Assignment 4: Recomendation Systems

Jaden Jefferson

5/7/2022

## Recomendation System Project

This project is practice to build a simple recommendation list focused on ITEM based collaborative recommendation system. The project can be found here, <https://data-flair.training/blogs/data-science-r-movie-recommendation/>

(2)The dataset can be found here, <https://drive.google.com/file/d/1Dn1BZD3YxgBQJSIjbfNnmCFlDW2jdQGD/view>

## Business Goal (1)

Building a recommendation system is the best way to keep a customer on a platform.The goal is to predict what USER 1 would like.

With collaborative recommendation we may understand USER 1 and USER 2 are very similar what they view in the past and like, so if USER 2 liked something then USER 1 likely would too. This can go vice versa.

## Required Libraries

Downloading the necessary libraries

library(recommenderlab)

## Loading required package: Matrix

## Loading required package: arules

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

## Loading required package: proxy

##   
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':  
##   
## as.matrix

## The following objects are masked from 'package:stats':  
##   
## as.dist, dist

## The following object is masked from 'package:base':  
##   
## as.matrix

## Loading required package: registry

## Registered S3 methods overwritten by 'registry':  
## method from   
## print.registry\_field proxy  
## print.registry\_entry proxy

library(ggplot2)   
library(data.table)  
library(reshape2)

##   
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':  
##   
## dcast, melt

## Importing Dataset (3)

setwd("C:\\Users\\Owner\\Documents\\Documents\\ANA515\\Assignment\\IMDB-Dataset")  
movie\_data <- read.csv("movies.csv",stringsAsFactors = FALSE)  
rating\_data <- read.csv("ratings.csv")  
str(movie\_data)

## 'data.frame': 10329 obs. of 3 variables:  
## $ movieId: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ title : chr "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting to Exhale (1995)" ...  
## $ genres : chr "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fantasy" "Comedy|Romance" "Comedy|Drama|Romance" ...

I used read.csv because it is faster. Had to change the directory using setwd() first to shorten the string in the unpackaging process.Made to data frames movie data and reading data. Changed movie\_data to string because of the titles.

## Dataset Description for Movies (4)

summary (movie\_data)

## movieId title genres   
## Min. : 1 Length:10329 Length:10329   
## 1st Qu.: 3240 Class :character Class :character   
## Median : 7088 Mode :character Mode :character   
## Mean : 31924   
## 3rd Qu.: 59900   
## Max. :149532

There are 10329 rows and 3 columns in the movie dataset. The columns in movie data are named movie id, title and genre.

## Dataset Description for Ratings (4)

summary (rating\_data)

## userId movieId rating timestamp   
## Min. : 1.0 Min. : 1 Min. :0.500 Min. :8.286e+08   
## 1st Qu.:192.0 1st Qu.: 1073 1st Qu.:3.000 1st Qu.:9.711e+08   
## Median :383.0 Median : 2497 Median :3.500 Median :1.115e+09   
## Mean :364.9 Mean : 13381 Mean :3.517 Mean :1.130e+09   
## 3rd Qu.:557.0 3rd Qu.: 5991 3rd Qu.:4.000 3rd Qu.:1.275e+09   
## Max. :668.0 Max. :149532 Max. :5.000 Max. :1.452e+09

There are 105339 rows and 4 columns in the rating dataset. The columns in ratings are userid, movieid, rating, and timestamp.

## Data Pre-processing/Cleaning (5)

These steps are necessary to allow the recommendation system to function. The data lacks missing values luckily.

1. It is necessary to convert genres into an understable format by creating a matrix of films and genres
2. Creating a search matrix to help perform a search of films by specifying the genre
3. Convert the matrix to a sparse matrix. This allows recommendations comprehend everything for our recommendation system

