Northeastern University
College *of* Engineering

Data Mining in Engineering
IE 7275 Course Project

Movie Recommendation System

Group 3
Min Gong
Wei Chieh Wang

(8573505361)
(8579303520)

gong.mi@husky.neu.edu
wang.weichi@husky.neu.edu

Percentage of Effort Contributed by Student 1: _____50%_____

Percentage of Effort Contributed by Student 2: _____50%_____

Signature of Student 1:_____

Signature of Student 2:_____

Submission Date: _____12/2/2019_____

Contents

1. Problem Setting
2. Data Definition
3. Data Sources
4. Data description
5. Data Exploration
6. Data Mining Tasks
7. Data Mining Models
8. Performance Evaluation
9. Project Results
10. Impact of Project Outcomes
11. Appendix: R codes of project

1. Problem Setting

Nowadays, auto recommendations are everywhere online in our daily life. Movies, TV programs, music, online shopping and so on.

We chose the movie recommendation system as our data mining project.

2. Project Definition

By using unsupervised learning methods, we want to analyze each user's watching and rating history, find patterns and use proper models to recommend potentially liked movies for him/her. Specifically, for a new user, who have no rating history in our data set, we take our own-built searching and sorting based on user's input preferences method to find potential recommendations; for an old user, we take user-based collaborative filtering to find "neighbors" who have similar tastes and then recommend the movies they have watched but our client (the old user who wants recommendations) have not yet.

3. Data Description

We have two files for this project.

The file **movies.csv** contains information about moiveId, title, year(the years it was released), genres(animation, adventure, action…) for 10327 movies.

The file **ratings.csv** contains information about userId, movieId(the movie that was rated by the user), ratings(the user give to the movie) for 668 users.

4. Data sources

https://drive.google.com/file/d/1Dn1BZD3YxgBQJSIjbfNnmCFlDW2jdQGD/view

5. Data Exploration

Since the format of raw data in **movies.csv** is hard to handle in the following data mining process, we transformed it in excel at first. And for following processing, we create dummy variables for predictors of genres (categorical). See Figure 1.

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance

movieId	title	year	genres				
1	Toy Story	1995	Adventure	Animation	Children	Comedy	Fantasy
2	Jumanji	1995	Adventure	Children	Fantasy		
3	Grumpier Ol	1995	Comedy	Romance			
4	Waiting to E	1995	Comedy	Drama	Romance		
5	Father of the	1995	Comedy				
6	Heat	1995	Action	Crime	Thriller		
7	Sabrina	1995	Comedy	Romance			
8	Tom and Huc	1995	Adventure	Children			
9	Sudden Deat	1995	Action				
10	GoldenEye	1995	Action	Adventure	Thriller		

Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama
0	1	1	1	1	0	0	0
0	1	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0

Figure1. data preprocessing for movies.csv

In **ratings.csv**, we have a variable called timestamp. We think it not important for our analysis, so ignore it in the following process.

After preprocessing, we want to find best movies (based on ratings by users) in each genre or in each time period. This could do a help for us to understand raw data and offer us a reference for recommendations.

we set a filter for candidate movies in the rankings in case of a small number of ratings dominates the result. For example, if a movie was just rated by one user, but got a 5, it might get a really high rank which is not convincible. Thus, we only take the movies that received

more than 5% number of ratings in the same genre or time period. For candidates over the threshold, we take average rating scores as the measure to rank them.

We take boxplots to visualize the ratings for top10 movies which can not only give average but also variation information for each movie's ratings.

Here are some examples of best movies in some time period or of some genres in figure 2 and figure 3.

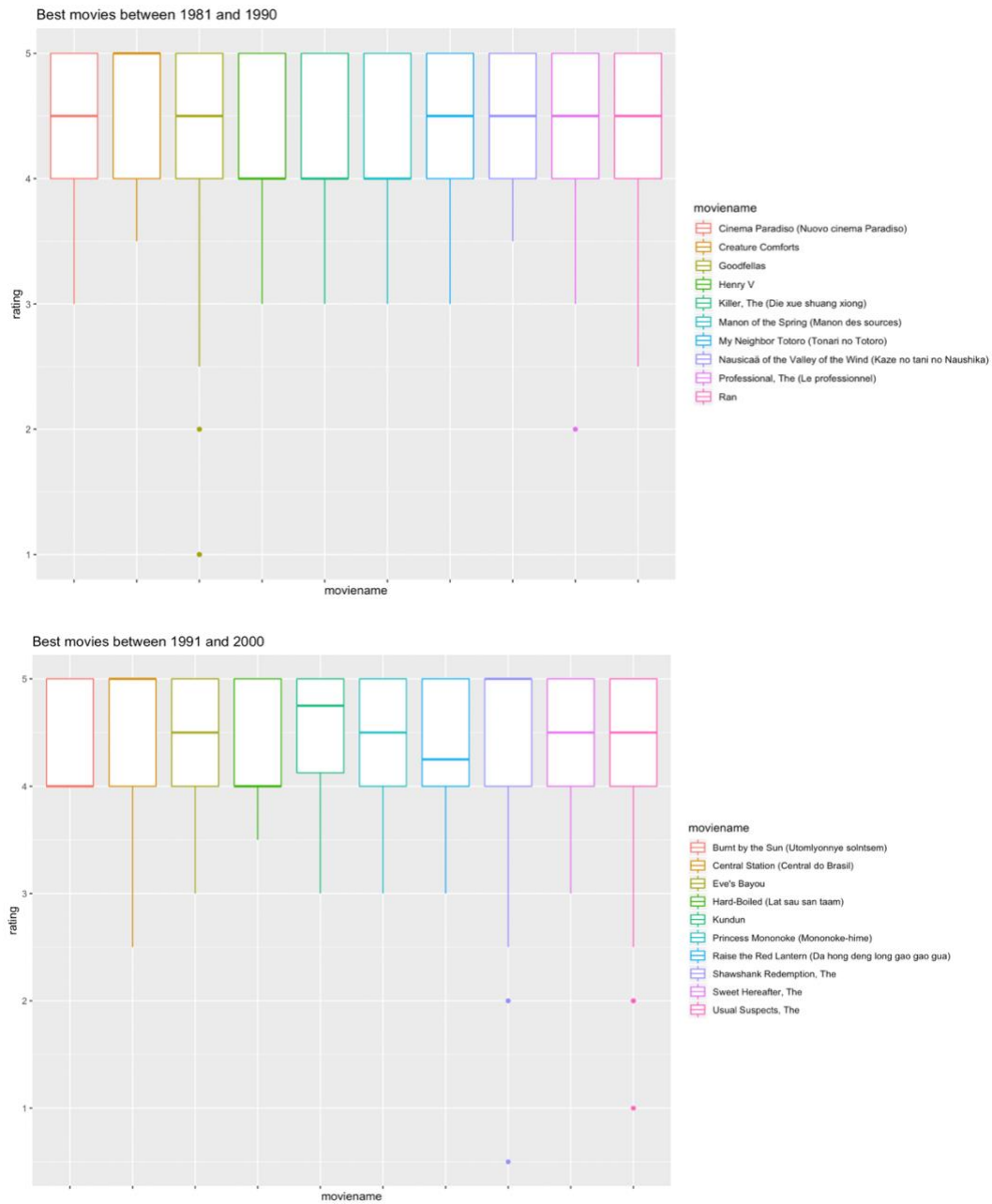
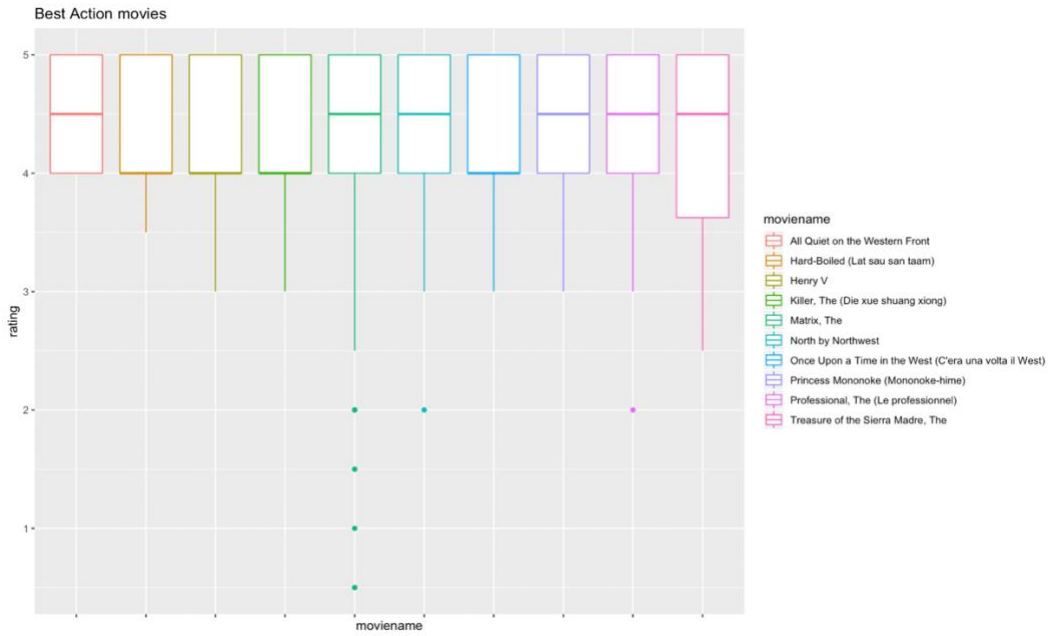


