Rmarkdown Lab 1

## R Markdown

# Load the packages —-

#install.packages("pacman")  
pacman::p\_load(pacman, dplyr, GGally, ggplot2, ggthemes, data.table, reshape2, recommenderlab, arules, arulesViz,  
 ggvis, httr, lubridate, plotly, rio, rmarkdown, shiny,   
 stringr, tidyr)  
  
if(! "arules" %in% installed.packages()) install.packages("arules", depend = TRUE)  
library(arules)  
  
if(! "arulesViz" %in% installed.packages()) install.packages("arulesViz", depend = TRUE)  
library(arulesViz)  
  
if(! "recommenderlab" %in% installed.packages()) install.packages("recommenderlab", depend = TRUE)  
library("recommenderlab")  
  
if(! "reshape2" %in% installed.packages()) install.packages("reshape2", depend = TRUE)  
library("reshape2")  
  
if(! "data.table" %in% installed.packages()) install.packages("data.table", depend = TRUE)  
library(data.table)  
library(rmarkdown)

# Load the datasets from CSV file —-

movies\_d<-read.csv("movies.csv",header=TRUE)  
ratings\_d<-read.csv("ratings.csv",header=TRUE)

# check the structure of csv files —-

str(movies\_d)

## 'data.frame': 9742 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" ...

str(ratings\_d)

## 'data.frame': 100836 obs. of 4 variables:  
## $ userId : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ movieId : int 1 3 6 47 50 70 101 110 151 157 ...  
## $ rating : num 4 4 4 5 5 3 5 4 5 5 ...  
## $ timestamp: int 964982703 964981247 964982224 964983815 964982931 964982400 964980868 964982176 964984041 964984100 ...

# Check the data sets dimension and summary of statistic —-

dim(movies\_d)

## [1] 9742 3

summary(movies\_d)

## movieId title genres   
## Min. : 1 Length:9742 Length:9742   
## 1st Qu.: 3248 Class :character Class :character   
## Median : 7300 Mode :character Mode :character   
## Mean : 42200   
## 3rd Qu.: 76232   
## Max. :193609

dim(ratings\_d)

## [1] 100836 4

summary(ratings\_d)

## userId movieId rating timestamp   
## Min. : 1.0 Min. : 1 Min. :0.500 Min. :8.281e+08   
## 1st Qu.:177.0 1st Qu.: 1199 1st Qu.:3.000 1st Qu.:1.019e+09   
## Median :325.0 Median : 2991 Median :3.500 Median :1.186e+09   
## Mean :326.1 Mean : 19435 Mean :3.502 Mean :1.206e+09   
## 3rd Qu.:477.0 3rd Qu.: 8122 3rd Qu.:4.000 3rd Qu.:1.436e+09   
## Max. :610.0 Max. :193609 Max. :5.000 Max. :1.538e+09

# check first 10 rows of data sets —-

head(movies\_d, 10)

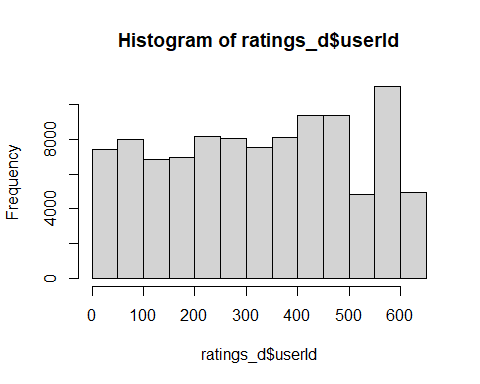
## movieId title  
## 1 1 Toy Story (1995)  
## 2 2 Jumanji (1995)  
## 3 3 Grumpier Old Men (1995)  
## 4 4 Waiting to Exhale (1995)  
## 5 5 Father of the Bride Part II (1995)  
## 6 6 Heat (1995)  
## 7 7 Sabrina (1995)  
## 8 8 Tom and Huck (1995)  
## 9 9 Sudden Death (1995)  
## 10 10 GoldenEye (1995)  
## genres  
## 1 Adventure|Animation|Children|Comedy|Fantasy  
## 2 Adventure|Children|Fantasy  
## 3 Comedy|Romance  
## 4 Comedy|Drama|Romance  
## 5 Comedy  
## 6 Action|Crime|Thriller  
## 7 Comedy|Romance  
## 8 Adventure|Children  
## 9 Action  
## 10 Action|Adventure|Thriller

head(ratings\_d, 10)

## userId movieId rating timestamp  
## 1 1 1 4 964982703  
## 2 1 3 4 964981247  
## 3 1 6 4 964982224  
## 4 1 47 5 964983815  
## 5 1 50 5 964982931  
## 6 1 70 3 964982400  
## 7 1 101 5 964980868  
## 8 1 110 4 964982176  
## 9 1 151 5 964984041  
## 10 1 157 5 964984100

# check the behavior pattern of userid, moveid and ratings —-

hist(ratings\_d$userId) # check the high-level pattern of userid



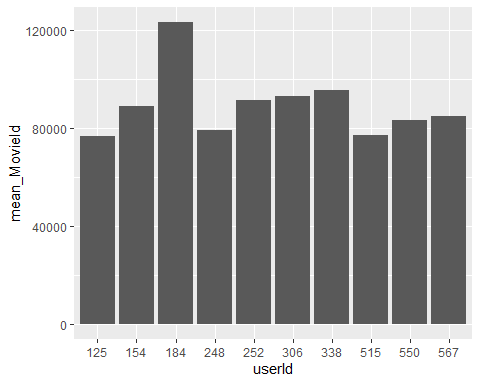
ratings\_histogram\_UID <- ratings\_d %>%  
 mutate(userId = factor(userId)) %>%  
 group\_by(userId) %>% # group by userId  
 summarize(mean\_MovieId = round(mean(movieId), 2)) %>%  
 arrange(desc(mean\_MovieId))

## `summarise()` ungrouping output (override with `.groups` argument)

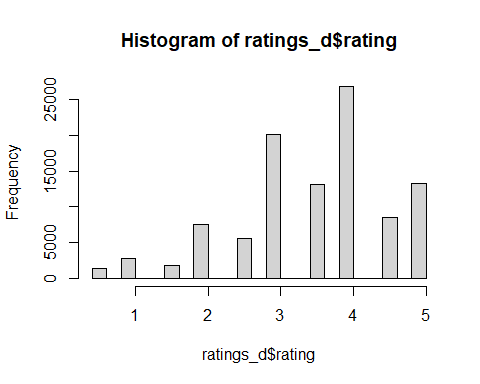
ratings\_histogram\_UID # check values

## # A tibble: 610 x 2  
## userId mean\_MovieId  
## <fct> <dbl>  
## 1 184 123329.  
## 2 338 95710.  
## 3 306 93239.  
## 4 252 91263.  
## 5 154 89048.  
## 6 567 84713.  
## 7 550 83108.  
## 8 248 79146.  
## 9 515 77186.  
## 10 125 76717.  
## # ... with 600 more rows

ggplot(ratings\_histogram\_UID[1:10,], aes(x = userId, y = mean\_MovieId)) +  
 geom\_bar(stat = "identity") # visualization



hist(ratings\_d$rating) # check the high-level pattern of user ratings



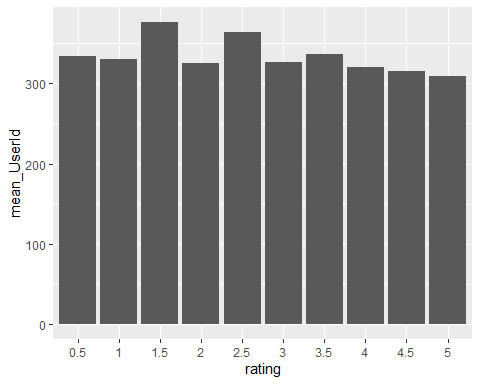
ratings\_histogram\_Rat <- ratings\_d %>%  
 mutate(rating = factor(rating)) %>%  
 group\_by(rating) %>% # group by rating  
 summarize(mean\_UserId = round(mean(userId), 2)) %>%  
 arrange(desc(mean\_UserId))

## `summarise()` ungrouping output (override with `.groups` argument)

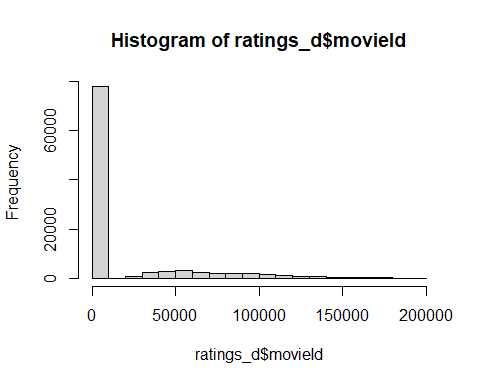
ratings\_histogram\_Rat # check the values

## # A tibble: 10 x 2  
## rating mean\_UserId  
## <fct> <dbl>  
## 1 1.5 377.  
## 2 2.5 364.  
## 3 3.5 337.  
## 4 0.5 335.  
## 5 1 330.  
## 6 3 327.  
## 7 2 325.  
## 8 4 320.  
## 9 4.5 315.  
## 10 5 309.

