ADA LAB 7

K VIDYASAGAR

19BCD7183

```
# Building a recommendation engine that recommends movies to users.
# Item Based Collaborative Filter recommendation system
library(readr)

library(recommenderlab)
library(ggplot2)
library(data.table)
library(reshape2)
# Retrieve and display data
#setwd("/Users/arpitabhattacharya/Desktop/Warwick /Internship/Github uploads/Movie
recommendation - R/IMDB-Dataset") movie_data <- read_csv("IMDB-
Dataset/movies.csv",stringsAsFactors=FALSE) rating_data <- read_csv("IMDB-
Dataset/ratings.csv")

str(movie_data) #
Overview the summary
summary(movie_data)
head(movie_data)
summary(rating_data)
head(rating_data)
# Data pre-processing
# Creating a one-hot encoding to create a matrix that comprises of corresponding genres for
each of the films.
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)
# Creating a ‘search matrix’ - searching films by specifying the genre SearchMatrix <-
cbind(movie_data[,1:2], genre_mat2[]) head(SearchMatrix) 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
# Overview some important parameters for building recommendation systems for movies
recommendation_model <- recommenderRegistry$get_entries(dataType =
"realRatingMatrix") names(recommendation_model) lapply(recommendation_model, "[[",
"description")
# Implementing a single model in the R project – Item Based Collaborative Filtering
recommendation_model$IBCF_realRatingMatrix$parameters
# Collaborative Filtering involves suggesting movies to the users that are based on collecting
preferences from many other users.
# With the help of recommenderlab, we can compute similarities between users
similarity_mat <- similarity(ratingMatrix[1:4, ],
                             method = "cosine",
                             which = "users")
as.matrix(similarity_mat) image(as.matrix(similarity_mat),
main = "User's Similarities") # Portray the similarity that is
shared between the films movie_similarity <-
similarity(ratingMatrix[, 1:4], method =
                               "cosine", which = "items")
as.matrix(movie_similarity)
image(as.matrix(movie_similarity), main = "Movies similarity")
rating_values <- as.vector(ratingMatrix@data)
unique(rating_values) # extracting unique ratings
Table_of_Ratings <- table(rating_values) # creating a count of movie ratings
Table_of_Ratings
# Most viewed movies visualization 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,]
# Visualize a bar plot for the total number of views of the top films
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")
# Heatmap of Movie Ratings
# Visualize a heatmap of the movie ratings image(ratingMatrix[1:20, 1:25], axes = FALSE,
main = "Heatmap of the first 25 rows and 25 columns") # Data Preparation movie_ratings
<- ratingMatrix[rowCounts(ratingMatrix) > 50, colCounts(ratingMatrix) > 50]
movie_ratings
# describing matrix of relevant users 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") #
Visualizing the distribution of the average ratings per user
average_ratings <- rowMeans(movie_ratings)
qplot(average_ratings, fill=I("steelblue"), col=I("red")) +
  ggtitle("Distribution of the average rating per user")
# Data Normalization normalized_ratings <- normalize(movie_ratings)
sum(rowMeans(normalized_ratings) > 0.00001)
image(normalized_ratings[rowCounts(normalized_ratings) >
minimum_movies, colCounts(normalized_ratings) > minimum_users],
     main = "Normalized Ratings of the Top Users")
# Data Binarization
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")
# Collaborative Filtering System
```

```
# Splitting the dataset into 80% training set and 20% test set
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[!sampled_data, ]
# Building the Recommendation System
recommendation_system <- recommenderRegistry$get_entries(dataType
="realRatingMatrix") recommendation_system$IBCF_realRatingMatrix$parameters
recommen_model <- Recommender(data = training_data,
                              method = "IBCF",
                              parameter = list(k = 30))
recommen_model
class(recommen_model)
# Exploring the data science recommendation system model
model_info <- getModel(recommen_model)
class(model_info$sim) dim(model_info$sim) top_items <- 20
image(model_info$sim[1:top_items, 1:top_items], main =
"Heatmap of the first rows and columns")
# Visualize sum of rows and columns with the similarity of the objects above 0 sum_rows
<- rowSums(model_info$sim > 0) table(sum_rows) sum_cols <-
colSums(model_info$sim > 0) qplot(sum_cols, fill=I("steelblue"), col=I("red"))+
ggtitle("Distribution of the column count")
# the number of items to recommend to each user
top_recommendations <- 10 predicted_recommendations <-
predict(object = recommen_model,
                                  newdata = testing_data, n
                                  = top_recommendations)
predicted_recommendations
# recommendation for the first user user1 <-
predicted_recommendations@items[[1]] 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
# matrix with the recommendations for each user
recommendation_matrix <- sapply(predicted_recommendations@items,
function(x){ as.integer(colnames(movie_ratings)[x]) })
#dim(recc_matrix)
recommendation_matrix[,1:4]
```

```
# Distribution of the Number of Items for IBCF number_of_items <-
factor(table(recommendation_matrix)) chart_title <- "Distribution of the
Number of Items for IBCF" qplot(number_of_items, fill=I("steelblue"),
col=I("red")) + ggtitle(chart_title)

number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
number_of_items_top <- head(number_of_items_sorted, n = 4)
table_top <- data.frame(as.integer(names(number_of_items_top)),
number_of_items_top)
for(i in 1:4) {
  table_top[i,1] <- as.character(subset(movie_data,
                                movie_data$movieId == table_top[i,1])$title)
}

colnames(table_top) <- c("Movie Title", "No. of Items")
```

head(table_top) Outputs :

```
> # Build a recommendation engine that recommends movies to users.
