Movielens Capstone Project

Chanya Limdamnern

3/27/2021

Contents

Introduction

This capstone project is the first part of PH125.9x: Data Science: Capstone course. For this project, movie recommendation system will be created using MovieLens dataset included in the dslabs package.

The goal of this project is to develop model for movie rating prediction which give the best RMSE.

Overview

MovieLens dataset is separated to EDX and validation dataset which is 90:10 portion respectively. You can see data structure of these 2 datasets below.

EDX dataset :

```
str(edx,width=70,strict.width="cut")
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392..
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (".."
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action"..
##  - attr(*, ".internal.selfref")=<externalptr>
```

Validation dataset :

```
str(validation,width=70,strict.width="cut")
```

```
## Classes 'data.table' and 'data.frame':   999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645..
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "H"..
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Chi"..
##  - attr(*, ".internal.selfref")=<externalptr>
```

There are 6 variables in EDX and validation dataset; userId, movieId, rating, timestamp, title and genres.

The EDX dataset will be used for model development and the validation dataset will be used for final model performance testing.

Executive summary

From two datasets we have; edx and validation dataset, the models to predict movie rating will be developed using EDX dataset. First step is data preparation including data manipulation and data cleaning.

After data preparation, second step is data exploration and visualization to validate which variable have relationship with movie rating and explore its distribution. Then, those variables will be used in prediction model development step.

Once we get the variables for our model, next step is to split EDX dataset to be train set for model development and test set for model validation by RMSE. The model which give the best RMSE will be the final model. This final model will give the final performance with validation dataset.

Methods and analysis

Data preparation

First step, let's see first six rows of EDX dataset.

```
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:          Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:        Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

Data manipulation

From first six row printed, you will see movie year is indicated in title variable : inside bracket after movie name. I will extract the movie year to be additional variable and explore this data in next section to see if there are any relationship to rating.

```
## extract movie year from title ##
edx <- edx %>%
  mutate(movie_year=str_sub(title,str_length(title)-4,str_length(title)-1))
## Change movie_year type to numeric
edx <- edx %>% mutate(movie_year=as.numeric(movie_year))
```

```
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres movie_year
## 1:                Comedy|Romance       1992
## 2:          Action|Crime|Thriller       1995
## 3:  Action|Drama|Sci-Fi|Thriller       1995
## 4:        Action|Adventure|Sci-Fi       1994
## 5: Action|Adventure|Drama|Sci-Fi       1994
## 6:       Children|Comedy|Fantasy       1994
```

Data cleaning

Next, we will check if there are any NA in EDX dataset.

