MovieLensProjectSubmission

L. Chenin

14/02/2020

Introduction

This is a report on the MovieLens Project. The data set was obtained from http://files.grouplens.org/datasets/movielens/ml-10m.zip and partitioned in an edx set and a validation set by eDX Harvard Data Science Team. This data set is the basis to check the proposed predicted movie ratings and RMSE score(s), as derived from the Netflix challenge, and is targeted to be below 0.86490.

The edX course 8 material was used to take into account movie, and user effects with penalized least squares to investigate the impact on the RMSE scores. However, as the scores obtained were close to the requirement, we investigated the effect of date when movies are rated, the genres the movie is classified and the year when movie was available to check further their impacts on the RMSE scores.

They are summarized as follows:

method	RMSE	edX RMSE
Just the average	1.0605613	1.0612018
Movie Effect Model	0.9439868	0.9439087
Movie + Effect Model	0.8666408	0.8653488
Regularized Movie Effect Model	0.9439217	0.9438521
Regularized Movie + User Effect Model	0.8659626	0.8648201
Reg Movie + User + Date Effect Model	0.8658438	0.8647012
Reg Movie + User + Date + Genres Effect Model	0.8655123	0.8643146
Reg Movie + User + Date + Genres + Year Effect Model	0.8653194	0.8641310

A further step of prediction was tried with the SVD and PCA analysis but remains to be worked out, as questions arised that were not solved, i.e. recompose the data frame with the predictions, apply the predictions to the test set and compute the RMSE score. Request for assistance will be posted to get hints.

Analysis

The analysis of the edx set shows that it has 9,000,055 observations of 6 variables, namely

- userId for 69,878 distinct users
- movieId for 10,677 distinct movies
- 9,000,055 rating distributed in 10 sequenced ratings, 0.5 to 5.0 as follows:

rating	quantity
0.5	85,374

rating	quantity
1.0	345,679
1.5	106,426
2.0	711,422
2.5	330,010
3.0	2,121,240
3.5	791,624
4.0	2,588,430
4.5	526,736
5.0	1,390,114

- timestamp for 6,519,590 distinct timestamps
- title for 10,676 distinct titles
- genres for 797 distinct combinations of genres.

```
################################
# 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")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
# MovieLens 10M dataset:
# 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"))
library(stringr)
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.ch
                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")
```

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

```r
# 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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)

n_distinct(edx$movieId)
```

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

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
n_distinct(edx$genres)
```

```
## [1] 797
```

```
n_distinct(edx$timestamp)
```

```
## [1] 6519590
```

```
n_distinct(edx$rating)
```

```
## [1] 10
```

```
n_distinct(edx$title)
```

```
## [1] 10676
```

```
# let us create an additional partition of training and test sets from the provided edx datasset

set.seed(755, sample.kind = "Rounding") #  in R 3.6 or later
```

We check the models to implement with a data partition of edx, and the first steps of checking the movie effect, and combined movie and user effects:

```
## Warning in set.seed(755, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

# to make sure we don't include users and movies in the test set that do not appear in
# the training set,
```

```r
# 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")

# Define the RMSE as per the Netflix challenge

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

# Define mu_hat, the average of all ratings

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

```
## [1] 3.512527
```

```r
# perform a prediction with all unknown ratings equal to mu_hat

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

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

```r
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)

# compute movie effect b_i. Drop the hat of mu_hat

mu <- mean(train_set$rating)

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

# predict with y_hat = mu_hat + b_i

predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

RMSE(test_set$rating, predicted_ratings)
```

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

```r
model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
```

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

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

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868

```
# compute user effect b_u

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

# predict with mu + b_i + b_u

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

RMSE(test_set$rating, predicted_ratings)
```

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

```
model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408

```
lambda <- 2.25
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

# Do we improve our results? not that much !

predicted_ratings <- test_set %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
```

```
  mutate(pred = mu + b_i) %>%
  pull(pred)

RMSE(predicted_ratings, test_set$rating)
```

Regularization with penalized least squares. Let's compute these regularized estimates of b_i using an optimal = 2.25. See below for the code getting an optimal = 4.75 when combining movie and user effect.

```
## [1] 0.9439217
```

```
model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie Effect Model",
                                     RMSE = model_3_rmse ))

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

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Regularized Movie Effect Model	0.9439217

```
lambdas <- seq(4.0, 5.5, 0.25)

rmses <- sapply(lambdas, function(l){
        mu <- mean(train_set$rating)
        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))

