Data 612 Project# 2

Joseph Simone

03/01/2020

Contents


```
library(tidyverse)
library(kableExtra)
library(knitr)
library(recommenderlab)
library(dplyr)
```

Content-Based and Collaborative Filtering

Overview

For assignment 2, start with an existing dataset of user-item ratings, such as our toy booksdataset, MovieLens, Jester [http://eigentaste.berkeley.edu/dataset/] or another dataset of your choosing.

Implement at least two of these recommendation algorithms: - Content-Based Filtering - User-User Collaborative Filtering - Item-Item Collaborative Filtering

Data Importation

The dataset I chose for the project is the MovieLens (ml-latest-small) Data-Set. This dataset was created by 610 users between March 29, 1996 and September 24, 2018, emcompassing 9742 movies.

MovieLens DataSet: Posted to my Github

```
ratings <- read.csv(paste0("https://raw.githubusercontent.com/josephsimone/Data-612/master/project_2/Mov
movies <- read.csv(paste0("https://raw.githubusercontent.com/josephsimone/Data-612/master/project_2/Movi
```

```r
movie_matrix <- ratings %>%
  select(-timestamp) %>%
  spread(movieId, rating)
```

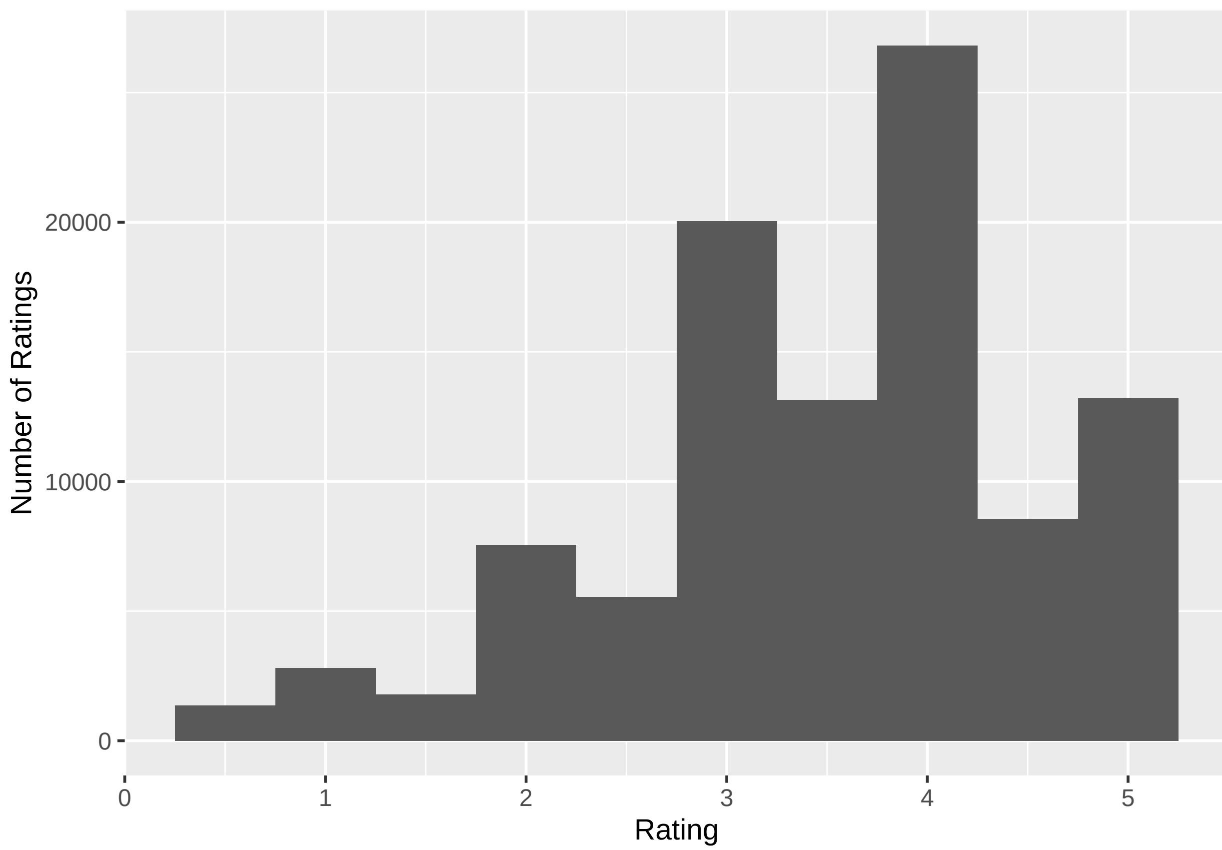
```r
row.names(movie_matrix) <- movie_matrix[,1]
movie_matrix <- movie_matrix[-c(1)]
movie_matrix <- as(as.matrix(movie_matrix), "realRatingMatrix")
```

```r
movie_matrix
```

```
## 610 x 9724 rating matrix of class 'realRatingMatrix' with 100836 ratings.
```

