MovieLens Project

Iuliia Shal

22 09 2021

Contents

1 Project goal

The aim of this project is to create a movie recommendation system based on ratings by movies and users provided for period 1995-2009.
We will have 2 big datasets:
- edx (9 ml records) - for modelling
- validation (1 ml records) - for testing the final model

Below you can find the example of data composition:

userId	movieId	rating	timestamp	title	genres
50805	1095	4.0	952366935	Glengarry Glen Ross (1992)	Drama
42011	6303	3.5	1121891132	Andromeda Strain, The (1971)	Mystery|Sci-Fi
45985	593	4.0	854490846	Silence of the Lambs, The (1991)	Crime|Horror|Thriller
19660	4210	4.0	1207988321	Manhunter (1986)	Action|Crime|Drama|Horror|Thriller
46073	186	4.0	1123689113	Nine Months (1995)	Comedy|Romance

We can see that in *edx* dataset we have:

Users	Movies
69878	10677

The most rated movies average ratings are:

title	N_ratings	Avg_rating
Pulp Fiction (1994)	31362	4.15
Forrest Gump (1994)	31079	4.01
Silence of the Lambs, The (1991)	30382	4.20
Jurassic Park (1993)	29360	3.66
Shawshank Redemption, The (1994)	28015	4.46
Braveheart (1995)	26212	4.08
Fugitive, The (1993)	25998	4.01
Terminator 2: Judgment Day (1991)	25984	3.93
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672	4.22
Apollo 13 (1995)	24284	3.89

2 Split edx into Train and Test sets

To build model on the edx dataset, we need to split it into *Train* and *Test* sets to avoid over fitting.

```
set.seed(755)
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_edx <- edx[-test_index,]
test_edx <- edx[test_index,]

test_edx <- test_edx %>%
  semi_join(train_edx, by = "movieId") %>%
  semi_join(train_edx, by = "userId")
```

3 Models testing

3.1 Simple Average

Let's start from the simple model where we estimate ratings as average movie rating.

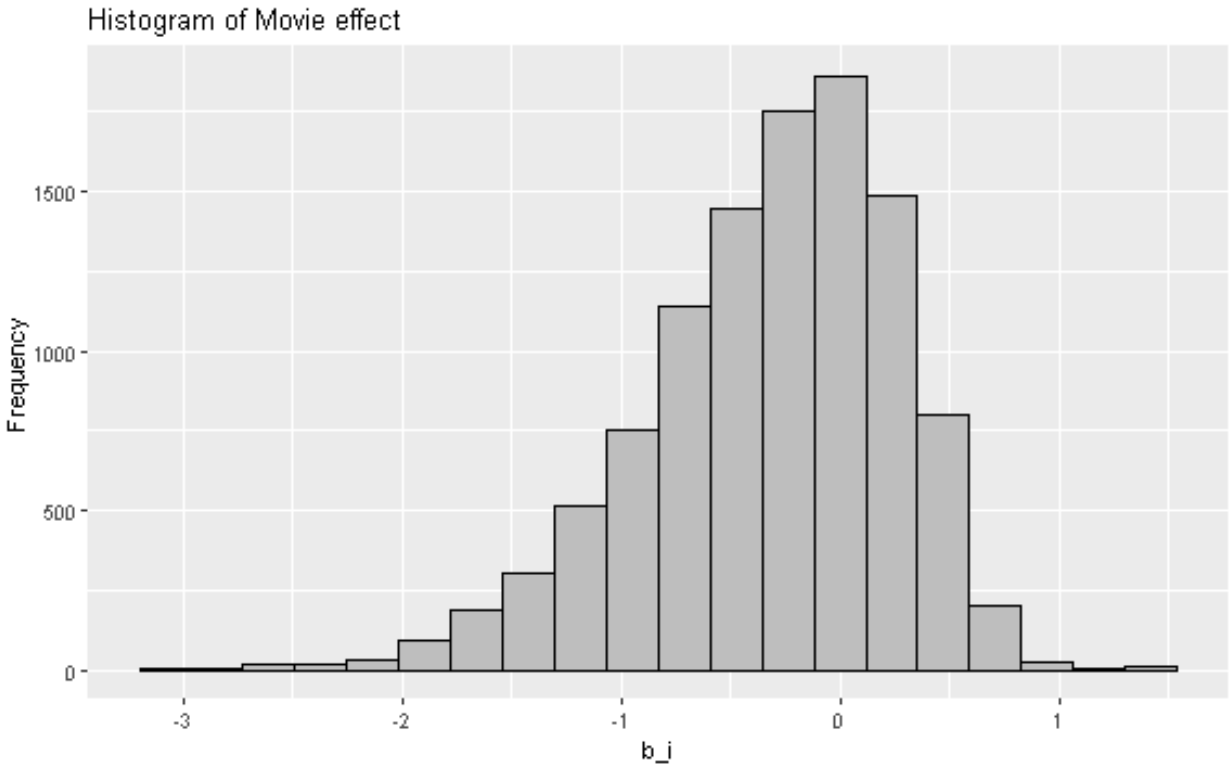
```
mu <- mean(train_edx$rating)
simple_average <- RMSE(test_edx$rating,mu)
print(simple_average)
```

```
## [1] 1.060561
```

$RMSE > 1$ meaning that our prediction may differ by 1 start rating. We need to improve the model and achieve $RMSE < 1$.

3.2 Movie effect model

Let's check the distribution of movies ratings excluding Average movie rating. By doing this, we center all ratings around average.



We can see that there only a few movies with 5 start rating (μ+1.5) and many movies equal to or below the average. We can explain it as there is a list of movies which everyone likes (blockbusters or Oscar winners) and a big tail of movies which are not that great. b_i will represent movie effect - on average how good the movie is in comparison to others.

It's time to check a new model where we will include avg movie rating (μ) and movie effect (b_i).

```
model_with_movie_effect_prediction <- mu + test_edx%>%
  left_join(movie_avg, by='movieId') %>%
  pull(b_i)

model_with_movie_effect_RMSE <-
  RMSE(test_edx$rating,model_with_movie_effect_prediction)

print(model_with_movie_effect_RMSE)
```