Figure2. Best Movies in 1980s and 1990s



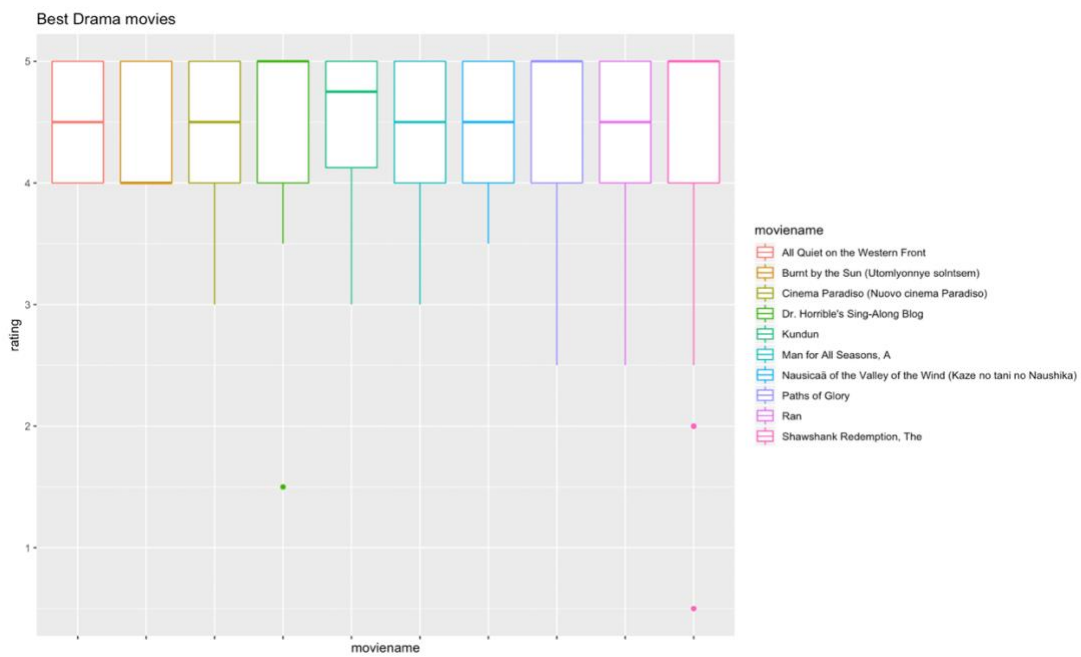


Figure3. best action movies & best drama movies

6. Data Mining Task

After data preprocessing and transformation in last step, we need to do the searching and soring for new users and do user-based collaborative filtering for old users.

For the new users, this model is very straight forward. We collect the input of new users which includes their preferred genre and time period. We search for all movies satisfied these requirement and sort them by average ratings. Then, we select top5 movies with highest average rating scores.

For the old users, we redefine a detailed data frame and compute RMS of each predictor values for each pair of users. Who has the least RMS are our neighbors. Then we can assume they have similar tastes of movies as our clients, so we can recommend the movies they have watched but our clients have not yet.

7. Data mining model

Recommendation system for new users

We will ask new users to input his/her preferred genres and years (we offer options).

After user input the genres and year, he/she will obtain an output, 5 recommended movies. (Priority: genres > years).

For example, the user input years for 2010, genres for Action and we will recommend him the top 5 actions movies in 2010; if there is not good action movies in 2010, we will give him/her an action movie 2009 or 2011 rather than some other type of movies in 2010.

Checking the intersections of genres and years selected, return top 5 movies with highest ratings, if no enough movies in the intersection, relax years to +- 1 year; if still no, +- 2 years.

Recommendation system for old users

We rebuild a data frame to store the information of each user's rating history. We have 55 variables in total, including user ID and number of rated movies, average rating score, mode of released years for each genre.

For example, for action movies:

First, we count the number of action movies that the user watched.

Second, we compute the mean of the ratings that s/he gave action movies.

Third, we compute the mode of years of the action movies that s/he rated.

After obtaining the information above, we need to do the normalization because the range of our variable is different.

We repeat these steps for all genres and we will get a data frame which has 55 columns. We call it as user_info. See figure 4.

userId	Action-number	Action-rating	Action-year	Adventure-number	Adventure-rating	Adventure-year	Animation-number
1	46	3.808385	2002	31	3.828638	1996	2
2	9	3.543215	1996	10	3.603448	1996	2
3	13	3.494274	1994	9	3.584900	1994	2
4	14	3.884750	1999	17	3.903794	1999	4
5	17	3.743478	1999	22	3.595424	1994	21

userId	Action-number	Action-rating	Action-year	Adventure-number	Adventure-rating	Adventure-year	Animation-number
1	0.052272727	0.9078574	0.9935484	0.050406504	0.9055285	0.9905707	0.01639344
2	0.010227273	0.8446452	0.9905707	0.016260163	0.8522678	0.9905707	0.01639344
3	0.014772727	0.8329784	0.9895782	0.014634146	0.8478808	0.9895782	0.01639344
4	0.015909091	0.9260615	0.9920596	0.027642276	0.9233040	0.9920596	0.03278688
5	0.019318182	0.8923847	0.9920596	0.035772358	0.8503699	0.9895782	0.17213114

Figure4. new built user_info data frame and the one after normalization

We use values in this data frame to compute distance (opposite of similarity, we use RMSE to denote it) of each pair of users.

We plan to find top 3 similar users according the distance between one specific user, then we find all movies the user hasn't watched but his/her neighbors have. We will recommend him/her these movies.

8. Performance Evaluation

Because our model is unsupervised learning, which is not like supervised learning model which has validation data set to evaluate the model, it is hard to test the performance. The only one way that we can test our model is inviting some volunteers to use our recommendation system and collect their feedbacks.

9. Result of Project

We made two models for new and old users successfully at last.

