DATA-643 Assignment - 02

Mohamed Elmoudni
Shazia Khan
Senthil Dhanapal

Contents

1 Requirements:

For project 2, you're asked to take some recommendation data (such as your toy movie dataset, MovieLens, or another dataset of your choosing), and implement at least two different recommendation algorithms on the data. For example, content-based, user-user CF, and/or itemitem CF are the three methods that we've covered in the course to date. You should evaluate different approaches, using different algorithms, normalization techniques, similarity methods, neighborhood sizes, etc. You don't need to be exhaustive-these are just some suggested possibilities. You may use whatever third party libraries you want. You should definitely tackle (at least) accuracy metrics. Please provide at least one graph, and a textual summary of your evaluation.

2 Data Load

In this project we will be using the MovieLense dataset that comes with the recommenderlab package.

3 Data Exploration

In this phase we will perform data analysis and summary of the main characteristics of a dataset. We will be describing the data by means of statistical and visualization techniques

Table 1: Data Summary

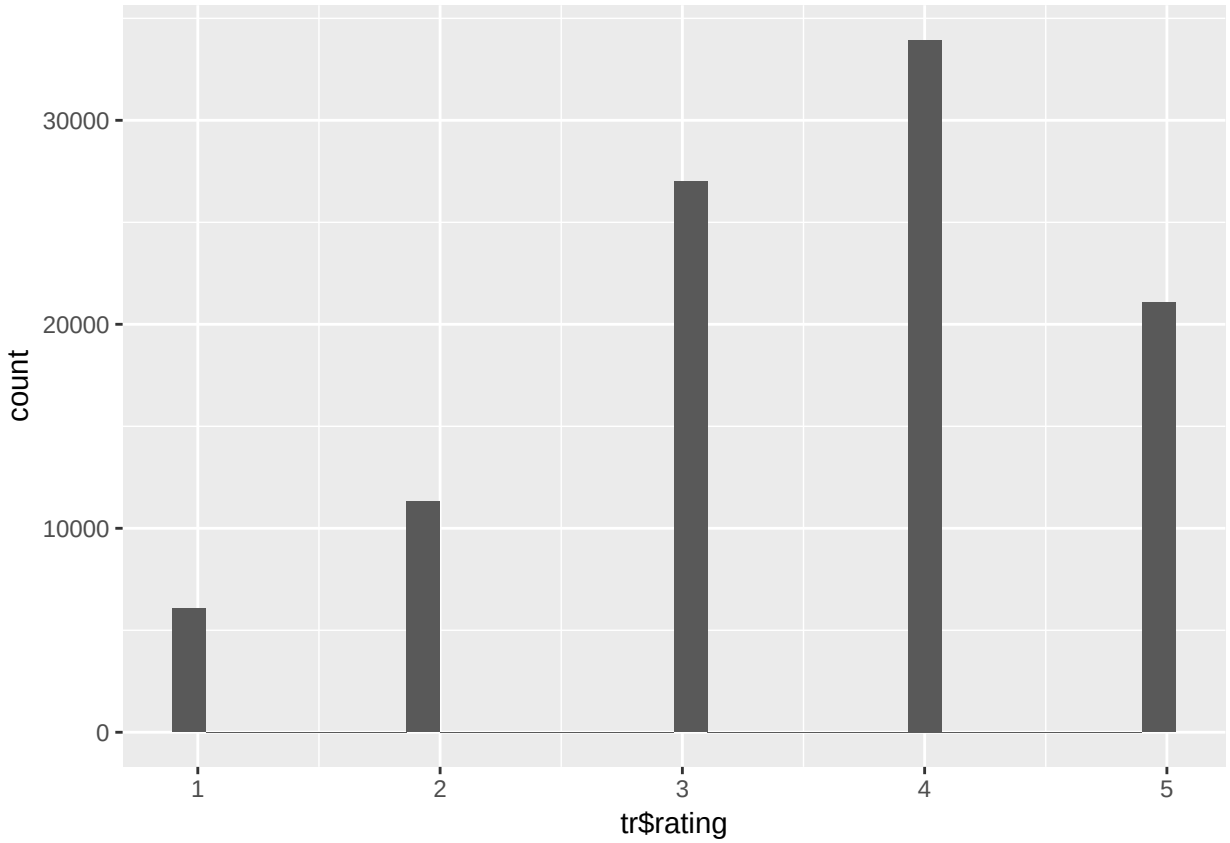
	user	item	rating
1	1	Toy Story (1995)	5
453	1	GoldenEye (1995)	3
584	1	Four Rooms (1995)	4
674	1	Get Shorty (1995)	3
883	1	Copycat (1995)	3
969	1	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	5