        predicted_ratings <- test_set %>%
           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_set$rating))
       })
```

compute RMSE with betwwen 4.0 and 5.5 that has been found optimal at 4.75 for b_i and b_u.

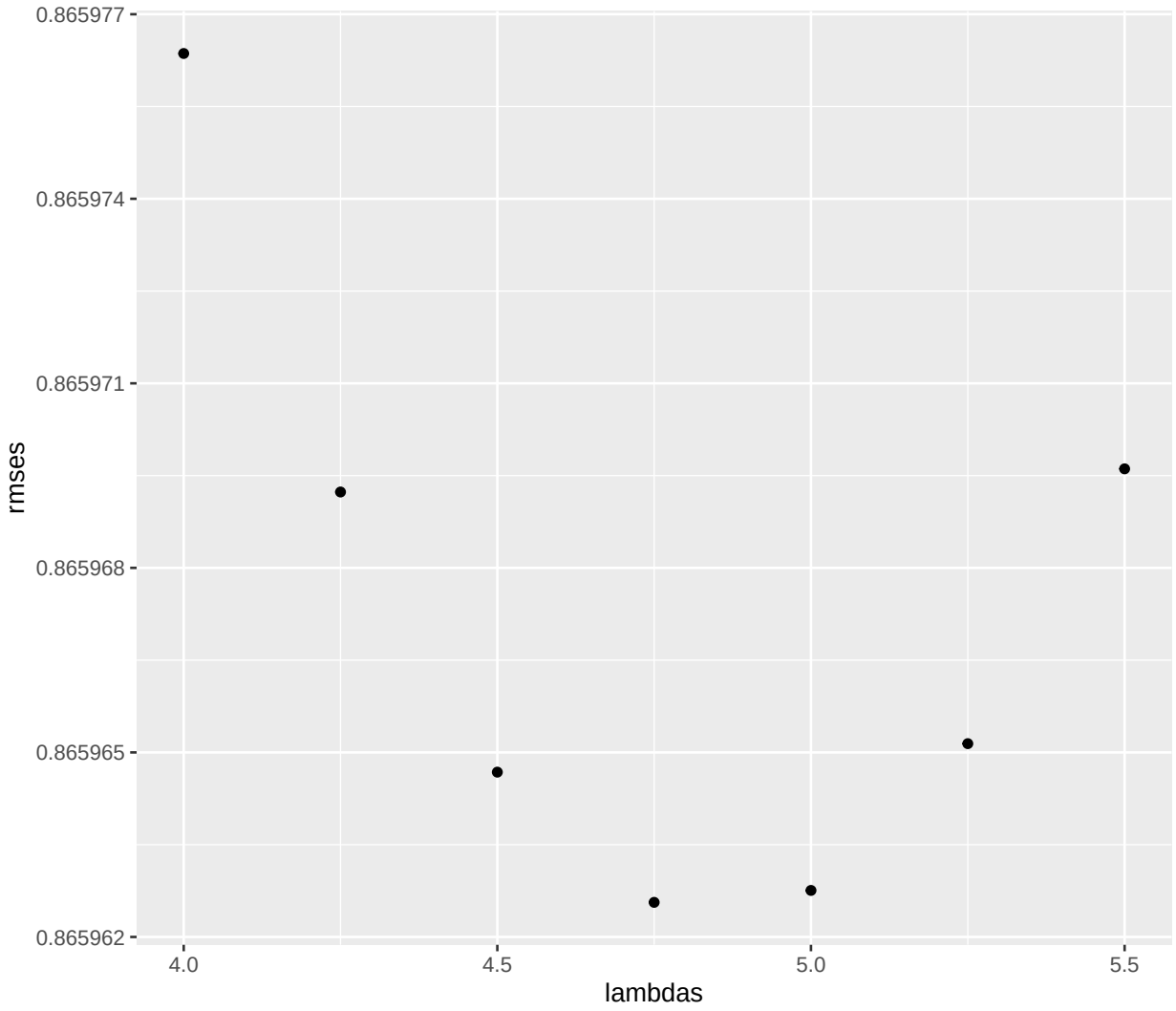

Figure 1: Plot optimal lambda

we plot the lambdas

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

For the full model, the optimal is:

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

```
lambda <- 4.75

mu <- mean(train_set$rating)

b_i <- train_set %>%
```

```
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

b_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

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

RMSE(predicted_ratings, test_set$rating)
```

```
## [1] 0.8659626
```

```
model_4_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User Effect Model",
                                     RMSE = model_4_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659626

```
# Date effect
# refine by adding  d_u, the effect of the date the movie was watched with groupings per week

library("lubridate")
```

to access the date we use the lubridate package on the timestamp to get the weeks when the movies are rated:

```
##
## 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
```

```
test_set <- test_set  %>% mutate(date = round_date(as_datetime(timestamp), unit = "week"))
train_set <- train_set %>% mutate(date = round_date(as_datetime(timestamp), unit = "week"))

# plot the average rating for each week against date
```