Data Exploration & Data Preporation

```r
num_ratings <- as.vector(movie_matrix@data)
num_ratings <- num_ratings[num_ratings != 0]
ggplot() + aes(num_ratings) +
  geom_histogram(binwidth = 0.5) +
  xlab("Rating") + ylab("Number of Ratings")
```



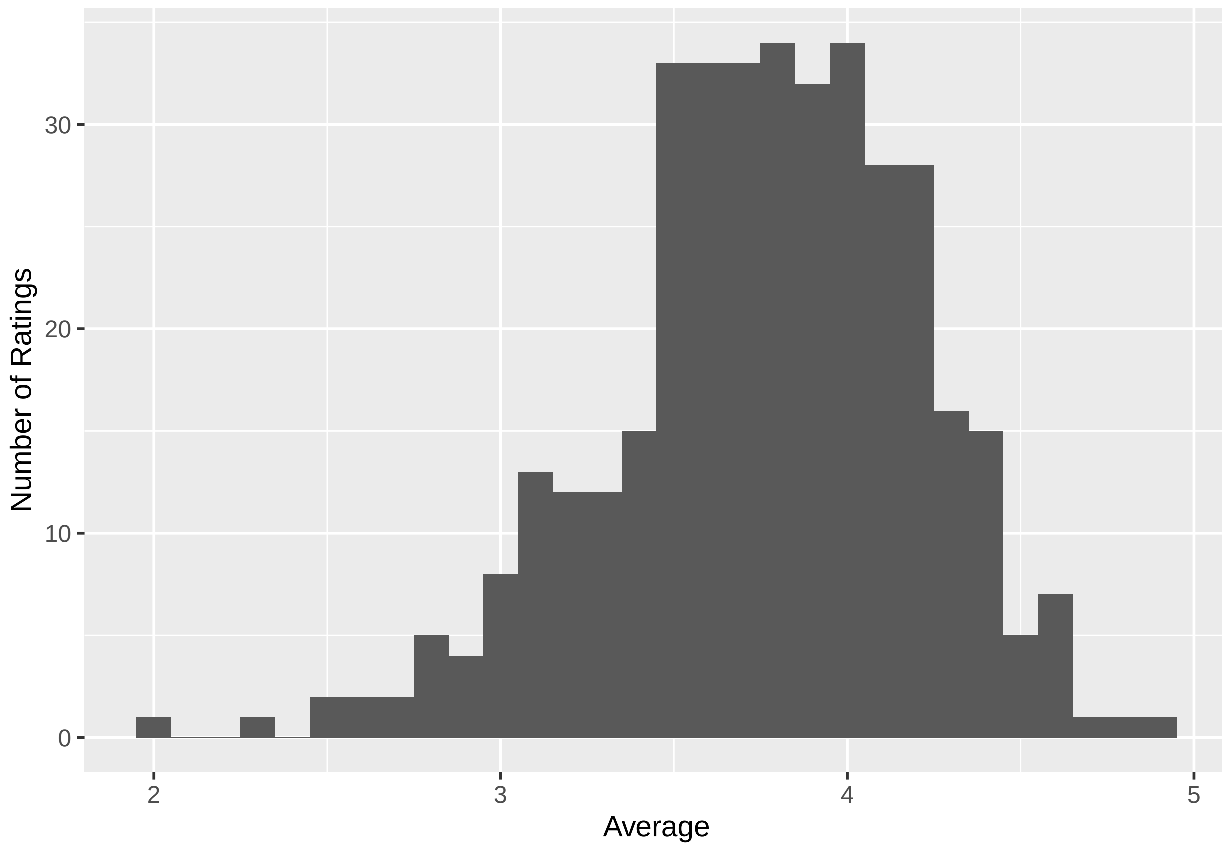
```r
films <- movie_matrix[rowCounts(movie_matrix) > 50, colCounts(movie_matrix) > 50]
films
```

```
## 378 x 436 rating matrix of class 'realRatingMatrix' with 36214 ratings.
```

According to the newly created Ratings Matrix, we may encounter some bias.