Table 2: Ratings Frequency Table

user	item	rating
405 : 735	Star Wars (1977) : 583	Min. :1.00
655 : 677	Contact (1997) : 509	1st Qu.:3.00
13 : 630	Fargo (1996) : 508	Median :4.00
450 : 538	Return of the Jedi (1983) : 507	Mean :3.53
276 : 515	Liar Liar (1997) : 485	3rd Qu.:4.00
416 : 487	English Patient, The (1996): 481	Max. :5.00
(Other):95810	(Other) :96319	NA

Var1	Freq
1	6059
2	11307

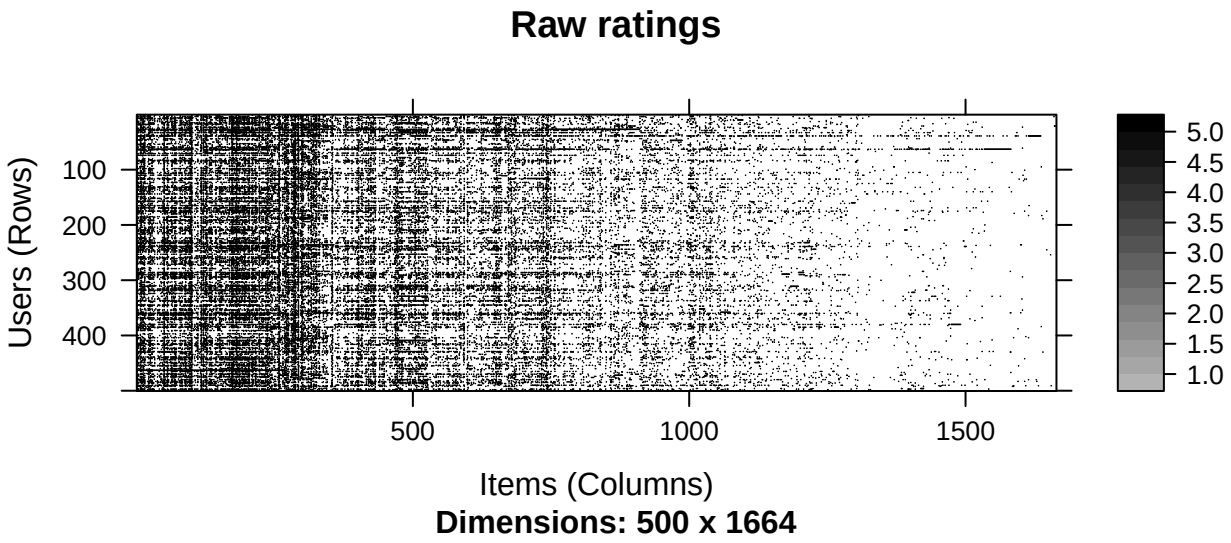
Var1	Freq
3	27002
4	33947
5	21077



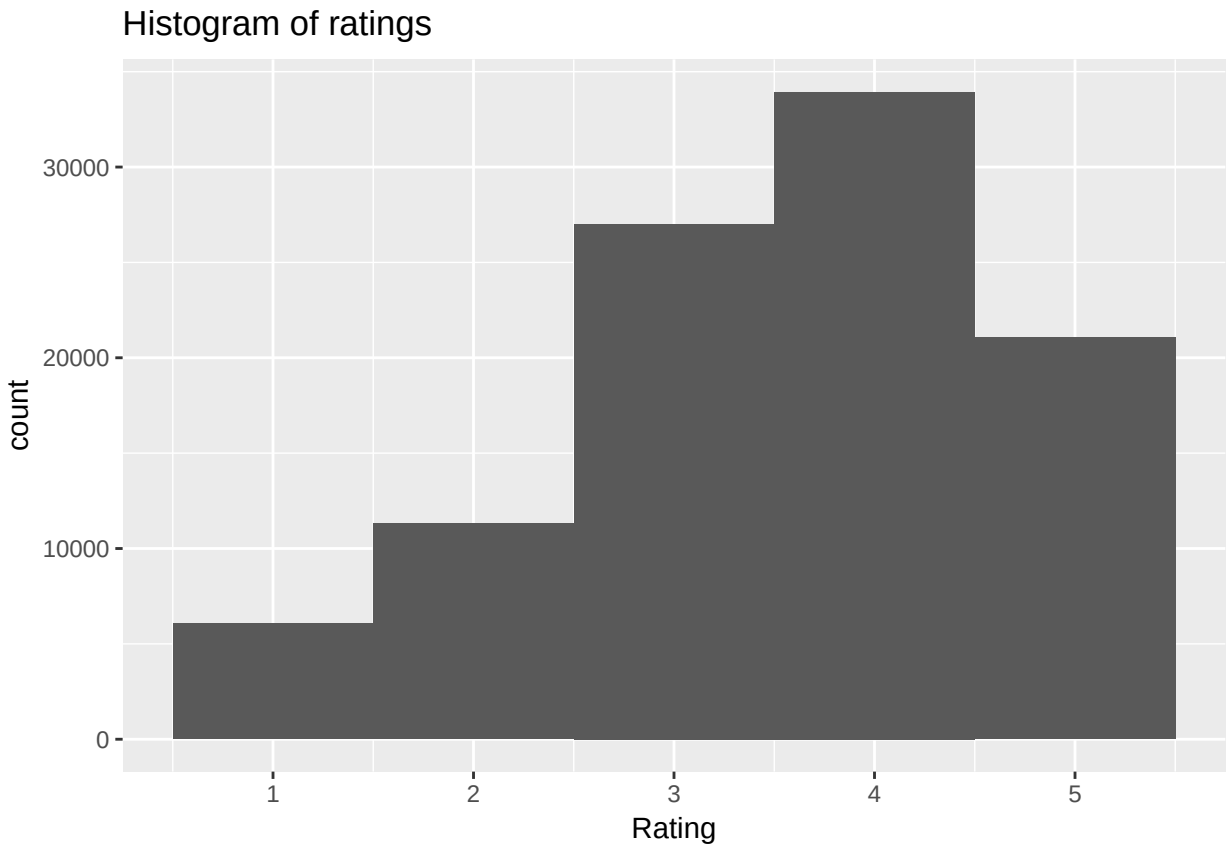
3.1 Utility Matrix

```
## [1] 5 4 0 3 1 2
```