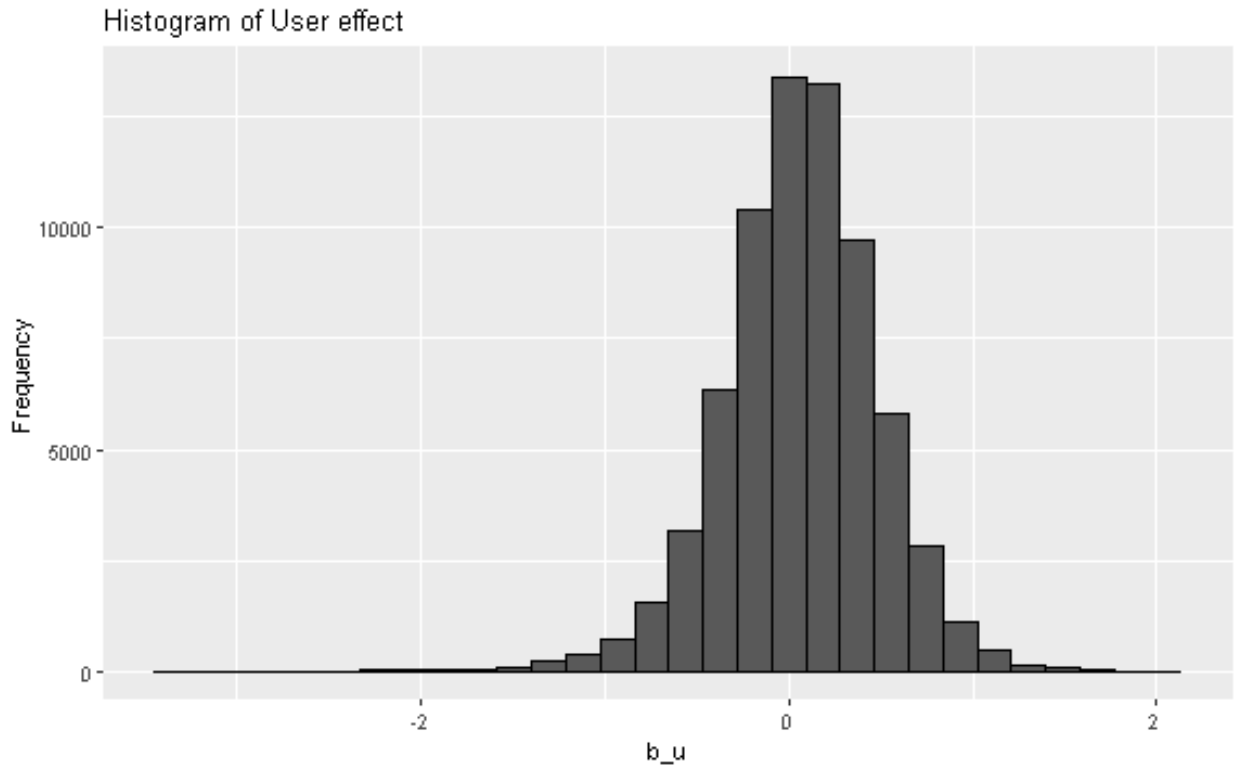
```
## [1] 0.9439868
```

RMSE is getting better but what about users? Do they differ from each other as well?

3.3 Movie and User effect model

Let's have a look how people differ from each other if we exclude avg rating and movie effect from ratings.

```
user_avg <- train_edx%>%
  left_join(movie_avg, by='movieId')%>%
  group_by(userId)%>%
  summarize(b_u = mean(rating-mu-b_i))

user_avg%>%
  ggplot(aes(b_u))+geom_histogram(bins = 30, color = "black")+
  ggtitle("Histogram of User effect")+
  ylab("Frequency")
```



Users preferences are normal distributed around average and we can see a left tail of very pessimistic viewers (who tend to rate movies lower) and a right tail of very optimistic viewers (tend to like everything they watch). We will call b_u as user effect - to show the difference in viewers behaviour.

Let's test a model which includes both movie and user effects:

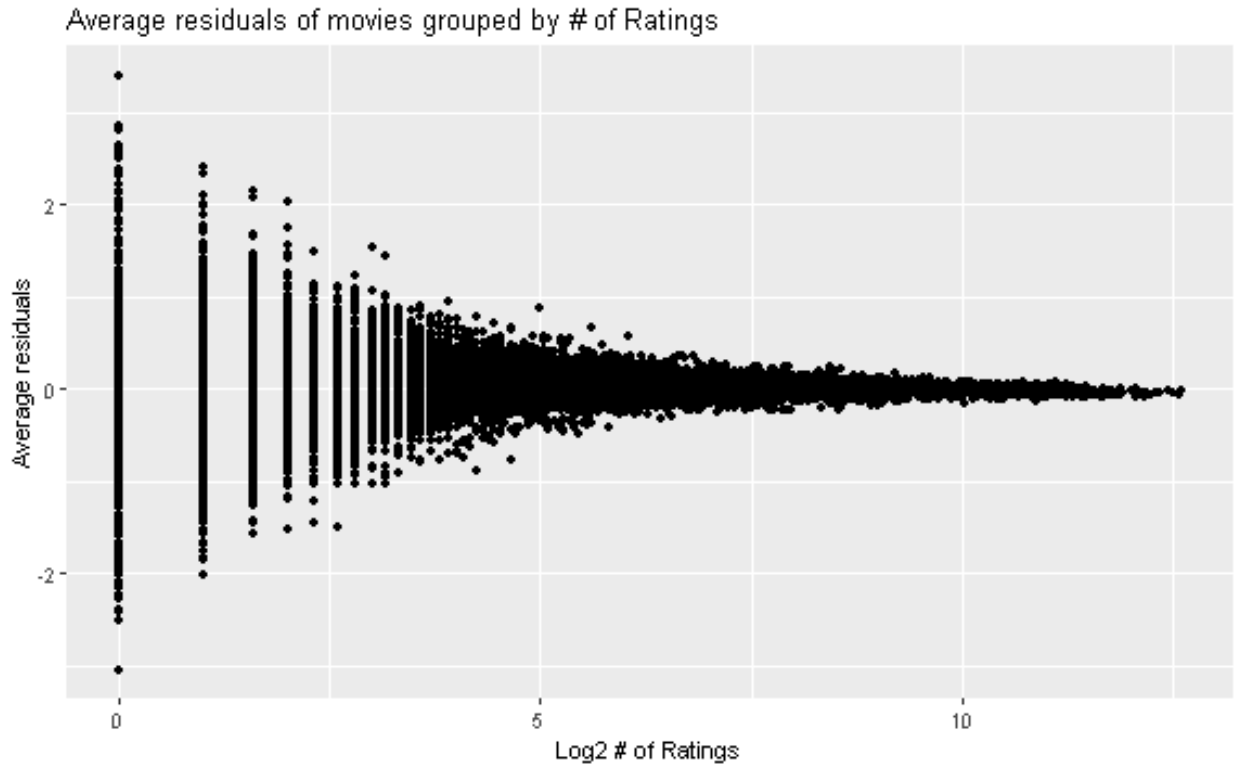
```
model_with_movie_user_effect_prediction <- mu + test_edx%>%
  left_join(movie_avg, by='movieId') %>% pull(b_i)+test_edx%>%
  left_join(user_avg, by='userId') %>% pull(b_u)

model_with_movie_user_effect_RMSE <-
  RMSE(test_edx$rating,model_with_movie_user_effect_prediction)

print(model_with_movie_user_effect_RMSE)
```

```
## [1] 0.8666408
```

We can see that RMSE has become significantly lower. Amazing results!

But can we make it even better?