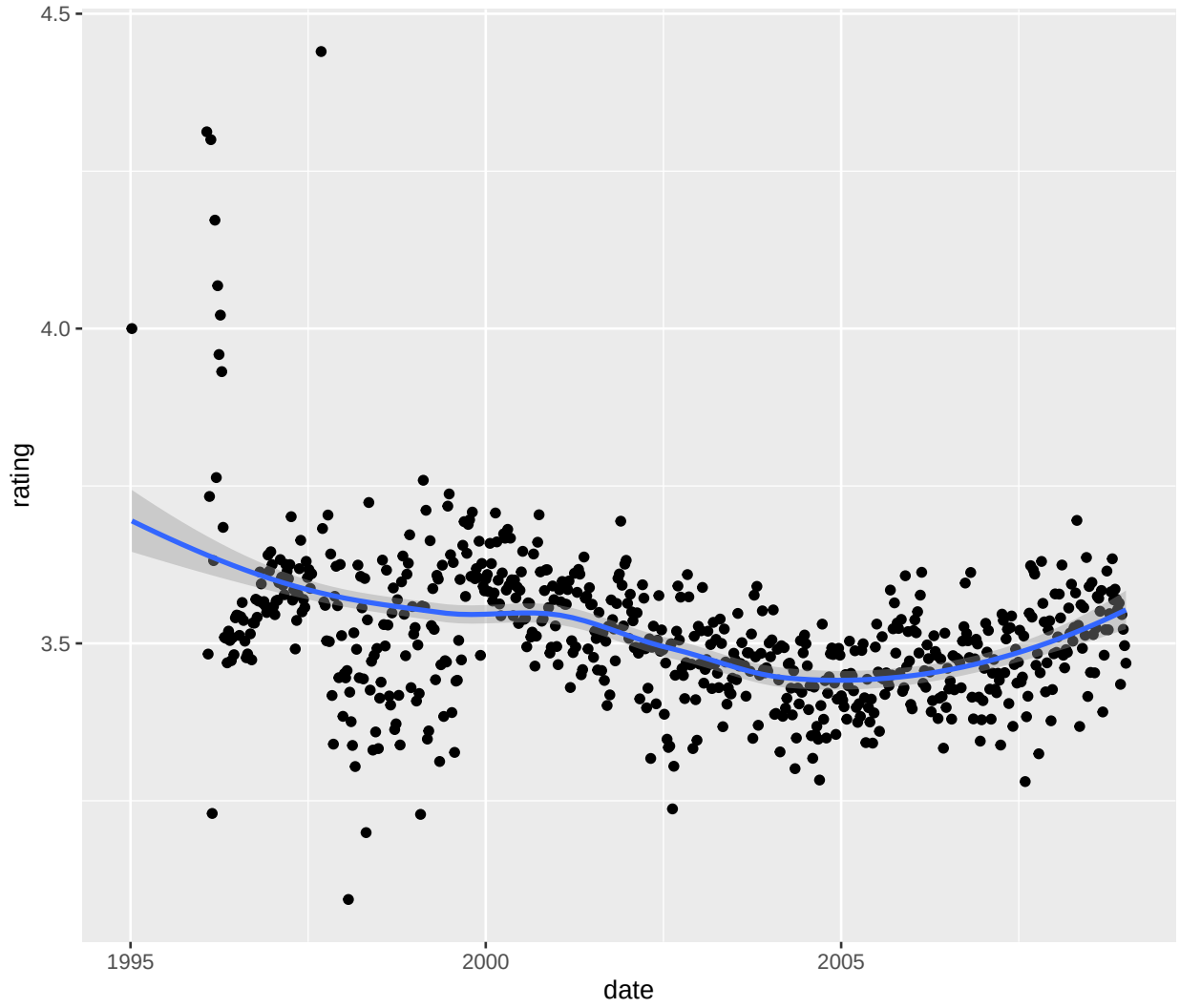

Figure 2: Rating versus date

plot date versus rating

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

```
d_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  group_by(date) %>%
  summarize(d_u = sum(rating - b_i - b_u -mu)/(n() + lambda))

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

RMSE(predicted_ratings, test_set$rating)
```

compute the date effect

```
## [1] 0.8658438
```

```
model_5_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Reg Movie + User + Date Effect Model",
                                     RMSE = model_5_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659626
Reg Movie + User + Date Effect Model	0.8658438

Genres Effect

plot the error bar plots of the average and standard error for each category of genres as there are 796 genres we limit to a sample of 80 genres, or 10%.

```
# final computation with optimal    =4.75

lambda <- 4.75

g_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(d_u, by="date") %>%
```

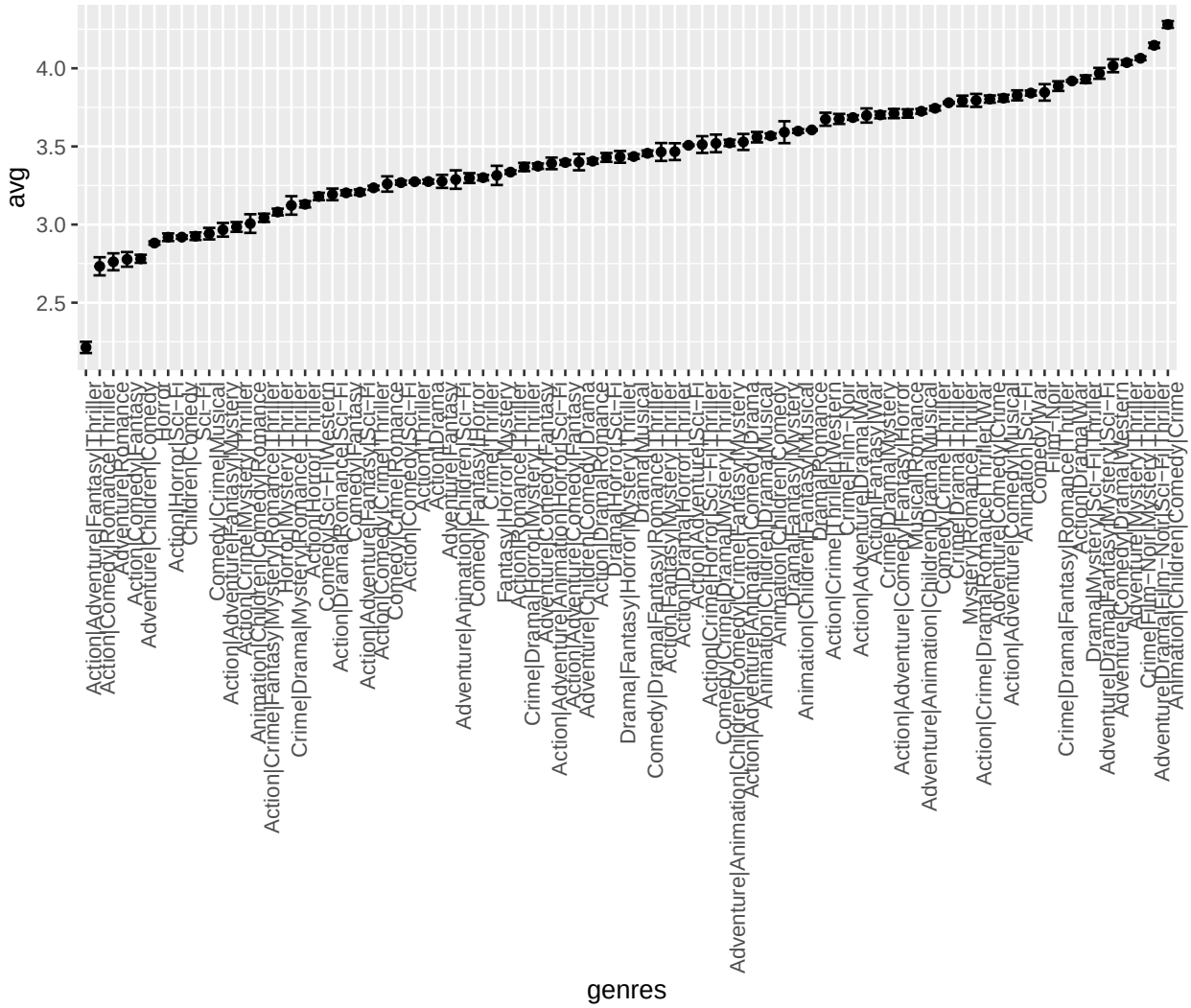

Figure 3: Average & std error versus 80 genres