Nevertheless, let's explore a Distribution Plot

```r
avg_rating <- rowMeans(films)
ggplot() + aes(avg_rating) +
  geom_histogram(binwidth = 0.1) +
  xlab("Average") + ylab("Number of Ratings")
```




```r
norm_films <- normalize(films)
avg_rating <- round(rowMeans(norm_films),5)
table(avg_rating)
```

Normalization

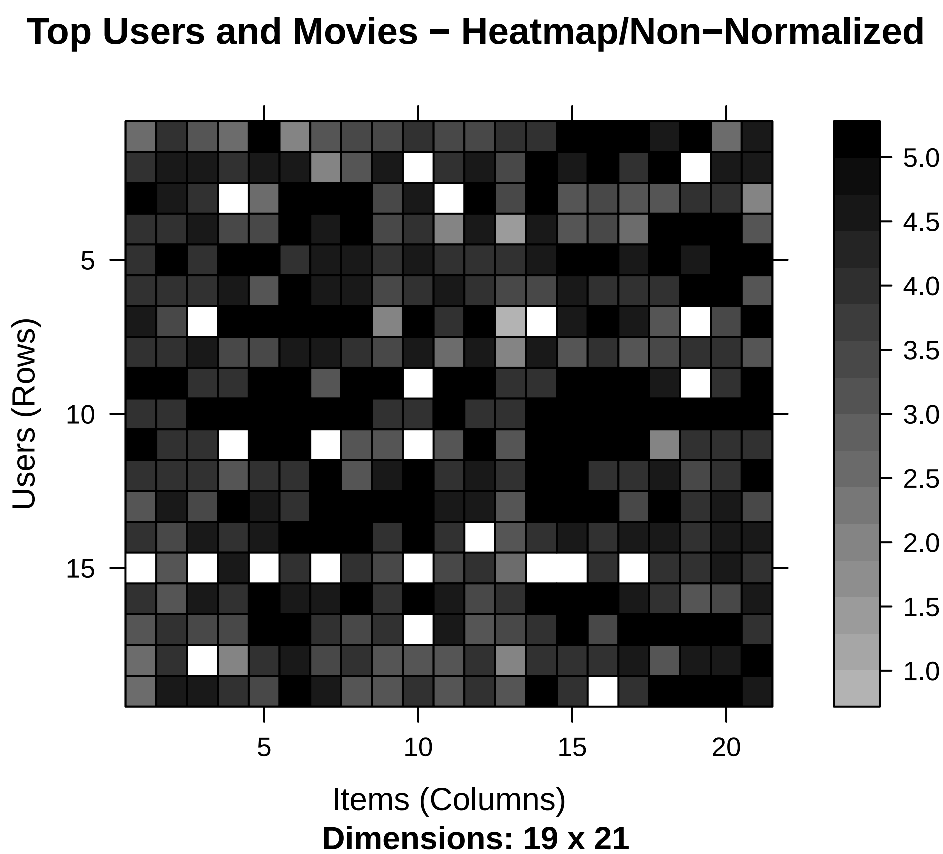
```
## avg_rating
##   0
## 378
```

Normalization of the mean to 0 on 378 rows.