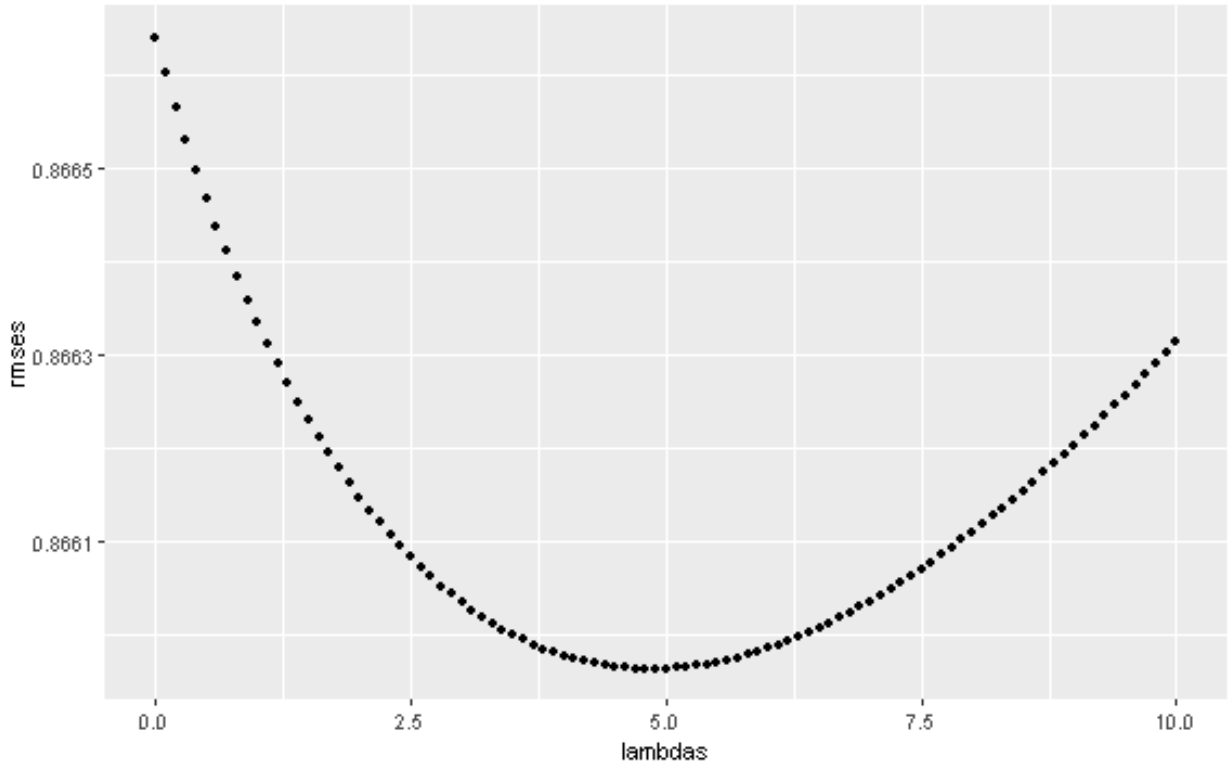
Let's have a look into highest discrepancies of the model.



We can see that highest discrepancies happen due to low # of ratings ($< 2^5$ # of ratings). The more ratings we have, the smoother projection we can get. To eliminate this noise and improve RMSE we need to penalize for extreme estimates which comes from small sample sizes. We will use regularization techniques for this case.

We are going to test set of λ and choose the one which minimize RMSE function the best:

```r
lambdas <- seq(0, 10, 0.1)

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

  mu <- mean(train_edx$rating)

  b_i <- train_edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l)) # Movie effect  and LAMBDA

  b_u <- train_edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l)) # User effect and LAMBDA

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

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

qplot(lambdas, rmses)
```



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

The best $\lambda = 4.9$ and we will use it as part of our regularized Movie and User effects model.

```
## [1] RMSE:  0.865962
```

We can see that RMSE hasn't changed much because the % ratings for not popular movies is very low in comparison the total population of ratings ($< 1\%$).

Table 4: % of ratings of Not popular movies, <= 32 ratings per movie

x
0.006233

Even though these movies create a big clusters out of total number of movies (30%).

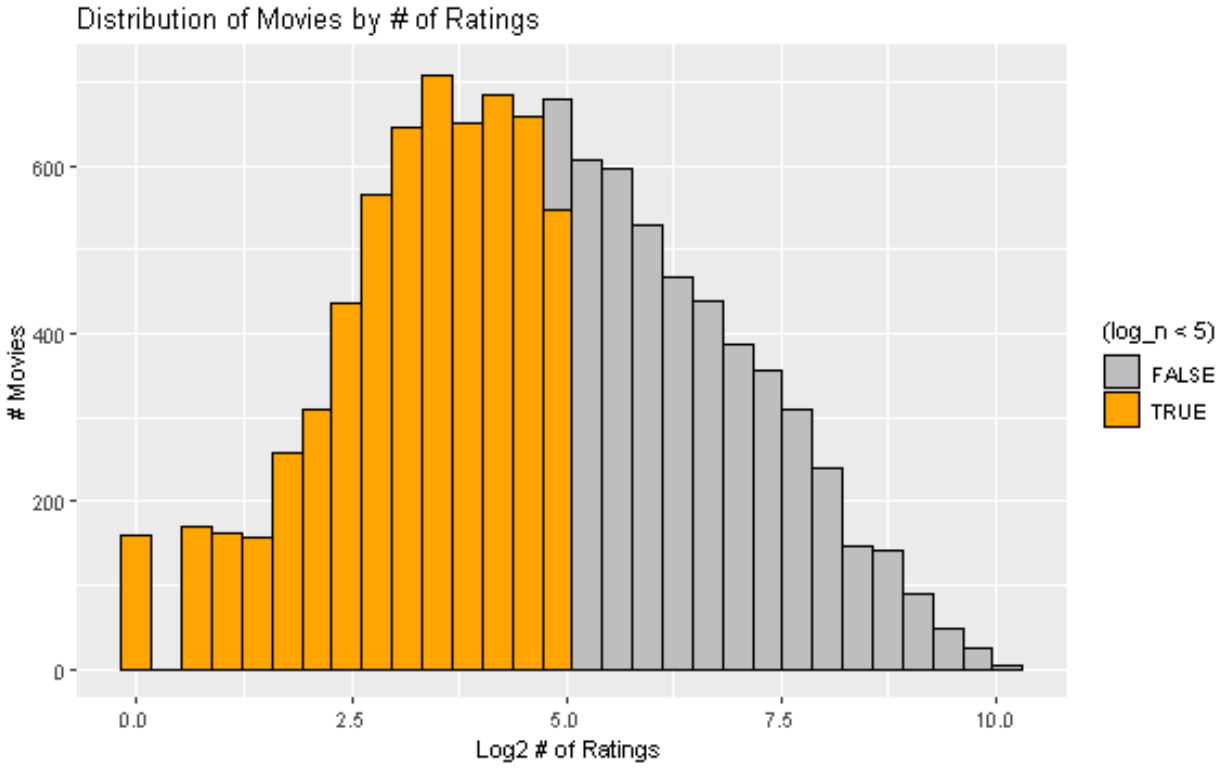
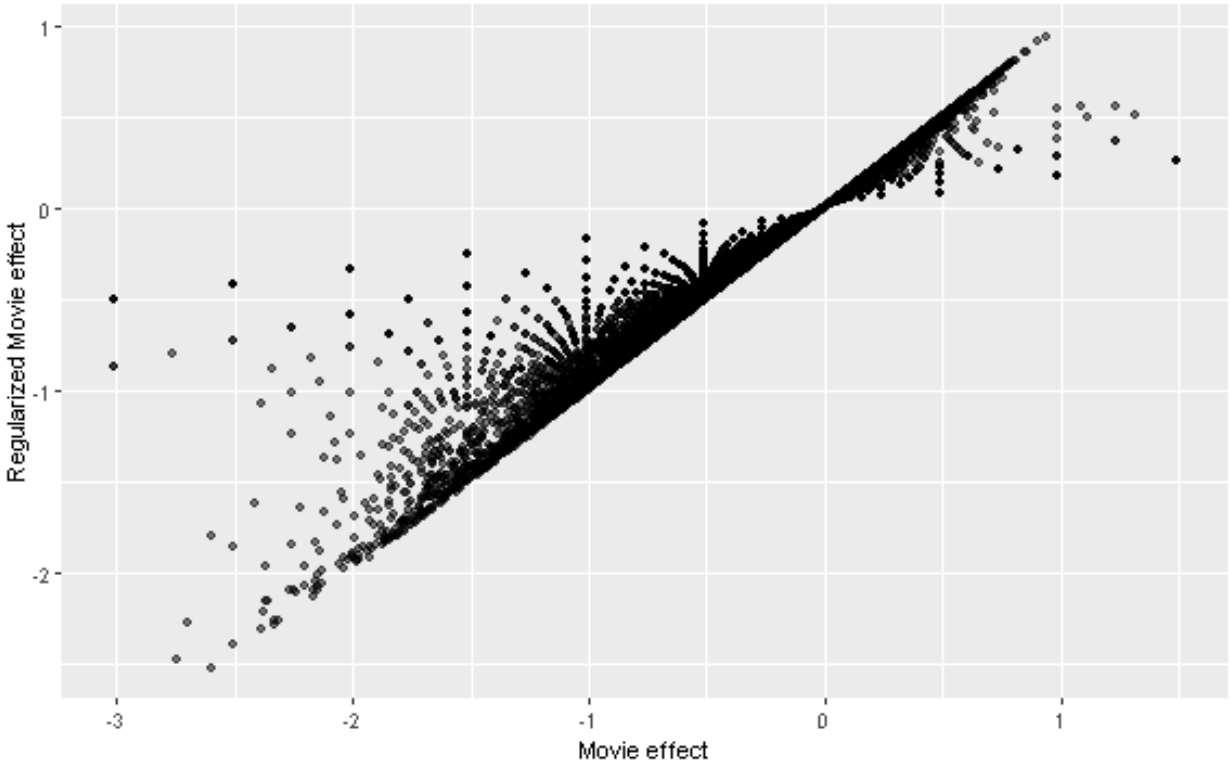