```
  group_by(genres) %>%
  summarize(g_u = sum(rating - d_u - b_i - b_u -mu)/(n() + lambda))

predicted_ratings <-
  test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(d_u, by ="date") %>%
  left_join(g_u, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + d_u + g_u) %>%
  pull(pred)

RMSE(predicted_ratings, test_set$rating)
```

compute genres effect

```
## [1] 0.8655123
```

```
model_6_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Reg Movie + User + Date + Genres Effect Model",
                                     RMSE = model_6_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659626
Reg Movie + User + Date Effect Model	0.8658438
Reg Movie + User + Date + Genres Effect Model	0.8655123

```
test_set <- test_set  %>% mutate(year = str_extract(str_extract(test_set$title, "\\(\\d{4}\\)"), "\\d{4}
train_set <- train_set  %>% mutate(year = str_extract(str_extract(train_set$title, "\\(\\d{4}\\)"), "\\
```

Year effect. We investigate effect of year when movie is issued. Extract the year and mutate it.

plot the year versus average rating

plot the number of movies rated per year

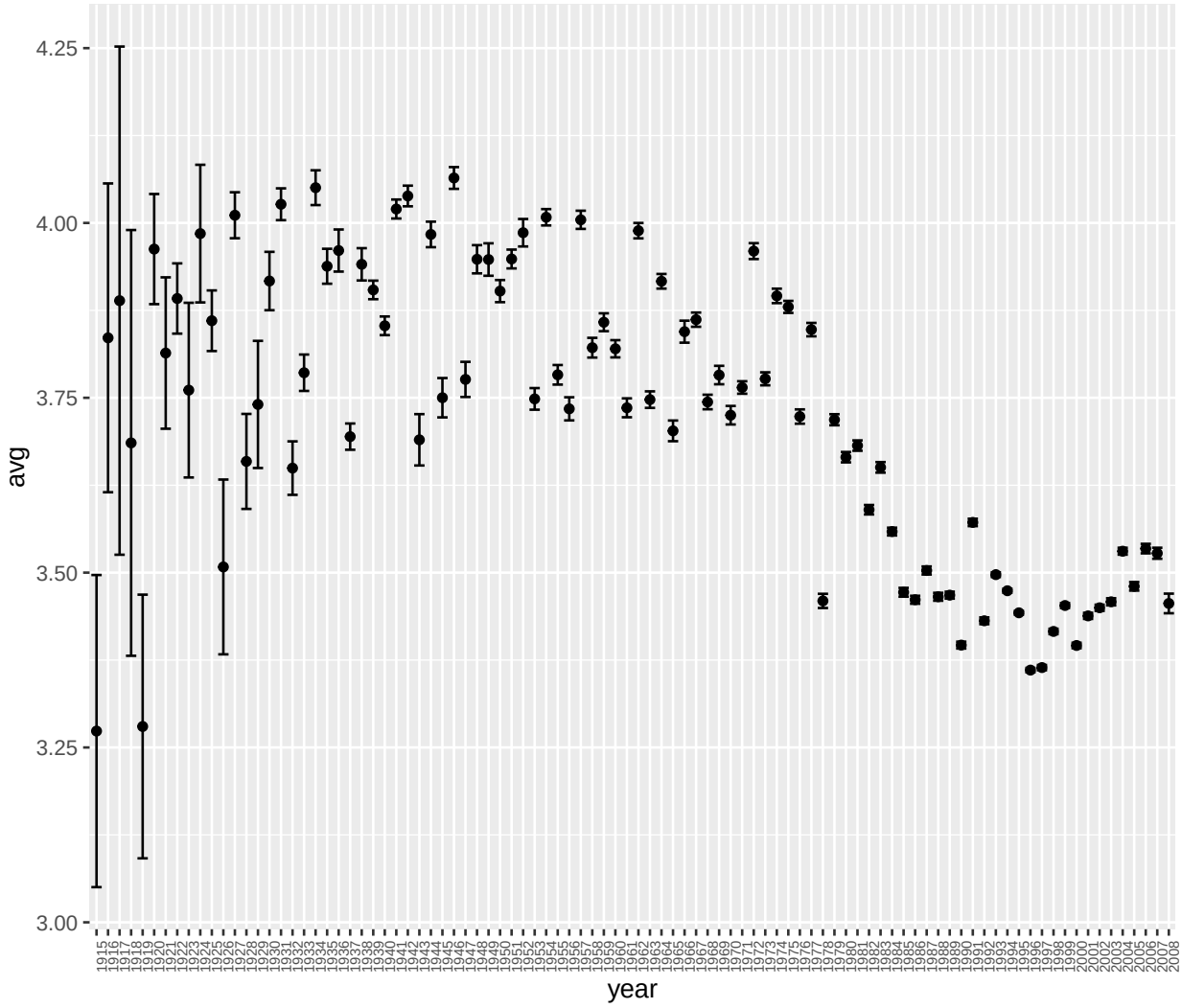

Figure 4: Rating average & std error versus year

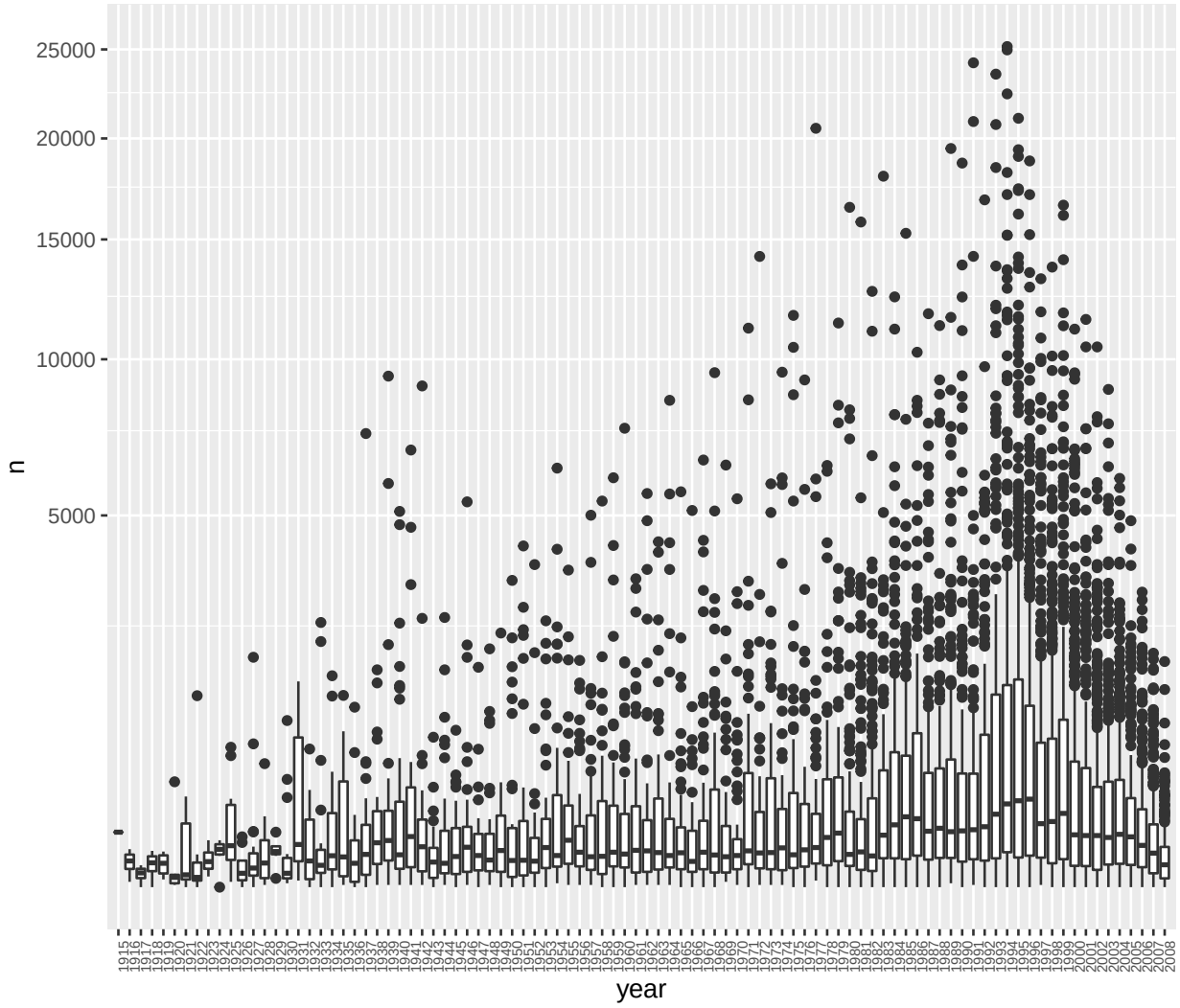

Figure 5: Number of movies, median & interquartile range versus year

```
y_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(d_u, by="date") %>%
  left_join(g_u, by="genres") %>%
  group_by(year) %>%
  summarize(y_u = sum(rating -g_u - d_u - b_i - b_u -mu)/(n() + lambda))

predicted_ratings <-
  test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(d_u, by ="date") %>%
  left_join(g_u, by = "genres") %>%
  left_join(y_u, by = "year") %>%