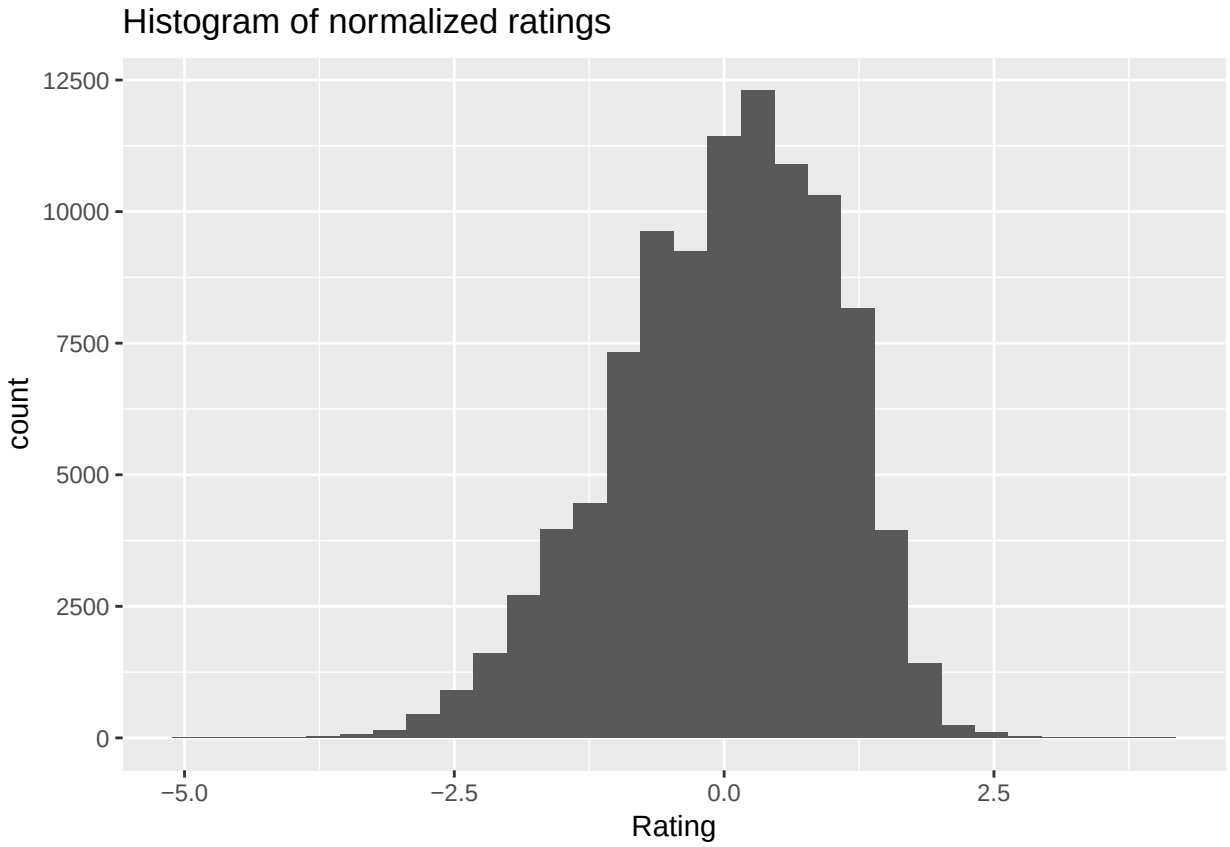
vector_ratings	Freq
1	6059
2	11307
3	27002
4	33947
5	21077


Raw ratings
Users (Rows)
100
200
300
400
500
1000
1500
Items (Columns)
Dimensions: 500 x 1664
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

