DATA 612 Final Project

Brian K. Liles

July 18, 2019

Final Project

The term **recommender system** may seem foreign to many, but we interact with these systems everyday. Years ago, when brick-and-mortar establishments reigned supreme, recommender systems existed on a human level. For example, a retail store clerk could recommend a pair of shoes to a customer after they selected a certain shirt. Currently, customers are offered recommendations at lightening speeds because of machine learning algorithms. Once a product is selected on Amazon, further options are at your disposal. Music streaming services provide their customers with similar songs based off prior selections. In addition, platforms like Netflix and Hulu do the same based off content based and collaborative filtering. During this project, we will look at two csv data sets that provide ratings and content for the **anime** genre.

Libraries

Data

From the popular database **Kaggle** the **Anime Recommendation Database** was downloaded from the following location https://www.kaggle.com/CooperUnion/anime-recommendations-database

The **anime** data set was uploaded to **GitHub** while the larger **ratings** data set will be uploaded from local machine; *data set will be uploaded to CUNY repository*.

After several failed attempts, the animme and ratings data created issues once the sparse matrix was being created. During the first run of attempts, error messages stating duplicate rows surfaced. After searching the internet for solutons, I tried using the reshape2 package and the melt function which didn't rectify the issue. Lastly, an attempt of grouping the data and creating an index also failed, an example of the code is listed below

convert to matrix rating <- rating %>% group_by(anime_id) %>% mutate(grouped_id = row_number())

head(rMtrx)

In order to see what datasets are available from the **recommenderlab** package we run the following code; we will be using the **MSWeb** data.

```
data_package <- data(package = "recommenderlab")
data_package$results[,c("Item","Title")]
```

```
##      Item
## [1,] "Jester5k"
## [2,] "JesterJokes (Jester5k)"
## [3,] "MSWeb"
## [4,] "MovieLense"
## [5,] "MovieLenseMeta (MovieLense)"
##      Title
## [1,] "Jester dataset (5k sample)"
## [2,] "Jester dataset (5k sample)"
## [3,] "Anonymous web data from www.microsoft.com"
## [4,] "MovieLense Dataset (100k)"
## [5,] "MovieLense Dataset (100k)"
```

Load the **MovieLense** data into R

```
data("MovieLense")
MovieLense
```

```
## 943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
```

```
# an alternative to obtaining the number of ratings is the nratings function
nratings(MovieLense)
```

```
## [1] 99392
```

Using the **summary** function in tandem with the **recommenderlab** package we can get an idea what is going on under the hood of the data.

```
summary(rowCounts(MovieLense)) # number of ratings per row
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    19.0    32.0    64.0   105.4   147.5   735.0
```

```
summary(rowMeans(MovieLense)) # row-wise rating means
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.497   3.325   3.619   3.588   3.871   4.870
```

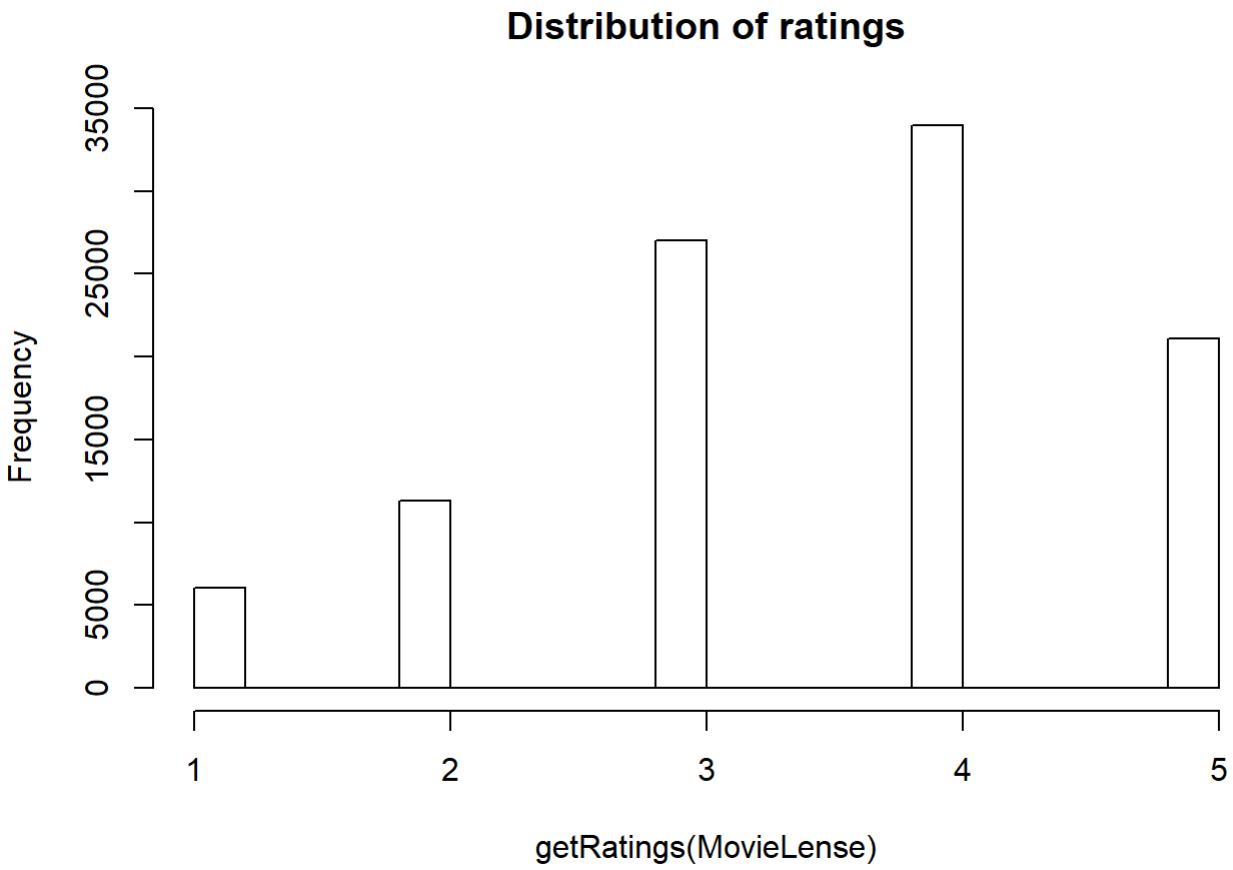
```
summary(colCounts(MovieLense)) # number of ratings per column
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    7.00   27.00   59.73   80.00  583.00
```

```
summary(colMeans(MovieLense)) # column-wise rating means
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.665   3.162   3.077   3.653   5.000
```

```
hist(getRatings(MovieLense), main="Distribution of ratings")
```



Based off **Building a Recommendation System with R** we can visualize the values of the ratings

```
ratings <- as.vector(MovieLense@data)
(tblRatings <- table(ratings))
```

```
## ratings
##       0       1       2       3       4       5
## 1469760    6059   11307   27002   33947   21077
```