## Data Pre-Processing Code Step 1 (5)

movie\_genre <- as.data.frame(movie\_data$genres, stringsAsFactors=FALSE)  
library(data.table)  
movie\_genre2 <- as.data.frame(tstrsplit(movie\_genre[,1], '[|]',   
 type.convert=TRUE),   
 stringsAsFactors=FALSE)  
colnames(movie\_genre2) <- c(1:10)  
  
list\_genre <- c("Action", "Adventure", "Animation", "Children",   
 "Comedy", "Crime","Documentary", "Drama", "Fantasy",  
 "Film-Noir", "Horror", "Musical", "Mystery","Romance",  
 "Sci-Fi", "Thriller", "War", "Western")  
genre\_mat1 <- matrix(0,10330,18)  
genre\_mat1[1,] <- list\_genre  
colnames(genre\_mat1) <- list\_genre  
  
for (index in 1:nrow(movie\_genre2)) {  
 for (col in 1:ncol(movie\_genre2)) {  
 gen\_col = which(genre\_mat1[1,] == movie\_genre2[index,col])  
 genre\_mat1[index+1,gen\_col] <- 1  
}  
}  
genre\_mat2 <- as.data.frame(genre\_mat1[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list  
for (col in 1:ncol(genre\_mat2)) {  
 genre\_mat2[,col] <- as.integer(genre\_mat2[,col]) #convert from characters to integers  
}   
str(genre\_mat2)

## 'data.frame': 10329 obs. of 18 variables:  
## $ Action : int 0 0 0 0 0 1 0 0 1 1 ...  
## $ Adventure : int 1 1 0 0 0 0 0 1 0 1 ...  
## $ Animation : int 1 0 0 0 0 0 0 0 0 0 ...  
## $ Children : int 1 1 0 0 0 0 0 1 0 0 ...  
## $ Comedy : int 1 0 1 1 1 0 1 0 0 0 ...  
## $ Crime : int 0 0 0 0 0 1 0 0 0 0 ...  
## $ Documentary: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Drama : int 0 0 0 1 0 0 0 0 0 0 ...  
## $ Fantasy : int 1 1 0 0 0 0 0 0 0 0 ...  
## $ Film-Noir : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Horror : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Musical : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Mystery : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Romance : int 0 0 1 1 0 0 1 0 0 0 ...  
## $ Sci-Fi : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Thriller : int 0 0 0 0 0 1 0 0 0 1 ...  
## $ War : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Western : int 0 0 0 0 0 0 0 0 0 0 ...

## Data Pre-Processing Code Step 1: Explained (5)

Basically to convert to a matrix we needed to isolate the genres from the movie\_data dataframe. Placing that in its own separate dataframe allows us to then split the genres from each other and turn it into a list.

From there we make the matrix to map the genres from the list with the movie ids. For each it gets a yes or no (or a 0 or 1)

## Data Pre-Processing Step 2 (5)

SearchMatrix <- cbind(movie\_data[,1:2], genre\_mat2[])  
head(SearchMatrix)

## movieId title Action Adventure Animation  
## 1 1 Toy Story (1995) 0 1 1  
## 2 2 Jumanji (1995) 0 1 0  
## 3 3 Grumpier Old Men (1995) 0 0 0  
## 4 4 Waiting to Exhale (1995) 0 0 0  
## 5 5 Father of the Bride Part II (1995) 0 0 0  
## 6 6 Heat (1995) 1 0 0  
## Children Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical  
## 1 1 1 0 0 0 1 0 0 0  
## 2 1 0 0 0 0 1 0 0 0  
## 3 0 1 0 0 0 0 0 0 0  
## 4 0 1 0 0 1 0 0 0 0  
## 5 0 1 0 0 0 0 0 0 0  
## 6 0 0 1 0 0 0 0 0 0  
## Mystery Romance Sci-Fi Thriller War Western  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 1 0 0 0 0  
## 4 0 1 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 1 0 0

This allows us to bind our previous results to our movie data.