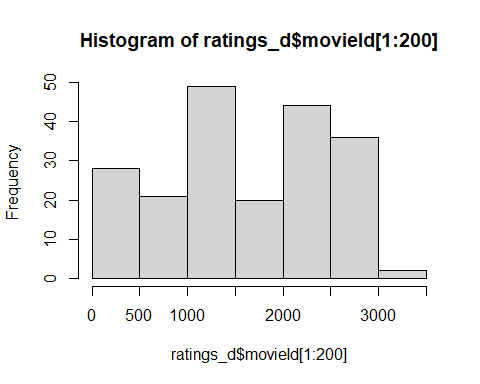
ggplot(ratings\_histogram\_Rat, aes(x = rating, y = mean\_UserId)) +  
 geom\_bar(stat = "identity") # visualization



hist(ratings\_d$movieId) # check the high-level pattern of movieId



hist(ratings\_d$movieId[1:200]) # checking 1-200 movies id



ratings\_histogram\_MID <- ratings\_d %>%  
 mutate(rating = factor(rating)) %>%  
 group\_by(rating) %>% # group by rating  
 summarize(mean\_movieId = round(mean(movieId), 2)) %>%  
 arrange(desc(mean\_movieId))

## `summarise()` ungrouping output (override with `.groups` argument)

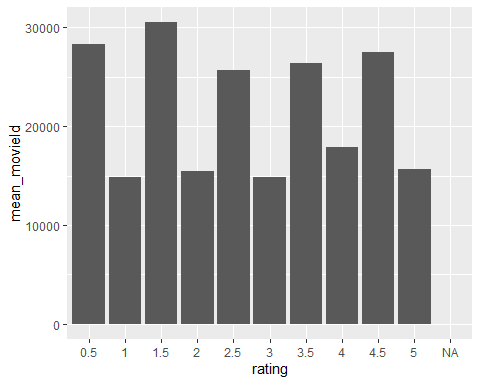
ratings\_histogram\_MID # check values

## # A tibble: 10 x 2  
## rating mean\_movieId  
## <fct> <dbl>  
## 1 1.5 30537.  
## 2 0.5 28268.  
## 3 4.5 27505.  
## 4 3.5 26388.  
## 5 2.5 25687.  
## 6 4 17858.  
## 7 5 15639.  
## 8 2 15497.  
## 9 3 14853.  
## 10 1 14820.

ggplot(ratings\_histogram\_MID[1:50,], aes(x = rating, y = mean\_movieId)) +  
 geom\_bar(stat = "identity") # visualization

## Warning: The `i` argument of ``[.tbl\_df`()` must lie in [0, rows] if positive, as of tibble 3.0.0.  
## Use `NA` as row index to obtain a row full of `NA` values.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

## Warning: Removed 40 rows containing missing values (position\_stack).



# group by genres to find missing genres —-

movies\_GenresGroup <- movies\_d %>%  
 group\_by(genres) %>% # group by genres  
 summarise(unique\_genres = n\_distinct(genres)) %>% # count by genres  
 arrange(desc(unique\_genres)) # arrange by genres

## `summarise()` ungrouping output (override with `.groups` argument)

head(movies\_GenresGroup) # check few top rows of dataset

## # A tibble: 6 x 2  
## genres unique\_genres  
## <chr> <int>  
## 1 (no genres listed) 1  
## 2 Action 1  
## 3 Action|Adventure 1  
## 4 Action|Adventure|Animation 1  
## 5 Action|Adventure|Animation|Children 1  
## 6 Action|Adventure|Animation|Children|Comedy 1

tail(movies\_GenresGroup) # check few bottom rows of dataset

## # A tibble: 6 x 2  
## genres unique\_genres  
## <chr> <int>  
## 1 Sci-Fi|IMAX 1  
## 2 Sci-Fi|Thriller 1  
## 3 Sci-Fi|Thriller|IMAX 1  
## 4 Thriller 1  
## 5 War 1  
## 6 Western 1

# remove missing genres from the movies datasets —-

movies\_d <- movies\_d %>% filter(genres != "(no genres listed)")  
dim(movies\_d) # check the row dimension

## [1] 9708 3

# check NA’s in moves datasets of specific field —-

na\_movies <- movies\_d %>%  
 filter(is.na(movieId)|is.na(title)|is.na(genres) | genres == "NA" )  
na\_movies # check the row - movies\_d <- na.omit(movies\_d) # becuase no NA's thats why we do not use this line

## [1] movieId title genres   
## <0 rows> (or 0-length row.names)

# Prepare specific list of genres (one-hot encoding) —-

genresList <- as.data.frame(movies\_d$genres, stringsAsFactors=FALSE)  
genresListColumn <- as.data.frame(tstrsplit(genresList[,1], '[|]',   
 type.convert=TRUE),   
 stringsAsFactors=FALSE)  
  
dim(genresListColumn) # check dimension of column

## [1] 9708 10

colnames(genresListColumn)

## [1] "c..Adventure....Adventure....Comedy....Comedy....Comedy....Action..."   
## [2] "c..Animation....Children....Romance....Drama...NA...Crime....Romance..."  
## [3] "c..Children....Fantasy...NA...Romance...NA...Thriller...NA..NA.."   
## [4] "c..Comedy...NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA.."   
## [5] "c..Fantasy...NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA.."   
## [6] "c.NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA.."   
## [7] "c.NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA...1"   
## [8] "c.NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA...2"   
## [9] "c.NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA...3"   
## [10] "c.NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA..NA...4"

colnames(genresListColumn) <- c(1:10) # based on no of dim values, assign values 1 to 10  
colnames(genresListColumn)

## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"

unique(genresListColumn[,1]) # get the unique list of genres

## [1] "Adventure" "Comedy" "Action" "Drama" "Crime"   
## [6] "Children" "Mystery" "Animation" "Documentary" "Thriller"   
## [11] "Horror" "Fantasy" "Western" "Film-Noir" "Romance"   
## [16] "Sci-Fi" "Musical" "War"

length(unique(genresListColumn[,1])) # give the no of genres

## [1] 18

genre\_Picklist <- c("Adventure", "Animation", "Children", "Comedy", "Fantasy", "Romance",  
 "Drama", "Thriller", "Action","Crime","Documentary",  
 "Film-Noir", "Horror", "Musical", "Mystery",  
 "Sci-Fi", "War", "Western") # total 18 genres  
genre\_Picklist # check list

## [1] "Adventure" "Animation" "Children" "Comedy" "Fantasy"   
## [6] "Romance" "Drama" "Thriller" "Action" "Crime"   
## [11] "Documentary" "Film-Noir" "Horror" "Musical" "Mystery"   
## [16] "Sci-Fi" "War" "Western"

# prepare genre list of matrix —-

matrix\_genre <- matrix(0,9709,18) # create empty matrix, 9708+1=no of movies+1 for calculation column, 18=no of genres  
matrix\_genre[1:10,] # check empty matrix

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]  
## [1,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [3,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [4,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [5,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [6,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [7,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [8,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [9,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [10,] 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [,14] [,15] [,16] [,17] [,18]  
## [1,] 0 0 0 0 0  
## [2,] 0 0 0 0 0  
## [3,] 0 0 0 0 0  
## [4,] 0 0 0 0 0  
## [5,] 0 0 0 0 0  
## [6,] 0 0 0 0 0  
## [7,] 0 0 0 0 0  
## [8,] 0 0 0 0 0  
## [9,] 0 0 0 0 0  
## [10,] 0 0 0 0 0

matrix\_genre[1,] <- genre\_Picklist # set first row to genre list  
matrix\_genre [1:10,] # check first of genre list