Next, let's take a look at the best and worst movies within the matrix

```
bestCol <- which.max(colMeans(MovieLense))
bestCol
```

```
## Great Day in Harlem, A (1994)
##                           808
```

```
worstCol <- which.min(colMeans(MovieLense))
worstCol
```

```
## 3 Ninjas: High Noon At Mega Mountain (1998)
##                                         312
```

An interesting visual technique is where the end user can explore the top percentile of users and movies.

```
cat("Top Five Percent of Movies","\n")
```

```
## Top Five Percent of Movies
```

```
(minMovies <- quantile(rowCounts(MovieLense), 0.95))
```

```
##   95%
## 309.7
```

```
cat("\n","Top Five Percent of Users","\n")
```

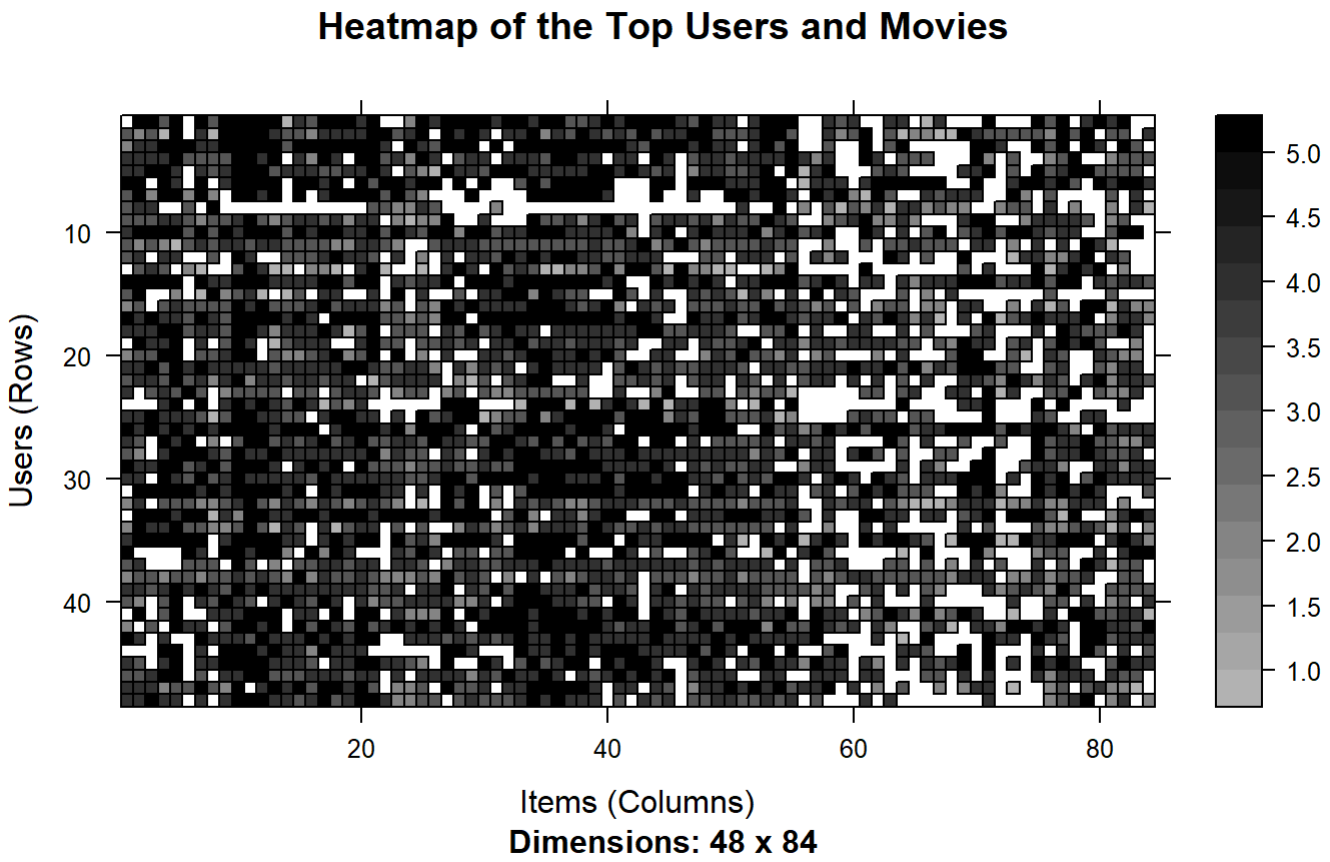
```
##
##  Top Five Percent of Users
```

```
(minUsers <- quantile(colCounts(MovieLense), 0.95))
```

```
##    95%
## 229.55
```

Next, we will visualize a heat map of the top users and movies

```
image(MovieLense[rowCounts(MovieLense) > minMovies,
                 colCounts(MovieLense) > minUsers],
                 main = "Heatmap of the Top Users and Movies")
```



Data Preparation

In the text, the recommendation models was constructed based off users who rated at least 50 movies. In our case, we will select users who have at least rated 25. We will call the data **silver**

```
(silver <- MovieLense[rowCounts(MovieLense) > 25,
                      colCounts(MovieLense) > 50])
```

```
## 799 x 591 rating matrix of class 'realRatingMatrix' with 80045 ratings.
```

Next, we will visualize the top 5% of users and movies

```
cat("Top Five Percent  of Movies","\n")
```

```
## Top Five Percent  of Movies
```

```
(minMovies <- quantile(rowCounts(silver), 0.95))
```

```
##   95%
## 259.2
```