```
any(is.na(edx))
```

```
## [1] FALSE
```

The result from code above is FALSE means there is no NA in our edx dataset. From this result, there is no further cleaning needed.

Data exploration and visualization

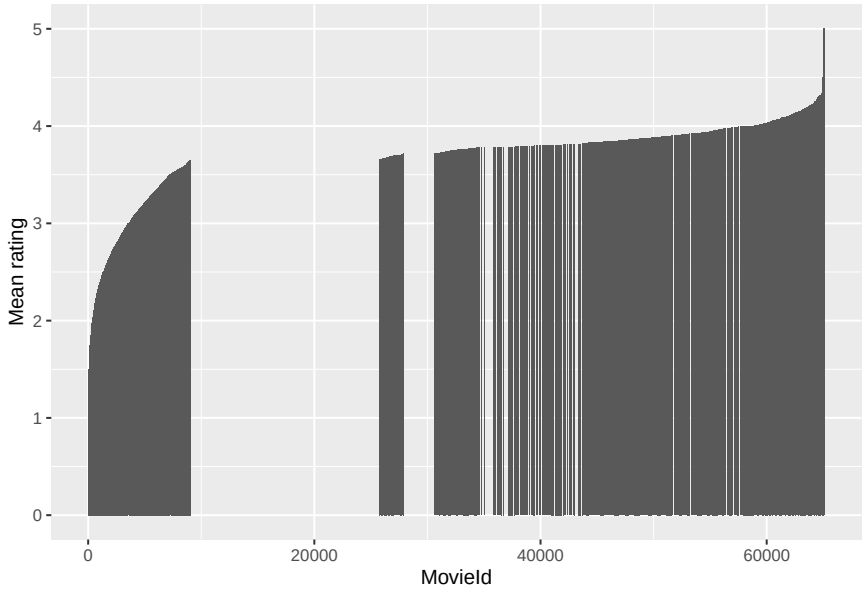
There are 2 main explorations in this section.

1. To check relationship of each variable with movie rating. The variable which effect movie rating will be used in modeling.
2. To analyze chosen variable from first exploration to see its distribution and consider if there are any adjustment needed for modeling.

Relationship of each variable with movie rating

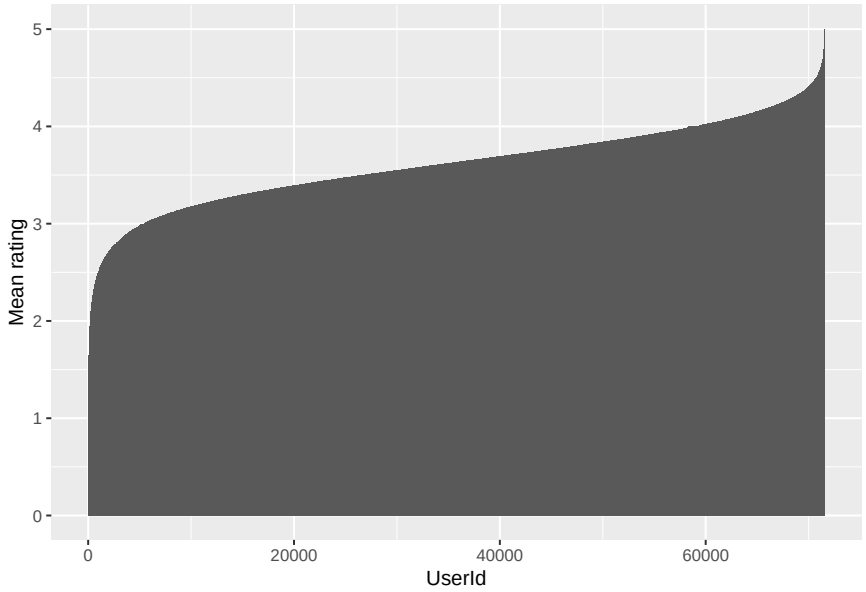
MovieId

Plot below show relationship between movieId and mean of rating. From the plot, we found that rating vary by movieId so movieId will be used in our prediction model.



UserId

Plot below show relationship between userId and mean of rating. From the plot, we found that rating vary by userId so userId will be used in our prediction model.



Movie year

Plot below show relationship between movie year and mean of rating. From the plot, we found that rating vary by movie year. This variable affect rating not much compare to userId and movieId, mean of rating only vary from about 3.3 to 4 but I will use this variable in prediction model and we will see if this variable can improve the prediction.

Timestamp :

This variable is possible to have in edx dataset but doesn't have in validation dataset so this variable isn't used in prediction model.

Title :

This variable reflect to movieId variable so this variable isn't used in prediction model.

Genres :

There are many genres included in this variables. To use this variable, we need to process data such as number of genres, ect. If we process this data, it is possible to have mismatch data between edx dataset and validation dataset so this variable isn't used in prediction model.

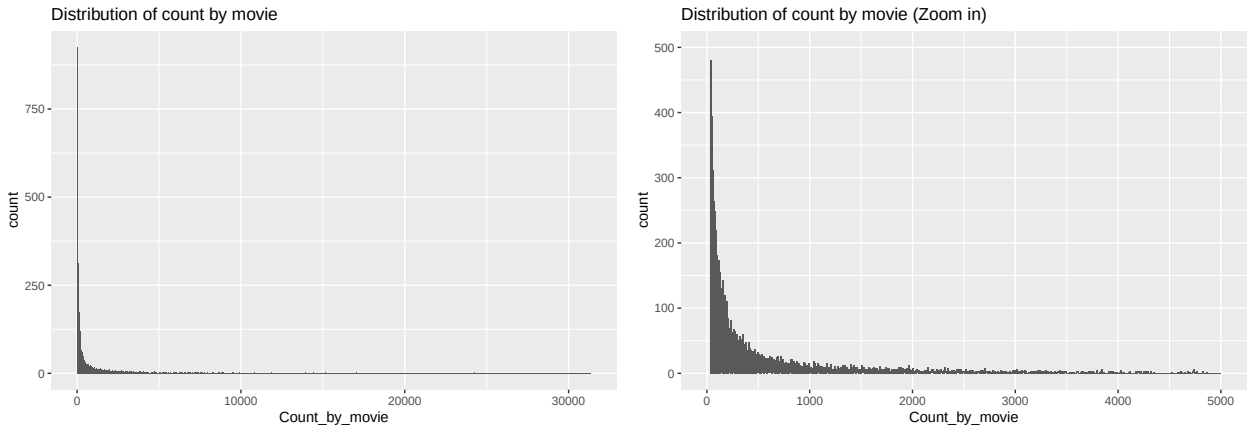
Summary : The variables which will be applied to the model are movieId, userId and movie year.

Distribution of chosen variables

MovieId

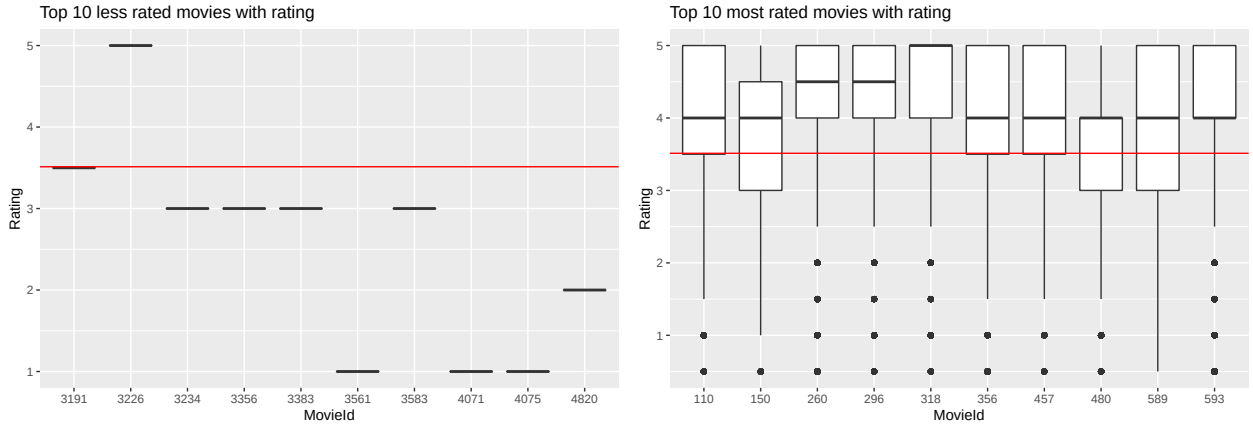
The distribution of count by movie below shows that many movies get rated less than others :

```
edx %>% group_by(movieId) %>%
  summarize(Count_by_movie=n()) %>% ggplot(aes(Count_by_movie))+
  geom_histogram(binwidth = 10)+
  ggtitle("Distribution of count by movie")

edx %>% group_by(movieId) %>%
  summarize(Count_by_movie=n()) %>% ggplot(aes(Count_by_movie))+
  geom_histogram(binwidth = 10)+
  ylim(0,500)+xlim(0,5000)+
  ggtitle("Distribution of count by movie (Zoom in)")
```



From this data, regularization which is to constrain the total variability of the effect sizes will be considered. Let's explore deeply on movies which get rated less compare to movies which get rated most.