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]   
## [1,] "Adventure" "Animation" "Children" "Comedy" "Fantasy" "Romance" "Drama"  
## [2,] "0" "0" "0" "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0" "0" "0" "0"   
## [,8] [,9] [,10] [,11] [,12] [,13] [,14]   
## [1,] "Thriller" "Action" "Crime" "Documentary" "Film-Noir" "Horror" "Musical"  
## [2,] "0" "0" "0" "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0" "0" "0" "0"   
## [,15] [,16] [,17] [,18]   
## [1,] "Mystery" "Sci-Fi" "War" "Western"  
## [2,] "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0"

colnames(matrix\_genre) <- genre\_Picklist # set column names to genre list  
matrix\_genre[1:10,]# check column name

## Adventure Animation Children Comedy Fantasy Romance Drama   
## [1,] "Adventure" "Animation" "Children" "Comedy" "Fantasy" "Romance" "Drama"  
## [2,] "0" "0" "0" "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0" "0" "0" "0"   
## Thriller Action Crime Documentary Film-Noir Horror Musical   
## [1,] "Thriller" "Action" "Crime" "Documentary" "Film-Noir" "Horror" "Musical"  
## [2,] "0" "0" "0" "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0" "0" "0" "0"   
## Mystery Sci-Fi War Western   
## [1,] "Mystery" "Sci-Fi" "War" "Western"  
## [2,] "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0"

nrow(genresListColumn) # no of rows in genre list

## [1] 9708

ncol(genresListColumn) # no of column in genre list

## [1] 10

# updating empty matrix with generesListColumn values —-

for (i in 1:nrow(genresListColumn)) {  
 for (c in 1:ncol(genresListColumn)) {  
 dummy\_col = which(matrix\_genre[1,] == genresListColumn[i,c])  
 matrix\_genre[i+1,dummy\_col] <- 1  
 }  
}  
  
matrix\_genre [1:10,]# check the assigned values in matrix with values of genresListColumn.

## Adventure Animation Children Comedy Fantasy Romance Drama   
## [1,] "Adventure" "Animation" "Children" "Comedy" "Fantasy" "Romance" "Drama"  
## [2,] "1" "1" "1" "1" "1" "0" "0"   
## [3,] "1" "0" "1" "0" "1" "0" "0"   
## [4,] "0" "0" "0" "1" "0" "1" "0"   
## [5,] "0" "0" "0" "1" "0" "1" "1"   
## [6,] "0" "0" "0" "1" "0" "0" "0"   
## [7,] "0" "0" "0" "0" "0" "0" "0"   
## [8,] "0" "0" "0" "1" "0" "1" "0"   
## [9,] "1" "0" "1" "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0" "0" "0" "0"   
## Thriller Action Crime Documentary Film-Noir Horror Musical   
## [1,] "Thriller" "Action" "Crime" "Documentary" "Film-Noir" "Horror" "Musical"  
## [2,] "0" "0" "0" "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0" "0" "0" "0"   
## [7,] "1" "1" "1" "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0" "0" "0" "0"   
## [10,] "0" "1" "0" "0" "0" "0" "0"   
## Mystery Sci-Fi War Western   
## [1,] "Mystery" "Sci-Fi" "War" "Western"  
## [2,] "0" "0" "0" "0"   
## [3,] "0" "0" "0" "0"   
## [4,] "0" "0" "0" "0"   
## [5,] "0" "0" "0" "0"   
## [6,] "0" "0" "0" "0"   
## [7,] "0" "0" "0" "0"   
## [8,] "0" "0" "0" "0"   
## [9,] "0" "0" "0" "0"   
## [10,] "0" "0" "0" "0"

summary(matrix\_genre) # check rows

## Adventure Animation Children Comedy   
## Length:9709 Length:9709 Length:9709 Length:9709   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## Fantasy Romance Drama Thriller   
## Length:9709 Length:9709 Length:9709 Length:9709   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## Action Crime Documentary Film-Noir   
## Length:9709 Length:9709 Length:9709 Length:9709   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## Horror Musical Mystery Sci-Fi   
## Length:9709 Length:9709 Length:9709 Length:9709   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## War Western   
## Length:9709 Length:9709   
## Class :character Class :character   
## Mode :character Mode :character

# convert into dataframe

matrix\_genre\_Removerow <- as.data.frame(matrix\_genre[-1,], stringsAsFactors=FALSE) # we removed first row as we generated earlier for calculation purpose  
summary(matrix\_genre\_Removerow) # check rows back to normal rows

## Adventure Animation Children Comedy   
## Length:9708 Length:9708 Length:9708 Length:9708   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## Fantasy Romance Drama Thriller   
## Length:9708 Length:9708 Length:9708 Length:9708   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## Action Crime Documentary Film-Noir   
## Length:9708 Length:9708 Length:9708 Length:9708   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## Horror Musical Mystery Sci-Fi   
## Length:9708 Length:9708 Length:9708 Length:9708   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
## War Western   
## Length:9708 Length:9708   
## Class :character Class :character   
## Mode :character Mode :character

nrow(matrix\_genre\_Removerow) # no of rows

## [1] 9708

ncol(matrix\_genre\_Removerow) # no of columns

## [1] 18

# convert matrix characters [“1” or “0”] to integers values [ 1 or 0] —-

for (c in 1:ncol(matrix\_genre\_Removerow)) {  
 matrix\_genre\_Removerow[,c] <- as.integer(matrix\_genre\_Removerow[,c])  
}   
  
summary(matrix\_genre\_Removerow) # check integer values in each column

## Adventure Animation Children Comedy   
## Min. :0.0000 Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.0000 Median :0.00000 Median :0.0000 Median :0.0000   
## Mean :0.1301 Mean :0.06294 Mean :0.0684 Mean :0.3869   
## 3rd Qu.:0.0000 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.00000 Max. :1.0000 Max. :1.0000   
## Fantasy Romance Drama Thriller   
## Min. :0.00000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.00000 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean :0.08024 Mean :0.1644 Mean :0.4492 Mean :0.1951   
## 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.00000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Action Crime Documentary Film-Noir   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.000000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.000000   
## Median :0.0000 Median :0.0000 Median :0.00000 Median :0.000000   
## Mean :0.1883 Mean :0.1235 Mean :0.04532 Mean :0.008962   
## 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:0.00000 3rd Qu.:0.000000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.000000   
## Horror Musical Mystery Sci-Fi   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.0000   
## Median :0.0000 Median :0.0000 Median :0.00000 Median :0.0000   
## Mean :0.1007 Mean :0.0344 Mean :0.05902 Mean :0.1009   
## 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:0.00000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## War Western   
## Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.00000 1st Qu.:0.0000   
## Median :0.00000 Median :0.0000   
## Mean :0.03935 Mean :0.0172   
## 3rd Qu.:0.00000 3rd Qu.:0.0000   
## Max. :1.00000 Max. :1.0000

matrix\_genre\_Removerow[1:10,] # check the matrix

## Adventure Animation Children Comedy Fantasy Romance Drama Thriller Action  
## 1 1 1 1 1 1 0 0 0 0  
## 2 1 0 1 0 1 0 0 0 0  
## 3 0 0 0 1 0 1 0 0 0  
## 4 0 0 0 1 0 1 1 0 0  
## 5 0 0 0 1 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 1 1  
## 7 0 0 0 1 0 1 0 0 0  
## 8 1 0 1 0 0 0 0 0 0  
## 9 0 0 0 0 0 0 0 0 1  
## 10 1 0 0 0 0 0 0 1 1  
## Crime Documentary Film-Noir Horror Musical Mystery Sci-Fi War Western  
## 1 0 0 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 0 0  
## 6 1 0 0 0 0 0 0 0 0  
## 7 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0  
## 9 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0

movies\_df <- cbind(as.data.frame(movies\_d), matrix\_genre\_Removerow) # combine both data (movie and matrix table)  
head(movies\_df)