## Data Pre-Processing Step 3 (5)

ratingMatrix <- dcast(rating\_data, userId~movieId, value.var = "rating", na.rm=FALSE)  
ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove userIds  
  
#Convert rating matrix into a recommenderlab sparse matrix  
  
ratingMatrix <- as(ratingMatrix, "realRatingMatrix")  
ratingMatrix

## Data Pre-Processing Step 4 (5)

We start the recommendation model using the recommenderRegistry to pull from our rating matrix

recommendation\_model <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")  
names(recommendation\_model)  
  
lapply(recommendation\_model, "[[", "description")

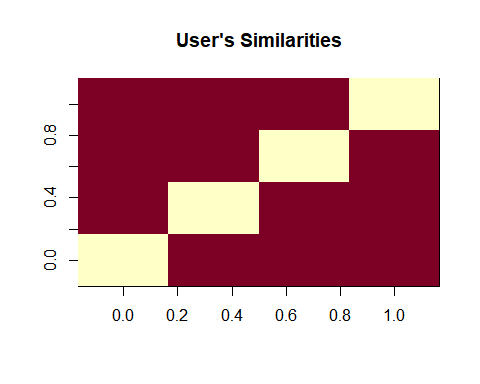
## [1] "HYBRID\_realRatingMatrix" "ALS\_realRatingMatrix"   
## [3] "ALS\_implicit\_realRatingMatrix" "IBCF\_realRatingMatrix"   
## [5] "LIBMF\_realRatingMatrix" "POPULAR\_realRatingMatrix"   
## [7] "RANDOM\_realRatingMatrix" "RERECOMMEND\_realRatingMatrix"   
## [9] "SVD\_realRatingMatrix" "SVDF\_realRatingMatrix"   
## [11] "UBCF\_realRatingMatrix"   
## $HYBRID\_realRatingMatrix  
## [1] "Hybrid recommender that aggegates several recommendation strategies using weighted averages."  
##   
## $ALS\_realRatingMatrix  
## [1] "Recommender for explicit ratings based on latent factors, calculated by alternating least squares algorithm."  
##   
## $ALS\_implicit\_realRatingMatrix  
## [1] "Recommender for implicit data based on latent factors, calculated by alternating least squares algorithm."  
##   
## $IBCF\_realRatingMatrix  
## [1] "Recommender based on item-based collaborative filtering."  
##   
## $LIBMF\_realRatingMatrix  
## [1] "Matrix factorization with LIBMF via package recosystem (https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html)."  
##   
## $POPULAR\_realRatingMatrix  
## [1] "Recommender based on item popularity."  
##   
## $RANDOM\_realRatingMatrix  
## [1] "Produce random recommendations (real ratings)."  
##   
## $RERECOMMEND\_realRatingMatrix  
## [1] "Re-recommends highly rated items (real ratings)."  
##   
## $SVD\_realRatingMatrix  
## [1] "Recommender based on SVD approximation with column-mean imputation."  
##   
## $SVDF\_realRatingMatrix  
## [1] "Recommender based on Funk SVD with gradient descend (https://sifter.org/~simon/journal/20061211.html)."  
##   
## $UBCF\_realRatingMatrix  
## [1] "Recommender based on user-based collaborative filtering."

## Data Exploration (8)

similarity\_mat <- similarity(ratingMatrix[1:4, ],  
 method = "cosine",  
 which = "users")  
as.matrix(similarity\_mat)

## 1 2 3 4  
## 1 0.0000000 0.9760860 0.9641723 0.9914398  
## 2 0.9760860 0.0000000 0.9925732 0.9374253  
## 3 0.9641723 0.9925732 0.0000000 0.9888968  
## 4 0.9914398 0.9374253 0.9888968 0.0000000

image(as.matrix(similarity\_mat), main = "User's Similarities")



This code allows us to see the similarities between users

## Data Exploration (8)

rating\_values <- as.vector(ratingMatrix@data)  
unique(rating\_values) # extracting unique ratings

## [1] 0.0 5.0 4.0 3.0 4.5 1.5 2.0 3.5 1.0 2.5 0.5

Table\_of\_Ratings <- table(rating\_values) # creating a count of movie ratings  
Table\_of\_Ratings

## rating\_values  
## 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5   
## 6791761 1198 3258 1567 7943 5484 21729 12237 28880 8187   
## 5   
## 14856

We need to create a rating values to make a table. This way we will be able to notice the breakdown of movie popularity. Notice that there is an absurd amount of 0 ratings. It will filtered out later that later.