We converge all models and algorithms into one function. And run the system automatically to give users recommendations. In figure 5,6 and 7, we show the working process of our system.

```
Welcome to our movies recommendation system!
Please enter your user ID. If you are a new user, please enter: new
Enter user ID:

Enter user ID: test
Sorry, we don't have record of this ID, please enter your user ID agian.
Welcome to our movies recommendation system!
Please enter your user ID. If you are a new user, please enter: new
Enter user ID:
```

Figure5. movie recommendation system login interface

```
Please enter your user ID. If you are a new user, please enter: new
Enter user ID: new
Hi, friend, so glad to have you here.
  Want some movies but don't know which one to start? Let us help you.
  Based on your preferences, we will offer you some amazing movies.
  Hope you enjoy it.
  Let's go!

 Let's begin by telling us your favorite genre.
Please select one of following genres:
Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Ho
rror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western
Your favorite genre is: Drama
Now, tell us your prefered released time of movies. Please select start and end year from 1902 to
2015.
Your favorite period is from: 1990
to: 2010
```

movieId <dbl>	moviename <fctr>	rating <dbl>
1730	Kundun	4.500000
318	Shawshank Redemption, The	4.454545
66934	Dr. Horrible's Sing-Along Blog	4.434783
213	Burnt by the Sun (Utomlyonnye solntsem)	4.400000
7371	Dogville	4.400000

Figure6. movie recommendation system working process for new users

```
> user_input()
Welcome to our movies recommendation system!
Please enter your user ID. If you are a new user, please enter: new
Enter user ID: 33
```

	movieId <dbl>	moviename <chr>	ratings <dbl>	count <dbl>
11	549	Thirty-Two Short Films About Glenn Gould	4.500000	3
22	1175	Delicatessen	4.500000	3
31	1208	Apocalypse Now	4.500000	3
56	2076	Blue Velvet	3.666667	3
6	318	Shawshank Redemption, The	3.166667	3

Figure6. movie recommendation system working process for old users

10. Impact of Project Outcomes

Recommendation System is a very popular type of machine learning application that is used in lots of areas like Movies, TV programs, music, online shopping and so on. To do this project help us a lot to understand this practical and useful tool. However, the algorithms in the real world are much more complex than ours. Hopefully, we can take a further study in this problem.

Based on our project, we have some practical suggestions for improvement in the future work. First, to define characteristics of a user or a movie, we need more predictors, like country, language, duration for a movie and age, gender, country, education level for a user. In addition, about time stamp, we didn't consider about it this time. In the future work, we may need to use it as well.

Project_team3

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)
library(scales)
```

read data

```
raw_movies <-
read.csv("/Users/cyrusgong/Desktop/NEU/Courses/IE7275/project/raw
data/movies_1.csv")
raw_ratings <-
read.csv("/Users/cyrusgong/Desktop/NEU/Courses/IE7275/project/raw
data/ratings.csv")
```

First, we want to make rankings of best movies in each decade and best movies in each genre. This could do a help for us to understand raw data and be a recourse for us to recommend specific movies to new users in our following model.

Ranking criteraia: 1. For annually best, we set a threshold 20%: only the movies which receive over 20% ratings from users can be ranked, for genre's best, we set it as 5%; 2. For candidates over the threshold, we take average rating scores as the measures to rank them.

Decades best:

```
# find how many years we have
temp1 <- c()
"%!in%" <- Negate("%in%")  #define a "not in" function

for (i in 1:length(raw_movies$year)){
  if (raw_movies$year[i] %!in% temp1){
    temp1 <- append(raw_movies$year[i], temp1)
  }
}

temp1 <- sort(temp1)
length(temp1)
```

```
## [1] 100

# bins: -1920, 10 years an interval, 2011-2015
decade_movie_id <- c()

# 1902-1920
for (i in 1902:1920){
  for (j in 1:length(raw_movies$year)){
    if(i == raw_movies$year[j]){
      decade_movie_id <- append(raw_movies$movieId[j], decade_movie_id)
    }
  }
}

# count number of ratings of each movie id
num_movies <- count(raw_ratings, movieId)

# select movies (in decade_movie_id) and their numbers of ratings between
1902 and 1920
num_ratings <- c()

for (each in decade_movie_id){
  if (length(which(num_movies$movieId == each) != 0)){
    num_ratings = append(c(each, num_movies$n[which(num_movies$movieId ==
each)]), num_ratings)
  }
  else{
    num_ratings = append(c(each, 0), num_ratings)
  }
}
```

Based on our criteria, only the movies with over 5% numbers of ratings in this decade can be the candidats for best movies.

```
threshold <- sum(num_ratings[seq(2, length(num_ratings),2)])*0.05
threshold

## [1] 1.05
```

Only the movies with more than 1.05 ratings can be candidates.

```
candidates <- c()
for (i in seq(2, length(num_ratings),2)){
  if (num_ratings[i] >= threshold){
    candidates = append(num_ratings[i-1],candidates)
  }
}
candidates

## [1]  6987  6988  7065 32898
```

Compute average rating score of each candidate

```
candidates_averating <- c()
for (each in candidates){
 candidates_averating = append(c(each,
mean(raw_ratings$rating[which(raw_ratings$movieId == each)])),
candidates_averating)
}
candidates_averating

## [1] 32898.000000     3.750000  7065.000000     2.166667  6988.000000
## [6]     3.250000  6987.000000     3.846154

ranks1 <- data.frame(matrix(candidates_averating,
nrow=length(candidates_averating)/2, byrow=2))
colnames(ranks1) <- c("movieId", "averating")

# sort by averating
ranks1902 = ranks1[with(ranks1, order(averating, decreasing = T)),]
ranks1902

##   movieId averating
## 4    6987  3.846154
## 1   32898  3.750000
## 3    6988  3.250000
## 2    7065  2.166667
```

best movies in a decade

compress above process to a function

```
decade_best <- function(year_s, year_e){
  # select all movieIds in this decade
  decade_movie_id <- c()
  for (i in year_s:year_e){
  for (j in 1:length(raw_movies$year)){
    if(i == raw_movies$year[j]){
      decade_movie_id <- append(raw_movies$movieId[j], decade_movie_id)
     }
    }
  }

  # count number of ratings of each movie id
  num_movies <- count(raw_ratings, movieId)

  # select movies (in decade_movie_id) and their numbers of ratings between
year_s and year_e
  num_ratings <- c()
```

```r
  for (each in decade_movie_id){
    if (length(which(num_movies$movieId == each) != 0)){
     num_ratings = append(c(each, num_movies$n[which(num_movies$movieId ==
each)]), num_ratings)
   }
   else{
     num_ratings = append(c(each, 0), num_ratings)
    }
  }

  # threshold for candidates
  threshold <- min(10, sum(num_ratings[seq(2, length(num_ratings), 2)])*0.05)

  candidates <- c()
  for (i in seq(2, length(num_ratings),2)){
    if (num_ratings[i] >= threshold){
      candidates = append(num_ratings[i-1],candidates)
    }
  }

  candidates_averating <- c()
  for (each in candidates){
  candidates_averating = append(c(each,
mean(raw_ratings$rating[which(raw_ratings$movieId == each)])),
candidates_averating)
  }
  ranks <- data.frame(matrix(candidates_averating,
nrow=length(candidates_averating)/2, byrow=2))
  colnames(ranks) <- c("movieId", "averating")