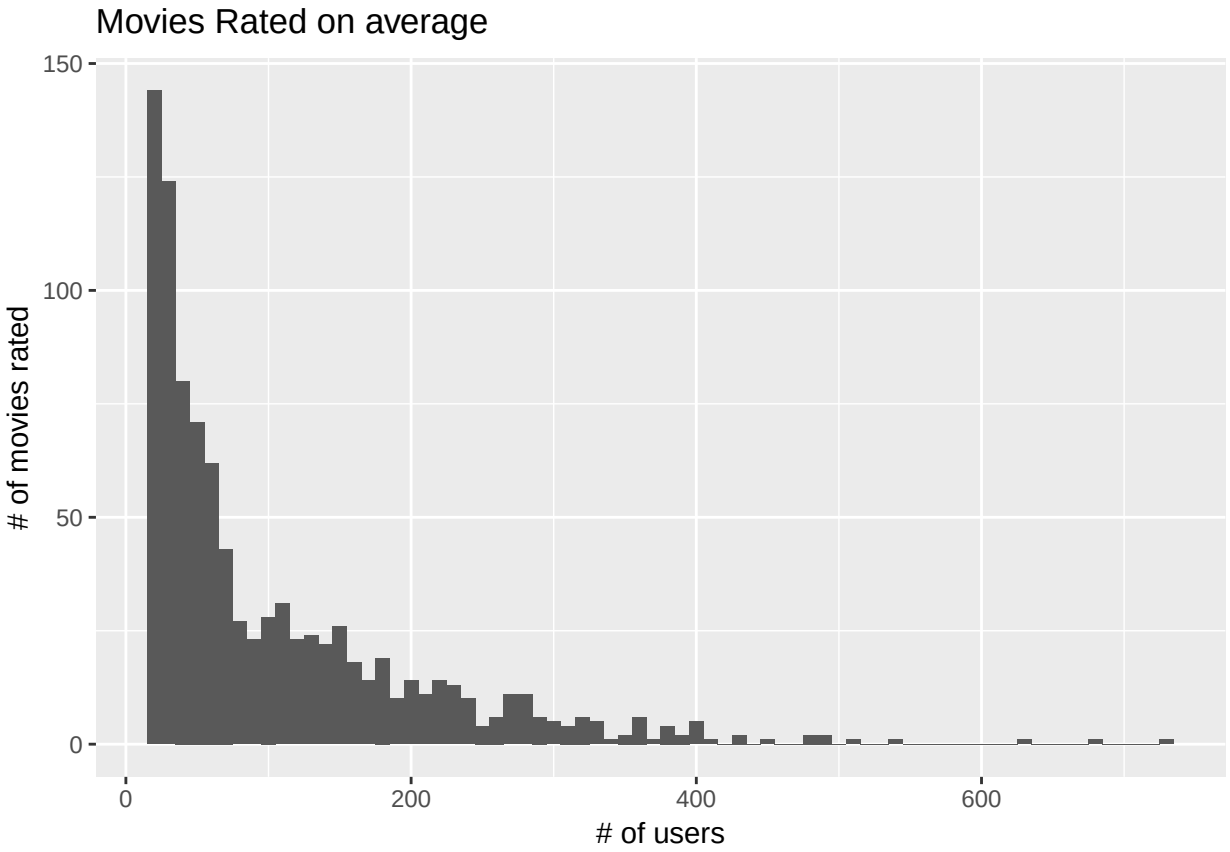
3.2 Data visualization

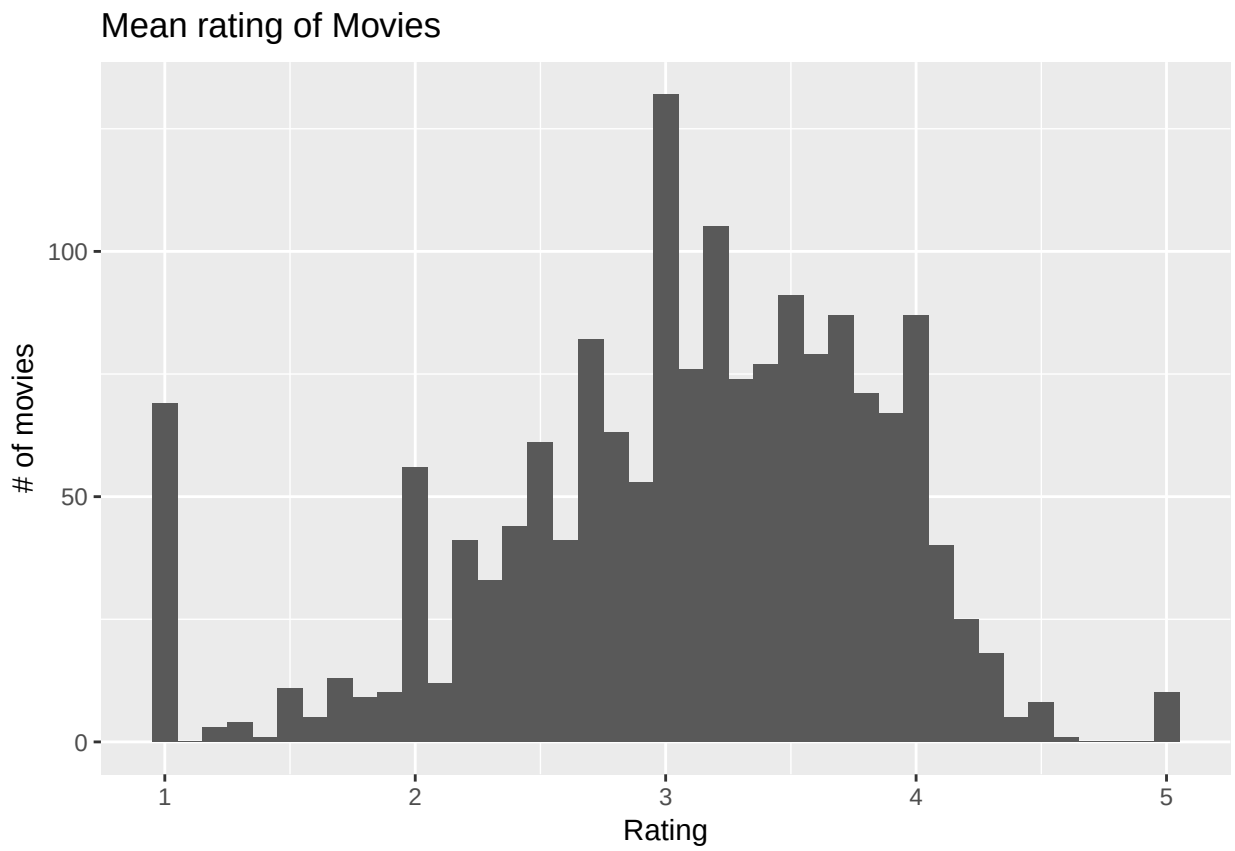



```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.00    4.00    3.53    4.00    5.00
```