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

# remove genres column from movie datasets —-

movies\_df <- movies\_df[,-3]   
head(movies\_df) # check the rows

## movieId title Adventure Animation Children  
## 1 1 Toy Story (1995) 1 1 1  
## 2 2 Jumanji (1995) 1 0 1  
## 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) 0 0 0  
## Comedy Fantasy Romance Drama Thriller Action Crime Documentary Film-Noir  
## 1 1 1 0 0 0 0 0 0 0  
## 2 0 1 0 0 0 0 0 0 0  
## 3 1 0 1 0 0 0 0 0 0  
## 4 1 0 1 1 0 0 0 0 0  
## 5 1 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 1 1 1 0 0  
## Horror Musical Mystery Sci-Fi War Western  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 0 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 0 0 0

dim(movies\_df) # check dimension

## [1] 9708 20

# create year field in rating data as new features —-

head(movies\_df$title, 20) # check 20 rows of title

## [1] "Toy Story (1995)"   
## [2] "Jumanji (1995)"   
## [3] "Grumpier Old Men (1995)"   
## [4] "Waiting to Exhale (1995)"   
## [5] "Father of the Bride Part II (1995)"   
## [6] "Heat (1995)"   
## [7] "Sabrina (1995)"   
## [8] "Tom and Huck (1995)"   
## [9] "Sudden Death (1995)"   
## [10] "GoldenEye (1995)"   
## [11] "American President, The (1995)"   
## [12] "Dracula: Dead and Loving It (1995)"   
## [13] "Balto (1995)"   
## [14] "Nixon (1995)"   
## [15] "Cutthroat Island (1995)"   
## [16] "Casino (1995)"   
## [17] "Sense and Sensibility (1995)"   
## [18] "Four Rooms (1995)"   
## [19] "Ace Ventura: When Nature Calls (1995)"  
## [20] "Money Train (1995)"

dim(movies\_df) # check dimension

## [1] 9708 20

years=substr(movies\_df$title, nchar(movies\_df$title)-5+1, nchar(movies\_df$title)-1) # pick year from the text string  
year <- as.data.frame(years) # convert year field into dataframe  
movies\_dfc <- cbind(movies\_df, year) # combine both data (rating and year table)  
head(movies\_dfc,20) # check 20 rows

## movieId title Adventure Animation Children  
## 1 1 Toy Story (1995) 1 1 1  
## 2 2 Jumanji (1995) 1 0 1  
## 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) 0 0 0  
## 7 7 Sabrina (1995) 0 0 0  
## 8 8 Tom and Huck (1995) 1 0 1  
## 9 9 Sudden Death (1995) 0 0 0  
## 10 10 GoldenEye (1995) 1 0 0  
## 11 11 American President, The (1995) 0 0 0  
## 12 12 Dracula: Dead and Loving It (1995) 0 0 0  
## 13 13 Balto (1995) 1 1 1  
## 14 14 Nixon (1995) 0 0 0  
## 15 15 Cutthroat Island (1995) 1 0 0  
## 16 16 Casino (1995) 0 0 0  
## 17 17 Sense and Sensibility (1995) 0 0 0  
## 18 18 Four Rooms (1995) 0 0 0  
## 19 19 Ace Ventura: When Nature Calls (1995) 0 0 0  
## 20 20 Money Train (1995) 0 0 0  
## Comedy Fantasy Romance Drama Thriller Action Crime Documentary Film-Noir  
## 1 1 1 0 0 0 0 0 0 0  
## 2 0 1 0 0 0 0 0 0 0  
## 3 1 0 1 0 0 0 0 0 0  
## 4 1 0 1 1 0 0 0 0 0  
## 5 1 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 1 1 1 0 0  
## 7 1 0 1 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0  
## 9 0 0 0 0 0 1 0 0 0  
## 10 0 0 0 0 1 1 0 0 0  
## 11 1 0 1 1 0 0 0 0 0  
## 12 1 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0  
## 14 0 0 0 1 0 0 0 0 0  
## 15 0 0 1 0 0 1 0 0 0  
## 16 0 0 0 1 0 0 1 0 0  
## 17 0 0 1 1 0 0 0 0 0  
## 18 1 0 0 0 0 0 0 0 0  
## 19 1 0 0 0 0 0 0 0 0  
## 20 1 0 0 1 1 1 1 0 0  
## Horror Musical Mystery Sci-Fi War Western years  
## 1 0 0 0 0 0 0 1995  
## 2 0 0 0 0 0 0 1995  
## 3 0 0 0 0 0 0 1995  
## 4 0 0 0 0 0 0 1995  
## 5 0 0 0 0 0 0 1995  
## 6 0 0 0 0 0 0 1995  
## 7 0 0 0 0 0 0 1995  
## 8 0 0 0 0 0 0 1995  
## 9 0 0 0 0 0 0 1995  
## 10 0 0 0 0 0 0 1995  
## 11 0 0 0 0 0 0 1995  
## 12 1 0 0 0 0 0 1995  
## 13 0 0 0 0 0 0 1995  
## 14 0 0 0 0 0 0 1995  
## 15 0 0 0 0 0 0 1995  
## 16 0 0 0 0 0 0 1995  
## 17 0 0 0 0 0 0 1995  
## 18 0 0 0 0 0 0 1995  
## 19 0 0 0 0 0 0 1995  
## 20 0 0 0 0 0 0 1995

dim(movies\_dfc) # check dimension

## [1] 9708 21

# searching 1 to 20 Crime movie by selecting year 2005 to find movie title —-

subset(movies\_dfc, Crime == 1 & years == 2005)$title[1:20]

## [1] "Elektra (2005)"   
## [2] "Assault on Precinct 13 (2005)"   
## [3] "Be Cool (2005)"   
## [4] "Hostage (2005)"   
## [5] "Miss Congeniality 2: Armed and Fabulous (2005)"   
## [6] "Sin City (2005)"   
## [7] "State Property 2 (2005)"   
## [8] "King's Ransom (2005)"   
## [9] "xXx: State of the Union (2005)"   
## [10] "Unleashed (Danny the Dog) (2005)"   
## [11] "Batman Begins (2005)"   
## [12] "Hustle & Flow (2005)"   
## [13] "Devil's Rejects, The (2005)"   
## [14] "Four Brothers (2005)"   
## [15] "Transporter 2 (2005)"   
## [16] "Lord of War (2005)"   
## [17] "Domino (2005)"   
## [18] "Man, The (2005)"   
## [19] "Exorcism of Emily Rose, The (2005)"   
## [20] "Green Street Hooligans (a.k.a. Hooligans) (2005)"

# searching 1 to 20 Horror movie by selecting year 1995 to find movie title —-

subset(movies\_dfc, Horror == 1 & years == 1995)$title[1:20]

## [1] "Dracula: Dead and Loving It (1995)"   
## [2] "Copycat (1995)"   
## [3] "Vampire in Brooklyn (1995)"   
## [4] "Addiction, The (1995)"   
## [5] "Lord of Illusions (1995)"   
## [6] "Prophecy, The (1995)"   
## [7] "Species (1995)"   
## [8] "Castle Freak (1995)"   
## [9] "Tales from the Crypt Presents: Demon Knight (1995)"   
## [10] "Tales from the Hood (1995)"   
## [11] "Village of the Damned (1995)"   
## [12] "In the Mouth of Madness (1995)"   
## [13] "Candyman: Farewell to the Flesh (1995)"   
## [14] "Halloween: The Curse of Michael Myers (Halloween 6: The Curse of Michael Myers) (1995)"  
## [15] "Carnosaur 2 (1995)"   
## [16] "Darkman II: Return of Durant, The (1995)"   
## [17] "Langoliers, The (1995)"   
## [18] "Leprechaun 3 (1995)"   
## [19] "Ice Cream Man (1995)"   
## [20] NA

# remove timestamp from rating dataset —-

head(ratings\_d) # check rows

## userId movieId rating timestamp  
## 1 1 1 4 964982703  
## 2 1 3 4 964981247  
## 3 1 6 4 964982224  
## 4 1 47 5 964983815  
## 5 1 50 5 964982931  
## 6 1 70 3 964982400

ratings\_d <- ratings\_d[,-4]  
head(ratings\_d) # check rows

## userId movieId rating  
## 1 1 1 4  
## 2 1 3 4  
## 3 1 6 4  
## 4 1 47 5  
## 5 1 50 5  
## 6 1 70 3

dim(ratings\_d) # check dimension

## [1] 100836 3

ratings\_df<-as.data.frame(ratings\_d) # convert data into data frame  
dim(ratings\_df) # check dimension of dataset