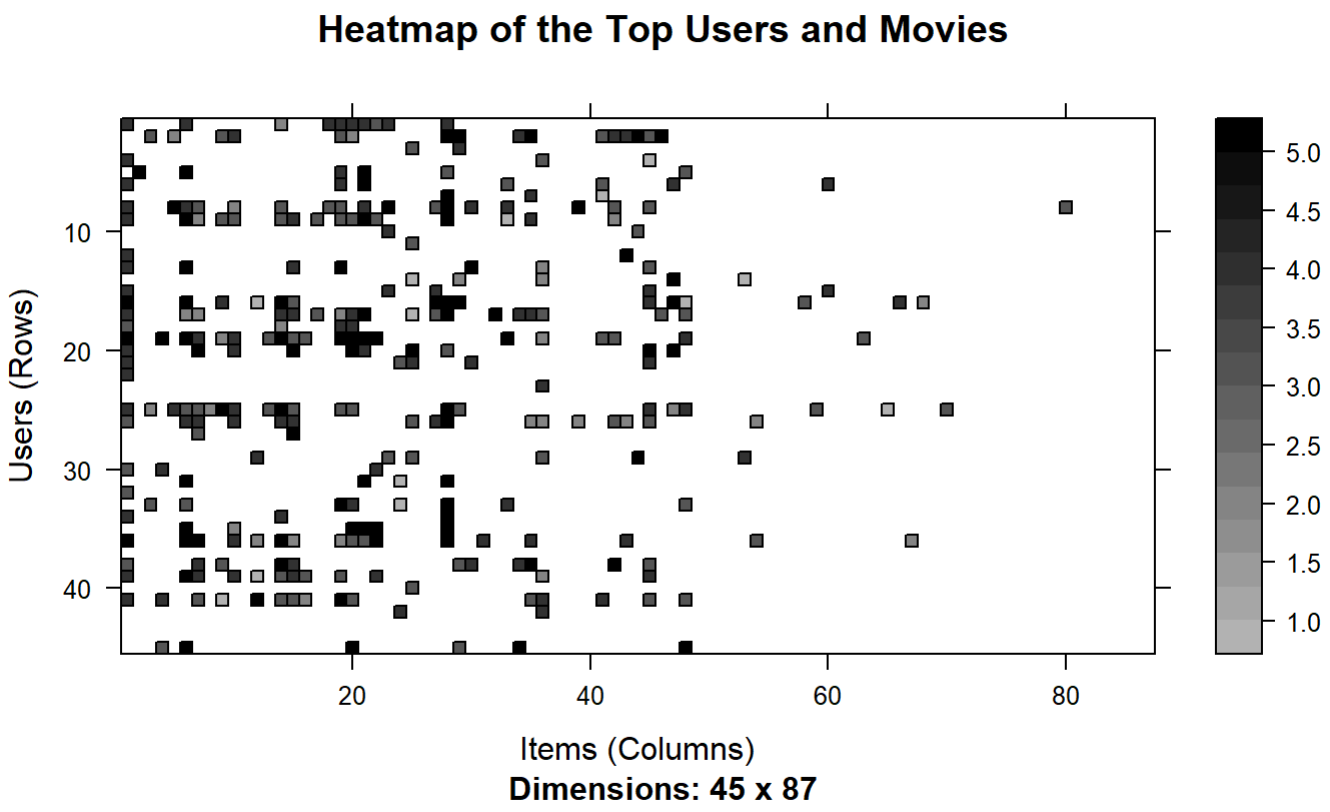
```
cat("\n","Top Five Percent of Users","\n")
```

```
##
##  Top Five Percent of Users
```

```
(minUsers <- quantile(colCounts(silver), 0.95))
```

```
## 95%
## 292
```

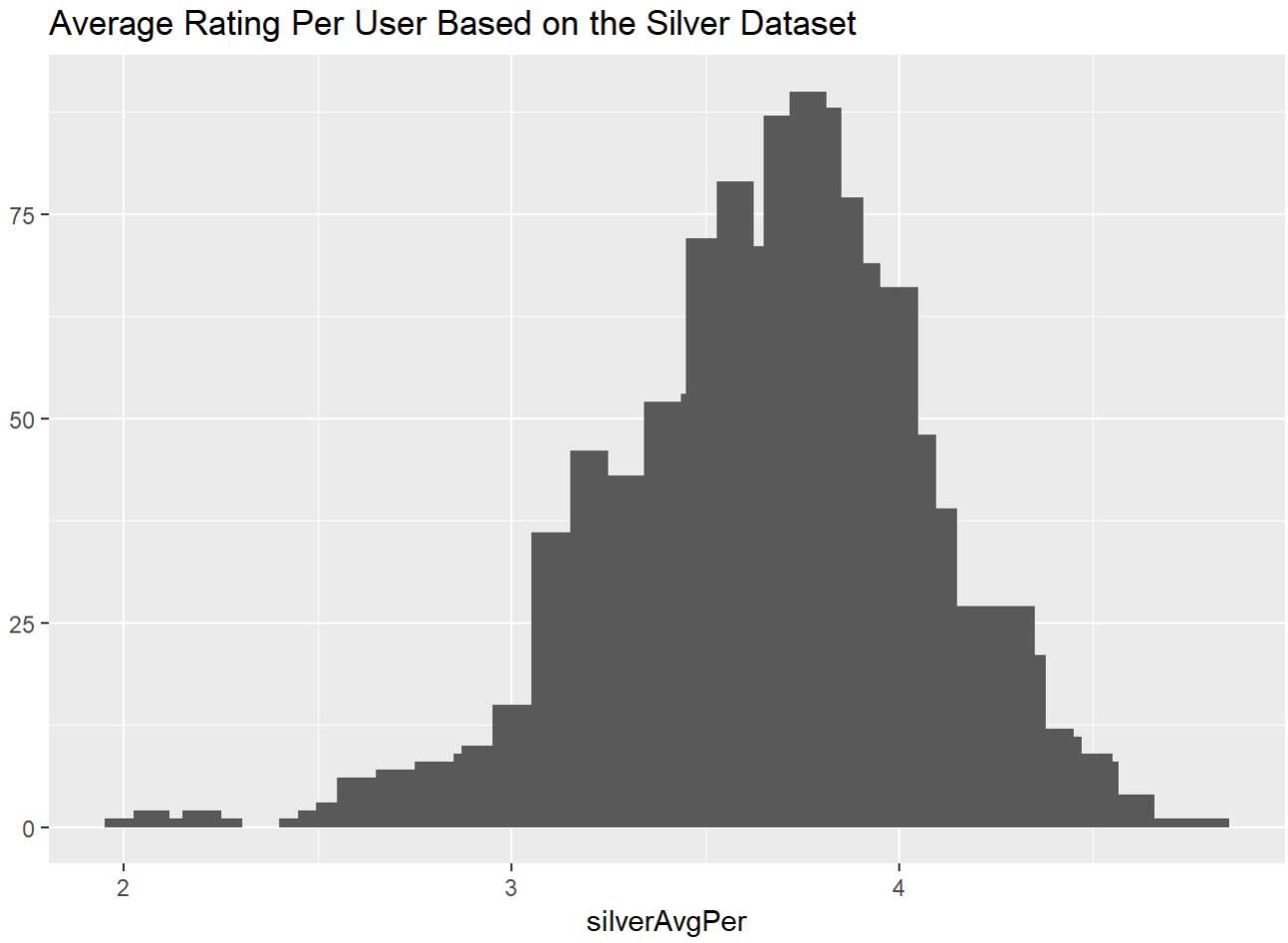
```
image(MovieLense[rowCounts(silver) > minMovies,
                 colCounts(silver) > minUsers],
                 main = "Heatmap of the Top Users and Movies")
```



Next, we will visualize the average rating per user in the silver data set

```
silverAvgPer <- rowMeans(silver)

qplot(silverAvgPer) +
    stat_bin(binwidth = 0.1) +
    ggtitle("Average Rating Per User Based on the Silver Dataset")
```