> # Item Based Collaborative Filter recommendation system
> library(readr)
>
> library(recommenderlab)
> library(ggplot2)
> library(data.table)
> library(reshape2)
> # Retrieve and display data
> #setwd("/Users/arpitabhattacharya/Desktop/Warwick /Internship/Github uploads/Movie
recommendation - R/IMDB-Dataset")
> movie_data <- read_csv("IMDB-Dataset/movies.csv",stringsAsFactors=FALSE)
Error in read_csv("IMDB-Dataset/movies.csv", stringsAsFactors = FALSE) :
  unused argument (stringsAsFactors = FALSE)
>
> rating_data <- read_csv("IMDB-Dataset/ratings.csv")
Rows: 105339 Columns: 4
-- Column specification --------------------------------------
Delimiter: "," dbl (4): userId, movieId, rating,
timestamp

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
> str(movie_data)
Error in str(movie_data) : object 'movie_data' not found
```

```
> # Overview the summary
> summary(movie_data)
Error in h(simpleError(msg, call)) : error in evaluating the argument 'object' in selecting a
  method for function 'summary': object
'movie_data' not found
> head(movie_data)
Error in h(simpleError(msg, call)) : error in evaluating the argument 'x' in selecting a
  method for function 'head': object
'movie_data' not found
> summary(rating_data)
     userId         movieId          rating
Min.   :  1.0   Min.   :     1   Min.   :0.500
1st Qu.:192.0   1st Qu.:  1073   1st Qu.:3.000
Median :383.0   Median :  2497   Median :3.500
Mean   :364.9   Mean   : 13381   Mean   :3.517
3rd Qu.:557.0   3rd Qu.:  5991   3rd Qu.:4.000
Max.   :668.0   Max.   :149532   Max.   :5.000 timestamp
Min.   :8.286e+08
1st Qu.:9.711e+08
Median :1.115e+09
Mean   :1.130e+09
3rd Qu.:1.275e+09 Max.
:1.452e+09
> head(rating_data)
# A tibble: 6 x 4 userId movieId
  rating  timestamp
   <dbl>   <dbl>  <dbl>      <dbl>
1      1      16    4   1217897793
2      1      24    1.5 1217895807
3      1      32    4   1217896246
4      1      47    4   1217896556
5      1      50    4   1217896523
6      1     110    4   1217896150
> # Data pre-processing
> # Creating a one-hot encoding to create a matrix that comprises of corresponding genres for
each of the films.
> movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors=FALSE)
Error in as.data.frame(movie_data$genres, stringsAsFactors = FALSE) :
  object 'movie_data' not found
> library(data.table)
> movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]',
+                                         type.convert=TRUE),
+                               stringsAsFactors=FALSE)
```

```
Error in strsplit(as.character(x), ...) : object 'movie_genre' not found
> colnames(movie_genre2) <- c(1:10)
Error in colnames(movie_genre2) <- c(1:10) :
  object 'movie_genre2' not found
> 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
+   }
+ }
Error in nrow(movie_genre2) : object 'movie_genre2' not found
> 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 0 0 0 0 0 ...
$ Adventure  : int  0 0 0 0 0 0 0 0 0 0 ...
$ Animation  : int  0 0 0 0 0 0 0 0 0 0 ...