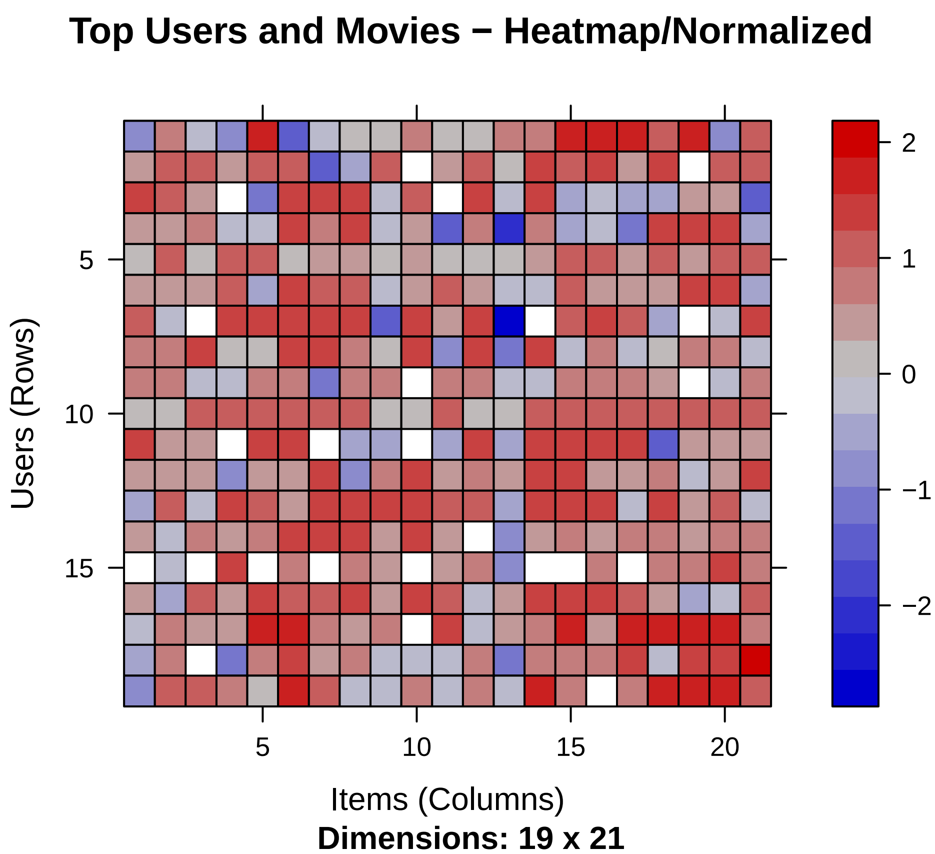
```
min_items <- quantile(rowCounts(films), 0.95)
min_users <- quantile(colCounts(films), 0.95)

image(films[rowCounts(films) > min_items,
                 colCounts(films) > min_users],
      main = "Top Users and Movies - Heatmap/Non-Normalized")
```

Comparision of Non-Normalized & Normalized DataSets.




```
image(norm_films[rowCounts(norm_films) > min_items,
                 colCounts(norm_films) > min_users],
      main = "Top Users and Movies - Heatmap/Normalized")
```



Item to Item Collaborative Filtering

Test & Training Sets Splitting Data - Training Set 80% & Testing Set 20%

```
set.seed(60)
temp_train <- sample(x = c(TRUE, FALSE), size = nrow(films),
                     replace = TRUE, prob = c(0.8, 0.2))
```

```
movie_train <- films[temp_train, ]
movie_test <- films[!temp_train, ]
```

```
movie_train
```

```
## 297 x 436 rating matrix of class 'realRatingMatrix' with 29337 ratings.
```

```
movie_test
```

```
## 81 x 436 rating matrix of class 'realRatingMatrix' with 6877 ratings.
```

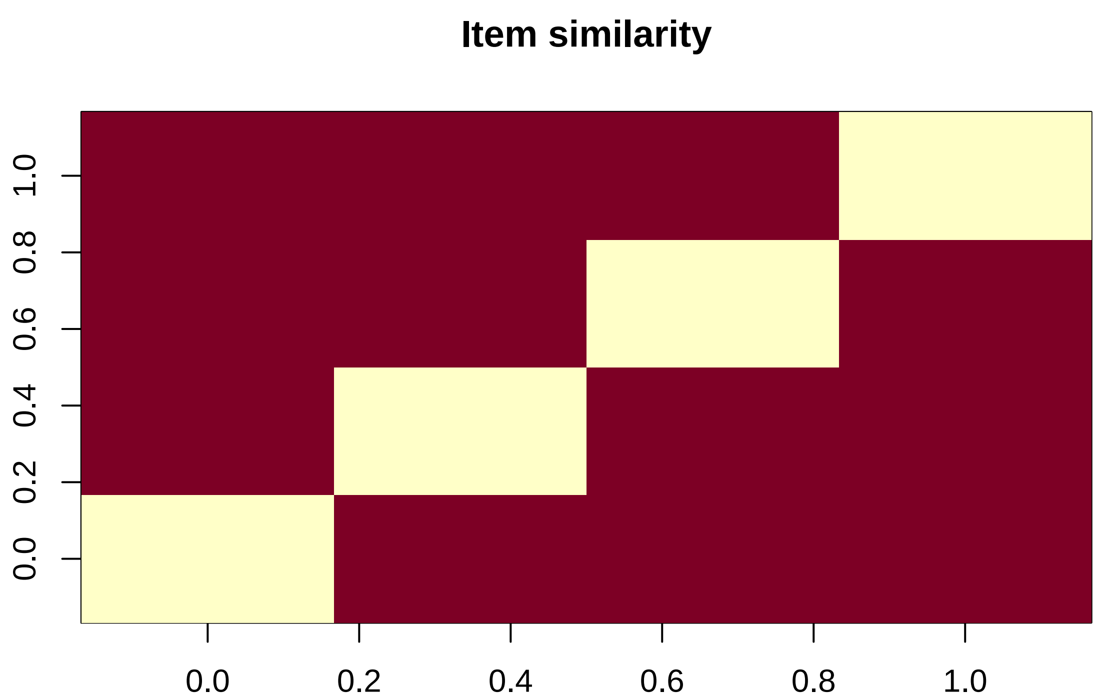
```
movieIBCF <- Recommender(movie_train, method = "IBCF" ,param=list(normalize = "Z-score",method="Jaccard
```