  # sort by averating
  ranks = ranks[with(ranks, order(averating, decreasing = T)),]
  return(ranks)
}

decade_best(1931,1940)
```

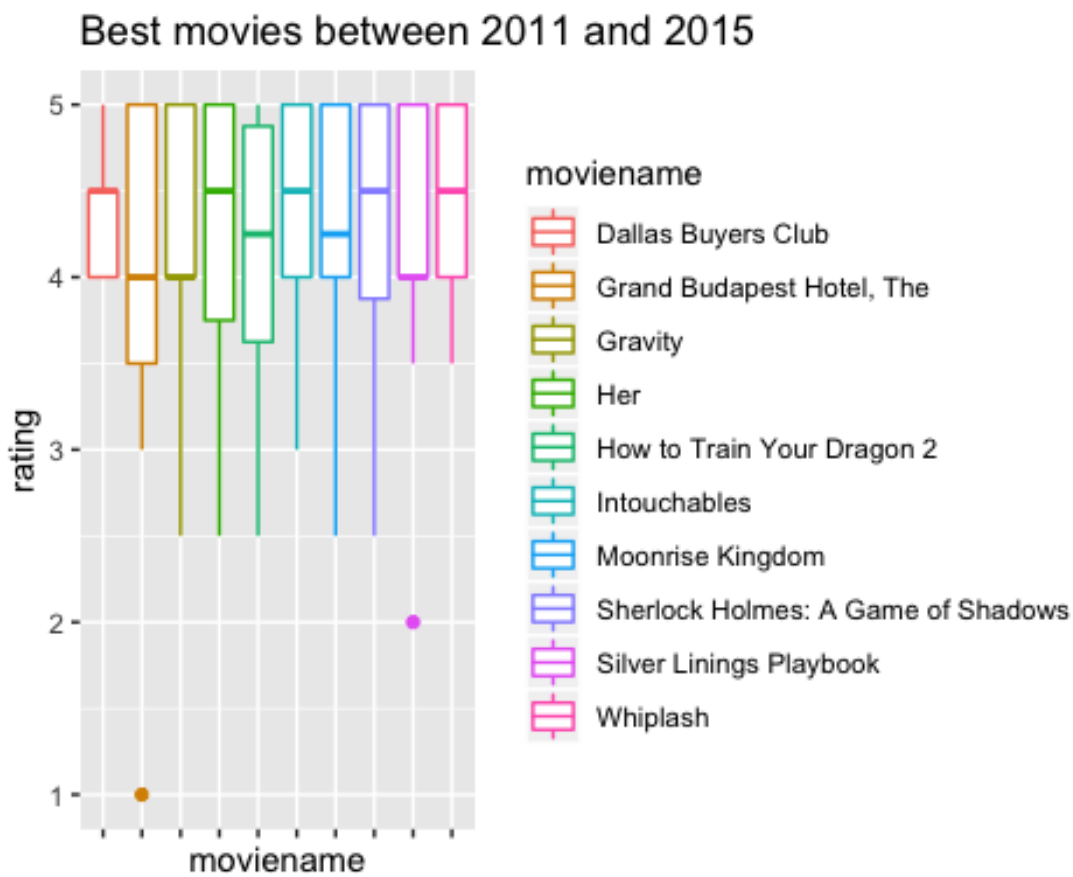
```
##    movieId averating
## 5     8228  4.346154
## 12     945  4.307692
## 1     1260  4.261905
## 26     928  4.173913
## 10     905  4.113636
## 13     965  4.108696
## 21     919  4.090090
## 30    3095  4.078947
## 20     955  4.029412
## 2     2644  4.000000
## 4     3307  4.000000
## 7     1256  4.000000
```

```
## 19      940  4.000000
## 11      950  3.973684
## 22      920  3.956522
## 17     3462  3.954545
## 28     1281  3.911765
## 16     7132  3.900000
## 23      954  3.892857
## 27      951  3.882353
## 25      898  3.851852
## 3      2648  3.825000
## 15     1931  3.821429
## 18      594  3.819444
## 29     1282  3.735294
## 14     1340  3.714286
## 24      596  3.580645
## 8      2366  3.532258
## 6      2633  3.400000
## 9      3932  3.318182
```

visualization: plot top 10 in each decade with boxplots

```
# define a function to select top 10 movieIds
top10_y <- function(year_s, year_r){
  df = decade_best(year_s, year_r)
  return(c(df[["movieId"]][1:min(10,length(df[,1]))]))
}


# define a function to a new dataframe to store top10 movieId, movie names
and their ratings
build_df_y <- function(year_s, year_e){
  df <- data.frame()
  for (each in top10_y(year_s, year_e)){
    temp <- data.frame("movieId" = each,
                       "moviename" =
raw_movies$title[which(raw_movies$movieId == each)],
                       "rating" =
raw_ratings$rating[which(raw_ratings$movieId == each)])
    df <- rbind(df, temp)
  }
return(df)
}

# define a function to build boxplots
top10_box_y <- function(year_s, year_e){
  box <- ggplot(data = build_df_y(year_s,year_e), aes(x=moviename, y=rating,
color = moviename)) +
  geom_boxplot() +
  theme(axis.text.x=element_blank())+
```

```
  ggtitle(paste("Best movies between",year_s,"and",year_e))

  return(box)
}

top10_box_y(2011,2015)
```



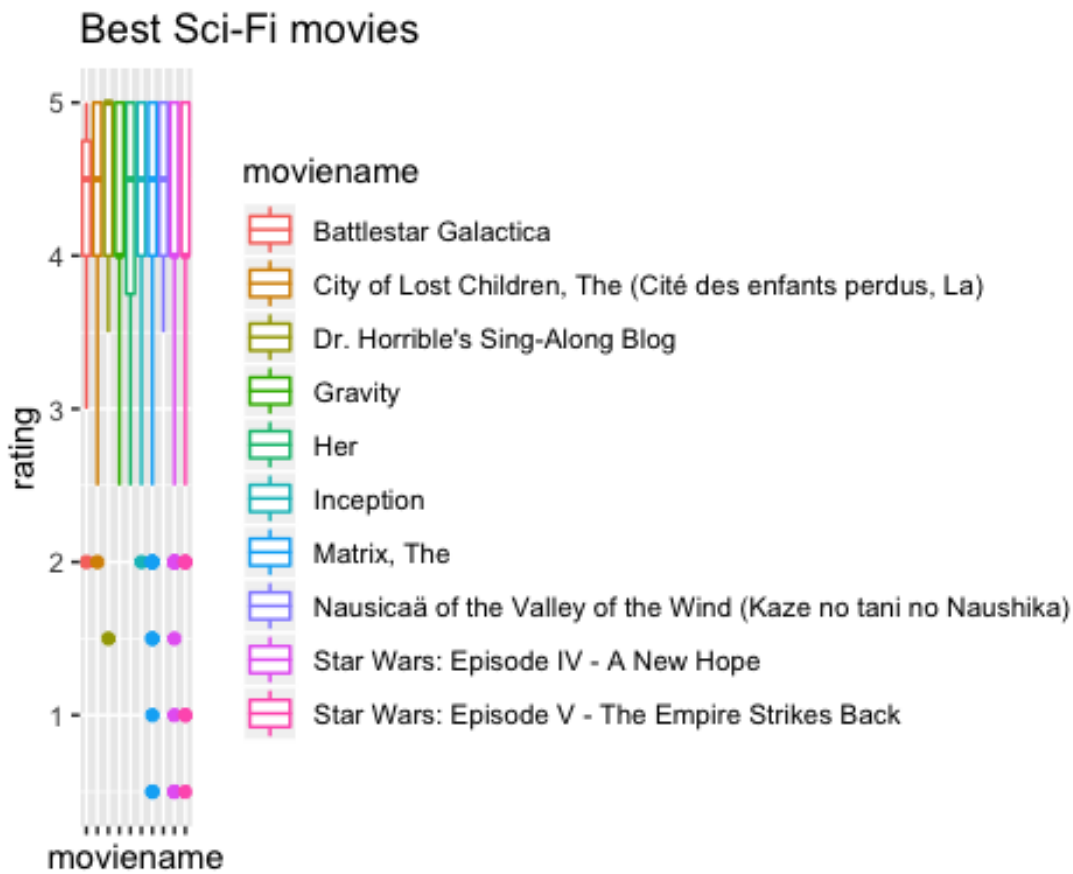
best movies in a genre

Create dummies for genres

```
list_genre <- c("Action", "Adventure", "Animation", "Children",
                "Comedy", "Crime","Documentary", "Drama", "Fantasy",
                "Film-Noir", "Horror", "Musical", "Mystery","Romance",
                "Sci-Fi", "Thriller", "War", "Western")

# add dummy' columns first
movies_dummy <- raw_movies
for (each in list_genre){
  movies_dummy[as.character(each)] <- rep(0, length(movies_dummy$movieId))
}
```