```
movie_less <- edx %>% group_by(movieId)%>%
  summarize(movie_count=n(),mean_rating=mean(rating)) %>%
  arrange(movie_count) %>% slice(0:10) %>% pull(movieId)
edx %>% filter(movieId %in% movie_less) %>% group_by(movieId)%>%
  ggplot(aes(as.factor(movieId),rating))+
  geom_boxplot()+
  geom_hline(yintercept=mean(edx$rating), color="red")+
  ggtitle("Top 10 less rated movies with rating")+
  xlab("MovieId")+
  ylab("Rating")

movie_most <- edx %>% group_by(movieId)%>%
  summarize(movie_count=n(),mean_rating=mean(rating)) %>%
  arrange(desc(movie_count)) %>% slice(0:10) %>% pull(movieId)
edx %>% filter(movieId %in% movie_most) %>% group_by(movieId)%>%
  ggplot(aes(as.factor(movieId),rating))+
  geom_boxplot()+
  geom_hline(yintercept=mean(edx$rating), color="red")+
  ggtitle("Top 10 most rated movies with rating")+
  xlab("MovieId")+
  ylab("Rating")
```



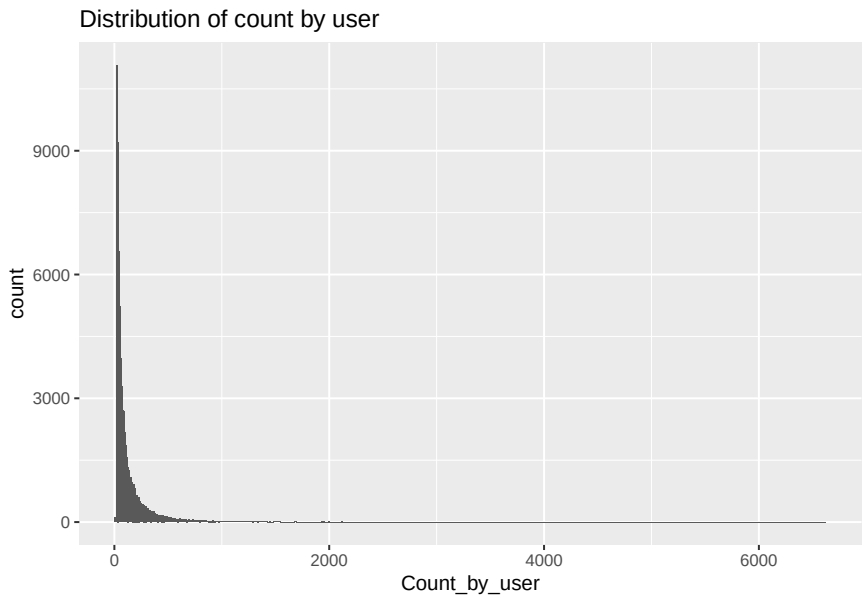
Left boxplot is top 10 less rated movies distribution with rating. Right boxplot is top 10 most rated movies distribution with rating. The red line is mean rating of EDX dataset.

Comparing both graphs, less rated movies distribution is out of the red line more than most rated movies which means the less rated movies may lead to overfitting prediction so regularized movie effect should be applied to our prediction model.

UserId

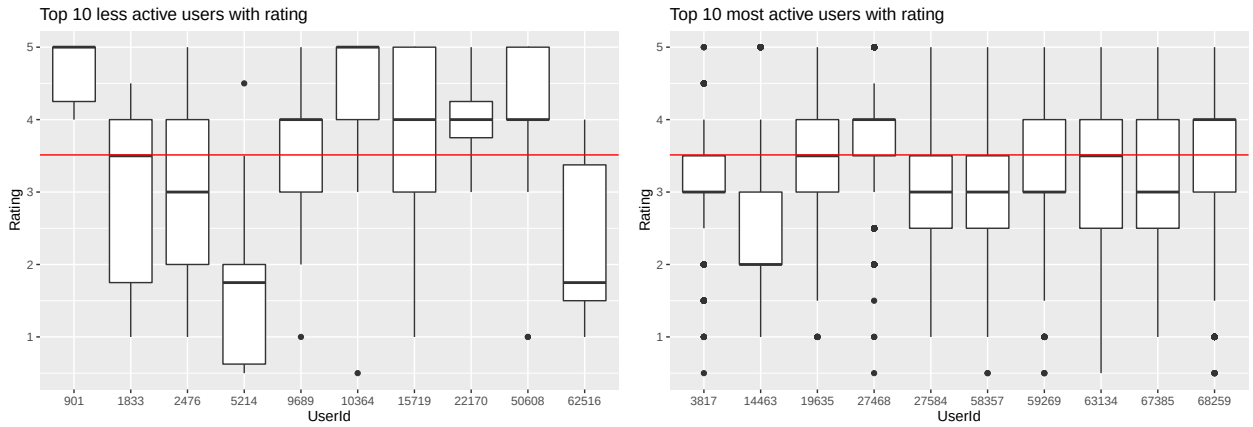
The distribution of count by user below shows that many users are less active than others at rating movies:

```
edx %>% group_by(userId) %>%
  summarize(Count_by_user=n()) %>% ggplot(aes(Count_by_user))+
  geom_histogram(binwidth = 10)+
  ggtitle("Distribution of count by user")
```



From this data, regularization which is to constrain the total variability of the effect sizes will be considered. Let's explore deeply on users which are less active compare to users which are most active.