Modeling

Similarity Matrix visualization of the Item Similarity Matrix

```r
similarity_items <- similarity(movie_train[, 1:4], method = "cosine", which = "items")

image(as.matrix(similarity_items), main = "Item similarity")
```



Top Ten Movies and other Movies that are similar.

```r
sim_model <- getModel(movieIBCF)$sim
top_pick <- order(colSums(sim_model > 0), decreasing = TRUE)[1:10]
top_films <- as.data.frame(as.integer(rownames(sim_model)[top_pick]))
```

```r
colnames(top_films) <- c("movieId")
movie_data <- top_films %>% inner_join(movies, by = "movieId") %>% select(Movie = "title")
knitr::kable(movie_data) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

Movie
Harry Potter and the Half-Blood Prince (2009)
Juno (2007)
Inglourious Basterds (2009)
Up (2009)
District 9 (2009)
Sherlock Holmes (2009)
Toy Story 3 (2010)
Inception (2010)
Social Network, The (2010)
The Hunger Games (2012)

```
preditors <- predict(movieIBCF, newdata = movie_test, n = 6)
preditors
```

Recommendations Using Test Set

```
## Recommendations as 'topNList' with n = 6 for 81 users.
```

Movie Ratings for the First User Taking in consideration the First User, pulling Movie Recommendations

```
first_user <- as.data.frame(movie_test@data[1,movie_test@data[1,]>0])
colnames(first_user) <- c("Rating")
first_user[c("movieId")] <- as.integer(rownames(first_user))
first_user_data <- movies %>%
  inner_join(first_user, by = "movieId") %>%
  select(Movie = "title", Rating) %>%
  arrange(desc(Rating))
knitr::kable(first_user_data) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