$ Children   : int  0 0 0 0 0 0 0 0 0 0 ...
$ Comedy     : int  0 0 0 0 0 0 0 0 0 0 ...
$ Crime      : int  0 0 0 0 0 0 0 0 0 0 ...
$ Documentary: int  0 0 0 0 0 0 0 0 0 0 ...
$ Drama      : int  0 0 0 0 0 0 0 0 0 0 ... $
Fantasy    : int  0 0 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 0 0 0 0 0 0 0 0 ...
$ Sci-Fi     : int  0 0 0 0 0 0 0 0 0 0 ... $
Thriller   : int  0 0 0 0 0 0 0 0 0 0 ...
$ War        : int  0 0 0 0 0 0 0 0 0 0 ...
$ Western    : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
> # Creating a ‘search matrix’ - searching films by specifying the genre
> SearchMatrix <- cbind(movie_data[,1:2], genre_mat2[]) Error in
cbind(movie_data[, 1:2], genre_mat2[]) :
  object 'movie_data' not found
> head(SearchMatrix)
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'x' in selecting a method for function 'head': object
'SearchMatrix' not found
> 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
668 x 10325 rating matrix of class ‘realRatingMatrix’ with 105339 ratings.
> # Overview some important parameters for building recommendation systems for movies
> recommendation_model <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
> names(recommendation_model)
[1] "HYBRID_realRatingMatrix"
[2] "ALS_realRatingMatrix"
[3] "ALS_implicit_realRatingMatrix"
[4] "IBCF_realRatingMatrix"
[5] "LIBMF_realRatingMatrix"
[6] "POPULAR_realRatingMatrix"
[7] "RANDOM_realRatingMatrix"
[8] "RERECOMMEND_realRatingMatrix"
[9] "SVD_realRatingMatrix"
[10]     "SVDF_realRatingMatrix"[11] "UBCF_realRatingMatrix"
> lapply(recommendation_model, "[[", "description")
$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."

> # Implementing a single model in the R project – Item Based Collaborative Filtering
> recommendation_model$IBCF_realRatingMatrix$parameters
$k
[1] 30

$method
[1] "Cosine"

$normalize
[1] "center"
$normalize_sim_matrix
[1] FALSE

$alpha
[1] 0.5

$na_as_zero
[1] FALSE

> # Collaborative Filtering involves suggesting movies to the users that are based on collecting preferences from many other users.

```
> # With the help of recommenderlab, we can compute similarities between users
> 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")
> # Portray the similarity that is shared between the films
> movie_similarity <- similarity(ratingMatrix[, 1:4], method =
+                                  "cosine", which = "items")
> as.matrix(movie_similarity)
          1         2         3         4
1 0.0000000 0.9669732 0.9559341 0.9101276
2 0.9669732 0.0000000 0.9658757 0.9412416
3 0.9559341 0.9658757 0.0000000 0.9864877
4 0.9101276 0.9412416 0.9864877 0.0000000
> image(as.matrix(movie_similarity), main = "Movies similarity")
> 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
6791761   1198   3258   1567   7943   5484  21729
   3.5      4    4.5      5
 12237  28880   8187  14856
> # Most viewed movies visualization
> 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)
+ }
```

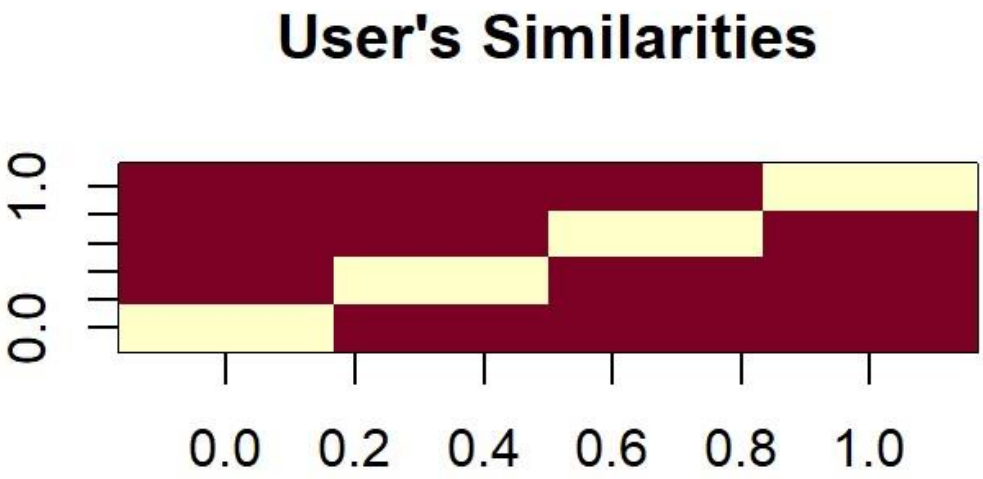
```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'x' in selecting a method for function 'subset': object
'movie_data' not found
> table_views[1:6,]
    movie views title
296   296   325    NA
356   356   311    NA
318   318   308    NA
480   480   294    NA
593   593   290    NA
260   260   273    NA
> # Visualize a bar plot for the total number of views of the top films
> 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")
> # Heatmap of Movie Ratings
> # Visualize a heatmap of the movie ratings
> image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25
columns")
> # Data Preparation
> movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
+                               colCounts(ratingMatrix) > 50]
> movie_ratings
420 x 447 rating matrix of class ‘realRatingMatrix’ with 38341 ratings.