```
user_less <- edx %>% group_by(userId)%>%
  summarize(user_count=n(),mean_rating=mean(rating)) %>%
  arrange(user_count) %>% slice(0:10) %>% pull(userId)
edx %>% filter(userId %in% user_less) %>% group_by(userId)%>%
  ggplot(aes(as.factor(userId),rating))+
  geom_boxplot()+
  geom_hline(yintercept=mean(edx$rating), color="red")+
  ggtitle("Top 10 less active users with rating")+
  xlab("UserId")+
  ylab("Rating")

user_most <- edx %>% group_by(userId)%>%
  summarize(user_count=n(),mean_rating=mean(rating)) %>%
  arrange(desc(user_count)) %>% slice(0:10) %>% pull(userId)
edx %>% filter(userId %in% user_most) %>% group_by(userId)%>%
  ggplot(aes(as.factor(userId),rating))+
  geom_boxplot()+
  geom_hline(yintercept=mean(edx$rating), color="red")+
  ggtitle("Top 10 most active users with rating")+
  xlab("UserId")+
  ylab("Rating")
```



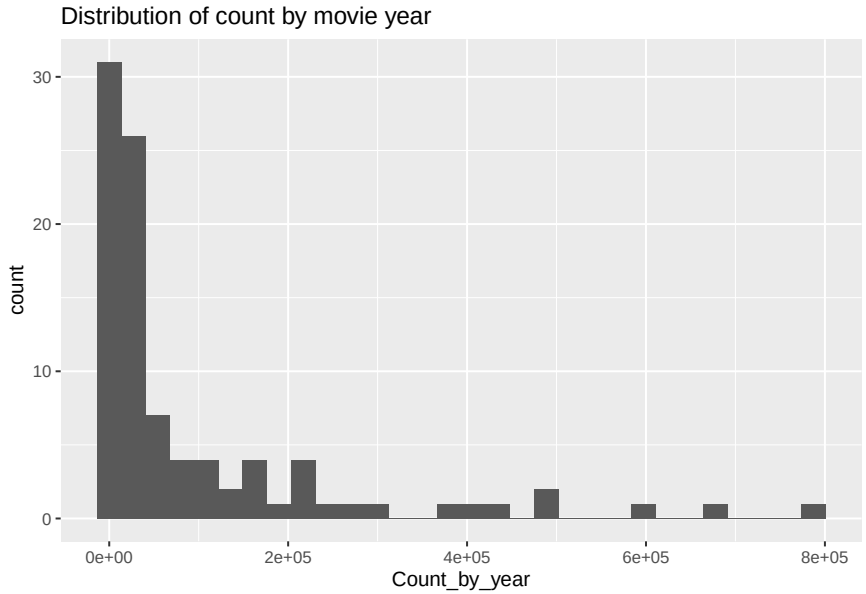
Left boxplot is top 10 less active users distribution with rating. Right boxplot is top 10 most active users distribution with rating. The red line is mean rating of EDX dataset.

Comparing both graphs, less active users distribution is out of the red line more than most active users which means the less active users may lead to overfitting prediction so regularized user effect should be applied to our prediction model.

Movie year

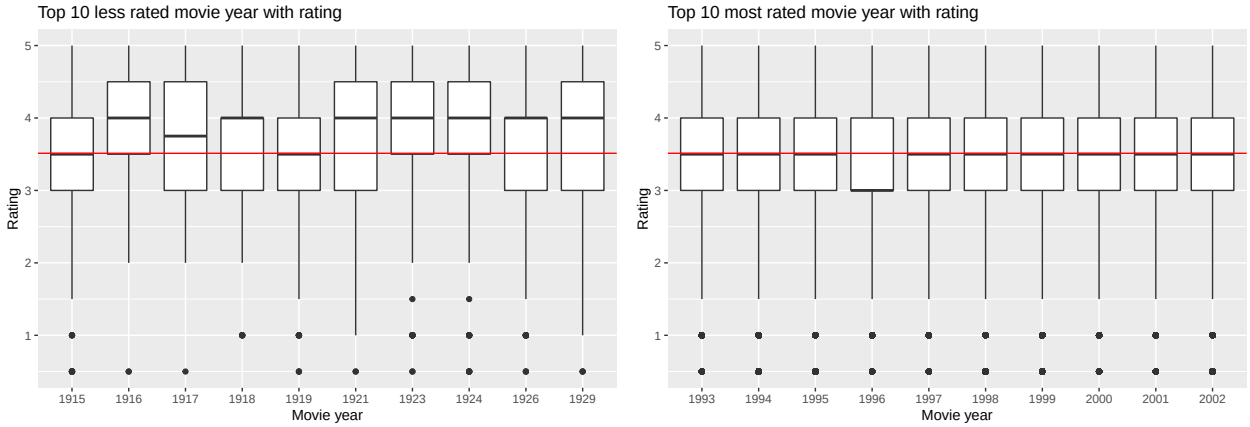
The distribution of count by movie year below shows that many movie year get rated less than others :

```
edx %>% group_by(movie_year) %>%
  summarize(Count_by_year=n()) %>% ggplot(aes(Count_by_year))+
  geom_histogram()+
  ggtitle("Distribution of count by movie year")
```



From this data, regularization which is to constrain the total variability of the effect sizes will be considered. Let's explore deeply on movie year which get rated less compare to movie year which get rated most.