```r
# create dummies
for (i in 1:length(movies_dummy$movieId)){
  for (j in 1:length(list_genre)){
    if (list_genre[j] %in% sapply(movies_dummy[i,][4:12], as.character)){
      movies_dummy[i,][12+j] = 1
    }
  }
}

genre_best <- function(genre){
  # select all movieIds in this genre
  genre_movie_id <- c()
  for (i in 1:length(movies_dummy$movieId)){
    if (movies_dummy[i,][genre] == 1){
      genre_movie_id <- append(movies_dummy$movieId[i], genre_movie_id)
     }
  }

  # count number of ratings of each movie id
  num_movies <- count(raw_ratings, movieId)

  # select movies (in genre_movie_id) and their numbers of ratings in genre
  num_ratings <- c()

  for (each in genre_movie_id){
    if (length(which(num_movies$movieId == each) != 0)){
     num_ratings = append(c(each, num_movies$n[which(num_movies$movieId ==
each)]), num_ratings)
   }
   else{
     num_ratings = append(c(each, 0), num_ratings)
    }
  }

  # threshold for candidates
  threshold <- min(10, sum(num_ratings[seq(2, length(num_ratings), 2)])*0.05)

  candidates <- c()
  for (i in seq(2, length(num_ratings),2)){
    if (num_ratings[i] >= threshold){
      candidates = append(num_ratings[i-1],candidates)
    }
  }

  candidates_averating <- c()
  for (each in candidates){
  candidates_averating = append(c(each,
mean(raw_ratings$rating[which(raw_ratings$movieId == each)])),
candidates_averating)
```

```
  }
  ranks <- data.frame(matrix(candidates_averating,
nrow=length(candidates_averating)/2, byrow=2))
  colnames(ranks) <- c("movieId", "averating")

  # sort by averating
  ranks = ranks[with(ranks, order(averating, decreasing = T)),]
  return(ranks)
}

# define a function to select top 10 movieIds
top10_g <- function(genre){
  df = genre_best(genre)
  return(c(df[["movieId"]][1:min(10,length(df[,1]))]))
}


# define a function to a new dataframe to store top10 movieId, movie names
and their ratings
build_df_g <- function(genre){
  df <- data.frame()
  for (each in top10_g(genre)){
    temp <- data.frame("movieId" = each,
                       "moviename" =
raw_movies$title[which(raw_movies$movieId == each)],
                       "rating" =
raw_ratings$rating[which(raw_ratings$movieId == each)])
    df <- rbind(df, temp)
  }
return(df)
}

# define a function to build boxplots
top10_box_g <- function(genre){
  box <- ggplot(data = build_df_g(genre), aes(x=moviename, y=rating, color =
moviename)) +
  geom_boxplot() +
  theme(axis.text.x=element_blank())+
  ggtitle(paste("Best", genre, "movies"))

  return(box)
}

top10_box_g("Sci-Fi")
```



We can take above information as reference recommendations for new users who don't have any rating history.

Recommendation system for new users

input: genres, years (we offer options) output: 5 recommended movies

priority: genres > years

logic: check the intersections of genres and years selected, return top 5 movies with highest ratings; if no enough movies in the intersection, relax years to +- 1 year; if still no, +- 2 years;

```
decade_best(1930,1940)

##    movieId averating
## 1     1927  4.500000
## 7     8228  4.346154
## 14     945  4.307692
## 3     1260  4.261905
## 28     928  4.173913
## 12     905  4.113636
```

```
## 15      965  4.108696
## 23      919  4.090090
## 32     3095  4.078947
## 22      955  4.029412
## 4      2644  4.000000
## 6      3307  4.000000
## 9      1256  4.000000
## 21      940  4.000000
## 13      950  3.973684
## 24      920  3.956522
## 19     3462  3.954545
## 2      4970  3.950000
## 30     1281  3.911765
## 18     7132  3.900000
## 25      954  3.892857
## 29      951  3.882353
## 27      898  3.851852
## 5      2648  3.825000
## 17     1931  3.821429
## 20      594  3.819444
## 31     1282  3.735294
## 16     1340  3.714286
## 26      596  3.580645
## 10     2366  3.532258
## 8      2633  3.400000
## 11     3932  3.318182
```

```
temp.df <- data.frame("movieId" = integer(), "moviename" = character(),
"rating" = double())
temp.df<- add_row(temp.df, movieId = 4, moviename= "apd", rating = 5)
```

new_recom()

run this function for new users

```
# function to return movie names by movie IDs
movies_return <- function(movieid_list, genre){
  temp.df <- data.frame("movieId" = integer(), "moviename" = character(),
"rating" = double())
  for (each in movieid_list){
    temp.df<- add_row(temp.df, movieId = each, moviename =
raw_movies$title[which(raw_movies$movieId == each)] , rating =
genre_best(genre)$averating[which(genre_best(genre)$movieId == each)])

  }
  return(temp.df)
}


# function to find top 5 in the intersection of selected genre and period
```

```r
top5_its <- function(genre, year_s, year_e){
  temp_g <- genre_best(genre)
  temp_y <- decade_best(year_s, year_e)
  temp_its <- intersect(temp_g, temp_y)

  # check whether we have 5 movies in the intersection
  if (length(temp_its$movieId) >= 5){
    top5_id = c(temp_its[,1][1:5])
    top5 = movies_return(top5_id, genre)
    return(top5)
  }
  # if no 5 movies, we select relax +- one year
  else{
    top5_its(genre, year_s-1, year_e+1)
  }
}


# recommendation top 5 for new users
new_recom <- function(){

  # starter

  cat("Hi, friend, so glad to have you here.
  Want some movies but don't know which one to start? Let us help you.
  Based on your preferences, we will offer you some amazing movies.
  Hope you enjoy it.
  Let's go!\n")

  cat("\n Let's begin by telling us your favorite genre.\n")
  cat("Please select one of following genres:\n")
  cat(list_genre, sep = ", ")
  favgenre <- readline(prompt = "Your favorite genre is: ")

  cat("Now, tell us your prefered released time of movies. Please select
start and end year from 1902 to 2015.")
  favperiod_s <- readline(prompt = "Your favorite period is from: ")
  favperiod_e <- readline(prompt = "to: ")


  # pick the top 5 rated movies in the intersection
  top5_its(favgenre, favperiod_s, favperiod_e)
}
```

old_recom()

We use 3 varaibles for each genre to denote a user's watching history: # movies watched, mean/mode of the ratings, mode of years of the movies

Build the data frame to store above information:

```
temp_name <- c("userId")

for (i in list_genre){
  for (j in c("number", "rating", "year")){   # number: # movies watched;
rating: mean of the ratings; year: mode of years of the movies.
    each = paste(i,j,sep = "-")
    temp_name = append(temp_name, each)
  }
}

user_info <- data.frame(matrix(ncol = 55, nrow = 1))
colnames(user_info) <- temp_name

# fill in userids into user_info
for (i in unique(raw_ratings$userId)){
  user_info[i,][1] = i
}

head(user_info)

##    userId Action-number Action-rating Action-year Adventure-number
## 1       1            NA            NA          NA               NA
## NA      2            NA            NA          NA               NA
## 3       3            NA            NA          NA               NA
## 4       4            NA            NA          NA               NA
## 5       5            NA            NA          NA               NA
## 6       6            NA            NA          NA               NA
##    Adventure-rating Adventure-year Animation-number Animation-rating
## 1                NA             NA               NA               NA
## NA               NA             NA               NA               NA
## 3                NA             NA               NA               NA
## 4                NA             NA               NA               NA
## 5                NA             NA               NA               NA
## 6                NA             NA               NA               NA
##    Animation-year Children-number Children-rating Children-year
## 1              NA              NA              NA            NA
## NA             NA              NA              NA            NA
## 3              NA              NA              NA            NA
## 4              NA              NA              NA            NA
```