> # describing matrix of relevant users
> 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")
> # Visualizing the distribution of the average ratings per user
> average_ratings <- rowMeans(movie_ratings)
> qplot(average_ratings, fill=I("steelblue"), col=I("red")) +
+   ggtitle("Distribution of the average rating per user")
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
> # Data Normalization
> normalized_ratings <- normalize(movie_ratings)
> sum(rowMeans(normalized_ratings) > 0.00001)
[1] 0
> image(normalized_ratings[rowCounts(normalized_ratings) > minimum_movies,
```

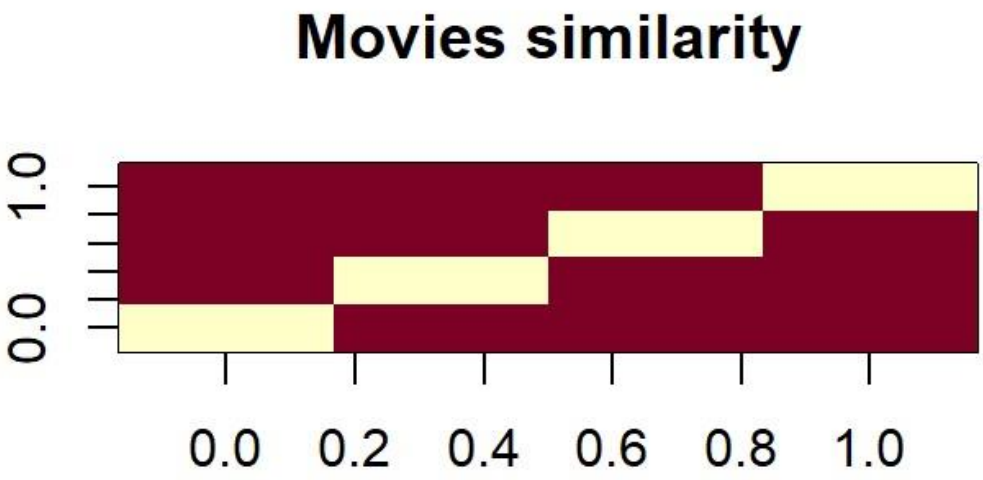
```
+                     colCounts(normalized_ratings) > minimum_users],
+      main = "Normalized Ratings of the Top Users")
> # Data Binarization
> binary_minimum_movies <- quantile(rowCounts(movie_ratings), 0.95)
> binary_minimum_users <- quantile(colCounts(movie_ratings), 0.95)
> #movies_watched <- 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")
> # Collaborative Filtering System
> # Splitting the dataset into 80% training set and 20% test set