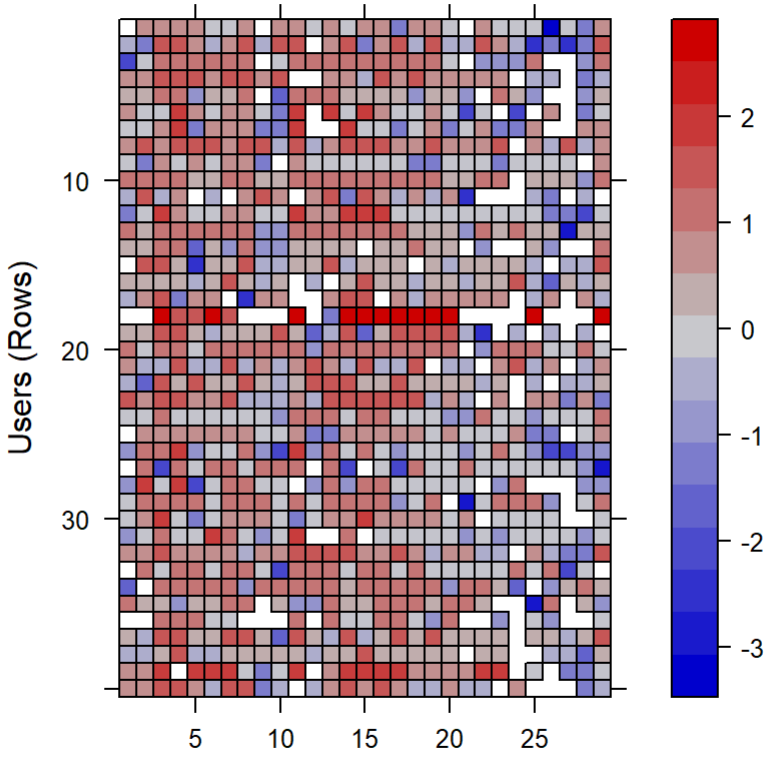
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Based off the graph, one can see that very few raters scored movies as a 2 while the bulk of the ratings were 3.5. Next, we will normalize the data

```
# normalizing the data
silverNorm <- normalize(silver)

image(silverNorm[rowCounts(silverNorm) > minMovies,
                 colCounts(silverNorm) > minUsers],
                 main = "Heatmap of the Top Users and Movies Based on Normalized Data")
```

Heatmap of the Top Users and Movies Based on Normalized Data



Dimensions: 40 x 29

Recommender Lab Models

In past recommendation models during the class, the **IBCF** and **UBCF** algorithm proved best so it will be the one used for this project.

```
recModels <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
recModels$IBCF_realRatingMatrix$parameters
```

```
## $k
## [1] 30
##
## $method
## [1] "Cosine"
##
## $normalize
## [1] "center"
##
## $normalize_sim_matrix
## [1] FALSE
##
## $alpha
## [1] 0.5
##
## $na_as_zero
## [1] FALSE
```

```
recModels$UBCF_realRatingMatrix$parameters
```

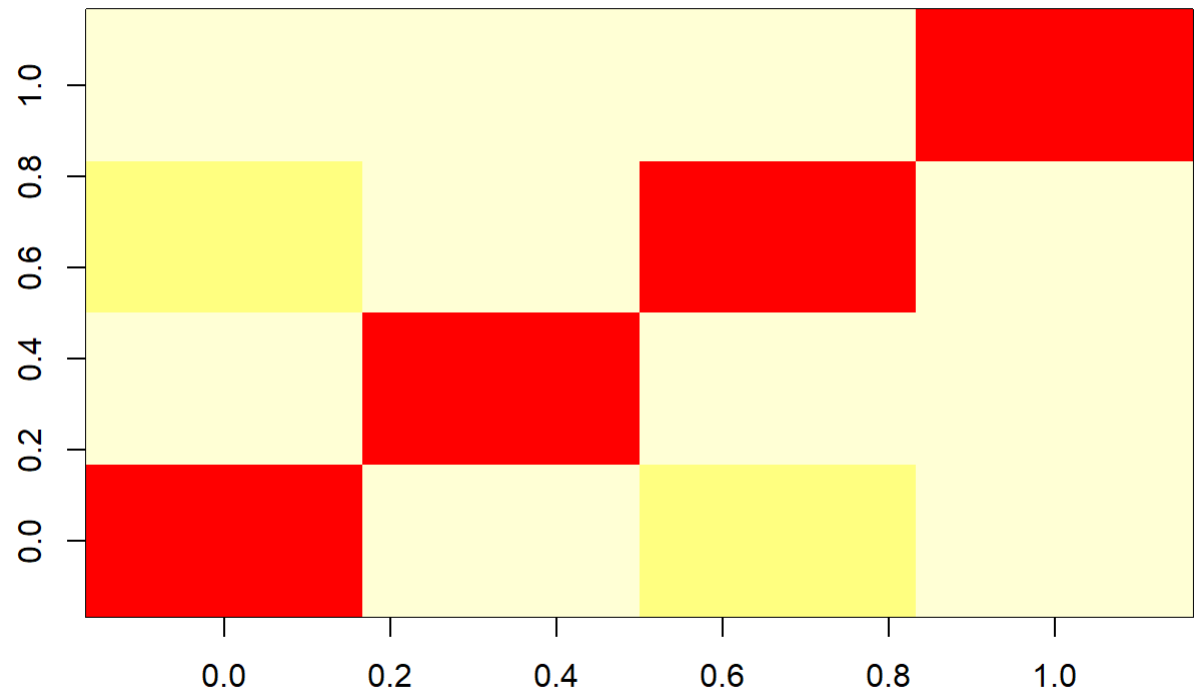
```
## $method
## [1] "cosine"
##
## $nn
## [1] 25
##
## $sample
## [1] FALSE
##
## $normalize
## [1] "center"
```

We will now check the similarity between users using the **similarity** function; this will be conducted based on the first four users based on the **cosine** method and then on the first four movies

```
simUsers <- similarity(silver[1:4, ],
                               method = "cosine",
                               which = "users")
as.matrix(simUsers) # view as a matrix
```

```
##           1         2         3         5
## 1 0.0000000 0.9605115 0.8339504 0.9309778
## 2 0.9605115 0.0000000 0.9268716 0.9848027
## 3 0.8339504 0.9268716 0.0000000 1.0000000
## 5 0.9309778 0.9848027 1.0000000 0.0000000
```