Movie	Rating
Brazil (1985)	5.0
Taxi Driver (1976)	4.5
Blade Runner (1982)	4.5
Fargo (1996)	4.5
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)	4.5
Full Metal Jacket (1987)	4.5
Chinatown (1974)	4.5
Memento (2000)	4.5
Spirited Away (Sen to Chihiro no kamikakushi) (2001)	4.5
Dark Knight, The (2008)	4.5
Toy Story 3 (2010)	4.5
Dark Knight Rises, The (2012)	4.5
Usual Suspects, The (1995)	4.0
LÃ©on: The Professional (a.k.a. The Professional) (LÃ©on) (1994)	4.0
Shawshank Redemption, The (1994)	4.0
Schindler's List (1993)	4.0
Reservoir Dogs (1992)	4.0
Monty Python and the Holy Grail (1975)	4.0
Wallace & Gromit: The Wrong Trousers (1993)	4.0
One Flew Over the Cuckoo's Nest (1975)	4.0
Princess Bride, The (1987)	4.0
12 Angry Men (1957)	4.0
To Kill a Mockingbird (1962)	4.0
Apocalypse Now (1979)	4.0
Alien (1979)	4.0
Annie Hall (1977)	4.0
Graduate, The (1967)	4.0
Cool Hand Luke (1967)	4.0
Requiem for a Dream (2000)	4.0
Amelie (Fabuleux destin d'AmÃ©lie Poulain, Le) (2001)	4.0
Pan's Labyrinth (Laberinto del fauno, El) (2006)	4.0
WALLÂ · E (2008)	4.0
Up (2009)	4.0
Seven (a.k.a. Se7en) (1995)	3.5
Forrest Gump (1994)	3.5
Rear Window (1954)	3.5
Casablanca (1942)	3.5
Citizen Kane (1941)	3.5
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	3.5
Goodfellas (1990)	3.5
L.A. Confidential (1997)	3.5
Good Will Hunting (1997)	3.5
American History X (1998)	3.5
Matrix, The (1999)	3.5
Sixth Sense, The (1999)	3.5
American Beauty (1999)	3.5
Fight Club (1999)	3.5
Donnie Darko (2001)	3.5
Lord of the Rings: The Fellowship of the Ring, The (2001)	3.5
Lord of the Rings: The Two Towers, The (2002)	3.5
Lord of the Rings: The Return of the King, The (2003)	3.5
Eternal Sunshine of the Spotless Mind (2004)	3.5
Star Wars: Episode IV - A New Hope (1977)	3.0
Pulp Fiction (1994)	3.0
Silence of the Lambs, The (1991)	3.0
Star Wars: Episode V - The Empire Strikes Back (1980)	3.0
Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (1966)	3.0

```r
first_recommendation <- preditors@itemLabels[preditors@items[[1]]]
first_recommendation <- as.data.frame(as.integer(first_recommendation))
colnames(first_recommendation) <- c("movieId")
first_recommendation_data <- first_recommendation %>% inner_join(movies, by = "movieId") %>% select(Movi
knitr::kable(first_recommendation_data) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

Movie
Stargate (1994)
Ace Ventura: Pet Detective (1994)
Lion King, The (1994)
Speed (1994)
Jurassic Park (1993)
Home Alone (1990)

Recommendations for First User

User to User Collaborative Filtering

```r
(movieUBCF <- Recommender(movie_train, method = "UBCF",param=list(normalize = "Z-score",method="Jaccard
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 297 users.
```

```r
( predicted_UBCF <- predict(movieUBCF, newdata = movie_test, n = 6) )
```

Modeling

```
## Recommendations as 'topNList' with n = 6 for 81 users.
```

Recommendations for First User Taking into account the first user again, let's explore some recommendations.

```r
user_recommendation <- predicted_UBCF@itemLabels[predicted_UBCF@items[[1]]]
user_recommendation <- as.data.frame(as.integer(user_recommendation))
colnames(user_recommendation) <- c("movieId")
user_data <- user_recommendation %>% inner_join(movies, by = "movieId") %>% select(Movie = "title")
knitr::kable(user_data) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

Movie
Fugitive, The (1993)
Casino Royale (2006)
Terminator, The (1984)
Back to the Future (1985)
Planet of the Apes (1968)
Jurassic Park (1993)

```
movie_UBCF <- Recommender(movie_train, method = "UBCF", parameter = list(normalize = NULL))
predicted_UBCF <- predict(movie_UBCF, newdata = movie_test, n = 6)
movie_recommendation <- predicted_UBCF@itemLabels[predicted_UBCF@items[[1]]]
movie_recommendation <- as.data.frame(as.integer(movie_recommendation))
colnames(movie_recommendation) <- c("movieId")
movie_data <- movie_recommendation %>% inner_join(movies, by = "movieId") %>% select(Movie = "title")
knitr::kable(movie_data) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

Movie
Toy Story (1995)
Apollo 13 (1995)
True Lies (1994)
Fugitive, The (1993)
Jurassic Park (1993)
Speed (1994)

Normalization

Evaluation of Models In this Section, we will be creating an evaluation scheme to evaluate the Recommendation System's Popularity.

Evaluated at:

- $Top1$
- $Top3$
- $Top5$
- $Top10$
- $Top15$
- $Top20$

Determining the amount of time it takes our Recommedation System to serve up 'n' Recommedations.

```
model_eval<- evaluationScheme(as(films, "realRatingMatrix"),
                         method = "split",
                         train = 0.7,
                         given = 3,
                         goodRating = 5)
```

```r
preditor1 <- predict(movieIBCF, getData(model_eval, "known"), type = "ratings")
preditor2 <- predict(movie_UBCF, getData(model_eval, "known"), type = "ratings")
```

```r
final_eval <- rbind(
  IBCF = calcPredictionAccuracy(preditor1, getData(model_eval, "unknown")),
  UBCF = calcPredictionAccuracy(preditor2, getData(model_eval, "unknown")))

final_eval
```

```
##          RMSE      MSE      MAE
## IBCF 1.284471 1.649866 1.021675
## UBCF 2.909079 8.462738 2.729438
```

```r
rec_scheme <- evaluationScheme(as(films, "realRatingMatrix"),
                        method = "cross",
                        k = 4,
                        given = 3,
                        goodRating=5)
```