## Visualizations: Movie Popularity (8)

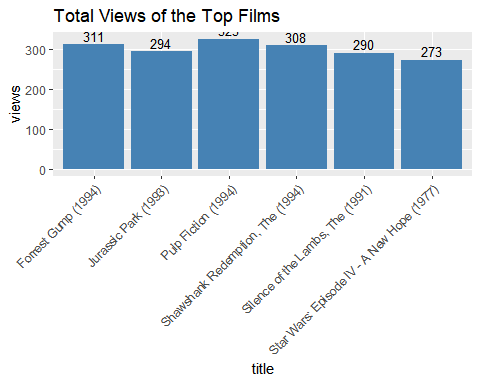
Similar to last chunk we can figure out the popularity of a movie in a table. we do that by creating a dataframe of movie views from the count of views for each movie in our rating matrix. Then create the a dataframe of views that will be sorted by decreasing order.

library(ggplot2)  
movie\_views <- colCounts(ratingMatrix) # count views for each movie  
table\_views <- data.frame(movie = names(movie\_views),  
 views = movie\_views) # create dataframe of views  
table\_views <- table\_views[order(table\_views$views,  
 decreasing = TRUE), ] # sort by number of views  
table\_views$title <- NA  
for (index in 1:10325){  
 table\_views[index,3] <- as.character(subset(movie\_data,  
 movie\_data$movieId == table\_views[index,1])$title)  
}  
table\_views[1:6,]

## movie views title  
## 296 296 325 Pulp Fiction (1994)  
## 356 356 311 Forrest Gump (1994)  
## 318 318 308 Shawshank Redemption, The (1994)  
## 480 480 294 Jurassic Park (1993)  
## 593 593 290 Silence of the Lambs, The (1991)  
## 260 260 273 Star Wars: Episode IV - A New Hope (1977)

## Visualizations: Movie Popularity Part 2 (8)

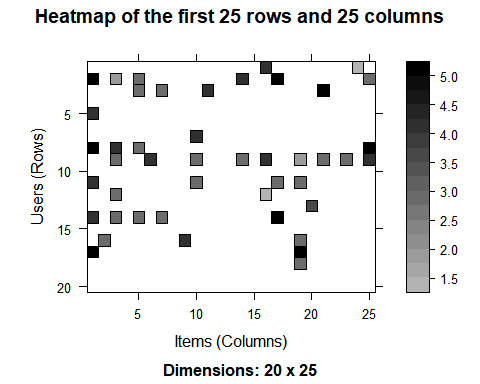
ggplot(table\_views[1:6, ], aes(x = title, y = views)) +  
 geom\_bar(stat="identity", fill = 'steelblue') +  
 geom\_text(aes(label=views), vjust=-0.3, size=3.5) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
  
 ggtitle("Total Views of the Top Films")



This creates a bar graph of the top popular views for each individual title. It is not really beneficial in my opinion. You can note position placements from the descending table just as fast after all.

##Visualizations: Ratings (8)

image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")



This is a heat map that shows the first 25 rows and 25 columns of movie ratings.Also no real usage. IT would have been interesting to potentially overlap Users ratings for the top movies. Maybe Pulp Fiction was the most viewed but maybe not the highest rated, etc.

##Data Preparation (5) For this part we want to find useful data, normalize, and binarize it.

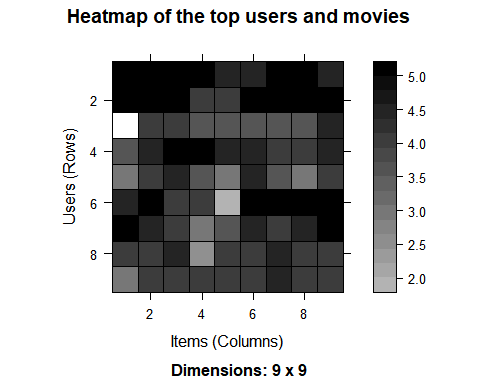
1. First is selecting 50 as minimum number of users who rated a film and minimum number that viewed a film. This will act like a filter to discard the least watched films or not relevant users. Then we can plot a heat map that visualizes the the top users and movies.
2. There can be a bias due to the variance of high and low ratings for all the watched films by individual users. Normalizing it will take it away and change the average value of ratings column to 0. Then we can plot a heat map that visualized the normalized ratings.
3. Binarizing the data makes our values either 1 or 0 making the system more efficient. 1 equals a rating above 3 and 0 otherwise.