## [1] 100836 3

summary(ratings\_df) # check statistic

## userId movieId rating   
## Min. : 1.0 Min. : 1 Min. :0.500   
## 1st Qu.:177.0 1st Qu.: 1199 1st Qu.:3.000   
## Median :325.0 Median : 2991 Median :3.500   
## Mean :326.1 Mean : 19435 Mean :3.502   
## 3rd Qu.:477.0 3rd Qu.: 8122 3rd Qu.:4.000   
## Max. :610.0 Max. :193609 Max. :5.000

head(ratings\_df) # check few records

## userId movieId rating  
## 1 1 1 4  
## 2 1 3 4  
## 3 1 6 4  
## 4 1 47 5  
## 5 1 50 5  
## 6 1 70 3

length(unique(ratings\_df$movieId)) # unique movieId number

## [1] 9724

length(unique(ratings\_df$userId)) # unique userId number

## [1] 610

# check NA’s in ratings datasets of specific field —-

na\_rating <- ratings\_df %>%  
 filter(is.na(ratings\_df$movieId) |is.na(ratings\_df$rating)|is.na(ratings\_df$userId) )  
na\_rating # check the row - movies\_d <- na.omit(movies\_d) # becuase no NA's thats why we do not use this line

## [1] userId movieId rating   
## <0 rows> (or 0-length row.names)

# Find # of MovieId in both CSV files and then will be removed those movies in ratings\_d.CSV which are not rated —-

movies\_MovieIdCount <- length(unique(movies\_dfc$movieId))   
movies\_MovieIdCount # check the movieId count in movies\_d csv

## [1] 9708

ratings\_MovieIdCount <- length(unique(ratings\_df$movieId))  
ratings\_MovieIdCount # check the movieId count in ratings\_d csv

## [1] 9724

# Find not rated rows in movieId and then removed rows from matrix\_genre\_Removerow —-

which((ratings\_df$movieId %in% movies\_dfc$movieId) == FALSE) # check which movieId is not noted in matrix

## [1] 3625 7354 7409 9102 9142 9148 14029 16887 16914 16926 17781 17805  
## [13] 17838 17880 17882 19528 27247 30172 30472 36356 37496 44461 49826 49852  
## [25] 58078 64960 64965 70345 70433 81450 81467 81925 81930 83107 86574 88076  
## [37] 88112 88137 90222 92129 92155 95015 95035 95055 95079 95958 95962

length(which((ratings\_df$movieId %in% movies\_dfc$movieId) == FALSE)) # count of movieId not rated

## [1] 47

dim(ratings\_df) # check dimension

## [1] 100836 3

length(unique(ratings\_df$movieId)) # check lengh

## [1] 9724

ratings\_df\_removeRows <- ratings\_df[-which((ratings\_df$movieId %in% movies\_dfc$movieId) == FALSE),] # removed not rated movieId  
dim(ratings\_df\_removeRows) # check dimension after removing

## [1] 100789 3

length(unique(ratings\_df\_removeRows$movieId)) # check length after removing

## [1] 9690

rownames(ratings\_df\_removeRows) <- NULL  
head(ratings\_df\_removeRows) #check rows

## userId movieId rating  
## 1 1 1 4  
## 2 1 3 4  
## 3 1 6 4  
## 4 1 47 5  
## 5 1 50 5  
## 6 1 70 3

# store the final restult of movies\_dfc in new searchmovie\_matrix.csv file for our shiny app—-

write.csv(movies\_dfc, "searchmovie\_matrix.csv")

# store the final restult of ratings\_matrix in new ratings\_matrix.csv file for our shiny app—-

write.csv(ratings\_df\_removeRows, "searchratings\_matrix.csv")

# User-Based Collaborative Filtering Approach —-

# we are using the behavior of user's preference in our case "generes"  
# based on genere's items we recommends an items to similar users in the same  
# group for views  
  
# Create sparse matrix for recommendation. Rows = userId, Columns = movieId  
ratings\_matrix <- dcast(ratings\_df\_removeRows, userId~movieId, value.var = "rating", na.rm=FALSE) # movieId for x-axix and UserId for y-axix  
ratings\_matrix [1:10,1:10]# check 1-10 rows (userId) and columns (movieId)

## userId 1 2 3 4 5 6 7 8 9  
## 1 1 4.0 NA 4 NA NA 4 NA NA NA  
## 2 2 NA NA NA NA NA NA NA NA NA  
## 3 3 NA NA NA NA NA NA NA NA NA  
## 4 4 NA NA NA NA NA NA NA NA NA  
## 5 5 4.0 NA NA NA NA NA NA NA NA  
## 6 6 NA 4 5 3 5 4 4 3 NA  
## 7 7 4.5 NA NA NA NA NA NA NA NA  
## 8 8 NA 4 NA NA NA NA NA NA NA  
## 9 9 NA NA NA NA NA NA NA NA NA  
## 10 10 NA NA NA NA NA NA NA NA NA

dim(ratings\_matrix) # check dimension of matrix

## [1] 610 9691

ncol(ratings\_matrix) # check column number - actual movieId is 9724 so we need to remove first column userId

## [1] 9691

# convert dataframe into matrix

ratings\_matrix <- as.matrix(ratings\_matrix[,-1])# removing userid col as userId represent Rows and movieId represent Columns  
ratings\_matrix[1:10,1:10] # check rows

## 1 2 3 4 5 6 7 8 9 10  
## [1,] 4.0 NA 4 NA NA 4 NA NA NA NA  
## [2,] NA NA NA NA NA NA NA NA NA NA  
## [3,] NA NA NA NA NA NA NA NA NA NA  
## [4,] NA NA NA NA NA NA NA NA NA NA  
## [5,] 4.0 NA NA NA NA NA NA NA NA NA  
## [6,] NA 4 5 3 5 4 4 3 NA 3  
## [7,] 4.5 NA NA NA NA NA NA NA NA NA  
## [8,] NA 4 NA NA NA NA NA NA NA 2  
## [9,] NA NA NA NA NA NA NA NA NA NA  
## [10,] NA NA NA NA NA NA NA NA NA NA

dim(ratings\_matrix) # check dimension

## [1] 610 9690

# Convert rating matrix into a recommenderlab sparse matrix

ratings\_matrix <- as(ratings\_matrix, "realRatingMatrix")  
ratings\_matrix # check rating values

## 610 x 9690 rating matrix of class 'realRatingMatrix' with 100789 ratings.

# recommendation model parameters use for our algorithm

recommender\_model <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")  
names(recommender\_model)

## [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"

lapply(recommender\_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."

# look into Item-based and user-based parameter for model preparation

recommender\_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

recommender\_model$UBCF\_realRatingMatrix$parameters

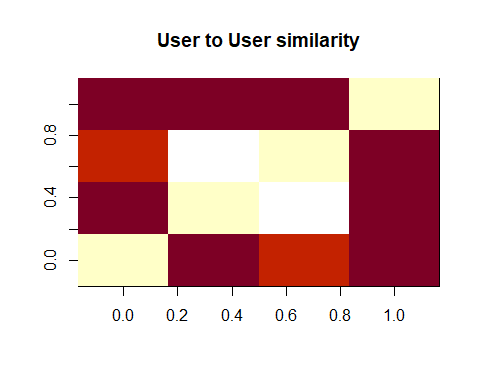
## $method  
## [1] "cosine"  
##   
## $nn  
## [1] 25  
##   
## $sample  
## [1] FALSE  
##   
## $weighted  
## [1] TRUE  
##   
## $normalize  
## [1] "center"  
##   
## $min\_matching\_items  
## [1] 0  
##   
## $min\_predictive\_items  
## [1] 0

# check the similarity between users or betweeb items. any one method is applied in model *cosine, pearson* and *jaccard*.

similarity\_users <- similarity(ratings\_matrix[1:4, ], method = "cosine", which = "users") # check 1 to 4 userid similarity  
as.matrix(similarity\_users)

## 1 2 3 4  
## 1 0.0000000 1 0.7919033 0.9328096  
## 2 1.0000000 0 NA 1.0000000  
## 3 0.7919033 NA 0.0000000 1.0000000  
## 4 0.9328096 1 1.0000000 0.0000000