```
movieY_less <- edx %>% group_by(movie_year)%>%
  summarize(movieY_count=n(),mean_rating=mean(rating)) %>%
  arrange(movieY_count) %>% slice(0:10) %>% pull(movie_year)
edx %>% filter(movie_year %in% movieY_less) %>% group_by(movie_year)%>%
  ggplot(aes(as.factor(movie_year),rating))+
  geom_boxplot()+
  geom_hline(yintercept=mean(edx$rating), color="red")+
  ggtitle("Top 10 less rated movie year with rating")+
  xlab("Movie year")+
  ylab("Rating")

movieY_most <- edx %>% group_by(movie_year)%>%
  summarize(movieY_count=n(),mean_rating=mean(rating)) %>%
  arrange(desc(movieY_count)) %>% slice(0:10) %>% pull(movie_year)
edx %>% filter(movie_year %in% movieY_most) %>% group_by(movie_year)%>%
  ggplot(aes(as.factor(movie_year),rating))+
  geom_boxplot()+
  geom_hline(yintercept=mean(edx$rating), color="red")+
  ggtitle("Top 10 most rated movie year with rating")+
  xlab("Movie year")+
  ylab("Rating")
```



Left boxplot is top 10 less rated movie year distribution with rating. Right boxplot is top 10 most rated movie year distribution with rating. The red line is mean rating of EDX dataset.

Comparing both graphs, less rated movie year distribution is out of the red line more than most rated movie year which means the less rated movie year may lead to overfitting prediction so regularized movie year effect should be applied to our prediction model.

Model Development

RMSE function

The residual mean squared error (RMSE) is used as model performance on test set. We define $y_{u,i,y}$ as the rating for movie i by user u and movie year y and denote our prediction with $\hat{y}_{u,i,y}$. The RMSE is then defined as:

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

The RMSE represents how far between predicted value and actual value.

The RMSE function coding :

```
RMSE <- function(true_data,pred_data){
  sqrt(mean((true_data-pred_data)^2))
}
```

Split Train and Test set

Prediction model will be created using EDX dataset. If there is only one dataset so we can't validate its performance. Let's split EDX dataset to be train set and test set. Train set is used for prediction model development and test set is used for performance validation.

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(edx$rating,times = 1,p=0.2,list=FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

To assure we don't include userId and movieId in the test set that do not appear in the train 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")
```

Models

1. Naive mean

Let's start with a simple prediction model to predict the same rating for all movies regardless of other parameters. The model assumes the same rating for all movies, users and movie year with all the differences explained by random variation would look like this:

$$Y_{u,i,y} = \mu + \varepsilon_{u,i,y}$$

with $\varepsilon_{u,i,y}$ independent errors sampled from the same distribution centered at 0 and μ the true rating for all movies which is average of all rating in this model.

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

```
## [1] 3.512482
```

```
naive_rmse <- RMSE(test_set$rating,mu_hat)
```

```
RMSE_result <- data.frame(Method="Naive mean",RMSE=naive_rmse)
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.059904

The RMSE for Naive mean model is shown in table below. The result is more than 1 which still doesn't satisfy. Let's see how we can improve our prediction model in next model.

2. Movie effect

From data exploration and visualization section, the parameters which effect movie rating are movieId, userId and movie year. Let's start to apply movieId effect to our model by adding b_i to the model :

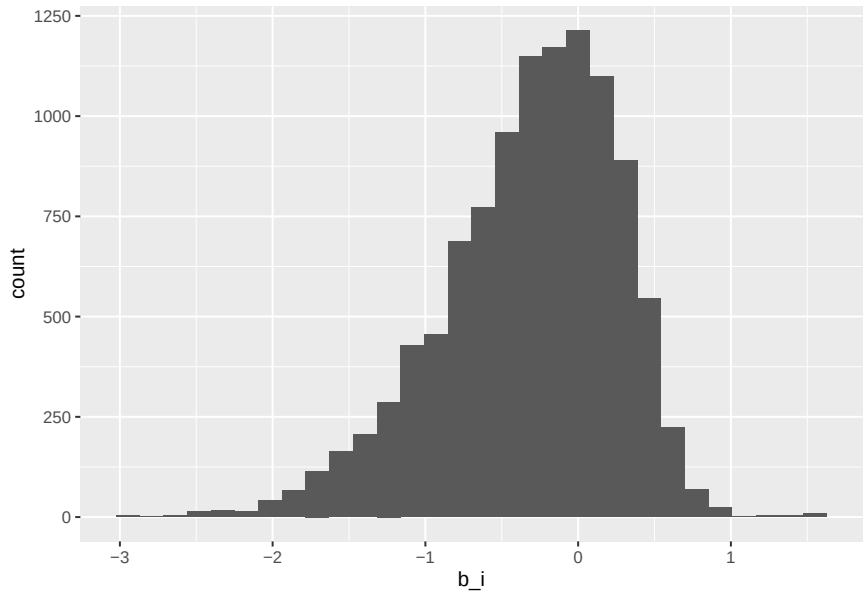
$$Y_{u,i,y} = \mu + b_i + \varepsilon_{u,i,y}$$

with b_i represent average ranking for movie i:

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

You can see distribution of b_i in histogram below which means there are bias from each movie.