##Data Perparation First Step (5)

movie\_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,  
 colCounts(ratingMatrix) > 50]  
movie\_ratings

## 420 x 447 rating matrix of class 'realRatingMatrix' with 38341 ratings.

minimum\_movies<- quantile(rowCounts(movie\_ratings), 0.98)  
minimum\_users <- quantile(colCounts(movie\_ratings), 0.98)  
image(movie\_ratings[rowCounts(movie\_ratings) > minimum\_movies,  
 colCounts(movie\_ratings) > minimum\_users],  
main = "Heatmap of the top users and movies")



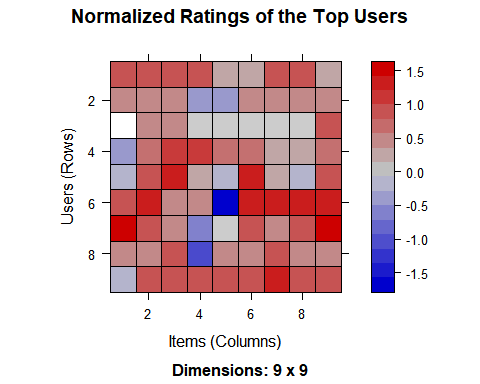
This provides us with people who actual watch and rate movies. It would be terrible to include people who aren’t really insightful. Also it will not include the least watched films. This is important if we want to make a top recommendation system with only ‘X’ amount of space available.

## Data Preparation Second Step (5)

normalized\_ratings <- normalize(movie\_ratings)  
sum(rowMeans(normalized\_ratings) > 0.00001)

## [1] 0

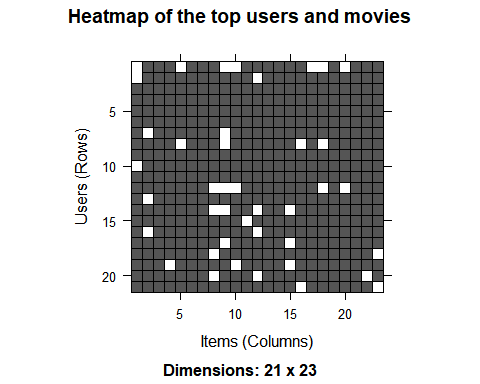
image(normalized\_ratings[rowCounts(normalized\_ratings) > minimum\_movies,  
 colCounts(normalized\_ratings) > minimum\_users],  
main = "Normalized Ratings of the Top Users")



This normalizes the movie ratings by using normalize() function.

## Data Preparation third Step (5)

binary\_minimum\_movies <- quantile(rowCounts(movie\_ratings), 0.95)  
binary\_minimum\_users <- quantile(colCounts(movie\_ratings), 0.95)  
#movies\_watched <- binarize(movie\_ratings, minRating = 1)  
  
good\_rated\_films <- binarize(movie\_ratings, minRating = 3)  
image(good\_rated\_films[rowCounts(movie\_ratings) > binary\_minimum\_movies,  
colCounts(movie\_ratings) > binary\_minimum\_users],  
main = "Heatmap of the top users and movies")



## Making the Collaborative Filter

For this projects the collaborative filter is looking for similarity of the items based on people’s rating. The filtering system will be split into a training set (80% of data) and a test set (20% of data)

## Collaborative Filtering System

sampled\_data<- sample(x = c(TRUE, FALSE),  
 size = nrow(movie\_ratings),  
 replace = TRUE,  
 prob = c(0.8, 0.2))  
training\_data <- movie\_ratings[sampled\_data, ]  
testing\_data <- movie\_ratings

## Recommendation System Explained

In the recommendation model there are parameters. K denotes the number of items for computing their similarities. In this project it is 30. The model method will be item based collaboration filtering (IBCF) and the data we are using is the trainding data.