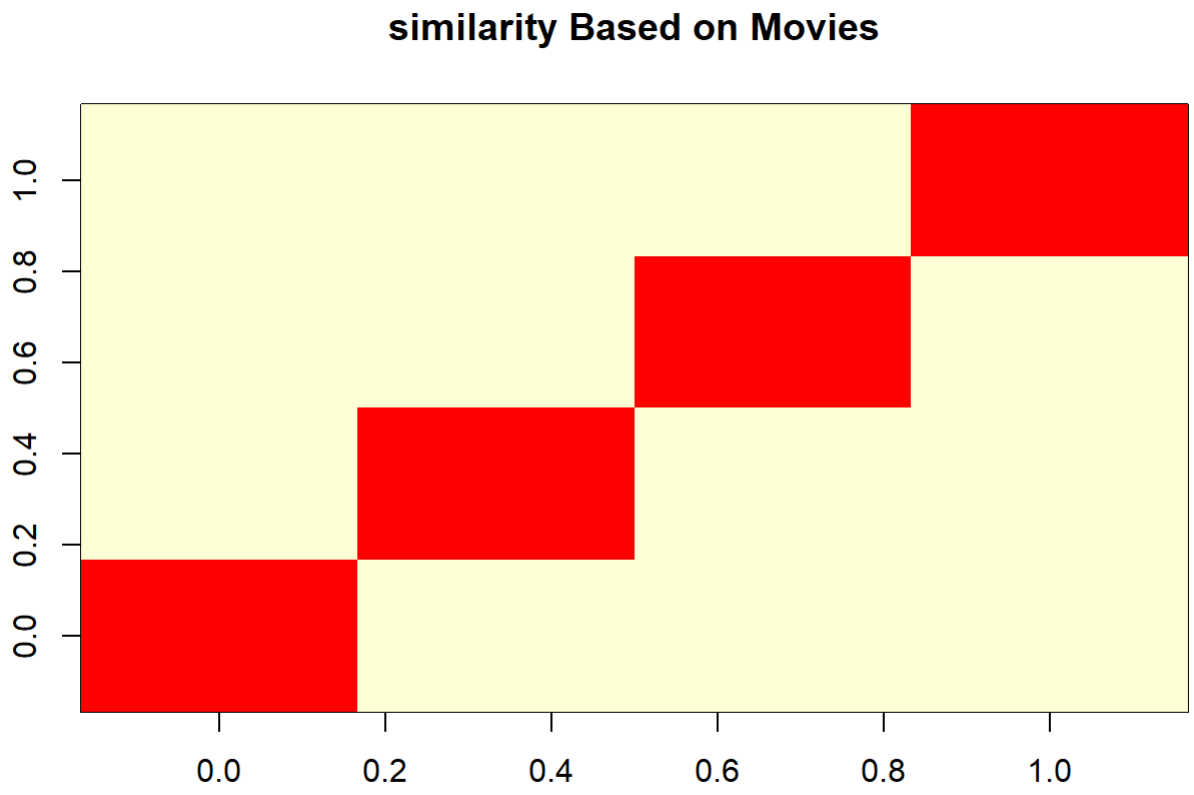
```
# view the similarity output
image(as.matrix(simUsers), main = "Similarity Based on Users")
```




```
simMovies <- similarity(silver[,1:4 ],
                            method = "cosine",
                            which = "items")
as.matrix(simMovies) # view as a matrix
```

```
##                   Toy Story (1995) GoldenEye (1995) Four Rooms (1995)
## Toy Story (1995)          0.0000000        0.9487374         0.9132997
## GoldenEye (1995)          0.9487374        0.0000000         0.9088797
## Four Rooms (1995)         0.9132997        0.9088797         0.0000000
## Get Shorty (1995)         0.9429069        0.9394926         0.8991940
##                   Get Shorty (1995)
## Toy Story (1995)          0.9429069
## GoldenEye (1995)          0.9394926
## Four Rooms (1995)         0.8991940
## Get Shorty (1995)         0.0000000
```

```
# view the similarity output
image(as.matrix(simMovies), main = "similarity Based on Movies")
```



Item Based Collaborative Filtering Model

Utilizing the collobarative filtering model which is based on the history of users preferences. The first step we will take is create the training/test sets based off the 80-20 method for the training and testing sets respectively.

```
set.seed(50)
sample <- sample.int(n = nrow(silver), size = floor(.80*nrow(silver)), replace = F)
silverTrain <- silver[sample,]
silverTest <- silver[-sample,]
```

```
# check the nratings for the training data set
nratings(silverTrain)
```

```
## [1] 64210
```

```
# check the nratings for the test data set
nratings(silverTest)
```

```
## [1] 15835
```

Within this model we will recommend 5 movies to each user

```
IBCF_model <- Recommender(data = silverTrain,
                          method = "IBCF",
                          parameter = list(k = 30))


n_recommended <- 5 # the number of items to recommend to each user
IBCF_predicted <- predict(object = IBCF_model,
                          newdata = silverTest,
                          n = n_recommended)
IBCF_predicted
```

```
## Recommendations as 'topNList' with n = 5 for 160 users.
```

Exploring the IBCF Model

```
IBCF_details <- getModel(IBCF_model)
IBCF_details$description
```

```
## [1] "IBCF: Reduced similarity matrix"
```

```
IBCF_details$k
```

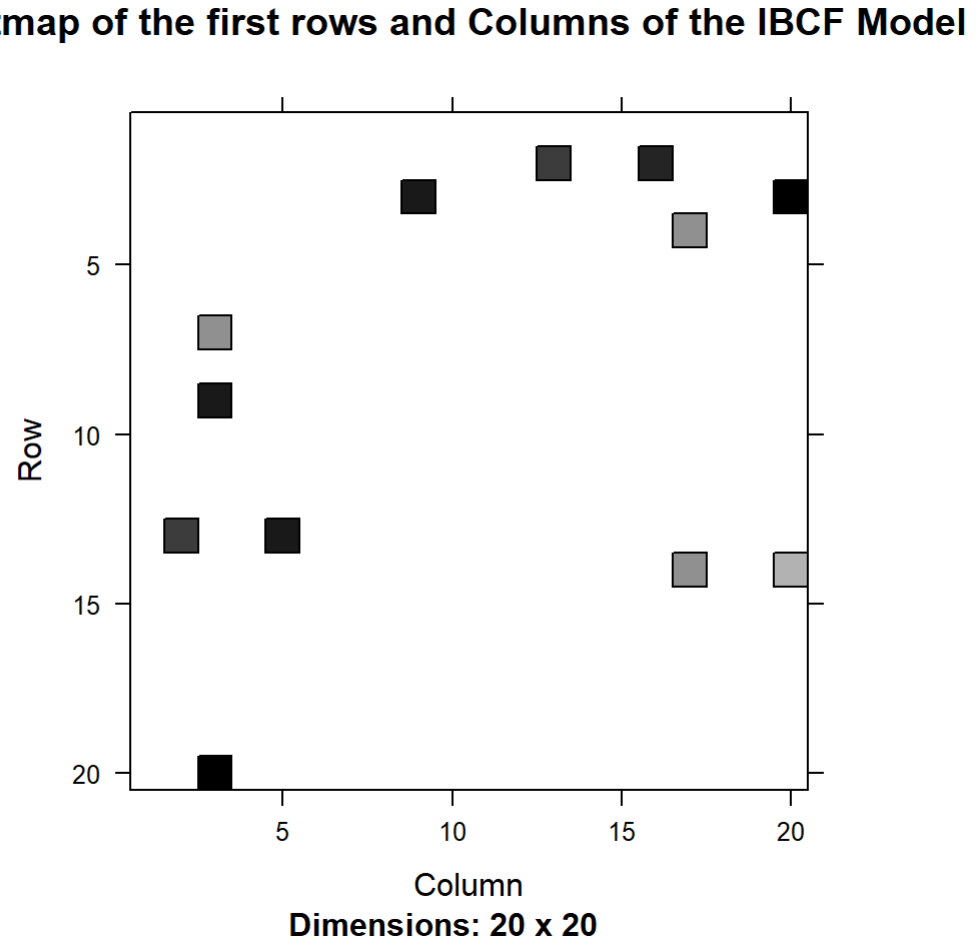
```
## [1] 30
```

```
class(IBCF_details$sim)
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

Next, we will create a heat map of the **IBCF** model