```r
solutions <- evaluate(rec_scheme,
                    method = "IBCF",
                    type = "topNList",
                    n = c(1, 3, 5, 10, 15, 20))
```

```
## IBCF run fold/sample [model time/prediction time]
##   1  [0.42sec/0.02sec]
##   2  [0.36sec/0.02sec]
##   3  [0.38sec/0.01sec]
##   4  [0.36sec/0.02sec]
```

```r
solutions2 <- evaluate(rec_scheme,
                    method = "UBCF",
                    type = "topNList",
                    n = c(1, 3, 5, 10, 15, 20))
```

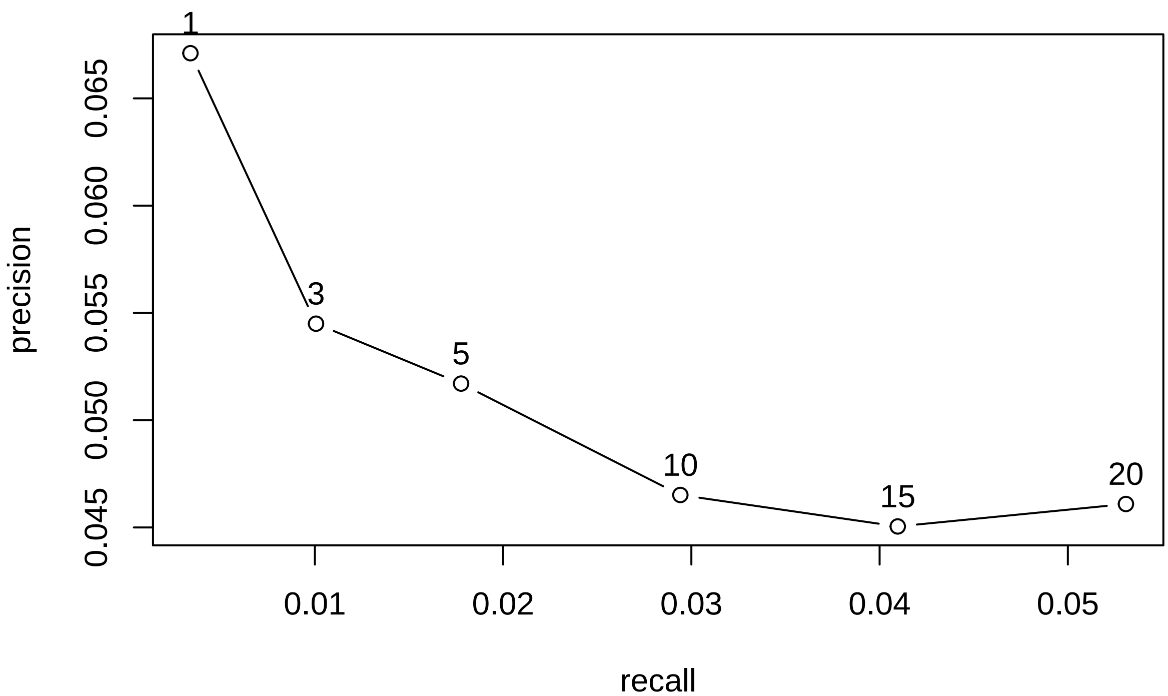
```
## UBCF run fold/sample [model time/prediction time]
##   1  [0sec/0.11sec]
##   2  [0sec/0.1sec]
##   3  [0sec/0.1sec]
##   4  [0sec/0.11sec]
```

```r
getConfusionMatrix(solutions)[[1]]
```

```
##            TP         FP       FN       TN  precision      recall         TPR
## 1  0.05208333  0.7916667 16.89583 415.2604 0.06172840 0.002496190 0.002496190
## 3  0.15625000  2.3750000 16.79167 413.6771 0.06172840 0.008208171 0.008208171
## 5  0.18750000  4.0312500 16.76042 412.0208 0.04444444 0.011047677 0.011047677
## 10 0.43750000  8.0000000 16.51042 408.0521 0.05185185 0.028774493 0.028774493
## 15 0.66666667 11.9583333 16.28125 404.0938 0.05267490 0.038592769 0.038592769
## 20 0.86458333 15.7708333 16.08333 400.2812 0.05149782 0.046787856 0.046787856
```