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.8520 -0.6466  0.1084  0.0000  0.7506  4.1280
```


Movies Rated on average
150
100
50
0
# of movies rated
0
200
400
600
# of users



4 Model Evaluation

In this section, we will evaluate IBCF and UBCF using Cosine, Jaccard, and Pearson methods.

Split dataset into Training and Testing

4.1 UBCF

UBCF: In user-based collaborative filtering predictions are based on aggregated ratings from the closest users (nearest neighbors). Nearest neighbors are defined based on similarity between users, which is calculated using available ratings. This method works under the assumption that the users with similar ratings will rate items similarly. In this case we will try UBCF using Cosine,Jaccard, and Pearson.

4.1.1 UBCF Cosine

```
##         RMSE       MSE       MAE
## 2  1.1779000 1.3874484 0.8534469
## 14 1.2654505 1.6013651 0.9155640
## 15 1.3512658 1.8259194 1.1936825
## 26 0.8827476 0.7792434 0.6704831
## 28 1.0426989 1.0872210 0.7919803
## 29 0.8045366 0.6472792 0.6542363
```

4.1.2 UBCF Jaccard

```
##         RMSE       MSE       MAE
## 2  1.1634233 1.3535537 0.8464815
## 14 1.2584868 1.5837891 0.9174970
## 15 1.3382890 1.7910174 1.1751823
## 26 0.8890289 0.7903723 0.6817192
## 28 1.0617028 1.1272128 0.8086684
## 29 0.7962437 0.6340040 0.6410867
```

4.1.3 UBCF Pearson

```
##         RMSE       MSE       MAE
## 2  1.1626436 1.3517403 0.8433780
## 14 1.2668504 1.6049100 0.9175953
## 15 1.3297240 1.7681660 1.1789731
## 26 0.8729242 0.7619966 0.6680795
## 28 1.0403326 1.0822919 0.7814768
## 29 0.7680313 0.5898721 0.6343311
```

4.2 IBCF

Item-based collaborative filtering where we consider set of movies rated by the user and computes item similarities with the targeted movie. Once similar movies are found, and then rating for the new movie is predicted by taking weighted average of the user's rating on these similar movies. in this case we will try IBCF using Cosine,Jaccard, and Pearson.

4.2.1 IBCF Cosine

```
##         RMSE      MSE       MAE
## 2  1.1547702 1.333494 0.6116706
## 14 1.3150123 1.729257 0.8175977
## 15 1.4698157 2.160358 1.1415666
## 26 0.7852414 0.616604 0.6143375
## 28 1.1300026 1.276906 0.9035216
## 29 1.1130071 1.238785 0.9494813
```

4.2.2 IBCF Jaccard

```
##         RMSE       MSE       MAE
## 2  1.1178853 1.2496676 0.7887729
## 14 1.3007794 1.6920271 0.8684994
## 15 1.3783470 1.8998404 1.1350760
## 26 0.7477717 0.5591626 0.6035208
## 28 1.1064164 1.2241571 0.8305630
## 29 0.7215553 0.5206420 0.5238772
```

4.2.3 IBCF Pearson