```
image(IBCF_details$sim[1:20,1:20], main = "Heatmap of the first rows and Columns of the IBCF Mode
l")
```



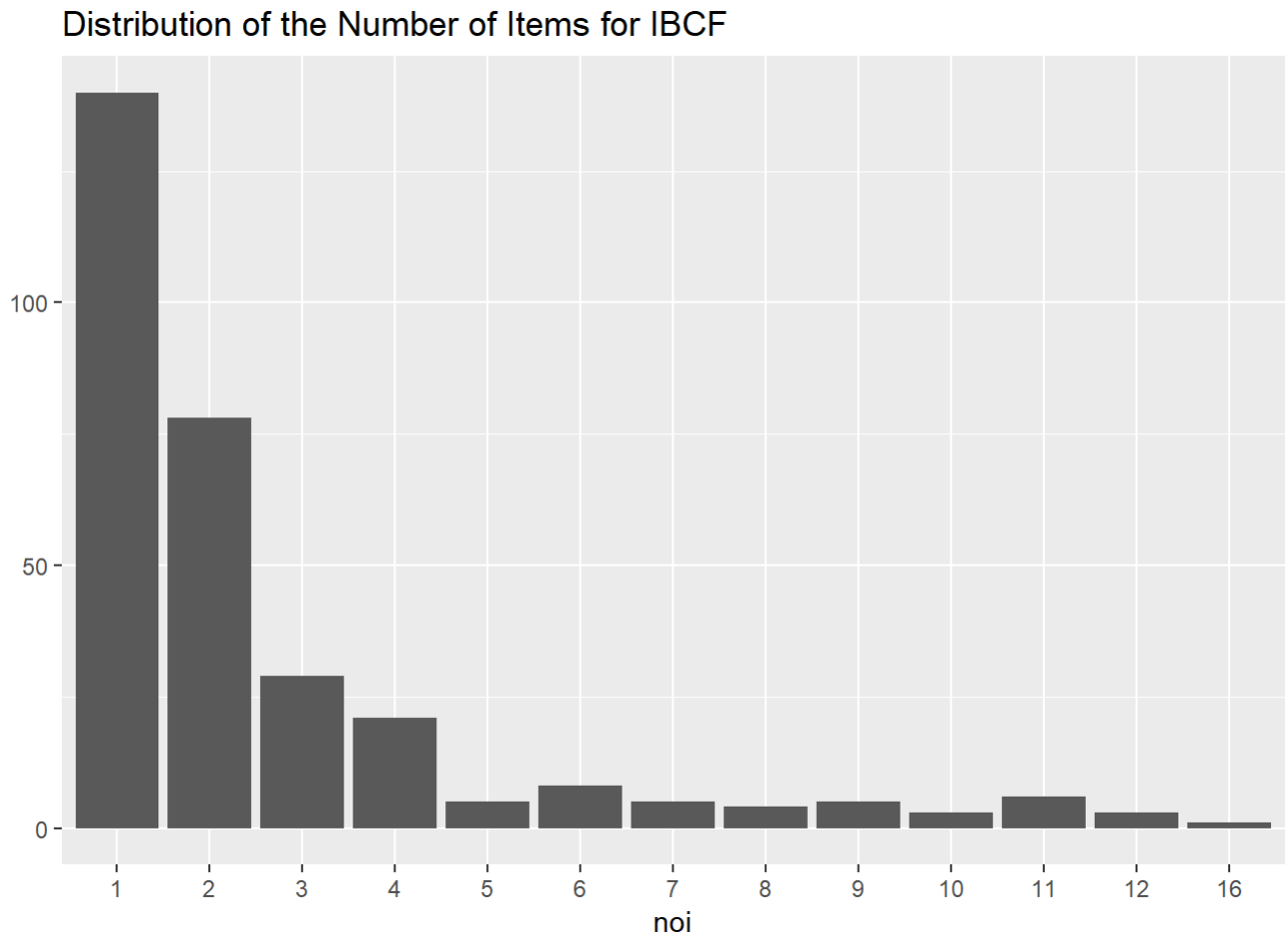
Next, we will see which movies have the most elements

```
col_sums <- colSums(IBCF_details$sim > 0)

IBCF_max <- order(col_sums, decreasing = TRUE)[1:6]
rownames(IBCF_details$sim)[IBCF_max]
```

```
## [1] "Flubber (1997)"        "Excess Baggage (1997)" "Postman, The (1997)"
## [4] "Apostle, The (1997)"   "Soul Food (1997)"      "Red Corner (1997)"
```

Next, we will create a distribution chart

```
IBCF_matrix <- sapply(IBCF_predicted@items, function(x){
    colnames(silver)[x]})

# number of items
noi <- factor(table(IBCF_matrix))
chart_title <- "Distribution of the Number of Items for IBCF"

# distribution chart
qplot(noi) +
    ggtitle(chart_title)
```



The top rated movies based on the **IBCF** model

```
noiSort <- sort(noi, decreasing = TRUE)
noiTop <- head(noiSort, n = 4)
(topFour <- data.frame(names(noiTop), noiTop))
```

```
##                                       names.noiTop. noiTop
## Rumble in the Bronx (1995) Rumble in the Bronx (1995)     16
## Cable Guy, The (1996)           Cable Guy, The (1996)     12
## Legends of the Fall (1994) Legends of the Fall (1994)     12
## Mr. Holland's Opus (1995)   Mr. Holland's Opus (1995)     12
```

User Based Collaborative Filtering Model

```
UBCF_model <- Recommender( data = silverTrain,
                           method = "UBCF")
UBCF_model
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 639 users.
```

```
UBCF_details <- getModel(UBCF_model)
names(UBCF_details)
```

```
## [1] "description" "data"        "method"      "nn"          "sample"
## [6] "normalize"   "verbose"
```

Applyting the Recommender Model

```
UBCF_predicted <- predict(object = UBCF_model, newdata = silverTest, n = 6)
UBCF_predicted
```