```
##            FPR
## 1  0.001904684
## 3  0.005719084
## 5  0.009714988
## 10 0.019283259
## 15 0.028802298
## 20 0.037975083
```

```r
plot(solutions, "prec/rec", annotate=TRUE)
```

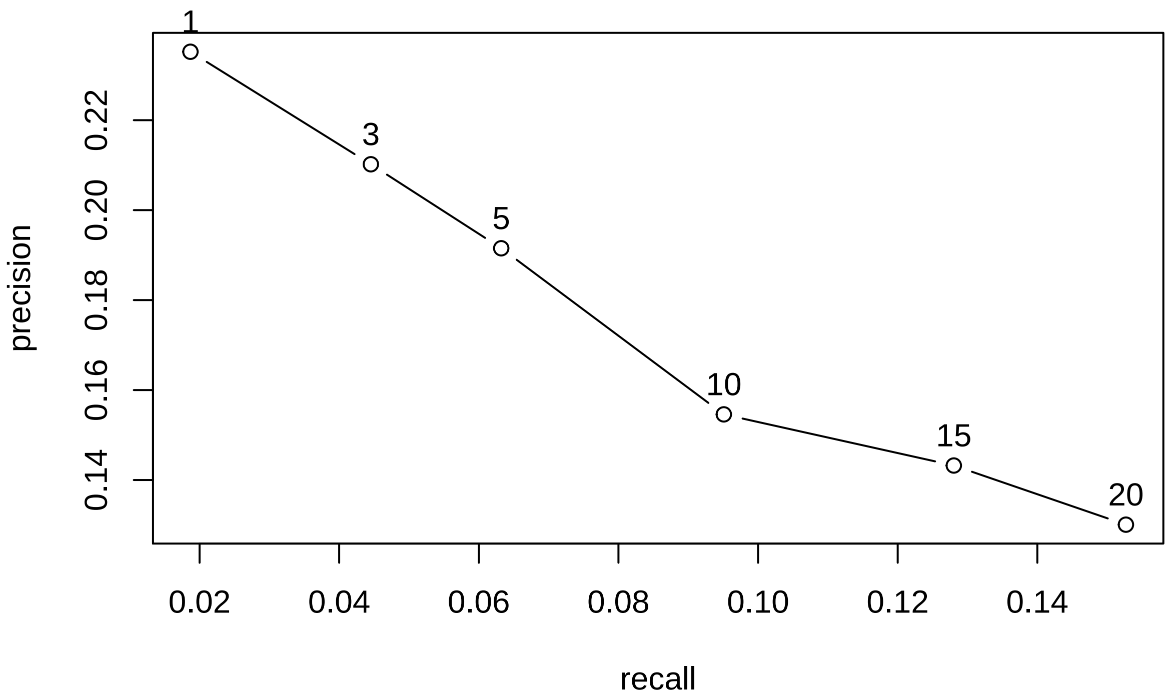



```r
getConfusionMatrix(solutions2)[[1]]
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.2500000  0.6666667 16.69792 415.3854 0.2727273 0.02548612 0.02548612
## 3  0.6562500  2.0937500 16.29167 413.9583 0.2386364 0.04890186 0.04890186
## 5  0.9583333  3.6250000 15.98958 412.4271 0.2090909 0.06568089 0.06568089
## 10 1.5937500  7.5729167 15.35417 408.4792 0.1738636 0.10115876 0.10115876
## 15 2.1875000 11.5625000 14.76042 404.4896 0.1590909 0.14613839 0.14613839
## 20 2.6354167 15.6979167 14.31250 400.3542 0.1437500 0.16910529 0.16910529
##            FPR
## 1  0.001594478
## 3  0.005008661
## 5  0.008675291
## 10 0.018131446
```

```
## 15 0.027709125
## 20 0.037628282
```

```r
plot(solutions2, "prec/rec", annotate=TRUE)
```



Conclusion

I am an avid IMDB and Netflix User and my peers often refer to me as a cinephile, or lover of motion picture cinema. Thererfore, when tasked with this project allows using the MovieLens DataSet to constructRecommender Systems, this was the first option for me. This offered me the opportunitry to explore Recommender Systems offered in the CRAN R package recommenderlab. I would like to further explore not only this package in future projects but this package as well.