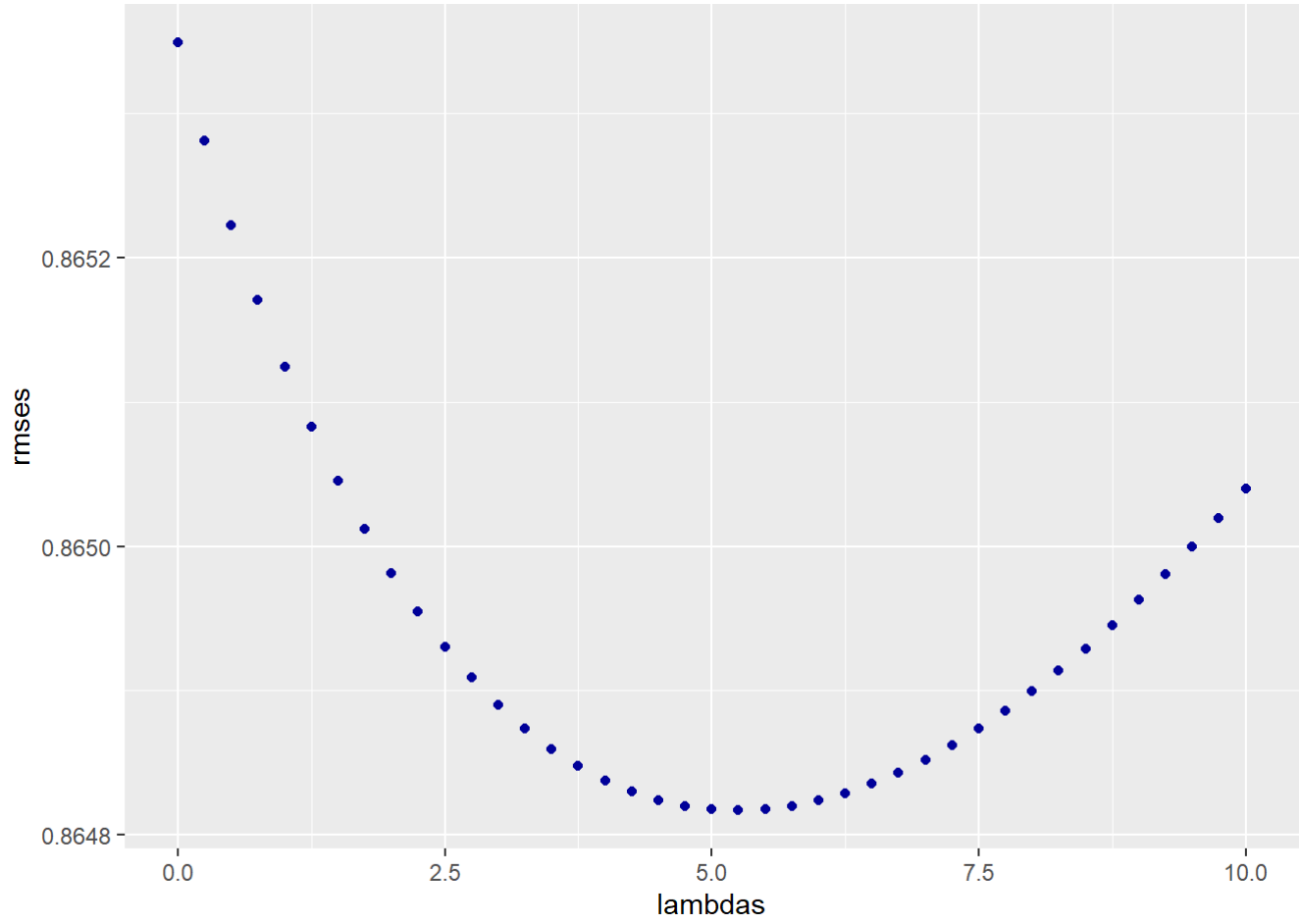
Lamdba is a measure of proportional reduction in error in cross tabulation analysis. For any sample with a nominal independent variable and dependent variable (or ones that can be treated nominally), it indicates the extent to which the modal categories and frequencies for each value of the independent variable differ from the

overall modal category and frequency, i.e. for all values of the independent variable together (www.rdocumentation.org/packages/DescTools/versions/0.99.30/topics/Lambda)

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

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

  mu <- mean(edx$rating)

  Deviation <- edx %>%
    group_by(movieId) %>%
    summarize(Deviation = sum(rating - mu)/(n()+l))

  b_u <- edx %>%
    left_join(Deviation, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - Deviation - mu)/(n()+l))

predicted_ratings <-
    validation %>%
    left_join(Deviation, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + Deviation + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation_CM$rating))
})
```

The below is a plot of the RMSE against the Lambda.