> 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[!sampled_data, ]
> # Building the Recommendation System
> 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
```

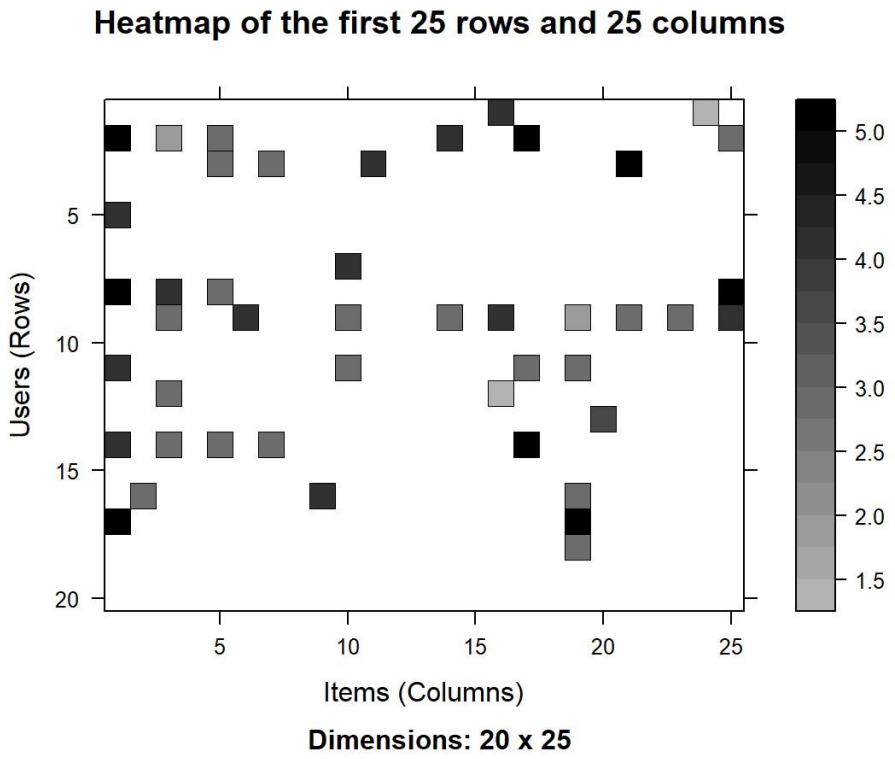
```
Recommender of type ‘IBCF’ for ‘realRatingMatrix’
learned using 357 users.
> class(recommen_model)
[1] "Recommender"
attr(,"package")
[1] "recommenderlab"
> # Exploring the data science recommendation system model
> model_info <- getModel(recommen_model)
> class(model_info$sim)
[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"
> dim(model_info$sim)
[1] 447 447
> top_items <- 20
> image(model_info$sim[1:top_items, 1:top_items],
+       main = "Heatmap of the first rows and columns")
> # Visualize sum of rows and columns with the similarity of the objects above 0
> sum_rows <- rowSums(model_info$sim > 0)
> table(sum_rows)
sum_rows 30
447
> sum_cols <- colSums(model_info$sim > 0)
> qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> # the number of items to recommend to each user
> top_recommendations <- 10
> predicted_recommendations <- predict(object = recommen_model,
+                                      newdata = testing_data,