```
movie_avgs %>% ggplot(aes(b_i))+geom_histogram()
```



Let's see how much our prediction improves once we apply movieId effect to our model.

```
predicted_rating_movie <- mu + test_set %>%
  left_join(movie_avgs,by="movieId") %>%
  pull(b_i)
movie_rmse <- RMSE(test_set$rating,predicted_rating_movie)

RMSE_result <- rbind(RMSE_result,data.frame(Method="Movie Effect",RMSE=movie_rmse))
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.0599043
Movie Effect	0.9437429

Now Movie Effect RMSE is better than Naive mean RMSE. We just apply movieId effect to the model. There are others two parameter(userId and movie year) which also effect movie rating. Next model will add userId effect to the model.

3. UserId and MovieId effect

This model will add userId effect to the model :

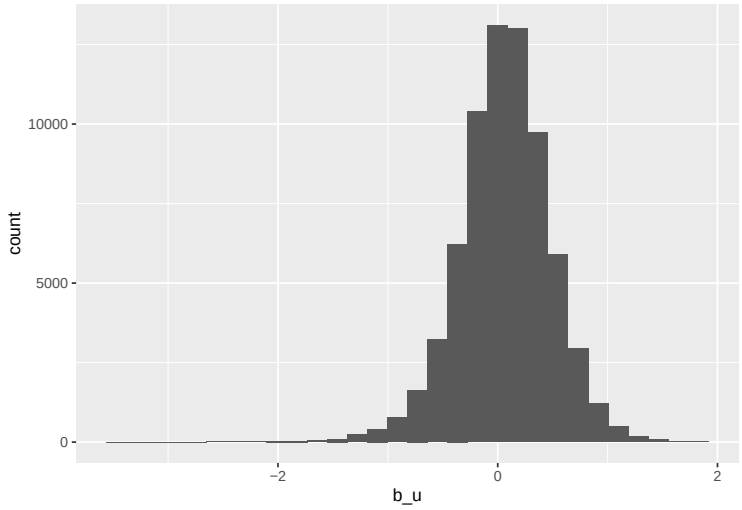
$$Y_{u,i,y} = \mu + b_i + b_u + \varepsilon_{u,i,y}$$

with b_u represent a user-specific effect:

```
movie_user_avgs <- train_set %>%
                   left_join(movie_avgs,by="movieId") %>%
                   group_by(userId)%>%
                   summarize(b_u=mean(rating-mu-b_i))
```

You can see distribution of b_u in histogram below which means there are bias from each user.

```
movie_user_avgs %>% ggplot(aes(b_u))+geom_histogram()
```



Let's see how much our prediction improves once we add userId effect to our model.

```
predicted_rating_movie_user <- test_set %>%
                               left_join(movie_avgs,by="movieId")%>%
                               left_join(movie_user_avgs,by="userId")%>%
                               mutate(pred=mu+b_i+b_u)%>%
                               pull(pred)
movie_user_rmse <- RMSE(predicted_rating_movie_user,test_set$rating)

RMSE_result <- rbind(RMSE_result,data.frame(Method="Movie and User Effect",
                                            RMSE=movie_user_rmse))
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.0599043
Movie Effect	0.9437429
Movie and User Effect	0.8659320

Now "Movie and User Effect" RMSE is better than first two models. Next model will add movie year effect to the model.

4. UserId, MovieId and Movie year effect

This model will add movie year effect to the model :

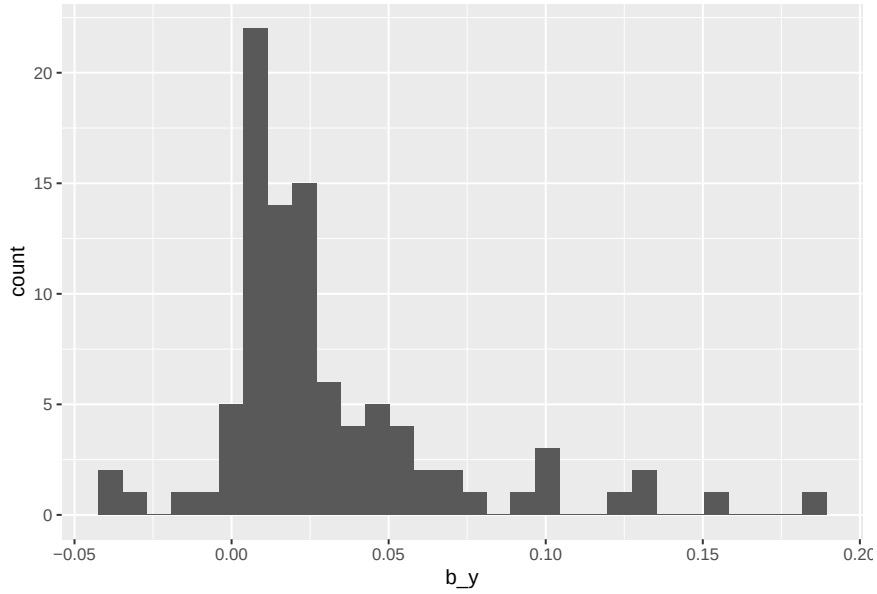
$$Y_{u,i,y} = \mu + b_i + b_u + b_y + \varepsilon_{u,i,y}$$

with b_y represent a movie year-specific effect:

```
movie_user_movieYear_avgs <- train_set %>%
  left_join(movie_avgs,by="movieId") %>%
  left_join(movie_user_avgs,by="userId") %>%
  group_by(movie_year)%>%
  summarize(b_y=mean(rating-mu-b_i-b_u))
```

You can see distribution of b_y in histogram below which means there are bias from each movie year.