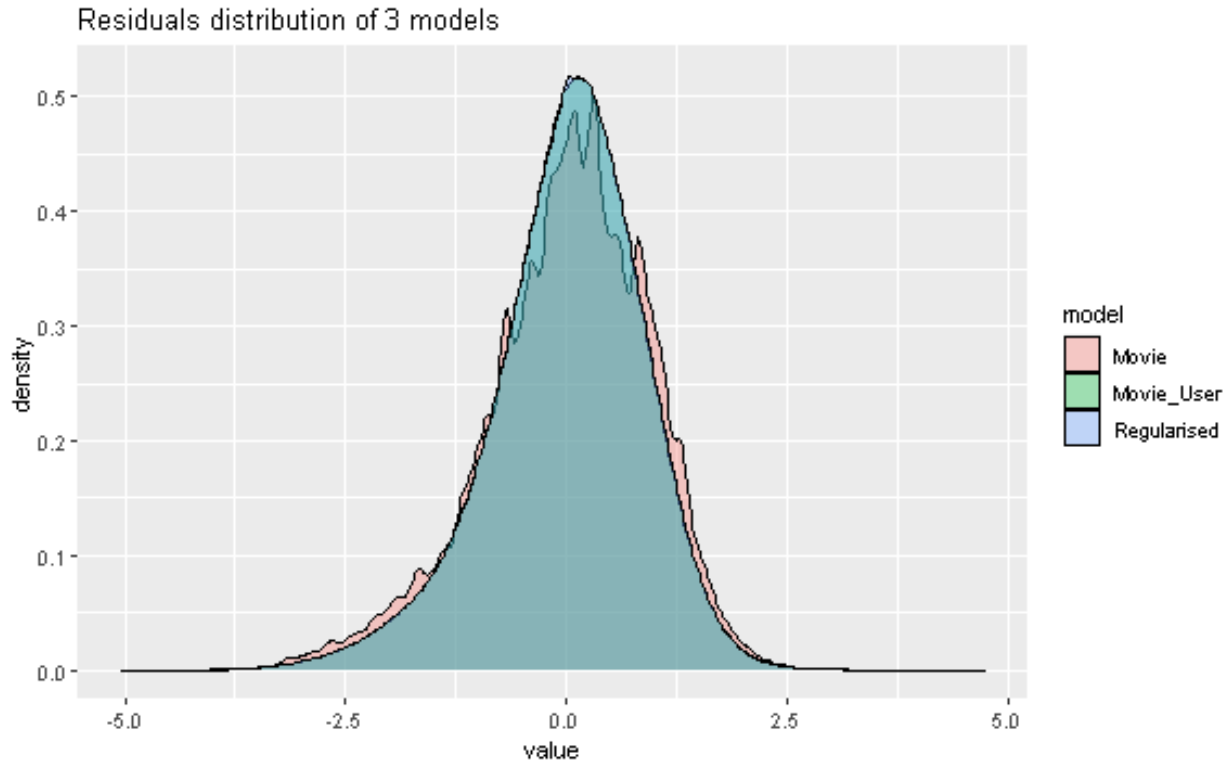
Table 5: % of Movies with # ratings <=32