  mutate(pred = mu + b_i + b_u + d_u + g_u + y_u) %>%
  pull(pred)

RMSE(predicted_ratings, test_set$rating)
```

compute year effect

```
## [1] 0.8653194
```

```
model_7_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                data_frame(method="Reg Movie + User + Date + Genres + Year Effect Model",  RMSE = model_
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659626
Reg Movie + User + Date Effect Model	0.8658438
Reg Movie + User + Date + Genres Effect Model	0.8655123
Reg Movie + User + Date + Genres + Year Effect Model	0.8653194

```
# Define the RMSE as per the Netflix challenge

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

# Define mu_hat, the average of all ratings

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

issue the final predictions for the edx and validation data sets and compute the final RMSE scores

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

```
# perform a prediction with all unknown ratings equal to mu_hat

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

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

```
rmse_results_f <- tibble(method = "Just the average", edX_RMSE = naive_rmse)

# compute movie effect b_i. Drop the hat of mu_hat

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

# predict with y_hat = mu_hat + b_i

predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

RMSE(validation$rating, predicted_ratings)
```

```
## [1] 0.9439087
```

```
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                          data_frame(method="Movie Effect Model",
                                     edX_RMSE = model_1_rmse ))
rmse_results_f %>% knitr::kable()
```

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