```
## Recommendations as 'topNList' with n = 6 for 160 users.
```

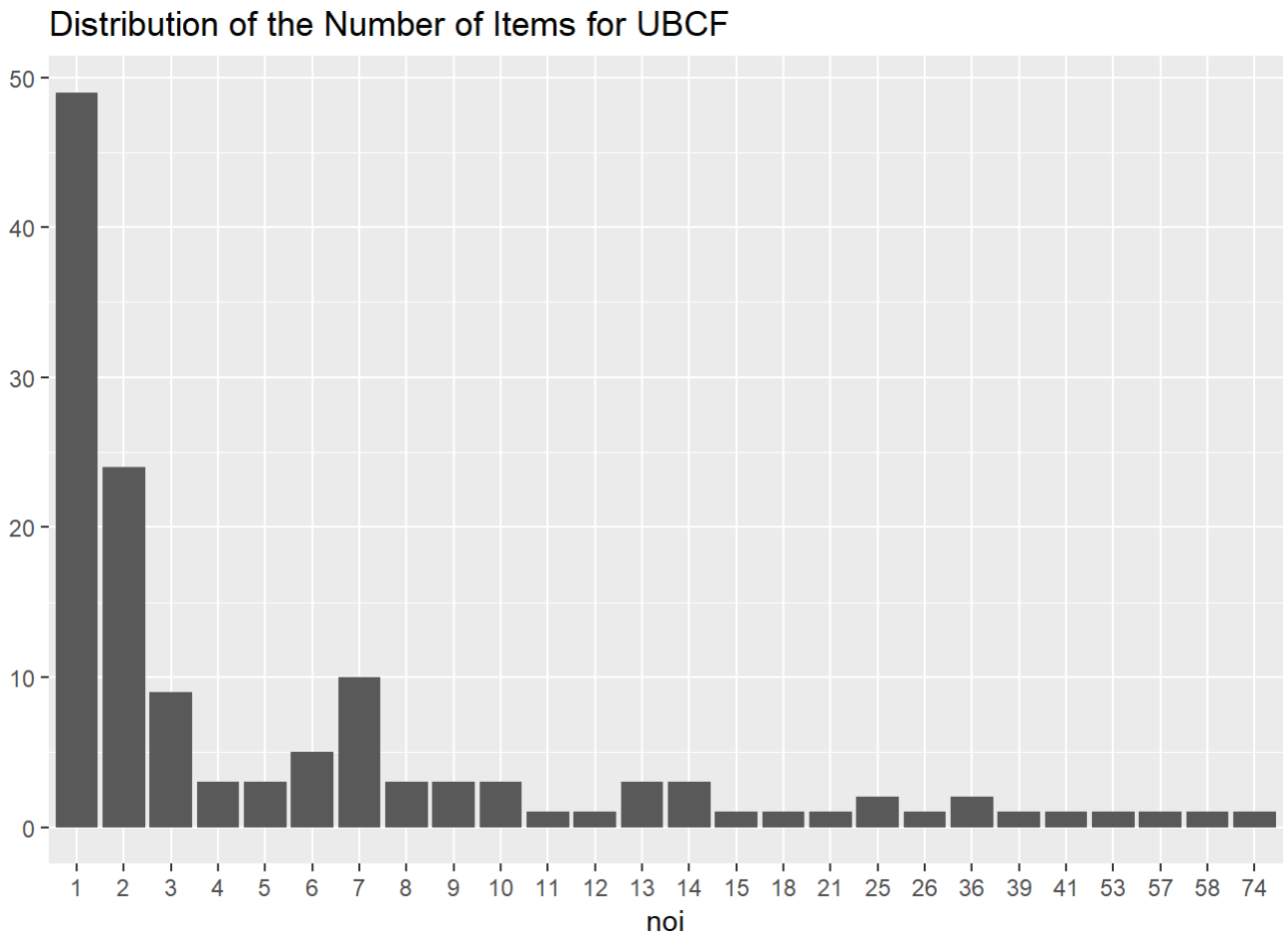
Next, we will create a matrix with the recommended data

```
UBCF_matrix <- sapply(UBCF_predicted@items, function(x){colnames(silver)[x]})
UBCF_matrix[,1:4]
```

```
##      8                            12
## [1,] "L.A. Confidential (1997)"   "Titanic (1997)"
## [2,] "Apt Pupil (1998)"           "Good Will Hunting (1997)"
## [3,] "Kolya (1996)"               "L.A. Confidential (1997)"
## [4,] "Full Monty, The (1997)"     "Full Monty, The (1997)"
## [5,] "Postino, Il (1994)"         "English Patient, The (1996)"
## [6,] "English Patient, The (1996)" "Apostle, The (1997)"
##      13                           31
## [1,] "Third Man, The (1949)"      "Star Wars (1977)"
## [2,] "Men in Black (1997)"        "Fargo (1996)"
## [3,] "Raging Bull (1980)"         "Godfather, The (1972)"
## [4,] "Chasing Amy (1997)"         "Schindler's List (1993)"
## [5,] "Fried Green Tomatoes (1991)" "Contact (1997)"
## [6,] "Gone with the Wind (1939)"  "Return of the Jedi (1983)"
```

Next, we will create a distribution chart

```
# number of items
noi <- factor(table(UBCF_matrix))
chart_title <- "Distribution of the Number of Items for UBCF"

# distribution chart
qplot(noi) +
    ggtitle(chart_title)
```



The top rated movies based on the **UBCF** model

```
noiSort <- sort(noi, decreasing = TRUE)
noiTop <- head(noiSort, n = 4)
(topFour <- data.frame(names(noiTop), noiTop))
```

```
##                                   names.noiTop. noiTop
## L.A. Confidential (1997) L.A. Confidential (1997)     74
## Titanic (1997)                     Titanic (1997)     58
## Full Monty, The (1997)     Full Monty, The (1997)     57
## Good Will Hunting (1997) Good Will Hunting (1997)     53
```

Evaluating the Ratings

```
models_to_evaluate <- list(
IBCF_cos = list(name = "IBCF",
                param = list(method = "cosine")),
IBCF_cor = list(name = "IBCF",
                param = list(method = "pearson")),
UBCF_cos = list(name = "UBCF",
                param = list(method = "cosine")),
UBCF_cor = list(name = "UBCF",
                param = list(method = "pearson")),
random = list(name = "RANDOM", param=NULL)
)
```

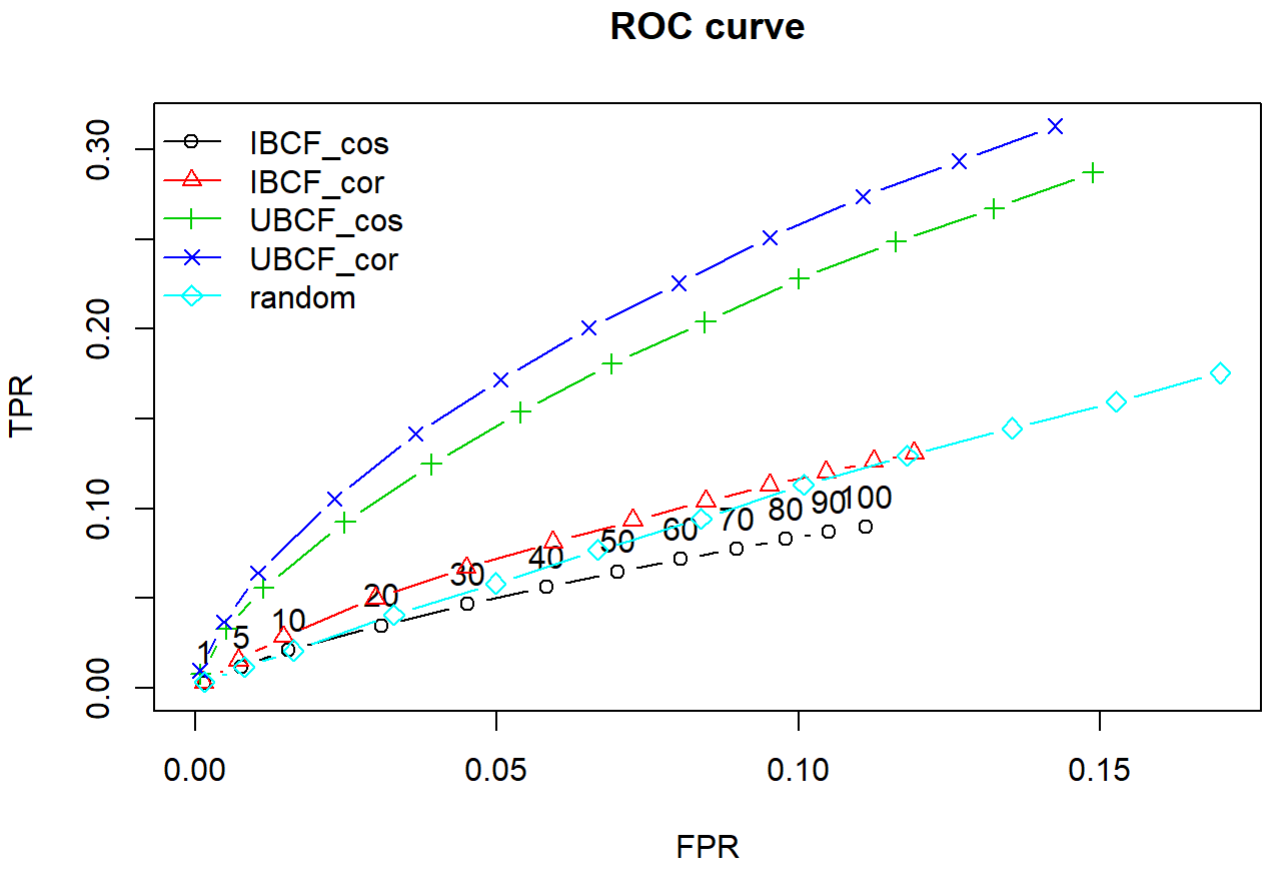
```
eval_sets <- evaluationScheme(data = silver,
                              method = "cross-validation",
                              k = 4,
                              given = 5,
                              goodRating = 3)