```
movie_user_movieYear_avgs %>% ggplot(aes(b_y))+geom_histogram()
```



Let's see how much our prediction improves once we add movie year effect to our model.

```
predicted_rating_movie_user_movieYear <- test_set %>%
  left_join(movie_avgs,by="movieId")%>%
  left_join(movie_user_avgs,by="userId")%>%
  left_join(movie_user_movieYear_avgs,by="movie_year")%>%
  mutate(pred=mu+b_i+b_u+b_y)%>%
  pull(pred)
movie_user_movieYear_rmse <- RMSE(predicted_rating_movie_user_movieYear,test_set$rating)

RMSE_result <- rbind(RMSE_result,data.frame(Method="Movie,User and Movie year Effect",
                                            RMSE=movie_user_movieYear_rmse))
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.0599043
Movie Effect	0.9437429
Movie and User Effect	0.8659320
Movie,User and Movie year Effect	0.8656117

Now "Movie, User and Movie year Effect" RMSE is better than first three models. We notice that this model improve from previous model(Movie and User effect) not much and it relates to our "data exploration and visualization" section which shows that movie year least effect to movie rating.

Next model will apply regularization concept to constrain the total variability of the effect sizes.

5. Regularized movieId, userId and movie year Effect

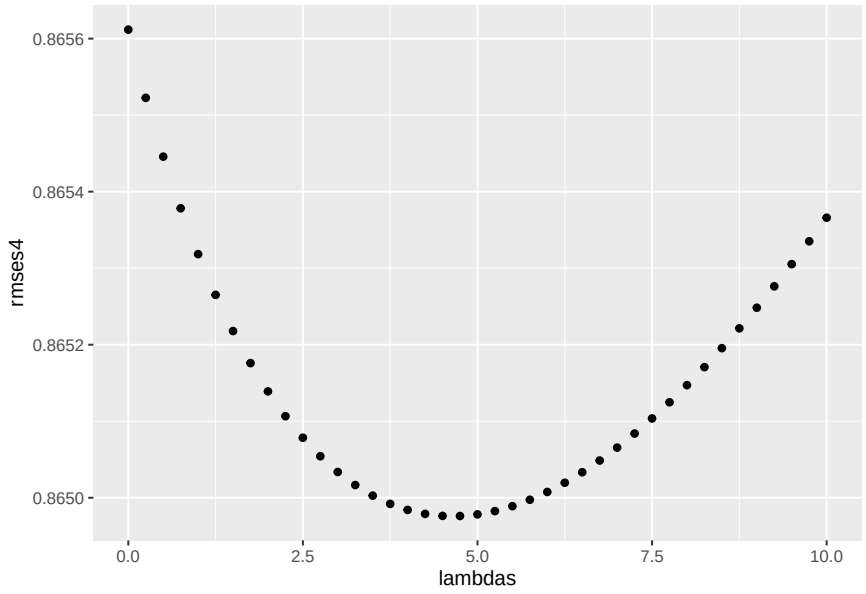
This model, λ which is penalty parameter of low numbers of ratings effect for movies, users and movie year will be added to reduce the effect of overfitting shown in data exploration and visualization section.

Note that λ is a tuning parameter. We can use cross-validation to choose it.

```
lambdas <- seq(0,10,0.25)
rmses4 <- sapply(lambdas,function(l){
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i=sum(rating-mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i,by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u=sum(rating-b_i-mu)/(n()+l))
  b_y <- train_set %>%
    left_join(b_i,by="movieId") %>%
    left_join(b_u,by="userId")%>%
    group_by(movie_year) %>%
    summarize(b_y=sum(rating-b_i-b_u-mu)/(n()+l))
  predicted_rating <- test_set %>%
    left_join(b_i,by="movieId") %>%
    left_join(b_u,by="userId") %>%
    left_join(b_y,by="movie_year") %>%
    mutate(pred = mu + b_i + b_u + b_y) %>%
    pull(pred)
  return(RMSE(predicted_rating,test_set$rating))
})
```

Plot below shows RMSE result from each λ value. λ which get the best RMSE is 4.75.

```
qplot(lambdas,rmses4)
```



```
lambdas[which.min(rmses4)]
```

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

```
Reg_movie_user_movieYear_rmse <- min(rmses4)
```