```
# compute user effect b_u

user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```r
# predict with y_hat = mu_hat + b_i + b_u

predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

RMSE(validation$rating, predicted_ratings)
```

```
## [1] 0.8653488
```

```r
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                          data_frame(method="Movie + User Effects Model",
                                     edX_RMSE = model_2_rmse ))
rmse_results_f %>% knitr::kable()
```

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488

```r
# Regularization

# Let's compute these regularized estimates of b_i using   =  2.25

lambda <- 2.25
mu <- mean(edx$rating)
movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())


# Do we improve our results? not much

predicted_ratings <- validation %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.9438521
```

```r
model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                          data_frame(method="Regularized Movie Effect Model",
                                     edX_RMSE = model_3_rmse ))

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

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438521

```
# compute RMSE with  optimal at 4.75

lambda <- 4.75

mu <- mean(edx$rating)

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))

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

RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8648201
```

```
model_4_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                          data_frame(method="Regularized Movie + User Effect Model",
                                     edX_RMSE = model_4_rmse ))
rmse_results_f %>% knitr::kable()
```

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438521
Regularized Movie + User Effect Model	0.8648201

```
# refine by adding  d_u with groupings per week

validation <- validation %>% mutate(date = round_date(as_datetime(timestamp), unit = "week"))
edx <- edx %>% mutate(date = round_date(as_datetime(timestamp), unit = "week"))
```

```
d_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  group_by(date) %>%
  summarize(d_u = sum(rating - b_i - b_u -mu)/(n() + lambda))

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

RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8647012
```

```
model_5_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                            data_frame(method="Reg Movie + User + Date Effect Model",
                                       edX_RMSE = model_5_rmse ))
rmse_results_f %>% knitr::kable()
```

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438521
Regularized Movie + User Effect Model	0.8648201
Reg Movie + User + Date Effect Model	0.8647012

```
# refine by adding  g_u with genres

g_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(d_u, by="date") %>%
  group_by(genres) %>%
  summarize(g_u = sum(rating - d_u - b_i - b_u -mu)/(n() + lambda))

predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(d_u, by ="date") %>%
  left_join(g_u, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + d_u + g_u) %>%
  pull(pred)
```

```
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8643146
```

```
model_6_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                            data_frame(method="Reg Movie + User + Date + Genres Effect Model",
                                       edX_RMSE = model_6_rmse ))
rmse_results_f %>% knitr::kable()
```

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438521
Regularized Movie + User Effect Model	0.8648201
Reg Movie + User + Date Effect Model	0.8647012
Reg Movie + User + Date + Genres Effect Model	0.8643146

```
# Year effect

# investigate effect of year when movie is issued. Extract the year and mutate it.

validation <- validation  %>% mutate(year = str_extract(str_extract(validation$title, "\\(\\d{4}\\)"),
edx <- edx  %>% mutate(year = str_extract(str_extract(edx$title, "\\(\\d{4}\\)"), "\\d{4}"))

# compute year effect

y_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(d_u, by="date") %>%
  left_join(g_u, by="genres") %>%
  group_by(year) %>%
  summarize(y_u = sum(rating -g_u - d_u - b_i - b_u -mu)/(n() + lambda))

predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(d_u, by ="date") %>%
  left_join(g_u, by = "genres") %>%
  left_join(y_u, by = "year") %>%
  mutate(pred = mu + b_i + b_u + d_u + g_u + y_u) %>%
  pull(pred)

RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.864131
```

```r
model_7_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results_f <- bind_rows(rmse_results_f,
                            data_frame(method="Reg Movie + User + Date + Genres + Year Effect Model",
                                       edX_RMSE = model_7_rmse ))
rmse_results_f %>% knitr::kable()
```

method	edX_RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438521
Regularized Movie + User Effect Model	0.8648201
Reg Movie + User + Date Effect Model	0.8647012
Reg Movie + User + Date + Genres Effect Model	0.8643146
Reg Movie + User + Date + Genres + Year Effect Model	0.8641310

```r
# Matrix factorization

# edit a workable subset of the train_set as the whole train_set failed.

train_mat <- train_set %>%
  group_by(movieId) %>%
  filter(n() >= 1000) %>% ungroup() %>%
  group_by(userId) %>%
  filter(n() >= 100) %>% ungroup()

# assemble the matrix

y <- train_mat %>%
  select(userId, movieId, rating) %>%
  spread(movieId, rating) %>%
  as.matrix()

#  add row names and column names

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

# get distinct movieId and title

movie_titles <- train_set %>%
  select(movieId, title) %>%
  distinct()

colnames(y) <- with(movie_titles, title[match(colnames(y), movieId)])

# convert y to residuals substracting b_i and b_u

y <- sweep(y, 2, colMeans(y, na.rm = TRUE))
```

```r
y <- sweep(y, 1, rowMeans(y, na.rm = TRUE))

# compute the decomposition making the residuals with NAs equal to 0.

y[is.na(y)] <- 0

pca <- prcomp(y)    # command failed as "Error: cannot allocate vector of size 3.4 Gb but worked with fi

dim(pca$rotation)
```

investigate SVD and PCA with the following piece of code

```
## [1] 1654 1654
```

```r
dim(pca$x)
```

```
## [1] 17011  1654
```