x
0.3005357

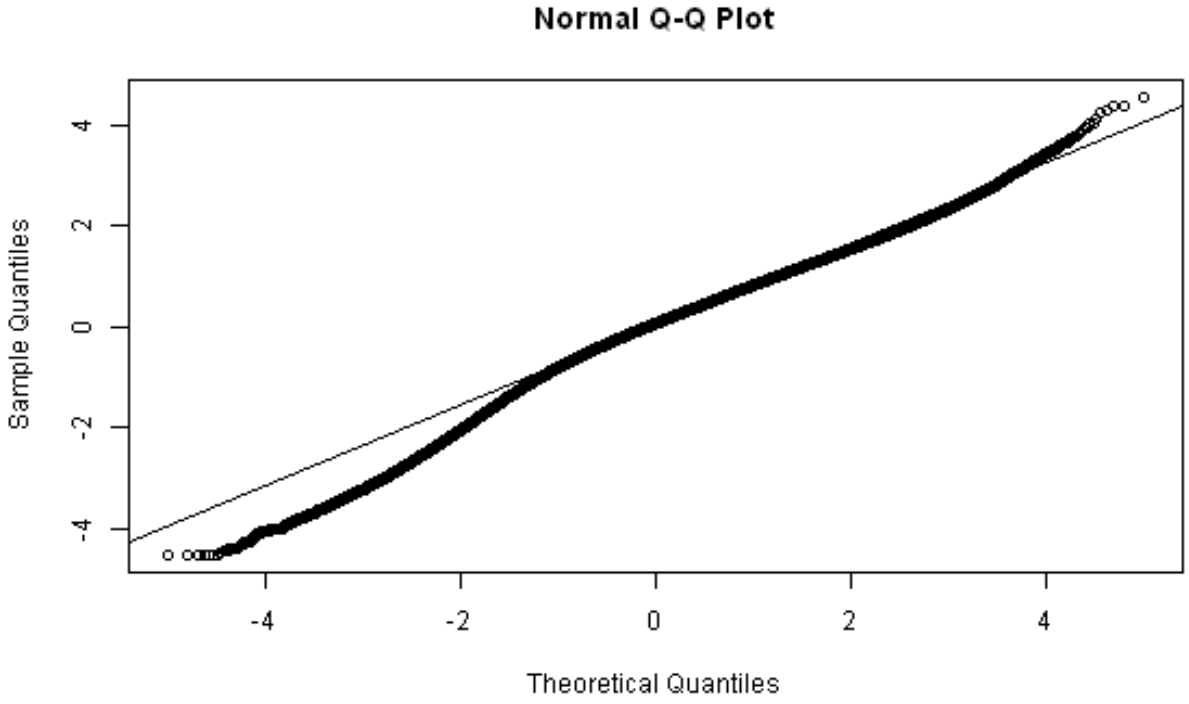
The graph below will show how movie effect has been minimized for movies with low number of ratings.



Now let's compare the distribution of residuals for 3 models:
1. Movie effect
2. Movie and User effects
3. Regularized Mode with Movie and User effect.



We can see the improvement we made by adding User effect and minor change by adding regularization. Let's also understand how far we are form the normal distribution of residuals in our final model.

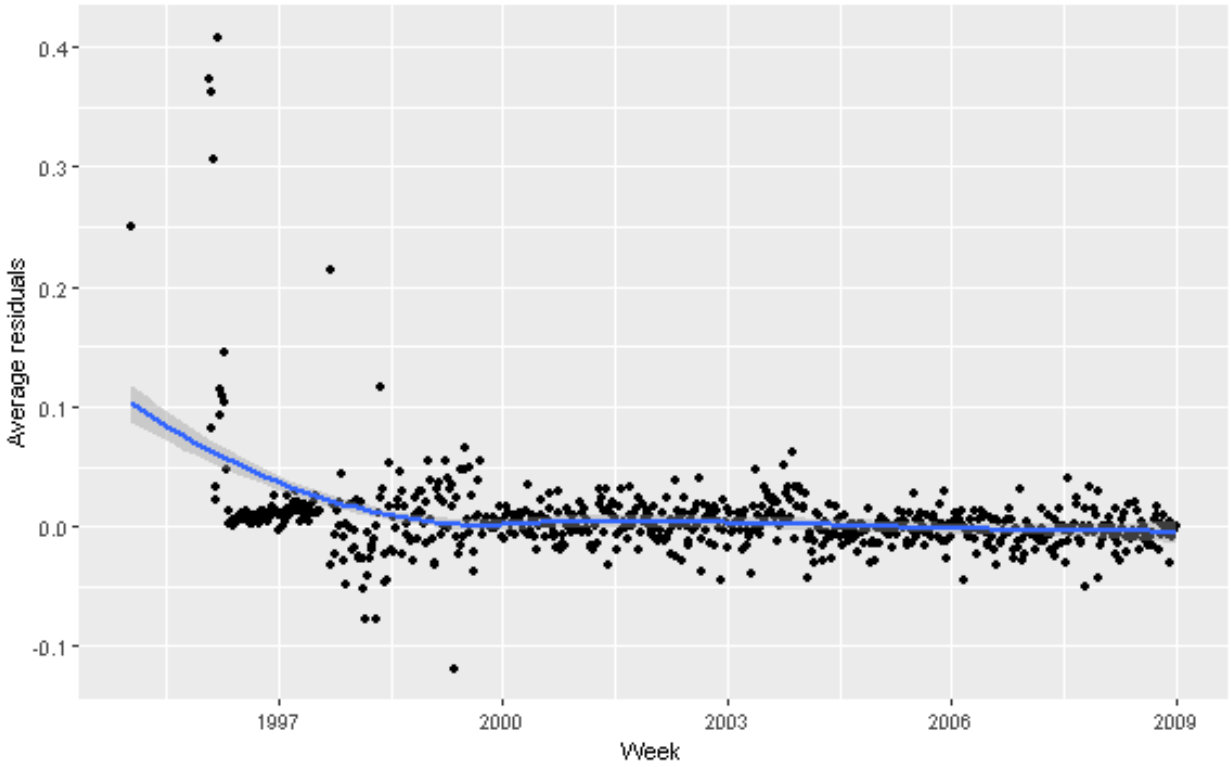


It's clear from the graph above that we have a big tail of overestimated ratings.
What else can we do to fix this problem?

3.4 Time effect

3.4.1 Week effect

By putting model's residuals against the week of rating, we can observe much higher residuals for ratings made earlier 1997. It means that ratings are better than we estimate.



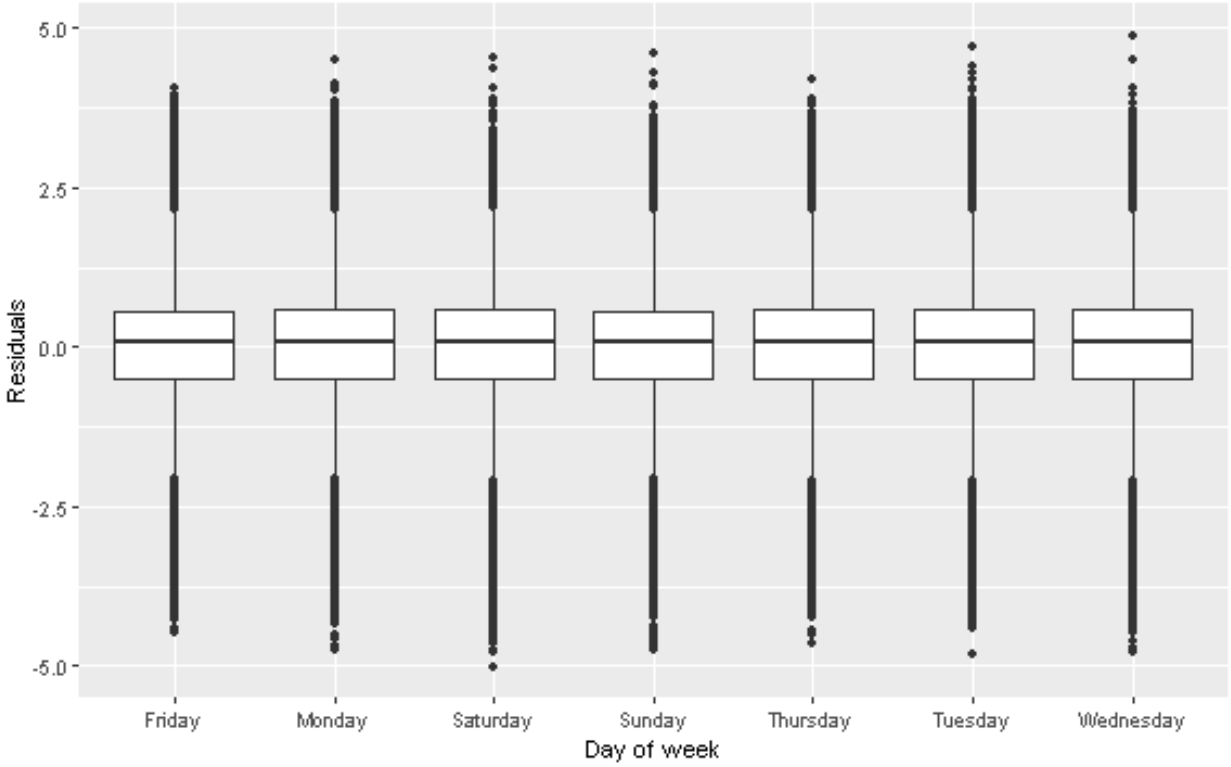
It might be one of reasons why our model overestimates other ratings. Moreover, we can see that after 1997 there is no time effect.
We will exclude this data from our train dataset treating it as outliers.