```
##        RMSE   MSE  MAE
```

```
## 2  2.573908 6.625 2.25
## 14      NaN   NaN  NaN
## 15      NaN   NaN  NaN
## 26      NaN   NaN  NaN
## 28      NaN   NaN  NaN
## 29      NaN   NaN  NaN
```

5 Calculate Average of all Errors

Below we will be calculating averages for both IBCF and UBCF for all three methods (Cosine, Jaccard, and Pearson)

Table 5: Evaluation Summary

	UBCF-Cosine	UBCF-Jaccard	UBCF-Pearson	IBCF-Cosine	IBCF-Jaccard	IBCF-Pearson
RMSE	1.0731782	1.0801493	1.0768082	1.1978534	1.2043480	1.525366
MSE	1.2047777	1.2197427	1.2136408	1.5727406	1.5522071	3.119162
MAE	0.8677387	0.8750378	0.8711948	0.9026599	0.9283885	1.365865
AverageOfAllErrors	1.0485649	1.0583099	1.0538813	1.2244180	1.2283145	2.003464

6 Conclusion

From the Evaluation Summary table above, UBCF Cosine shows the least average error for RMSE, MSE, and MAE compared with other methods. Therefore, UBCF-Cosine is the choosen method for recommendation. Next we will run the recommendation based on the best chosen method - UBCF Cosine- and show the result of one test user before and after prediction:

Table 6: Test Using UBCF Cosine

	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)
Before	NA	NA	NA
After	3.704918	3.68259	3.704918

Appendix A: DATA643 Assignment 02 R Code

```
if (!require("ggplot2",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.us.r-pro
if (!require("recommenderlab",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.
if (!require("reshape2",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.us.r-p
if (!require("knitr",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.us.r-proj

library(recommenderlab)
library(reshape2)
library(ggplot2)
library(knitr)


set.seed(1)
data("MovieLense")


# Just look at first few lines of this file


tr <- (as(MovieLense,'data.frame'))
kable(head(tr))

#############
kable(summary(tr), caption ="Data Summary")


kable(as.data.frame(table(tr$rating)), caption = "Ratings Frequency Table")

#qplot(tr$rating, tr$user)

qplot(tr$rating)

r <- MovieLense
#slotNames(r)

vector_ratings <- as.vector(r@data)
unique(vector_ratings)

# we will assume that a rating of 0 represents a missing value, so we can remove them from vector_ratin
vector_ratings <- vector_ratings[vector_ratings != 0]
table_ratings <- table(vector_ratings)
kable(data.frame(table_ratings))


# Visualizing a sample
image(sample(r, 500), main = "Raw ratings")



##########################
# Visualizing ratings
qplot(getRatings(r), binwidth = 1,
```

```
      main = "Histogram of ratings", xlab = "Rating")

summary(getRatings(r))

############################
# How about after normalization... "It looks better"
qplot(getRatings(normalize(r, method = "Z-score")),
      main = "Histogram of normalized ratings", xlab = "Rating")
summary(getRatings(normalize(r, method = "Z-score")))

##############################

# How many movies did people rate on average
qplot(rowCounts(r), binwidth = 10,
      main = "Movies Rated on average",
      xlab = "# of users",
      ylab = "# of movies rated")


###############


# What is the mean rating of each movie
qplot(colMeans(r), binwidth = .1,
      main = "Mean rating of Movies",
      xlab = "Rating",
      ylab = "# of movies")


#df <- data.frame(name=character(),RMSE=numeric,MSE=numeric,MAE=numeric)