##Recommendation System Code

recommendation\_system <- recommenderRegistry$get\_entries(dataType ="realRatingMatrix")  
recommendation\_system$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

recommen\_model <- Recommender(data = training\_data,  
 method = "IBCF",  
 parameter = list(k = 30))  
recommen\_model

## Recommender of type 'IBCF' for 'realRatingMatrix'   
## learned using 327 users.

class(recommen\_model)

## [1] "Recommender"  
## attr(,"package")  
## [1] "recommenderlab"

## Model

After retrieveing the model with getModel() function, the class and dimensions of our similarity matrix that is contained with model\_info. Then generate a heat map, that contain the top 20 items and visualize the similarity shared between them.

## Model Continued

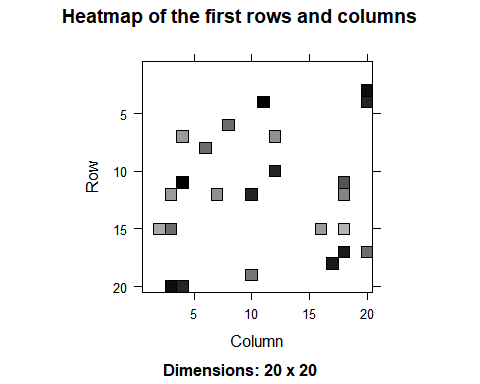
model\_info <- getModel(recommen\_model)  
class(model\_info$sim)

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

dim(model\_info$sim)

## [1] 447 447

top\_items <- 20  
image(model\_info$sim[1:top\_items, 1:top\_items],  
 main = "Heatmap of the first rows and columns")

 -This shows that some films are very similar to each other by their shade.

## Building the Top Recommendations System

In this step, a top\_recommendation variable that will be initialized to 10 fillms for each user. the predict function will identify similar items and will rank them. The rating is used as a weight in the ranking with each multiplied with related similarities and finally everything is added in the end.

## Building the Top Recommendations System Part 2

top\_recommendations <- 10 # the number of items to recommend to each user  
predicted\_recommendations <- predict(object = recommen\_model,  
 newdata = testing\_data,  
 n = top\_recommendations)  
predicted\_recommendations

## Recommendations as 'topNList' with n = 10 for 420 users.

## Building the Top Recommendations System Part 3

user1 <- predicted\_recommendations@items[[1]] # recommendation for the first user  
movies\_user1 <- predicted\_recommendations@itemLabels[user1]  
movies\_user2 <- movies\_user1  
for (index in 1:10){  
 movies\_user2[index] <- as.character(subset(movie\_data,  
 movie\_data$movieId == movies\_user1[index])$title)  
}  
movies\_user2

## [1] "Pan's Labyrinth (Laberinto del fauno, El) (2006)"  
## [2] "Juno (2007)"   
## [3] "WALLÂ·E (2008)"   
## [4] "No Country for Old Men (2007)"   
## [5] "300 (2007)"   
## [6] "Trainspotting (1996)"   
## [7] "Christmas Story, A (1983)"   
## [8] "Up (2009)"   
## [9] "It's a Wonderful Life (1946)"   
## [10] "Amadeus (1984)"

This code completes the idea that “User 1 = User 2” pretty much.

## Building the Top Recommendations System Part 4 (7)

recommendation\_matrix <- sapply(predicted\_recommendations@items,  
 function(x){ as.integer(colnames(movie\_ratings)[x]) }) # matrix with the recommendations for each user  
#dim(recc\_matrix)  
recommendation\_matrix[,1:4]

## [,1] [,2] [,3] [,4]  
## [1,] 48394 16 5 25  
## [2,] 56367 25 17 62  
## [3,] 60069 32 36 223  
## [4,] 55820 111 39 474  
## [5,] 51662 235 158 508  
## [6,] 778 265 160 529  
## [7,] 2804 337 235 541  
## [8,] 68954 541 317 590  
## [9,] 953 858 339 593  
## [10,] 1225 903 357 608

Here is the matrix of the Recommendation system. It shows the top 10 movie IDs for four users so looks like it is working right. It would be better if the movieID could be the name in this, however, when the actual system is implemented in the business the movieID would tack on to all info to populate it on the person’s recommendation list.