3.4.2 Day of the week effect

Do we have effect of rating per week day (Monday-Sunday?)
Do people like movies more when you're more relaxed on weekends?

```
train_edx %>%
  left_join(b_i_regularized, by='movieId')%>%
  left_join(b_u_regularized, by='userId')%>%
  mutate(date_rate=as_datetime(timestamp))%>% mutate(week=round_date(date_rate,"week"))%>%
  filter(week>"1997-01-01")%>%
  mutate(day_of_week=weekdays(date_rate))%>%
```

```
  mutate(pred = mu + b_i + b_u)%>%
  mutate(residual_regular=rating-pred)%>%
  ggplot(aes(x=day_of_week,y=residual_regular))+geom_boxplot()+
  xlab("Day of week")+
  ylab("Residuals")
```



As per graph above, there is no difference between day of the week, meaning that there is no such effect.

We will call a new database excluding ratings made <1997 as *train_edx_wide* and will use it in further analysis.

```
train_edx_wide <- train_edx%>%
  left_join(b_i_regularized, by='movieId')%>%
  left_join(b_u_regularized, by='userId')%>%
  mutate(date_rate=as_datetime(timestamp))%>%mutate(week=round_date(date_rate,"week"))%>%
  filter(week>"1997-01-01")%>%
  mutate(pred = mu + b_i + b_u)%>%
  mutate(residual_regular=rating-pred)
```

3.5 Principal Component Analysis

Before we perform Principal Component Analysis (PCA), we need to have a look at movies with highest number of residuals < -1 (movies which we overestimates by > 1 star).

```
total_negative <- train_edx_wide%>%
  filter(residual_regular<(-1))%>%summarize(n_negative=n())%>%.$n_negative

train_edx_wide %>%
  filter(residual_regular<(-1))%>%
  group_by(movieId) %>% summarize(n_negative=n())%>%
  arrange(desc(n_negative))%>%
  mutate(cumul_sum = cumsum(n_negative)/total_negative)%>%
  summarize(top_movies_50_percent = sum(cumul_sum<=0.5))
```

top_movies_50_percent
533

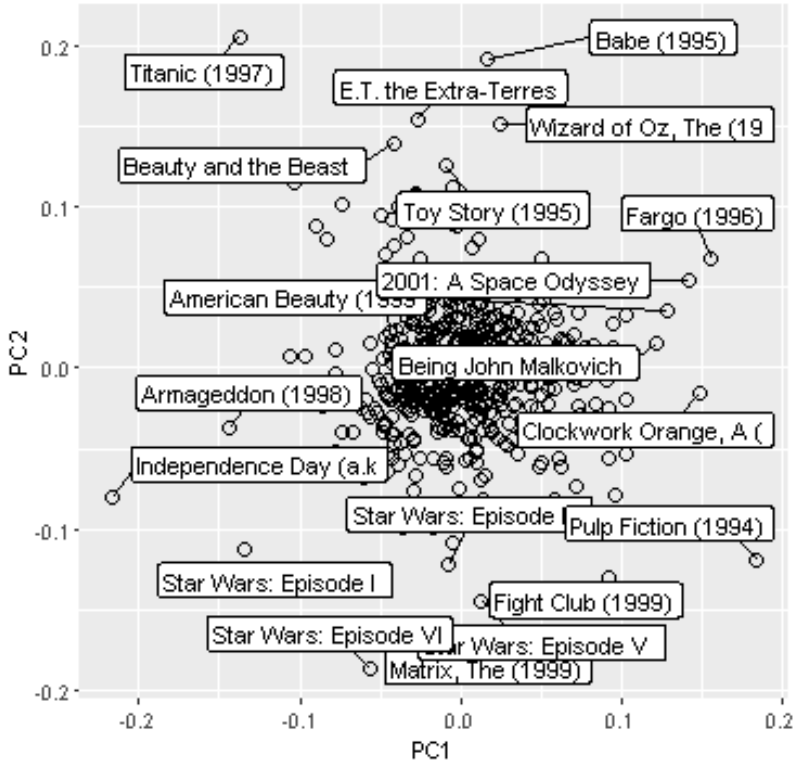
By performing the code above, we can find that 533 movies have 50% of all negative residuals.
We will choose them to improve prediction and create PCA model.

```
high_res_by_movie <- train_edx_wide%>%
  filter(residual_regular<(-1))%>%
  group_by(movieId)%>%
  summarize(n_negative=n())%>%
  top_n(533,n_negative)%>%
  .$movieId
```

We will also choose Users who rated $>= 20$ movies, we assume that this number is enough to understand their preferences.