plot variability for the vectors

```r
s <- svd(y)
names(s)
```

compute SVD decomposition of y

```
## [1] "d" "u" "v"
```

```r
# What proportion of the total variability is explained by the first 1,500 columns of  YV ?

sum(s$d[1:1500]^2) / sum(s$d^2)    # to get to 98% !!!
```

```
## [1] 0.9814304
```

```r
y_hat <- with(s,sweep(u[, 1:1500], 2, d[1:1500], FUN="*") %*% t(v[, 1:1500]))


#### there remains to convert this matrix to the useful dataframe of movie, users and ratings to arrive

# call to ggrepel

library(ggrepel)


pcs <- data.frame(pca$rotation, name = colnames(y))
```

plot clusters

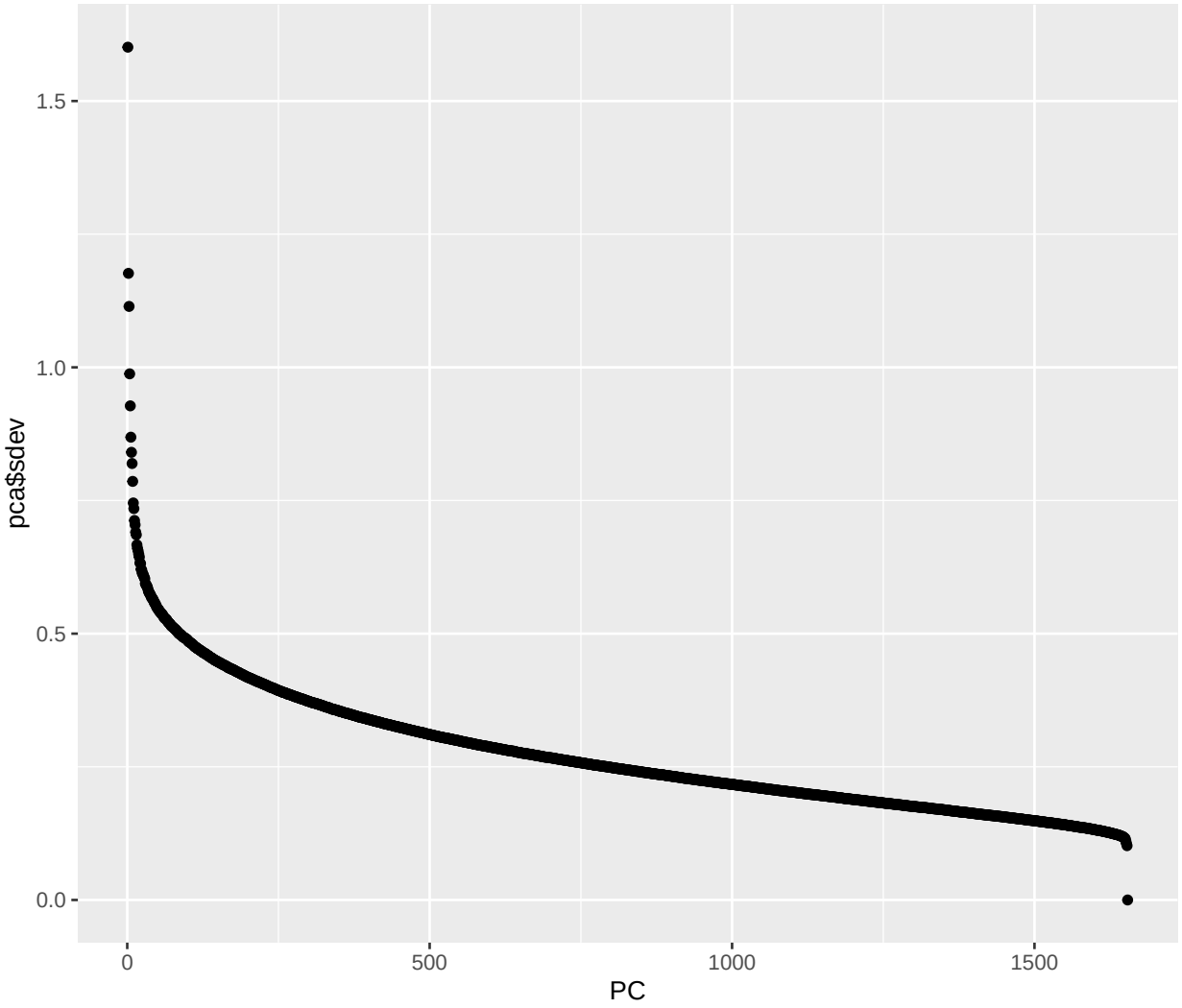

Figure 6: Principal Component variability

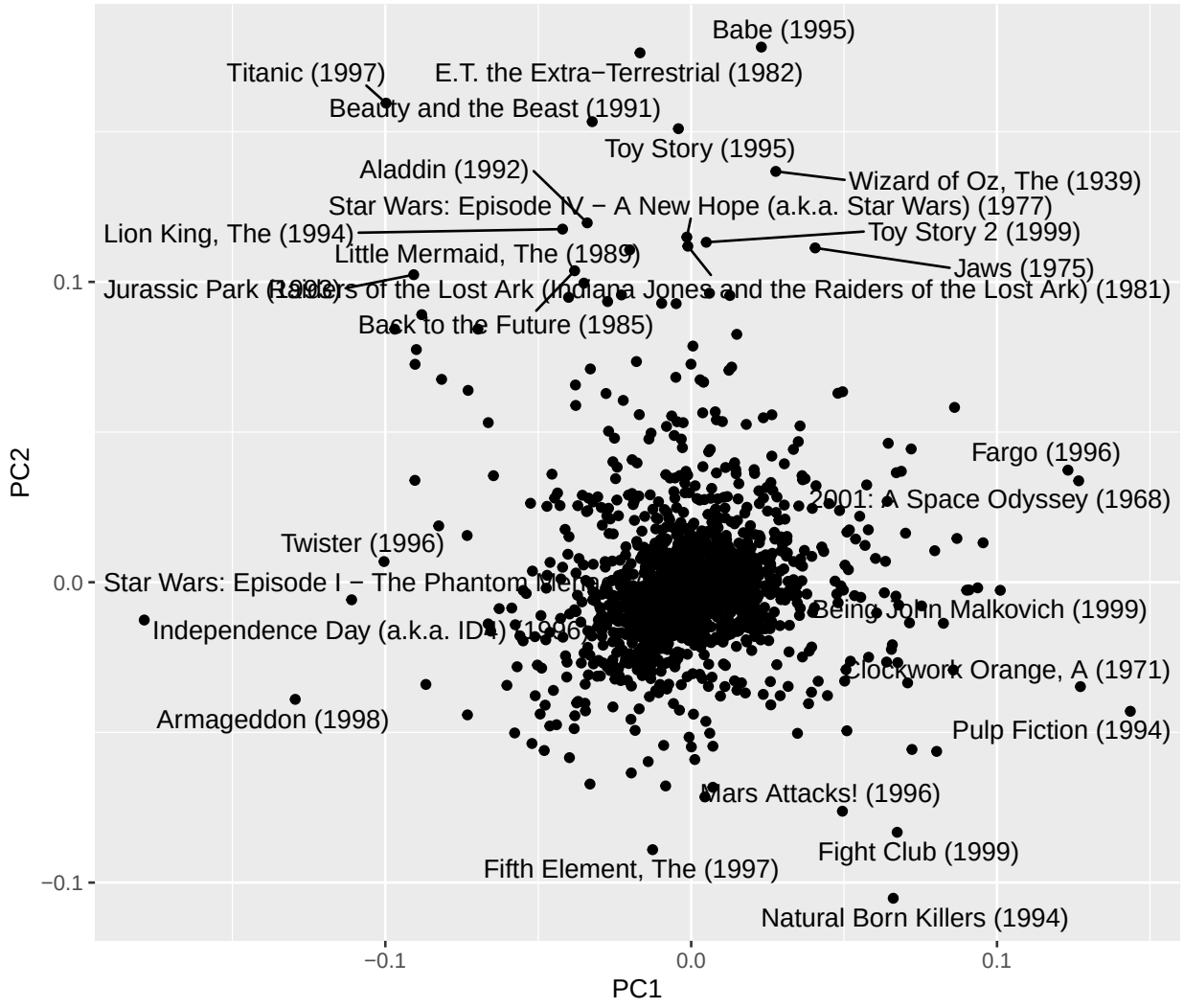

Figure 7: First two components movies trends

```
pcs %>% select(name, PC1) %>% arrange(PC1) %>% slice(1:10)
```

identify 10 movies in the possible clusters out of the central cloud

```
##                                                 name         PC1
## 1               Independence Day (a.k.a. ID4) (1996) -0.17883360
## 2                                  Armageddon (1998) -0.12945577
## 3  Star Wars: Episode I - The Phantom Menace (1999) -0.11101816
## 4                                    Twister (1996) -0.10042993
## 5                                    Titanic (1997) -0.09976858
## 6                                Pretty Woman (1990) -0.09693314
## 7                               Jurassic Park (1993) -0.09072221
## 8                                    Top Gun (1986) -0.09032904
## 9                                      Ghost (1990) -0.09028207
## 10                                     Speed (1994) -0.08985496
```

```
pcs %>% select(name, PC1) %>% arrange(desc(PC1)) %>% slice(1:10)
```

```
##                                                                                 name
## 1                                                                 Pulp Fiction (1994)
## 2                                                       Clockwork Orange, A (1971)
## 3                                                  2001: A Space Odyssey (1968)
## 4                                                                 Fargo (1996)
## 