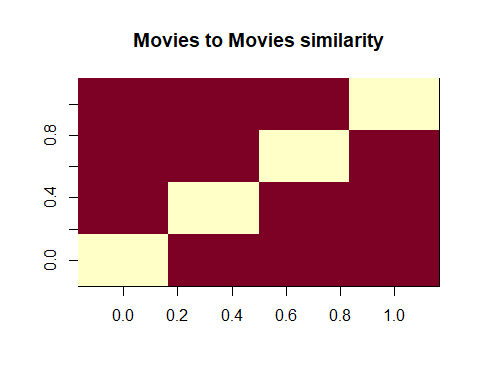
image(as.matrix(similarity\_users), main = "User to User similarity") # row and column represent to a user and each cell represent to the similarity, more red the cell = more similarity b/w users, diagonal is always red (each user itself comparing)



similarity\_items <- similarity(ratings\_matrix[, 1:4], method = "cosine", which = "items") # check 1 to 4 moviesid similarity  
as.matrix(similarity\_items)

## 1 2 3 4  
## 1 0.0000000 0.9644641 0.9715415 0.9838699  
## 2 0.9644641 0.0000000 0.9389013 0.9609877  
## 3 0.9715415 0.9389013 0.0000000 1.0000000  
## 4 0.9838699 0.9609877 1.0000000 0.0000000

image(as.matrix(similarity\_items), main = "Movies to Movies similarity")



# find the rating count in each head based on total rating view 5,931,640

ratings\_vector <- as.vector(ratings\_matrix@data) # data shows no of rating   
unique(ratings\_vector) # unique rating values

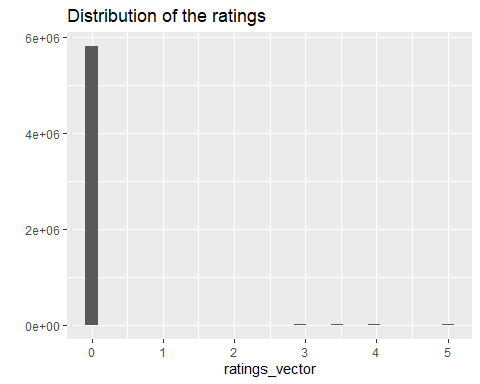
## [1] 4.0 0.0 4.5 2.5 3.5 3.0 5.0 0.5 2.0 1.5 1.0

ratingstable\_vector <- table(ratings\_vector) # put into table for each rating value count  
ratingstable\_vector # 0 rating values are so high 5,830,822 it represent missing values which will be removed from data sets

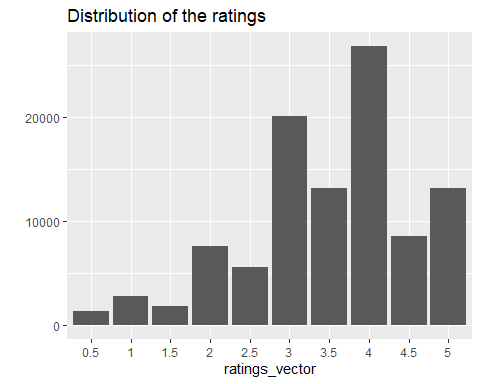
## ratings\_vector  
## 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5   
## 5810111 1368 2809 1791 7549 5544 20041 13130 26810 8543   
## 5   
## 13204

qplot(ratings\_vector) +   
 ggtitle("Distribution of the ratings") # 0 rating values so high and now it removed from rating datasets

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ratings\_vector <- ratings\_vector[ratings\_vector != 0] # rating == 0 are missing values in rating datasets and now removed  
ratings\_vector <- factor(ratings\_vector) # extract unique values of rating for graph  
  
qplot(ratings\_vector) +   
 ggtitle("Distribution of the ratings") # after removing 0 rating from graph. majority of rating values fall under > 3



# find top movies views in rating datasets

count\_Permovie <- colCounts(ratings\_matrix) # no of movie view for each movie  
topmovies\_views <- data.frame(movies\_names = names(count\_Permovie),  
 moviesviews = count\_Permovie) # create movie view dataframe, assign movie view count as variable name  
head(topmovies\_views, 10) # check raw data

## movies\_names moviesviews  
## 1 1 215  
## 2 2 110  
## 3 3 52  
## 4 4 7  
## 5 5 49  
## 6 6 102  
## 7 7 54  
## 8 8 8  
## 9 9 16  
## 10 10 132

topmovies\_views <- topmovies\_views[order(topmovies\_views$moviesviews, decreasing = TRUE),] # sort Desc   
head(topmovies\_views,10) # check the sorting

## movies\_names moviesviews  
## 356 356 329  
## 318 318 317  
## 296 296 307  
## 593 593 279  
## 2571 2571 278  
## 260 260 251  
## 480 480 238  
## 110 110 237  
## 589 589 224  
## 527 527 220

topmovies\_views$title <- NA # create new variable called title and assigned NA  
head(topmovies\_views, 10) # check the rows

## movies\_names moviesviews title  
## 356 356 329 NA  
## 318 318 317 NA  
## 296 296 307 NA  
## 593 593 279 NA  
## 2571 2571 278 NA  
## 260 260 251 NA  
## 480 480 238 NA  
## 110 110 237 NA  
## 589 589 224 NA  
## 527 527 220 NA

dim(topmovies\_views) # check the dimension and rows

## [1] 9690 3

nrow(topmovies\_views) # no of rows

## [1] 9690

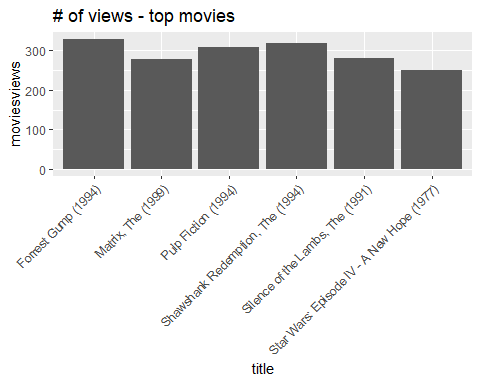
head(movies\_dfc) # few rows

## movieId title Adventure Animation Children  
## 1 1 Toy Story (1995) 1 1 1  
## 2 2 Jumanji (1995) 1 0 1  
## 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) 0 0 0  
## Comedy Fantasy Romance Drama Thriller Action Crime Documentary Film-Noir  
## 1 1 1 0 0 0 0 0 0 0  
## 2 0 1 0 0 0 0 0 0 0  
## 3 1 0 1 0 0 0 0 0 0  
## 4 1 0 1 1 0 0 0 0 0  
## 5 1 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 1 1 1 0 0  
## Horror Musical Mystery Sci-Fi War Western years  
## 1 0 0 0 0 0 0 1995  
## 2 0 0 0 0 0 0 1995  
## 3 0 0 0 0 0 0 1995  
## 4 0 0 0 0 0 0 1995  
## 5 0 0 0 0 0 0 1995  
## 6 0 0 0 0 0 0 1995

for (r in 1:nrow(topmovies\_views)){  
 topmovies\_views[r,3] <- as.character(subset(movies\_dfc, movies\_dfc$movieId == topmovies\_views[r,1])$title)  
} # pick title from movies\_dfc and added into topmovies\_views  
topmovies\_views[1:6,] # check 1 to 6 rows

## movies\_names moviesviews title  
## 356 356 329 Forrest Gump (1994)  
## 318 318 317 Shawshank Redemption, The (1994)  
## 296 296 307 Pulp Fiction (1994)  
## 593 593 279 Silence of the Lambs, The (1991)  
## 2571 2571 278 Matrix, The (1999)  
## 260 260 251 Star Wars: Episode IV - A New Hope (1977)

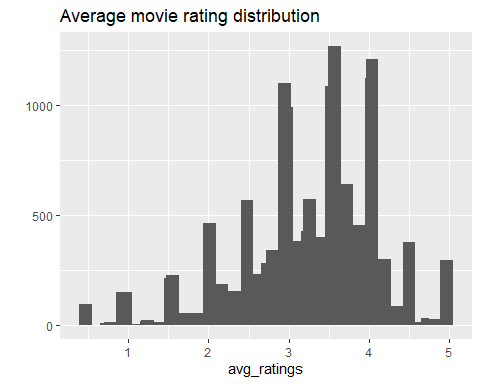
ggplot(topmovies\_views[1:6, ], aes(x = title, y = moviesviews)) +  
 geom\_bar(stat="identity") +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +   
ggtitle("# of views - top movies")