EvaluationSummary <- function(acc_matrix, methodName)
{
  df1<-as.data.frame(apply(acc_matrix,2,mean,na.rm=T))
  names(df1) <- methodName
  return(df1)
}


ratings_movies <- MovieLense
percentage_training <- 0.8
items_to_keep <- 10
rating_threshold <- 3
eval_sets <- evaluationScheme(data = ratings_movies, method = "split",
                              train = percentage_training,
                              given = items_to_keep,
                              goodRating =rating_threshold, k = NULL)




rec <- Recommender(data = getData(eval_sets, "train"),
```

```
                                method = 'UBCF',
                                param=list(normalize = "Z-score",method="Cosine"))

pred <- predict(object = rec,
                            newdata =getData(eval_sets, "known"),
                            n = 10, type = "ratings")

acc <- calcPredictionAccuracy(x = pred,
                                        data = getData(eval_sets, "unknown"),
                                        byUser =TRUE)

head(acc)

dfEvalSummary <- EvaluationSummary(acc,'UBCF-Cosine')







rec <- Recommender(data = getData(eval_sets, "train"),
                                method = 'UBCF',
                                param=list(normalize = "Z-score",method="Jaccard"))

pred <- predict(object = rec,
                            newdata =getData(eval_sets, "known"),
                            n = 10, type = "ratings")

acc <- calcPredictionAccuracy(x = pred,
                                        data = getData(eval_sets, "unknown"),
                                        byUser =TRUE)

head(acc)

dfEvalSummary <- cbind(dfEvalSummary,EvaluationSummary(acc,'UBCF-Jaccard'))







rec <- Recommender(data = getData(eval_sets, "train"),
                                method = 'UBCF',
                                param=list(normalize = "Z-score",method="Pearson"))

pred <- predict(object = rec,
                            newdata =getData(eval_sets, "known"),
                            n = 10, type = "ratings")

acc <- calcPredictionAccuracy(x = pred,
                                        data = getData(eval_sets, "unknown"),
                                        byUser =TRUE)
```

```r
head(acc)

dfEvalSummary <- cbind(dfEvalSummary,EvaluationSummary(acc,'UBCF-Pearson'))




# IBCF Cosine

rec <- Recommender(data = getData(eval_sets, "train"),
                                method = 'IBCF',
                                param=list(normalize = "Z-score",method="Cosine"))

pred <- predict(object = rec,
                            newdata =getData(eval_sets, "known"),
                            n = 10, type = "ratings")

acc <- calcPredictionAccuracy(x = pred,
                                        data = getData(eval_sets, "unknown"),
                                        byUser =TRUE)

head(acc)

dfEvalSummary <- cbind(dfEvalSummary,EvaluationSummary(acc,'IBCF-Cosine'))


rec <- Recommender(data = getData(eval_sets, "train"),
                                method = 'IBCF',
                                param=list(normalize = "Z-score",method="Jaccard"))

pred <- predict(object = rec,
                            newdata =getData(eval_sets, "known"),
                            n = 10, type = "ratings")

acc <- calcPredictionAccuracy(x = pred,
                                        data = getData(eval_sets, "unknown"),
                                        byUser =TRUE)

head(acc)

dfEvalSummary <- cbind(dfEvalSummary,EvaluationSummary(acc,'IBCF-Jaccard'))


rec <- Recommender(data = getData(eval_sets, "train"),
                                method = 'IBCF',
                                param=list(normalize = "Z-score",method="Pearson"))

pred <- predict(object = rec,
                            newdata =getData(eval_sets, "known"),
                            n = 10, type = "ratings")
```

```
acc <- calcPredictionAccuracy(x = pred,
                                        data = getData(eval_sets, "unknown"),
                                        byUser =TRUE)

head(acc)

dfEvalSummary <- cbind(dfEvalSummary,EvaluationSummary(acc,'IBCF-Pearson'))



avg <- apply(as.matrix(dfEvalSummary),2,mean)

df1 <- as.data.frame(t(avg))
rownames(df1) <- 'AverageOfAllErrors'
kable(rbind(dfEvalSummary,df1), caption = 'Evaluation Summary')


final_rec <- Recommender(data = MovieLense,
                                 method = 'UBCF',
                                 param=list(normalize = "Z-score",method="Cosine"))

pred <- predict(object = final_rec, newdata =MovieLense, n = 10, type = "ratings")

m1 <- head(as(pred,'matrix'))
m1 <- m1['2',2:4,drop=FALSE]

m <- as(MovieLense,'matrix')
m2 <- m['2',2:4,drop=FALSE]

df2 <- rbind(as.data.frame(m2), as.data.frame(m1))
rownames(df2) <- c('Before','After')
kable(df2, caption ='Test Using UBCF Cosine')

##
```