Now "Regularized Movie, User and Movie year Effect" RMSE is better than first four models.

```
RMSE_result <- rbind(RMSE_result,data.frame(Method="Regularized Movie,User and Movie year Effect",
                                            RMSE=Reg_movie_user_movieYear_rmse))
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.0599043
Movie Effect	0.9437429
Movie and User Effect	0.8659320
Movie,User and Movie year Effect	0.8656117
Regularized Movie,User and Movie year Effect	0.8649761

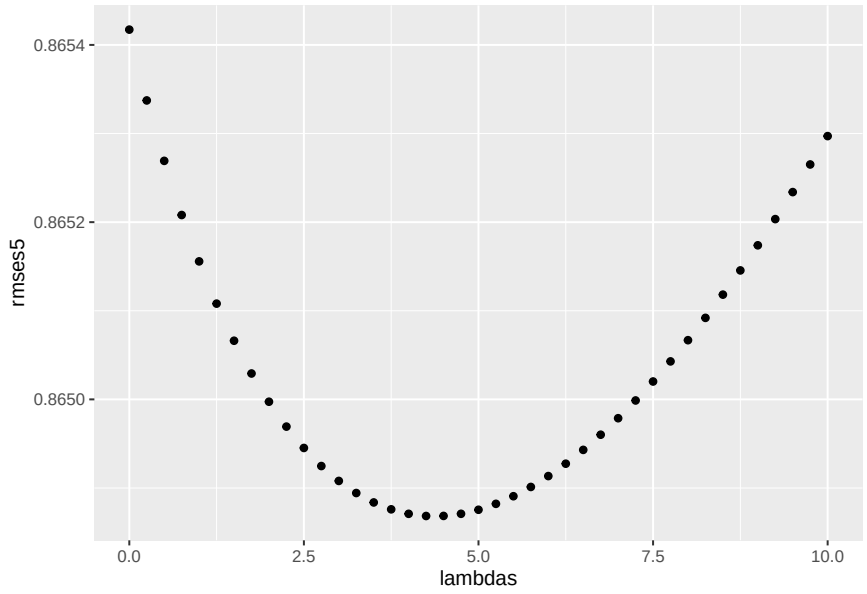
6. Regularized Movie,User and Movie year with adjusting Effect

This model will improve previous prediction model by adjusting the prediction which less than 0 to be minimum rating of train set and the prediction which more than 5 to be maximum rating of train set.

```
rmses5 <- sapply(lambdas,function(l){
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i=sum(rating-mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i,by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u=sum(rating-b_i-mu)/(n()+l))
  b_y <- train_set %>%
    left_join(b_i,by="movieId") %>%
    left_join(b_u,by="userId")%>%
    group_by(movie_year) %>%
    summarize(b_y=sum(rating-b_i-b_u-mu)/(n()+l))
  predicted_rating <- test_set %>%
    left_join(b_i,by="movieId") %>%
    left_join(b_u,by="userId") %>%
    left_join(b_y,by="movie_year") %>%
    mutate(pred = mu + b_i + b_u + b_y) %>%
    mutate(pred1=ifelse(pred>5,max(train_set$rating),pred))%>%
    mutate(pred2=ifelse(pred1<0,min(train_set$rating),pred1))%>%
    pull(pred2)
  return(RMSE(predicted_rating,test_set$rating))
})
```

Plot below shows RMSE result from each λ value. λ which get the best RMSE is 4.25 which is kept for final validation with validation dataset.

```
qplot(lambdas,rmses5)
```




```
lambdas[which.min(rmses5)]
```

```
## [1] 4.25
```

```
Reg_movie_user_movieYear_round_rmse <- min(rmses5)
```

```
lambda <- lambdas[which.min(rmses5)]
```

Now "Regularized Movie, User and Movie year with round Effect" RMSE is the best so this model will be used as final model.

```
RMSE_result <-
  rbind(RMSE_result,data.frame(Method="Regularized Movie,User and Movie year with adjusting Effect",
                               RMSE=Reg_movie_user_movieYear_round_rmse))
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.0599043
Movie Effect	0.9437429
Movie and User Effect	0.8659320
Movie,User and Movie year Effect	0.8656117
Regularized Movie,User and Movie year Effect	0.8649761
Regularized Movie,User and Movie year with adjusting Effect	0.8648684

Results

Test with validation dataset

```
## extract movie year from title for validation set ##
validation <- validation %>%
  mutate(movie_year=str_sub(title,str_length(title)-4,str_length(title)-1))
## Change movie_year type to numeric for validation set ##
validation <- validation %>% mutate(movie_year=as.numeric(movie_year))
```

```
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i=sum(rating-mu)/(n()+lambda))
b_u <- edx %>%
  left_join(b_i,by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u=sum(rating-b_i-mu)/(n()+lambda))
b_y <- edx %>%
  left_join(b_i,by="movieId") %>%
  left_join(b_u,by="userId")%>%
  group_by(movie_year) %>%
  summarize(b_y=sum(rating-b_i-b_u-mu)/(n()+lambda))
predicted_rating_validation <- validation %>%
  left_join(b_i,by="movieId") %>%
  left_join(b_u,by="userId") %>%
  left_join(b_y,by="movie_year") %>%
  mutate(pred = mu + b_i + b_u + b_y) %>%
  mutate(pred1=ifelse(pred>5,max(edx$rating),pred))%>%
  mutate(pred2=ifelse(pred1<0,min(edx$rating),pred1))%>%
  pull(pred2)
rmse_validation <- RMSE(validation$rating,predicted_rating_validation)

RMSE_result <- rbind(RMSE_result,data.frame(Method="Test with validation dataset",
                                            RMSE=rmse_validation))
knitr::kable(RMSE_result)
```

Method	RMSE
Naive mean	1.0599043
Movie Effect	0.9437429
Movie and User Effect	0.8659320
Movie,User and Movie year Effect	0.8656117
Regularized Movie,User and Movie year Effect	0.8649761
Regularized Movie,User and Movie year with adjusting Effect	0.8648684
Test with validation dataset	0.8644216

The RMSE when applied “Regularized Movie,User and Movie year with adjusting Effect” to validation dataset is 0.8644216.

Conclusion

The model which give the best RMSE is “Regularized Movie,User and Movie year with adjusting Effect” to predict rating from movieId, userId and movie year variables. The model also apply regularization to reduce the effect of overfitting and improve model by adjusting prediction which less than 0 to be minimum rating of train set and the prediction which more than 5 to be maximum rating of train set.