# Average and Relevant movie rating distribution

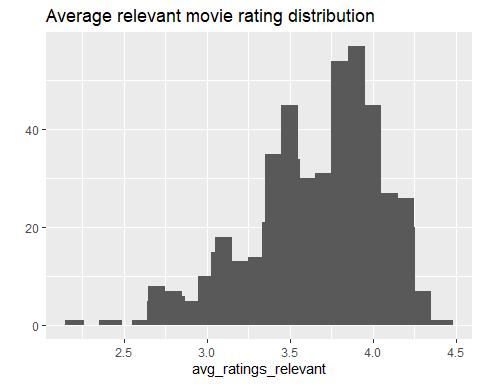
avg\_ratings <- colMeans(ratings\_matrix) # average movie column rating in rating matrix   
  
qplot(avg\_ratings) +   
 stat\_bin(binwidth = 0.1) +  
 ggtitle("Average movie rating distribution") # represent in graph, highest values fall btw 3 and 4 in average rating and 1, 2 and 5 have a fewest rating

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



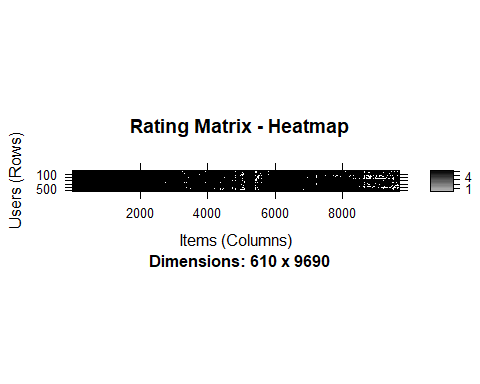
avg\_ratings\_relevant <- avg\_ratings[count\_Permovie > 50] # set minimum criteria 50 view per movie out of 1302 views.  
  
qplot(avg\_ratings\_relevant) +   
 stat\_bin(binwidth = 0.1) +  
 ggtitle(paste("Average relevant movie rating distribution")) # represent in graph, people rated the movies b/w 3 and 4.5

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

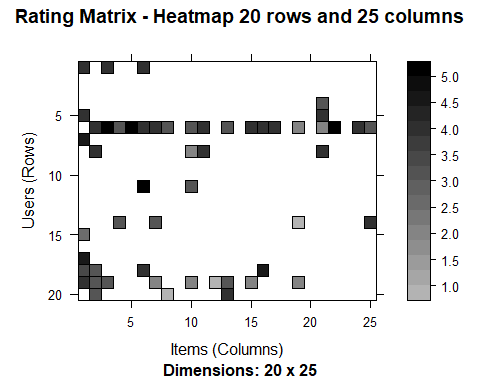


# create Heatmap graph for rating matrix —-

# row represent userId, column represent movieId and cell represent to ratings  
image(ratings\_matrix, main = "Rating Matrix - Heatmap") # difficult to read the rating dimensions



image(ratings\_matrix[1:20, 1:25], main = "Rating Matrix - Heatmap 20 rows and 25 columns") # close look of 20 rows and 25 columns



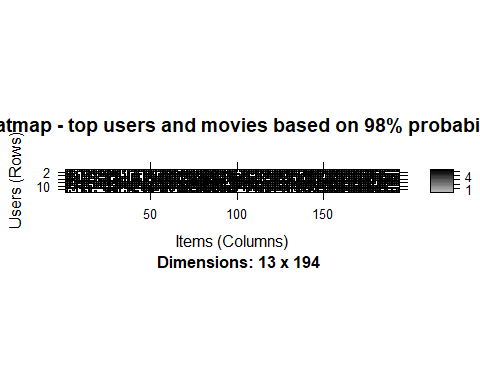
# Now I set the criteria to shorten rating data set because some user saw more movie and but not rated or some movie seen by many users but not rated and viceversa  
  
min\_nof\_movies <- quantile(rowCounts(ratings\_matrix), 0.98) # set the min values of movie per user and apply 0.98% probability out of 1 , row represent user   
min\_nof\_movies

## 98%   
## 976.64

min\_nof\_users <- quantile(colCounts(ratings\_matrix), 0.98) # set the min values of user per movie and apply 0.98% probability out of 1 , column represent movie  
min\_nof\_users

## 98%   
## 83.22

image(ratings\_matrix[rowCounts(ratings\_matrix) > min\_nof\_movies,  
 colCounts(ratings\_matrix) > min\_nof\_users],   
 main = "Heatmap - top users and movies based on 98% probability") # apply user and movies matching criteria, dark represent high-rated movies and user also giving high ratings

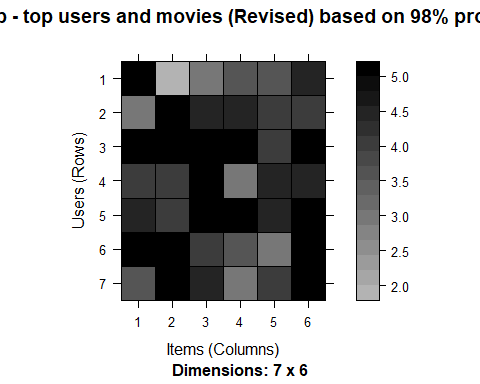


# customize our rating matrix data set —-

# two steps 1) select more relevant data (user and movies) 2) Normalize the data  
# 1) select more relevant data (user and movies)  
# assume 60 (10% of total user 610) minimum number of users per rated movie and 65 (5% of total views 1302) minimum views number per movie  
movies\_ratings <- ratings\_matrix[rowCounts(ratings\_matrix) > 60,  
 colCounts(ratings\_matrix) > 65]  
movies\_ratings # compared to 610 user x 9724 movies with total ratings 100,818

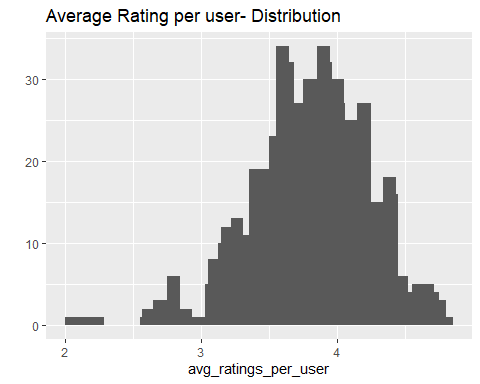
## 334 x 283 rating matrix of class 'realRatingMatrix' with 26739 ratings.

# now select top 98% of users and movies in the next rating matrix   
Newmin\_nof\_movies <- quantile(rowCounts(movies\_ratings), 0.98) # set the min values of movie per user  
Newmin\_nof\_users <- quantile(colCounts(movies\_ratings), 0.98) # set the min values of user per movies  
  
image(movies\_ratings[rowCounts(movies\_ratings) > Newmin\_nof\_movies,  
 colCounts(movies\_ratings) > Newmin\_nof\_users],   
 main = "Heatmap - top users and movies (Revised) based on 98% probability") # present in graph



avg\_ratings\_per\_user <- rowMeans(movies\_ratings) # average row user rating in rating matrix  
  
qplot(avg\_ratings\_per\_user) + stat\_bin(binwidth = 0.1) +  
 ggtitle("Average Rating per user- Distribution") # average rating per user varies a lot

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

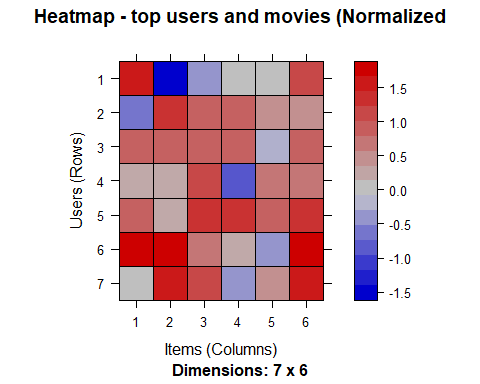


# 2) Normalize the data

# To remove the bias factor in movies rating as user gave, I normalized data to ensure that overall average to mean is Zero.  
movies\_ratings\_normalized <- normalize(movies\_ratings)  
sum(rowMeans(movies\_ratings\_normalized) > 0.00001) # check to see sum of user average rating should be zero.