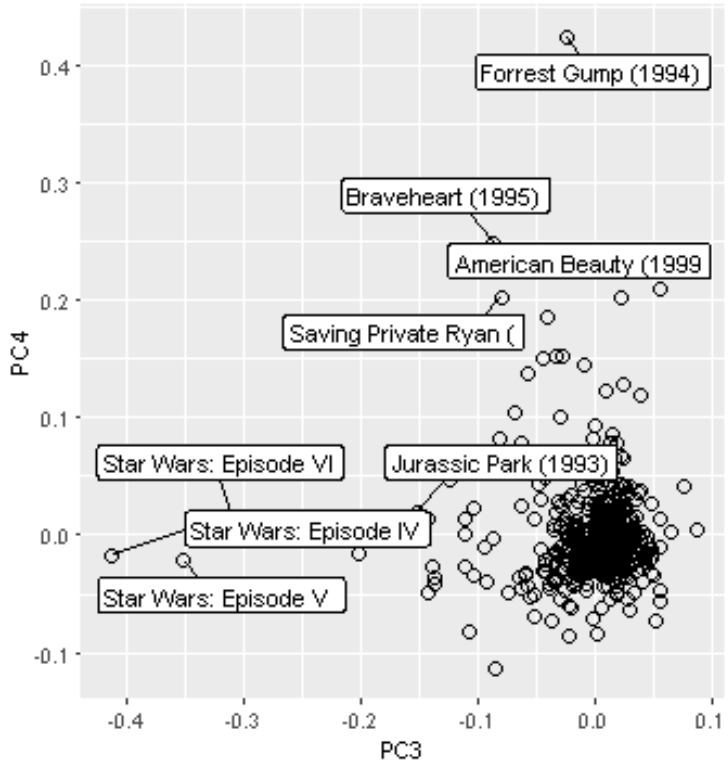
```
x <- train_edx_wide %>%
  filter(movieId %in% high_res_by_movie) %>%
  group_by(userId) %>%
  filter(n() >= 20) %>%
  ungroup() %>%
  select(title, userId, rating) %>%
  pivot_wider(names_from = "title", values_from = "rating") %>%
  as.matrix()

rownames(x)<- x[,1]
x <- x[,-1]

x <- sweep(x, 2, colMeans(x, na.rm = TRUE)) #Exclude Movie effect
x <- sweep(x, 1, rowMeans(x, na.rm = TRUE)) #Exclude User effect

x[is.na(x)] <- 0
pca <- prcomp(x)
```

Looking at Principal Component 1 (PC1) and Principal Component 2 (PC2), we can clearly say that PC1 explains differences between people who like Blockbusters (Armageddon, Independence Day) vs those who prefer Independent/Art movies (Pulp Fiction, Clockwork Orange). PC2 shows difference between movies for the whole family (Toy Story) vs movies for adults (Fight Club).

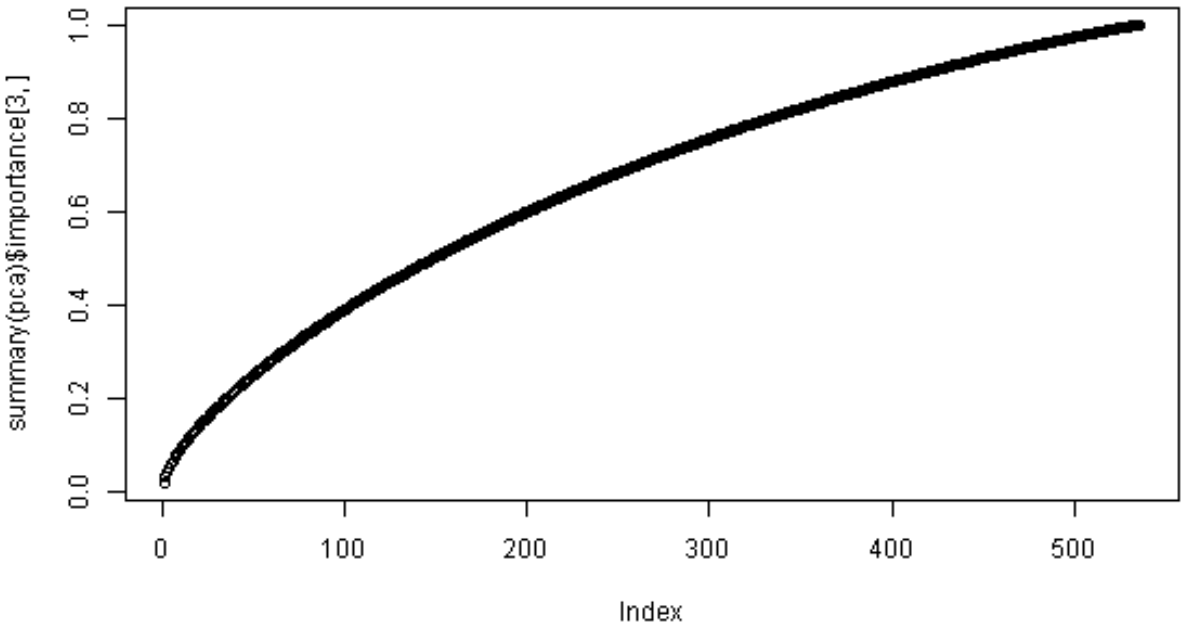


Meanwhile PC3 separates Nerd/Sci-Fi movies from others and PC4 - Oscar movies (tough ones) from Comedy movies.

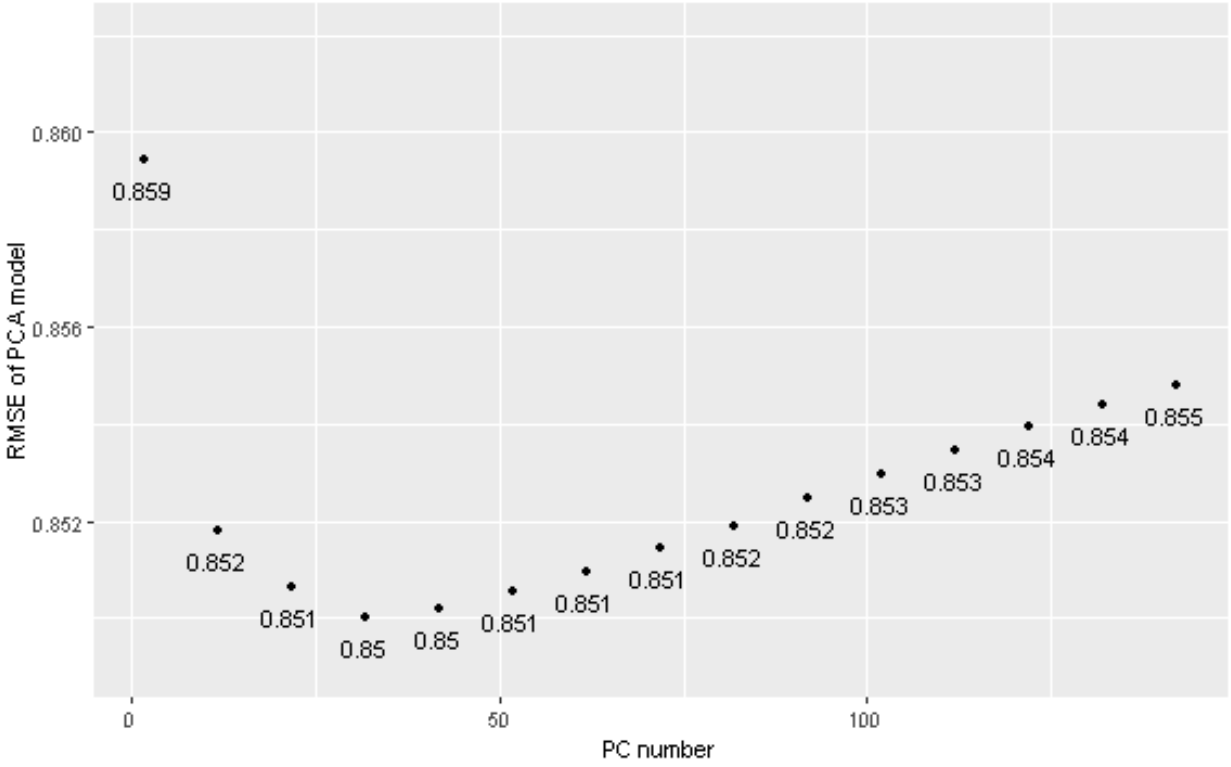
```
## Warning: ggrepel: 32 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



By plotting the importance of each Principal Component (PC), we can see that around 150 PCs can explain 50% of residuals fluctuations.



It's quite a high number of components to add to our model.
Let's check how RMSE improves when we add every 10 PCs to the model (starting with 2 PCs).



From the graph above we can find that RMSE is minimal when we use 32 principal components.