```
qplot(lambdas, rmses, color = I("#000099"))
```



To get the lowest lambda value we use the which.min.

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

```
## [1] 5.25
```

5.25 is the optimal lambda.

```
rmse_summary <- bind_rows(rmse_summary,
                          data_frame(method="Model 4: Regularized Movie with User Impact",
                                     RMSE = min(rmses))) %>% distinct
rmse_summary %>% knitr::kable()
```

method	RMSE
Model 1: Simple Mean	1.0612018
Model 2: Movie Raings Impact	0.9439087
Model 3: Movie with user effect	0.8653488
Model 4: Regularized Movie with User Impact	0.8648170

We note that the Model 4: Regularized Movie with User Impact has the lowest RMSE.

Conclusion

Using the training set and validation set provided by edx, we have successively built different models for predictions;
1. Model 1: Simple Mean gave us RMSE of 1.0612018
2. Model 2: Movie Ratings Impact gave us RMSE of 0.9439087
3. Model 3: Movie with user effect gave us RMSE of 0.8653488
4. Model 4: Regularized Movie with User Impact gave us RMSE of 0.8648170

Model 4 gave us the lowest RMSE of 0.8648170. This is an improvement of 19% compared to Model 1, Simple Mean. This linear regression model with regularized effects on movies and users seems to be the appropriate model to predict ratings on the validation set. We can further improve the RMSE by adding year, genre and year to our modelling. This would take up significant hardware and computing resources.

References

• Dan Jackson – “The Netflix Prize: How a $1 Million Contest Changed Binge-Watching Forever”, (https://www.thrillist.com/entertainment/nation/the-netflix-prize (https://www.thrillist.com/entertainment/nation/the-netflix-prize)), (2017)
• Irizzary,R., “Introduction to Data Science,github page,https://rafalab.github.io/dsbook/ (https://rafalab.github.io/dsbook/)”, (2018)
• “MovieLens 10M Dataset”, https://grouplens.org/datasets/movielens/10m/ (https://grouplens.org/datasets/movielens/10m/), (2019)

Links referred to;

• https://www.rdocumentation.org/packages/tibble/versions/1.4.2/topics/glimpse (https://www.rdocumentation.org/packages/tibble/versions/1.4.2/topics/glimpse)
• http://www.sthda.com/english/wiki/colors-in-r (http://www.sthda.com/english/wiki/colors-in-r)
• https://intellipaat.com/community/14998/how-do-i-change-the-background-color-of-a-plot-made-with-ggplot2 (https://intellipaat.com/community/14998/how-do-i-change-the-background-color-of-a-plot-made-with-ggplot2)
• https://stackoverflow.com/questions/14794599/how-to-change-line-width-in-ggplot (https://stackoverflow.com/questions/14794599/how-to-change-line-width-in-ggplot)
• https://www.datanovia.com/en/blog/ggplot-point-shapes-best-tips/ (https://www.datanovia.com/en/blog/ggplot-point-shapes-best-tips/)
• https://www.statisticshowto.datasciencecentral.com/rmse/ (https://www.statisticshowto.datasciencecentral.com/rmse/)
• https://stackoverflow.com/questions/18882206/change-outline-and-fill-colors-of-histogram-with-qplot (https://stackoverflow.com/questions/18882206/change-outline-and-fill-colors-of-histogram-with-qplot)
• https://www.rdocumentation.org/packages/DescTools/versions/0.99.30/topics/Lambda (https://www.rdocumentation.org/packages/DescTools/versions/0.99.30/topics/Lambda)