+                                      n = top_recommendations)
> predicted_recommendations
Recommendations as ‘topNList’ with n = 10 for 63 users.
> # recommendation for the first user
> user1 <- predicted_recommendations@items[[1]]
> movies_user1 <- predicted_recommendations@itemLabels[user1]
> movies_user2 <- movies_user1
> for (index in 1:10){
+   movies_user2[index] <- as.character(subset(movie_data,
+                                              movie_data$movieId == movies_user1[index])$title)
+ }
Error in h(simpleError(msg, call)) : error in evaluating the argument 'x' in selecting a
  method for function 'subset': object
'movie_data' not found
```

```
> movies_user2
[1] "39"   "288"  "339"  "357"  "440"  "474"  "708"  "1080"
[9] "1259" "1265"
> # matrix with the recommendations for each user
> recommendation_matrix <- sapply(predicted_recommendations@items,
+                                 function(x){ as.integer(colnames(movie_ratings)[x]) })
> #dim(recc_matrix)
> recommendation_matrix[,1:4]
     [,1] [,2] [,3]  [,4]
[1,]   39    6    7   596
[2,]  288   21   21  1183
[3,]  339   62  163  2078
[4,]  357  223  165  2302
[5,]  440  296  349  5669 [6,]
474  474  551   594
[7,]  708  529  588  3578
[8,] 1080  541  592 60069
[9,] 1259  593  648  5995
[10,] 1265  908  661  3255
> # Distribution of the Number of Items for IBCF
> number_of_items <- factor(table(recommendation_matrix))
> chart_title <- "Distribution of the Number of Items for IBCF"
> qplot(number_of_items, fill=I("steelblue"), col=I("red")) + ggtitle(chart_title)
>
> number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
> number_of_items_top <- head(number_of_items_sorted, n = 4)
> table_top <- data.frame(as.integer(names(number_of_items_top)),
+                         number_of_items_top)
> for(i in 1:4) {
+   table_top[i,1] <- as.character(subset(movie_data,
+                                         movie_data$movieId == table_top[i,1])$title)
+ }
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'x' in selecting a method for function 'subset': object
'movie_data' not found
>
> colnames(table_top) <- c("Movie Title", "No. of Items")
> head(table_top)
    Movie Title No. of Items