```
best_PC_number <- PC_n[which.min(rmses_pca)]
print(best_PC_number)
```

```
## [1] 32
```

We are going to use these 32 principal components for our model which are expected to improve the model by extra 0.01.
With code below we calculate $p * q$ matrix of estimated residuals.

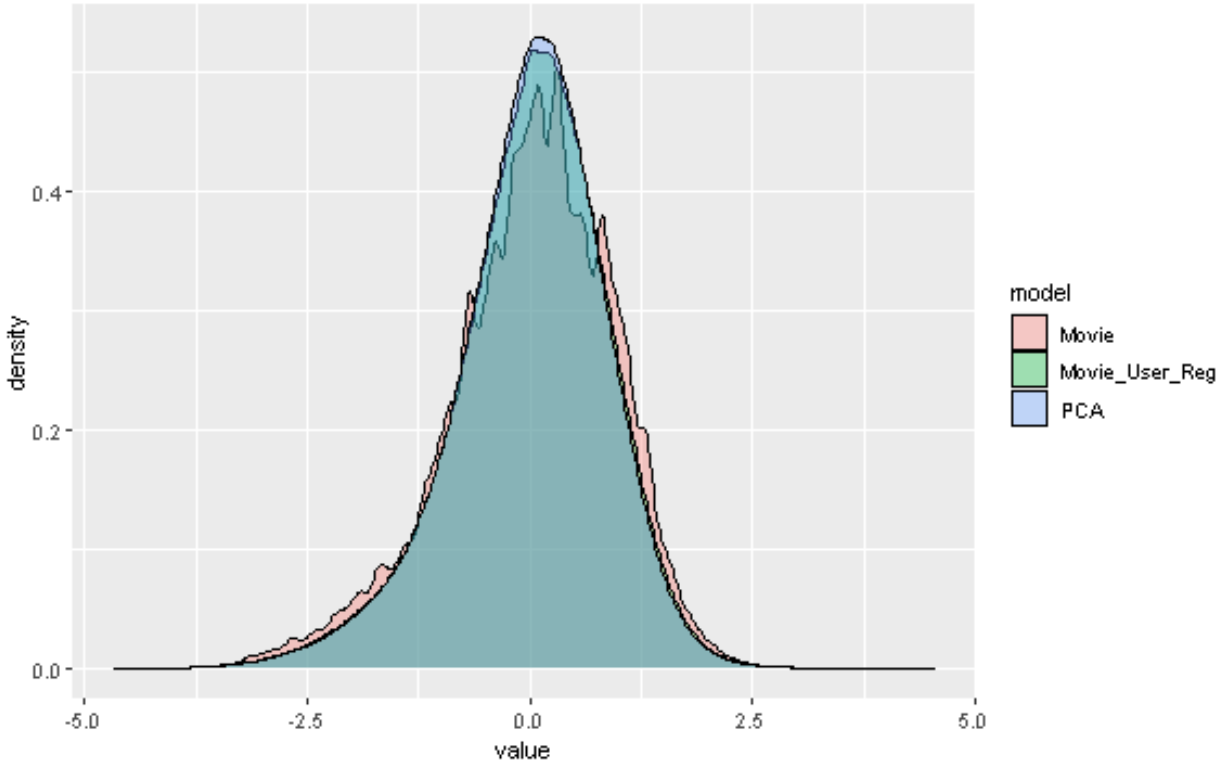
```
p<-32

p_q_matrix <- pca$x[,1:p]%*%t(pca$rotation[,1:p])
colnames(p_q_matrix) <- rownames(pca$rotation)

p_q_matrix <- tidy_matrix(p_q_matrix, row.name ="userId", col.name = "title",
                          value.name="residual_estimate")
p_q_matrix$userId <- as.integer(p_q_matrix$userId)
```

And our new model will look like:

$$Y_{u,i} = \mu + b_i + b_u + p_{u,1} * q_{1,i} + ... + p_{u,32} * q_{32,i} + e_{u,i}$$

where u-user, i-movie.

The residuals distribution of the new PCA model has improved on the *test_edx* dataset:



And RMSE of the final PCA model will be:

```
prediction_PCA <- test_edx%>%
  left_join(b_i_regularized, by='movieId')%>%
  left_join(b_u_regularized, by='userId')%>%
  left_join(p_q_matrix, by=c('userId','title'))%>%
  mutate(residual_estimate = replace(residual_estimate,is.na(residual_estimate),0))%>%
  mutate(pred = mu+b_i+b_u+residual_estimate)%>%
  mutate(if_else(pred>5,5,if_else(pred<0,0,pred)))%>% #Forecast can't be >5 or <0
  .$pred

prediction_PCA_RMSE<-RMSE(prediction_PCA, test_edx$rating)
print(prediction_PCA_RMSE)
```

```
## [1] 0.8500197
```

Prediction has been improved slightly, but still better than regularized model.

```
models <- t(data.frame(Simple_Avg = simple_average,
                       Movie_effect = model_with_movie_effect_RMSE,
                       Movie_User_effect = model_with_movie_user_effect_RMSE,
```

```
                      Regularized = regularized_model_prediction,
                      PCA = prediction_PCA_RMSE))
colnames(models)<-"RMSE"
knitr::kable(models, caption = "Models comparison by RMSE")
```

Table 7: Models comparison by RMSE

	RMSE
Simple_Avg	1.0605613
Movie_effect	0.9439868
Movie_User_effect	0.8666408
Regularized	0.8659624
PCA	0.8500197

We have found our best model ! Principal component model (PCA) !

4 Final model application

Now we are ready to apply our PCA model to the *Validation* dataset to see how well it predicts Movie ratings.

```
prediction_Final<-validation%>%
  left_join(b_i_regularized, by='movieId')%>%
  left_join(b_u_regularized, by='userId')%>%
  left_join(p_q_matrix, by=c('userId','title'))%>%
  mutate(residual_estimate=replace(residual_estimate,is.na(residual_estimate),0),
         b_i=replace(b_i,is.na(b_i),0),
         b_u=replace(b_u,is.na(b_u),0))%>%
  mutate(pred=mu+b_i+b_u+residual_estimate)%>%.$pred

prediction_Final_RMSE<-RMSE(prediction_Final,validation$rating)
print(prediction_Final_RMSE)
```

```
## [1] 0.8496295
```

$RMSE = 0.8496$ is a great improvement vs the model with Simple average and even Movie/User effects.