n_recommendations <- c(1, 5, seq(10, 100, 10))
list_results <- evaluate(x = eval_sets,
                         method = models_to_evaluate,
                         n = n_recommendations)
```

```
## IBCF run fold/sample [model time/prediction time]
##   1  [5.81sec/0.85sec]
##   2  [5.48sec/0.2sec]
##   3  [5.23sec/0.18sec]
##   4  [5sec/0.18sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [5.23sec/0.2sec]
##   2  [5.13sec/0.21sec]
##   3  [5.03sec/0.21sec]
##   4  [5.08sec/0.2sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.09sec/1.88sec]
##   2  [0.02sec/1.87sec]
##   3  [0.03sec/1.88sec]
##   4  [0.02sec/2.02sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.02sec/1.99sec]
##   2  [0.02sec/2.01sec]
##   3  [0.02sec/2.26sec]
##   4  [0.02sec/2.23sec]
## RANDOM run fold/sample [model time/prediction time]
##   1  [0sec/0.6sec]
##   2  [0.08sec/0.57sec]
##   3  [0sec/0.58sec]
##   4  [0sec/0.58sec]
```

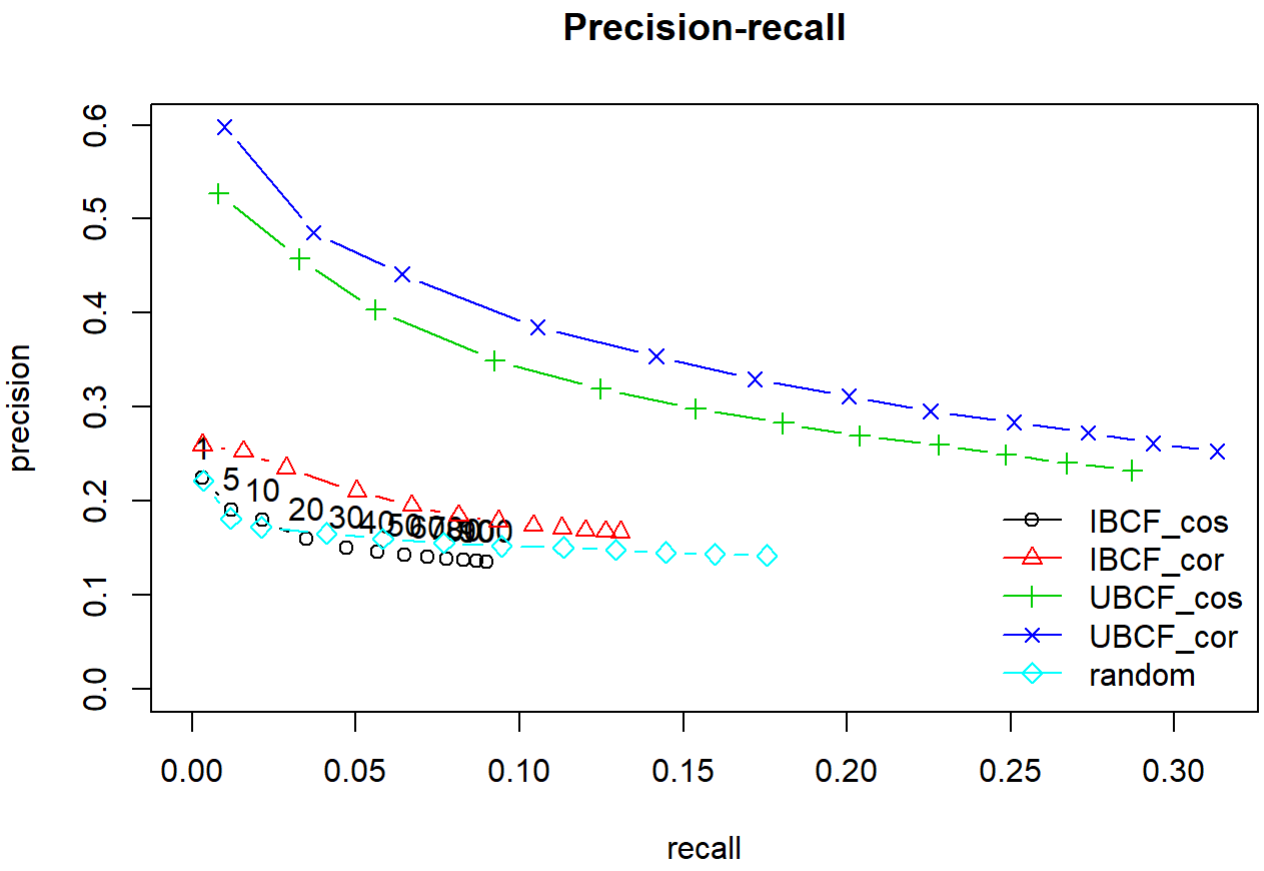
```
sapply(list_results, class) == "evaluationResults"
```

```
## IBCF_cos IBCF_cor UBCF_cos UBCF_cor   random
##     TRUE     TRUE     TRUE     TRUE     TRUE
```

```
plot(list_results, annotate = 1, legend = "topleft")
title("ROC curve")
```




```
plot(list_results, "prec/rec", annotate = 1, legend = "bottomright")
title("Precision-recall")
```



#Conclusion From prior interactions with the **IBCF** and **UBCF** algorithms I favored the prior. In this particular case I choose the **UBCF** algorithm.