25           25           8
923         923           8
21           21           7
260         260           7
```

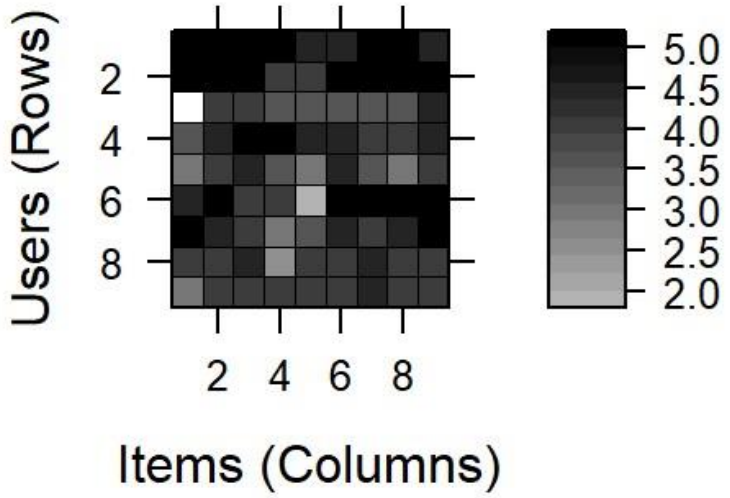
>


User's Similarities
1.0
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Movies similarity
0.0
1.0
0.0
0.2
0.4
0.6
0.8
1.0

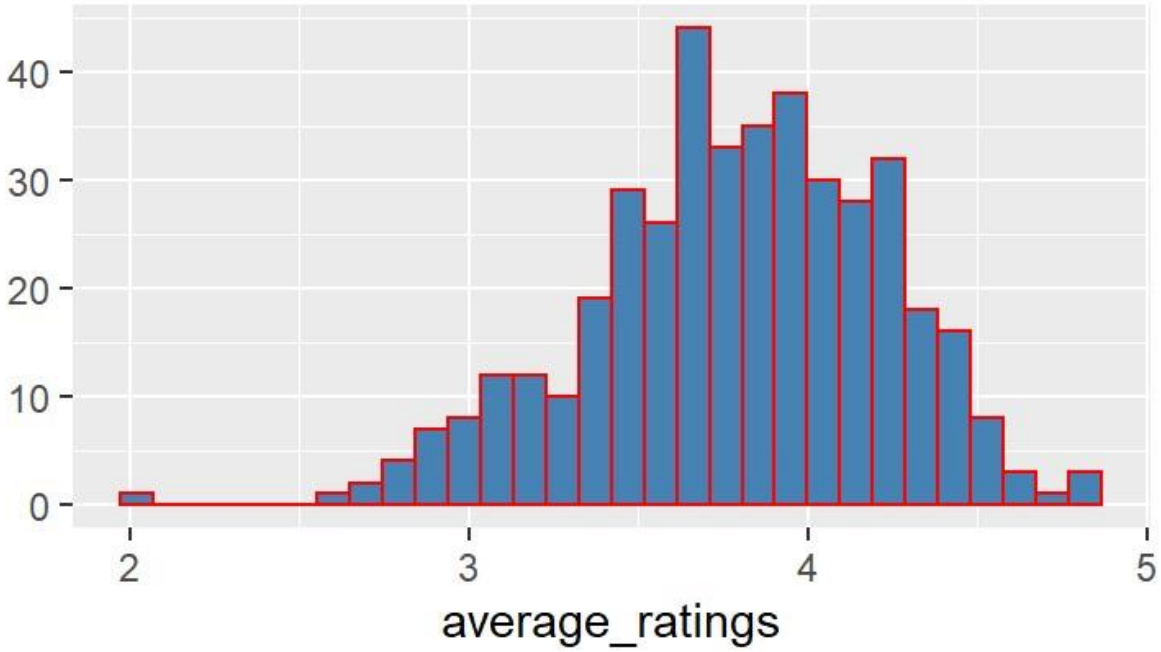

Heatmap of the first 25 rows and 25 columns
Users (Rows)
5
10
15
20
5
10
15
20
25
Items (Columns)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
Dimensions: 20 x 25

Heatmap of the top users and movies

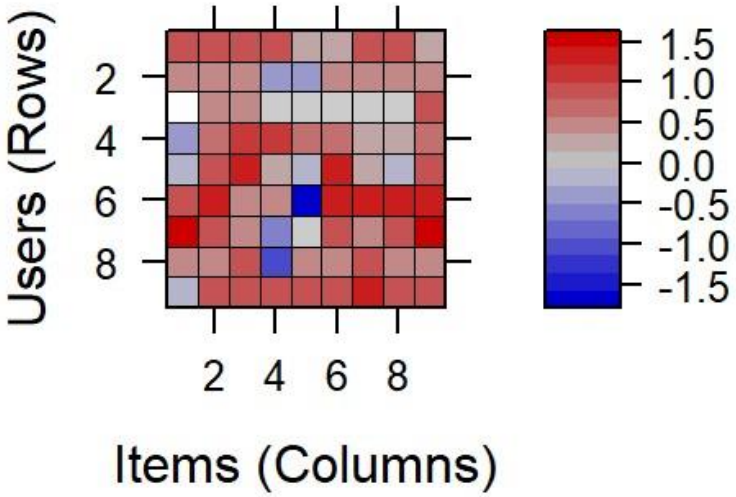


Dimensions: 9 x 9

Distribution of the average rating per user

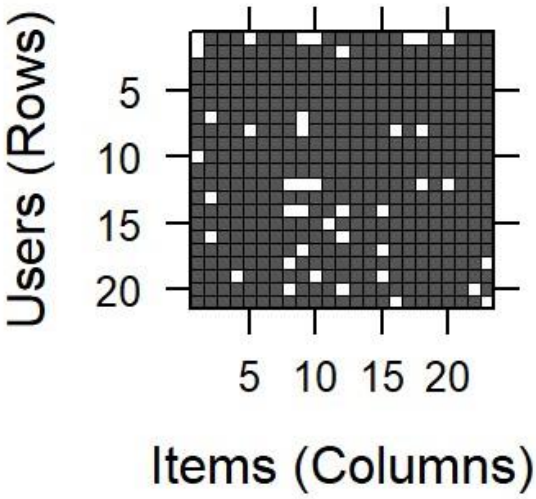


Normalized Ratings of the Top Users



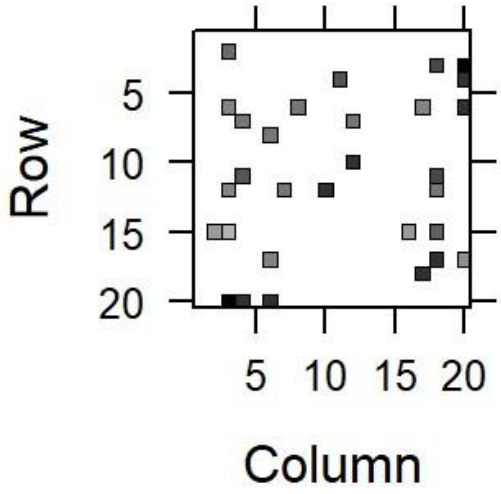
Dimensions: 9 x 9

Heatmap of the top users and movies



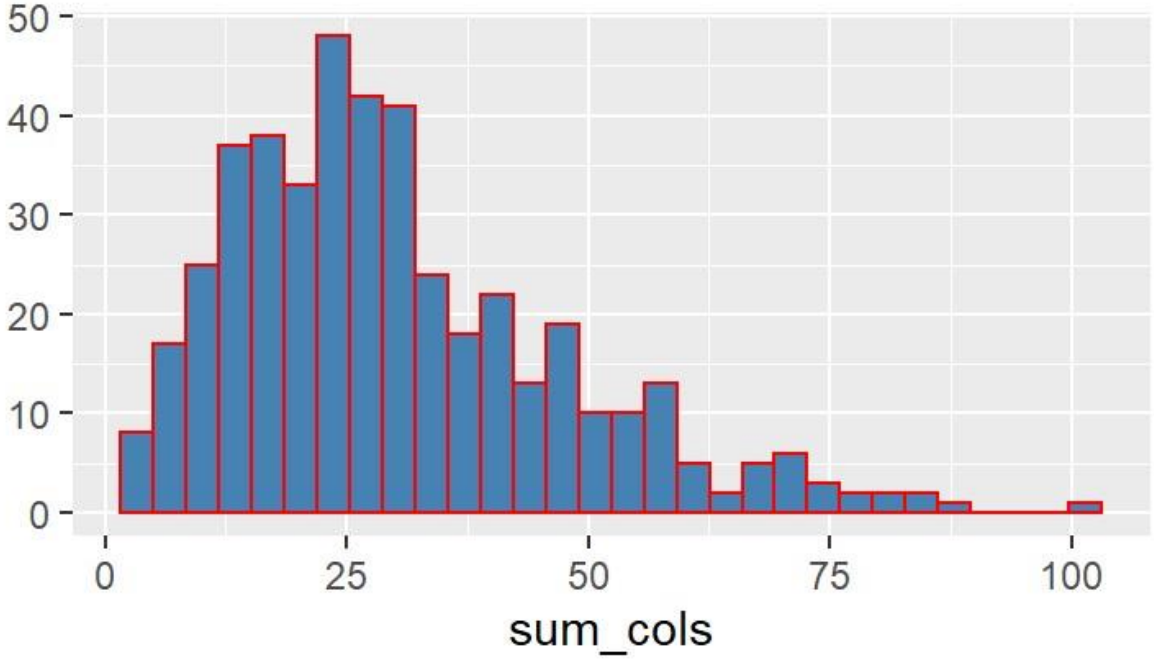
Dimensions: 21 x 23

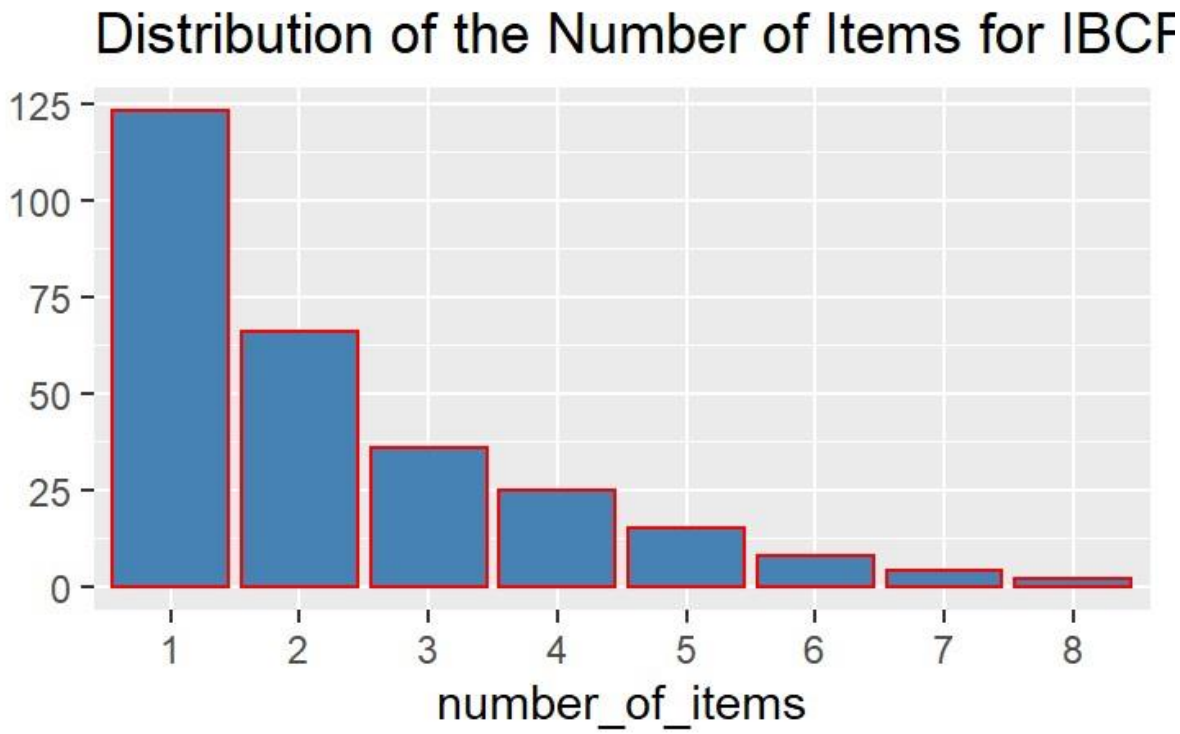
Heatmap of the first rows and columns



Dimensions: 20 x 20

Distribution of the column count




Distribution of the Number of Items for IBCF
125
100
75
50
25
0
1
2
3
4
5
6
7
8
number_of_items