```
## 5              NA              NA              NA            NA
## 6              NA              NA              NA            NA
##    Comedy-number Comedy-rating Comedy-year Crime-number Crime-rating
## 1             NA            NA          NA           NA           NA
## NA            NA            NA          NA           NA           NA
## 3             NA            NA          NA           NA           NA
## 4             NA            NA          NA           NA           NA
## 5             NA            NA          NA           NA           NA
## 6             NA            NA          NA           NA           NA
##    Crime-year Documentary-number Documentary-rating Documentary-year
## 1          NA                 NA                 NA               NA
## NA         NA                 NA                 NA               NA
## 3          NA                 NA                 NA               NA
## 4          NA                 NA                 NA               NA
## 5          NA                 NA                 NA               NA
## 6          NA                 NA                 NA               NA
##    Drama-number Drama-rating Drama-year Fantasy-number Fantasy-rating
## 1            NA           NA         NA             NA             NA
## NA           NA           NA         NA             NA             NA
## 3            NA           NA         NA             NA             NA
## 4            NA           NA         NA             NA             NA
## 5            NA           NA         NA             NA             NA
## 6            NA           NA         NA             NA             NA
##    Fantasy-year Film-Noir-number Film-Noir-rating Film-Noir-year
## 1            NA               NA              NA            NA
## NA           NA               NA              NA            NA
## 3            NA               NA              NA            NA
## 4            NA               NA              NA            NA
## 5            NA               NA              NA            NA
## 6            NA               NA              NA            NA
##    Horror-number Horror-rating Horror-year Musical-number Musical-rating
## 1             NA            NA          NA             NA             NA
## NA            NA            NA          NA             NA             NA
## 3             NA            NA          NA             NA             NA
## 4             NA            NA          NA             NA             NA
## 5             NA            NA          NA             NA             NA
## 6             NA            NA          NA             NA             NA
##    Musical-year Mystery-number Mystery-rating Mystery-year Romance-number
## 1            NA             NA             NA           NA             NA
## NA           NA             NA             NA           NA             NA
## 3            NA             NA             NA           NA             NA
## 4            NA             NA             NA           NA             NA
## 5            NA             NA             NA           NA             NA
## 6            NA             NA             NA           NA             NA
##    Romance-rating Romance-year Sci-Fi-number Sci-Fi-rating Sci-Fi-year
## 1              NA           NA            NA            NA          NA
## NA             NA           NA            NA            NA          NA
## 3              NA           NA            NA            NA          NA
## 4              NA           NA            NA            NA          NA
## 5              NA           NA            NA            NA          NA
```

```
## 6               NA            NA             NA             NA           NA
##    Thriller-number Thriller-rating Thriller-year War-number War-rating
## 1               NA               NA            NA         NA         NA
## NA              NA               NA            NA         NA         NA
## 3               NA               NA            NA         NA         NA
## 4               NA               NA            NA         NA         NA
## 5               NA               NA            NA         NA         NA
## 6               NA               NA            NA         NA         NA
##    War-year Western-number Western-rating Western-year
## 1        NA             NA             NA           NA
## NA       NA             NA             NA           NA
## 3        NA             NA             NA           NA
## 4        NA             NA             NA           NA
## 5        NA             NA             NA           NA
## 6        NA             NA             NA           NA
```

Define a function to find the genre based on movieID:

```
find_genre <- function(movieid){
  temp = which(movies_dummy$movieId == movieid)
  return(movies_dummy[temp,][13:30])
}
find_genre(1)[,1]

## [1] 0
```

Fill in user_info

```
# define a function to get the mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

# fill in the data frame
for (i in unique(raw_ratings$userId)){
  # fill in # movies watched into user_info
  temp <- raw_ratings$movieId[which(raw_ratings$userId == i)]

  user_info[which(user_info$userId==i),2] <-
sum(movies_dummy[,13][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),5] <-
sum(movies_dummy[,14][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),8] <-
sum(movies_dummy[,15][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),11] <-
sum(movies_dummy[,16][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),14] <-
sum(movies_dummy[,17][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),17] <-
sum(movies_dummy[,18][which(movies_dummy$movieId %in% temp)])
```

```
  user_info[which(user_info$userId==i),20] <-
sum(movies_dummy[,19][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),23] <-
sum(movies_dummy[,20][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),26] <-
sum(movies_dummy[,21][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),29] <-
sum(movies_dummy[,22][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),32] <-
sum(movies_dummy[,23][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),35] <-
sum(movies_dummy[,24][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),38] <-
sum(movies_dummy[,25][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),41] <-
sum(movies_dummy[,26][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),44] <-
sum(movies_dummy[,27][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),47] <-
sum(movies_dummy[,28][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),50] <-
sum(movies_dummy[,29][which(movies_dummy$movieId %in% temp)])
  user_info[which(user_info$userId==i),53] <-
sum(movies_dummy[,30][which(movies_dummy$movieId %in% temp)])


  # fill in average ratings for each genre into user_info
  temp1 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Action ==
1)])
  temp2 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Adventure
== 1)])
  temp3 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Animation
== 1)])
  temp4 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Children
== 1)])
  temp5 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Comedy ==
1)])
  temp6 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Crime ==
1)])
  temp7 <- intersect(temp,
movies_dummy$movieId[which(movies_dummy$Documentary == 1)])
  temp8 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Drama ==
1)])
  temp9 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Fantasy ==
1)])
  temp10 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$`Film-
Noir` == 1)])
  temp11 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Horror ==
1)])
  temp12 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Musical
```

```
== 1)])
  temp13 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Mystery
== 1)])
  temp14 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Romance
== 1)])
  temp15 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$`Sci-Fi`
== 1)])
  temp16 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Thriller
== 1)])
  temp17 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$War ==
1)])
  temp18 <- intersect(temp, movies_dummy$movieId[which(movies_dummy$Western
== 1)])

  user_info[which(user_info$userId==i),3] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp1)])
  user_info[which(user_info$userId==i),6] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp2)])
  user_info[which(user_info$userId==i),9] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp3)])
  user_info[which(user_info$userId==i),12] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp4)])
  user_info[which(user_info$userId==i),15] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp5)])
  user_info[which(user_info$userId==i),18] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp6)])
  user_info[which(user_info$userId==i),21] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp7)])
  user_info[which(user_info$userId==i),24] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp8)])
  user_info[which(user_info$userId==i),27] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp9)])
  user_info[which(user_info$userId==i),30] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp10)])
  user_info[which(user_info$userId==i),33] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp11)])
  user_info[which(user_info$userId==i),36] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp12)])
  user_info[which(user_info$userId==i),39] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp13)])
  user_info[which(user_info$userId==i),42] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp14)])
  user_info[which(user_info$userId==i),45] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp15)])
  user_info[which(user_info$userId==i),48] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp16)])
  user_info[which(user_info$userId==i),51] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp17)])
  user_info[which(user_info$userId==i),54] <-
mean(raw_ratings[,3][which(raw_ratings$movieId %in% temp18)])
```

```r
  # fill in mode of years into user_info
  user_info[which(user_info$userId==i),4] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp1)])
  user_info[which(user_info$userId==i),7] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp2)])
  user_info[which(user_info$userId==i),10] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp3)])
  user_info[which(user_info$userId==i),13] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp4)])
  user_info[which(user_info$userId==i),16] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp5)])
  user_info[which(user_info$userId==i),19] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp6)])
  user_info[which(user_info$userId==i),22] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp7)])
  user_info[which(user_info$userId==i),25] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp8)])
  user_info[which(user_info$userId==i),28] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp9)])
  user_info[which(user_info$userId==i),31] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp10)])
  user_info[which(user_info$userId==i),34] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp11)])
  user_info[which(user_info$userId==i),37] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp12)])
  user_info[which(user_info$userId==i),40] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp13)])
  user_info[which(user_info$userId==i),43] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp14)])
  user_info[which(user_info$userId==i),46] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp15)])
  user_info[which(user_info$userId==i),49] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp16)])
  user_info[which(user_info$userId==i),52] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp17)])
  user_info[which(user_info$userId==i),55] <-
getmode(movies_dummy[,3][which(movies_dummy$movieId %in% temp18)])
}

head(user_info)
```

```
##    userId Action-number Action-rating Action-year Adventure-number
## 1       1            46      3.808385        2002               31
## NA      2             9      3.543215        1996               10
## 3       3            13      3.494274        1994                9
## 4       4            14      3.884750        1999               17
## 5       5            17      3.743478        1999               22
## 6       6            12      3.840553        2006               14
```