5                                                  Being John Malkovich (1999)
## 6                                                       American Beauty (1999)
## 7                                                        Apocalypse Now (1979)
## 8                                                           Taxi Driver (1976)
## 9                                                         Blade Runner (1982)
## 10 Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
##           PC1
## 1  0.14358769
## 2  0.12727203
## 3  0.12670659
## 4  0.12319992
## 5  0.10106804
## 6  0.09545621
## 7  0.09367728
## 8  0.09084062
## 9  0.09007667
## 10 0.08689264
```

```
pcs %>% select(name, PC2) %>% arrange(PC2) %>% slice(1:10)
```

```
##                                  name         PC2
## 1           Natural Born Killers (1994) -0.10521525
## 2             Fifth Element, The (1997) -0.08903771
## 3                     Fight Club (1999) -0.08324411
## 4                Mars Attacks! (1996) -0.07623079
## 5                      AlienÂ³ (1992) -0.07148075
## 6            From Dusk Till Dawn (1996) -0.06826907
```

```
## 7               Starship Troopers (1997) -0.06782287
## 8  Ace Ventura: When Nature Calls (1995) -0.06719847
## 9                 Johnny Mnemonic (1995) -0.06349867
## 10      Robin Hood: Men in Tights (1993) -0.05971885
```

```
pcs %>% select(name, PC2) %>% arrange(desc(PC2)) %>% slice(1:10)
```

```
##                                                        name       PC2
## 1                                              Babe (1995) 0.1781759
## 2                          E.T. the Extra-Terrestrial (1982) 0.1762681
## 3                                           Titanic (1997) 0.1595142
## 4                                Beauty and the Beast (1991) 0.1533494
## 5                                          Toy Story (1995) 0.1510096
## 6                                Wizard of Oz, The (1939) 0.1367918
## 7                                           Aladdin (1992) 0.1196554
## 8                                      Lion King, The (1994) 0.1175703
## 9  Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 0.1149157
## 10                                        Toy Story 2 (1999) 0.1131931
```