```
##    Adventure-rating Adventure-year Animation-number Animation-rating
## 1          3.828638           1996                2         3.823077
## NA         3.603448           1996                2         3.789753
## 3          3.584900           1994                2         3.591317
## 4          3.903794           1999                4         3.761513
## 5          3.595424           1994               21         3.715227
## 6          3.823502           2006                6         3.935446
##    Animation-year Children-number Children-rating Children-year
## 1            2001               3        3.805825          1984
## NA           1995               3        3.788783          1995
## 3            1992               5        3.412371          1995
## 4            1999               6        3.812057          1999
## 5            2009              21        3.712099          2009
## 6            1989               8        3.882312          1971
##    Comedy-number Comedy-rating Comedy-year Crime-number Crime-rating
## 1             31      3.876258        1994           31     4.052491
## NA            11      3.687876        1995            3     4.064327
## 3             35      3.627171        1994           12     3.850273
## 4             46      3.894076        1999           18     4.193294
## 5             45      3.453895        2006            6     3.769972
## 6             28      3.778661        1998           11     3.935430
##    Crime-year Documentary-number Documentary-rating Documentary-year
## 1        1995                  1           3.000000             2006
## NA       1995                  0                NaN               NA
## 3        1995                  1           4.083333             1994
## 4        1994                  0                NaN               NA
## 5        1996                  0                NaN               NA
## 6        2001                  0                NaN               NA
##    Drama-number Drama-rating Drama-year Fantasy-number Fantasy-rating
## 1            45     4.043012       1994              8       3.996334
## NA           11     3.911483       1995              4       3.676877
## 3            36     3.823212       1994              5       3.235000
## 4            76     4.135638       1994              8       3.740476
## 5            19     3.868051       1998             16       3.632249
## 6            31     3.963038       2001              8       3.846084
##    Fantasy-year Film-Noir-number Film-Noir-rating Film-Noir-year
## 1          2001                2         4.069124           1997
## NA         1996                0              NaN             NA
## 3          1993                0              NaN             NA
## 4          1999                6         4.211429           1941
## 5          2001                0              NaN             NA
## 6          2006                0              NaN             NA
##    Horror-number Horror-rating Horror-year Musical-number Musical-rating
## 1              8      3.999326        1991              1       3.843434
## NA             0           NaN          NA              2       3.641711
## 3              2      4.142173        1994              3       3.586592
## 4              6      4.035846        1991              6       4.035556
## 5              4      3.868726        1975             11       3.579096
## 6              1      3.687500        2012              2       3.767857
##    Musical-year Mystery-number Mystery-rating Mystery-year Romance-number
```

```
## 1          1980             13       3.993393         1995             16
## NA         1996              2       3.713158         1995              8
## 3          1993              4       3.890220         1996             22
## 4          1952             10       3.986936         1995             37
## 5          1994              3       3.958140         1998             21
## 6          1971              2       3.901869         1992             15
##    Romance-rating Romance-year Sci-Fi-number Sci-Fi-rating Sci-Fi-year
## 1        3.785651         1994            25      3.773768        2002
## NA       3.509539         1995             5      3.767981        1996
## 3        3.618342         1994             3      3.751897        1995
## 4        3.988595         1940             3      4.081276        1995
## 5        3.442155         2001             6      3.835886        1993
## 6        3.907574         1999            12      3.850089        1993
##    Thriller-number Thriller-rating Thriller-year War-number War-rating
## 1               43        3.856816          1996         10   4.115609
## NA              12        3.558549          1996          0        NaN
## 3               21        3.735116          1995          3   4.200176
## 4               18        4.085504          1963         16   4.162949
## 5               11        3.790930          1999          2   4.020906
## 6               13        3.852529          2001          0        NaN
##    War-year Western-number Western-rating Western-year
## 1      1995              1       3.766169         1990
## NA       NA              0            NaN           NA
## 3      1994              3       3.752336         1994
## 4      1942              5       4.077236         1968
## 5      1995              0            NaN           NA
## 6        NA              0            NaN           NA
```

normalization

```
user_info_norm <- user_info

# replace NA with 0
user_info_norm[is.na(user_info_norm)] <- 0

# convert years to numeric
for (i in 2:55){
  temp <- rescale(as.numeric(user_info_norm[,i]))
  user_info_norm[,i] <- temp
}

head(user_info_norm)
```

```
##    userId Action-number Action-rating Action-year Adventure-number
## 1       1    0.05227273     0.9078574   0.9935484       0.05040650
## NA      2    0.01022727     0.8446452   0.9905707       0.01626016
## 3       3    0.01477273     0.8329784   0.9895782       0.01463415
## 4       4    0.01590909     0.9260615   0.9920596       0.02764228
## 5       5    0.01931818     0.8923847   0.9920596       0.03577236
```

```
## 6       6    0.01363636     0.9155258   0.9955335       0.02276423
##    Adventure-rating Adventure-year Animation-number Animation-rating
## 1         0.9055285      0.9905707       0.01639344        0.8688811
## NA        0.8522678      0.9905707       0.01639344        0.8613074
## 3         0.8478808      0.9895782       0.01639344        0.8162085
## 4         0.9233040      0.9920596       0.03278689        0.8548894
## 5         0.8503699      0.9895782       0.17213115        0.8443698
## 6         0.9043137      0.9955335       0.04918033        0.8944195
##    Animation-year Children-number Children-rating Children-year
## 1       0.9930521      0.01639344       0.9121716     0.9846154
## NA      0.9900744      0.01639344       0.9080869     0.9900744
## 3       0.9885856      0.02732240       0.8178694     0.9900744
## 4       0.9920596      0.03278689       0.9136652     0.9920596
## 5       0.9970223      0.11475410       0.8897075     0.9970223
## 6       0.9870968      0.04371585       0.9305038     0.9781638
##    Comedy-number Comedy-rating Comedy-year Crime-number Crime-rating
## 1    0.020287958     0.9323017   0.9895782  0.032460733    0.9235230
## NA   0.007198953     0.8869929   0.9900744  0.003141361    0.9262205
## 3    0.022905759     0.8723924   0.9895782  0.012565445    0.8774395
## 4    0.030104712     0.9365874   0.9920596  0.018848168    0.9556107
## 5    0.029450262     0.8307167   0.9955335  0.006282723    0.8591398
## 6    0.018324607     0.9088283   0.9915633  0.011518325    0.8968461
##    Crime-year Documentary-number Documentary-rating Documentary-year
## 1   0.9900744            0.00625          0.6000000        0.9955335
## NA  0.9900744            0.00000          0.0000000        0.0000000
## 3   0.9900744            0.00625          0.8166667        0.9895782
## 4   0.9895782            0.00000          0.0000000        0.0000000
## 5   0.9905707            0.00000          0.0000000        0.0000000
## 6   0.9930521            0.00000          0.0000000        0.0000000
##    Drama-number Drama-rating Drama-year Fantasy-number Fantasy-rating
## 1   0.012627781    0.7771335  0.7402597     0.02523659      0.9048303
## NA  0.002405292    0.6489160  0.7532468     0.01261830      0.8325006
## 3   0.009921828    0.5628665  0.7402597     0.01577287      0.7324528
## 4   0.021948286    0.8674276  0.7402597     0.02523659      0.8469003
## 5   0.004810583    0.6065772  0.7922078     0.05047319      0.8223960
## 6   0.008418521    0.6991729  0.8311688     0.02523659      0.8708114
##    Fantasy-year Film-Noir-number Film-Noir-rating Film-Noir-year
## 1     0.9935452       0.01086957        0.9411580      0.9930383
## NA    0.9910626       0.00000000        0.0000000      0.0000000
## 3     0.9895730       0.00000000        0.0000000      0.0000000
## 4     0.9925521       0.03260870        0.9740719      0.9651914
## 5     0.9935452       0.00000000        0.0000000      0.0000000
## 6     0.9960278       0.00000000        0.0000000      0.0000000
##    Horror-number Horror-rating Horror-year Musical-number Musical-rating
## 1    0.015151515     0.9533946   0.9890710    0.004329004      0.8666568
## NA   0.000000000     0.0000000   0.0000000    0.008658009      0.8211702
## 3    0.003787879     0.9874476   0.9905614    0.012987013      0.8087414
## 4    0.011363636     0.9621005   0.9890710    0.025974026      0.9099782
## 5    0.007575758     0.9222610   0.9811227    0.047619048      0.8070511
## 6    0.001893939     0.8790588   0.9995032    0.008658009      0.8496148
```

```
##    Musical-year Mystery-number Mystery-rating Mystery-year Romance-number
## 1     0.9831182    0.028017241      0.9083938    0.9920438    0.015579357
## NA    0.9910626    0.004310345      0.8446474    0.9920438    0.007789679
## 3     0.9895730    0.008620690      0.8849244    0.9925410    0.021421616
## 4     0.9692155    0.021551724      0.9069249    0.9920438    0.036027264
## 5     0.9900695    0.006465517      0.9003744    0.9935356    0.020447907
## 6     0.9786495    0.004310345      0.8875744    0.9905520    0.014605648
##    Romance-rating Romance-year Sci-Fi-number Sci-Fi-rating Sci-Fi-year
## 1       0.9073986    0.9900695   0.055187638     0.8893826   0.9935484
## NA      0.8412163    0.9905660   0.011037528     0.8880189   0.9905707
## 3       0.8672957    0.9900695   0.006622517     0.8842282   0.9900744
## 4       0.9560432    0.9632572   0.006622517     0.9618546   0.9900744
## 5       0.8250648    0.9935452   0.013245033     0.9040224   0.9890819
## 6       0.9366229    0.9925521   0.026490066     0.9073696   0.9890819
##    Thriller-number Thriller-rating Thriller-year  War-number War-rating
## 1      0.028839705       0.9028343     0.9905707 0.029673591  0.9579268
## NA     0.008048290       0.8330136     0.9905707 0.000000000  0.0000000
## 3      0.014084507       0.8743459     0.9900744 0.008902077  0.9776103
## 4      0.012072435       0.9563674     0.9741935 0.047477745  0.9689455
## 5      0.007377599       0.8874112     0.9920596 0.005934718  0.9358843
## 6      0.008718981       0.9018308     0.9930521 0.000000000  0.0000000
##     War-year Western-number Western-rating Western-year
## 1  0.9905660    0.005813953      0.8861574    0.9880834
## NA 0.0000000    0.000000000      0.0000000    0.0000000
## 3  0.9900695    0.017441860      0.8829027    0.9900695
## 4  0.9642502    0.029069767      0.9593496    0.9771599
## 5  0.9905660    0.000000000      0.0000000    0.0000000
## 6  0.0000000    0.000000000      0.0000000    0.0000000
```

We take this data frame to compute distance(opposite of similarity, we use RMSE to denote it) of each pair of users.

```
# function to find top3 similar users
find_3neighbors <- function(id){
  distance <- c()
  j <- which(user_info_norm$userId == id)
  for (i in 1:length(user_info_norm$userId)){
    temp <- sqrt(mean(as.numeric((user_info_norm[i,2:55]-
user_info_norm[j,2:55])^2)))
    distance <- append(distance, temp)
  }
  neighbors <- c()
  count <- 4
  while(count){
    temp1 <- user_info_norm$userId[which.min(distance)]
    neighbors <- append(neighbors, temp1)
    distance[which.min(distance)] = 1
    count = count - 1
  }
```

```
  neighbors = neighbors[2:4]
  return(neighbors)
}


# function to find movieids accoring to userid(watching history)
find_history <- function(id){
  movies <- c()
  temp <- which(raw_ratings$userId == id)
  for (i in temp){
    movies <- append(movies, raw_ratings$movieId[i])
  }
  return(movies)
}


# functions to find all movies the user hasn't watched but his/her neighbors
have.
find_new <- function(id){
  movies <- c()
  for (i in find_3neighbors(id)){
    movies <- append(movies, find_history(i))
  }
 # remove dulicates
  movies <- unique(movies)
 # remove his own history
  temp2 <- movies[! movies %in% find_history(id)]

  return(temp2)
}
```

After find_new(), we get a vector of recommended movie ids, we pick five movies with highest ratings by these neighbors (if more than one neighbor rated a movie, take the sum of ratings divided by 3). We assume if one user hasn't watched it means s/he didn't like it and gave a 0.

Attention: actually, I don't think this criteria is proper enough. There is another way: compute mean based on number of neighbors who watched it already. We choose the first standard.

```
find_top5mv <- function(id){
  temp_rating <- data.frame(matrix(ncol = 4, nrow = 1))
  colnames(temp_rating) <- c("movieId", "moviename", "ratings","count")

  # fill in new movie id first
```

```
  for (i in find_new(id)){
    temp_rating[nrow(temp_rating) + 1,] = c(i,0,0,0)
  }

  # compute rating averages by neighbors and count
  for (i in find_3neighbors(id)){
    for (j in find_history(i)){
      temp <- raw_ratings$rating[which((raw_ratings$userId == i) &
(raw_ratings$movieId == j))]
      temp_rating$ratings[which(temp_rating$movieId == j)] =
temp_rating$ratings[which(temp_rating$movieId == j)] + temp
      if (temp > 0){
        temp_rating$count[which(temp_rating$movieId == j)] =
temp_rating$count[which(temp_rating$movieId == j)] + 1
      }
    }
  }

  for (i in 1:length(temp_rating$ratings)){
    temp_rating$ratings[i] = temp_rating$ratings[i]/temp_rating$count[i]
  }

  # sort dataframe
  temp_rating <-temp_rating[order(temp_rating$count, temp_rating$ratings,
decreasing = TRUE),]
  temp_r <- temp_rating[1:5,]

  # fill in movie names
  for (i in 1:5){
    temp_r$moviename[i] <-
as.character(raw_movies$title[which(raw_movies$movieId ==
temp_r$movieId[i])])
  }

  return(temp_r)
}

find_top5mv(1)
```

```
##    movieId                  moviename  ratings count
## 42    1175              Delicatessen  3.833333     3
## 58    1610 Hunt for Red October, The  3.166667     3
## 7       36          Dead Man Walking  5.000000     2
## 28     493         Menace II Society  4.500000     2
## 35     778             Trainspotting  4.500000     2
```

old_recom()

```
old_recom <- function(id){
  find_top5mv(id)
}
```

start: get inputs from users

```
# define a functio to get inputs from users

# create a list to store all user ID
useridlist <- unique(raw_ratings$userId)

user_input <- function(){
  cat("Welcome to our movies recommendation system! \n")
  cat("Please enter your user ID. If you are a new user, please enter: new")
  userid <- readline(prompt="Enter user ID: ")

  if (userid =="new"){
    new_recom()
  }
  else if (userid %in% useridlist){
    old_recom(userid)
  }
  else {
    cat("Sorry, we don't have record of this ID, please enter your user ID
agian. \n")
    user_input()
  }
}
```

test

```
#user_input()
```