## [1] 0

image(movies\_ratings\_normalized[rowCounts(movies\_ratings\_normalized) > Newmin\_nof\_movies,   
 colCounts(movies\_ratings\_normalized) > Newmin\_nof\_users],  
 main = "Heatmap - top users and movies (Normalized") # present in graph for top user and movies, some cell shows red and blue color, however sum of average rating is ZERO



# Developing Training / Testing data sets —-

# In our recommendation system, we used collaborative filtering approach  
# Basic concepts is user's collaborating to each other to recommend other items  
# key Algorithm points are 1) similarity rating between two items by similar users 2) Identify each item of K (neighbor) most similar items  
# 3) Identify each user of simialr items for each users  
# I used 80% as trining data and 20% used as testing data in rating matrix  
  
sampled\_moviesratings <- sample(x = c(TRUE, FALSE),   
 size = nrow(movies\_ratings),  
 replace = TRUE,   
 prob = c(0.8, 0.2)) # 80% training and 20% testing  
  
Train\_data <- movies\_ratings[sampled\_moviesratings, ] # create training set  
Train\_data # check the rows and columns

## 266 x 283 rating matrix of class 'realRatingMatrix' with 21440 ratings.

Test\_data <- movies\_ratings[!sampled\_moviesratings, ] # create testing set  
Test\_data # check the rows and columns

## 68 x 283 rating matrix of class 'realRatingMatrix' with 5299 ratings.

# Create recommendation model —–

# we used IBCF (item-based) model and UBCF (user-based) model as comparison  
# in IBCF model paremeter, K = 30 (already define, # of items to computer similarity ) and  
# method = Cosine(Default) as alternative "Pearson" will be used  
# recommender\_models <- recommenderRegistry$get\_entries(dataType ="realRatingMatrix") (already mentioned in earlier steps)  
# recommender\_models$IBCF\_realRatingMatrix$parameters (already mentioned in earlier steps)  
  
Model\_Recommendation\_IBCF <- Recommender(data = Train\_data, method = "IBCF",  
 parameter = list(method="Cosine", k = 30)) # IBCF recommendation model  
Model\_Recommendation\_IBCF # check recommendation information

## Recommender of type 'IBCF' for 'realRatingMatrix'   
## learned using 266 users.

class(Model\_Recommendation\_IBCF) # check class of recommendation model

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

# exploring recommender model of IBCF —-

model\_info <- getModel(Model\_Recommendation\_IBCF)   
model\_info$description # model description

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

model\_info$k # k = 30

## [1] 30

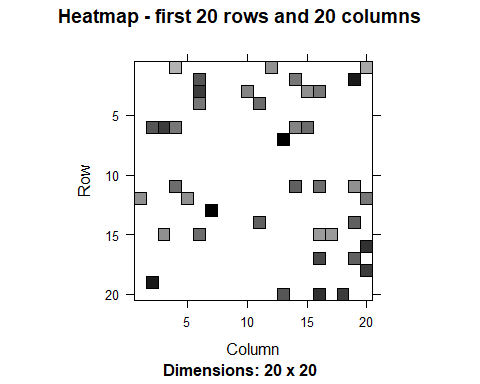
class(model\_info$sim) # has similarity matrix content information

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

dim(model\_info$sim) # check the dimension of similarity matrix

## [1] 283 283

no\_items <- 20 # first 20 rows and first 20 columns  
image(model\_info$sim[1:no\_items, 1:no\_items],   
 main = "Heatmap - first 20 rows and 20 columns") # represent in graph, In first 20 items many values are ZERO however few items in respect of K (30) neighbour element



# their values are > ZERO which means that matrix are not fully simmetric.  
  
row\_sums\_matrix <- rowSums(model\_info$sim > 0) # checking sum of rows in the similarity matrix above ZERO   
table(row\_sums\_matrix) # check in table

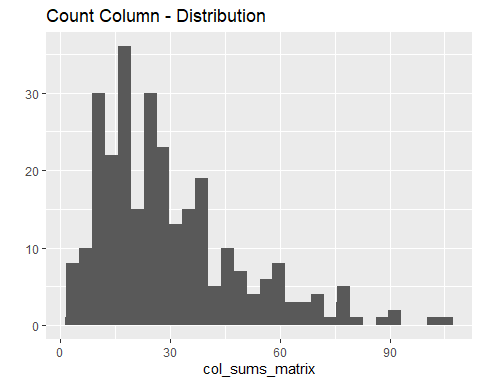
## row\_sums\_matrix  
## 30   
## 283

col\_sums\_matrix <- colSums(model\_info$sim > 0) # checking sum of columns in the similarity matrix above ZERO  
table(col\_sums\_matrix) # check in table

## col\_sums\_matrix  
## 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22   
## 1 2 5 3 3 4 7 6 8 9 7 7 8 7 8 9 12 4 5 6   
## 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 43   
## 6 8 9 7 8 6 9 4 2 2 5 2 6 7 4 3 6 6 4 1   
## 44 45 46 47 49 50 51 52 54 55 56 58 59 60 61 62 65 66 67 68   
## 2 2 2 4 1 4 2 1 3 1 3 2 2 3 3 1 2 1 1 1   
## 69 71 72 73 76 77 79 80 89 92 102 104   
## 1 2 1 1 3 1 1 1 1 2 1 1

qplot(col\_sums\_matrix) + stat\_bin(binwidth = 1) + ggtitle("Count Column - Distribution") # present in graph shows that few movies are similar to other items

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# check 1 to 6 most element movies  
targetCHK\_list <- order(col\_sums\_matrix, decreasing = TRUE)[1:6] # sort in desc   
targetCHK\_list # list movie id counts

## [1] 19 58 142 201 89 38

list\_Movies <- as.integer(rownames(model\_info$sim)[targetCHK\_list]) # assign list of movie's count and extract movid id   
list\_Movies # list movie id

## [1] 95 434 1544 3623 788 292

for (i in 1:6){  
list\_Movies[i] <- as.character(subset(movies\_dfc,movies\_dfc$movieId == list\_Movies[i])$title) # pick movie id title from movies\_dfc datasets  
}  
list\_Movies # print 1 to 6 top movie titles

## [1] "Broken Arrow (1996)"   
## [2] "Cliffhanger (1993)"   
## [3] "Lost World: Jurassic Park, The (1997)"  
## [4] "Mission: Impossible II (2000)"   
## [5] "Nutty Professor, The (1996)"   
## [6] "Outbreak (1995)"

# Applying recommender model IBCF —-

# let assume 10 number of movies recommend to each user based on 20 % of testing dataset  
# recommder model, first start extracting movies rating of each movies and then find similar movies   
# in the similarity matrix of movies for recommendation to users  
  
no\_recommendedMovies <- 10 # max 10 movies recommend to each user  
predicted\_model\_IBCF <- predict(object = Model\_Recommendation\_IBCF, newdata = Test\_data,   
 n = no\_recommendedMovies) # apply predication  
predicted\_model\_IBCF # check predication values

## Recommendations as 'topNList' with n = 10 for 68 users.

predicted\_model\_IBCF\_user\_1 <- predicted\_model\_IBCF@items[[1]] # extract 1-10 movies items from predication model for 1st user  
predicted\_model\_IBCF\_user\_1 # showing 1-10 movie items for user 1

## [1] 34 37 58 80 159 73 157 69 219 251

predicted\_model\_IBCF\_movie\_user\_1 <- predicted\_model\_IBCF@itemLabels[predicted\_model\_IBCF\_user\_1] # pick movie id based on 1-10 predication  
predicted\_model\_IBCF\_movie\_user\_1 # showing 1-10 movie items for user 1

## [1] "253" "288" "434" "595" "1968" "587" "1923" "539" "4886" "8636"

predicted\_model\_IBCF\_movie\_user\_2 <- predicted\_model\_IBCF\_movie\_user\_1   
for (i in 1:10){  
 predicted\_model\_IBCF\_movie\_user\_2[i] <- as.character(subset(movies\_dfc,   
 movies\_dfc$movieId == predicted\_model\_IBCF\_movie\_user\_1[i])$title)  
} # pick movie title from movie\_dfc data set for 10 movie items  
predicted\_model\_IBCF\_movie\_user\_2 # generate 1-10 movies list for user 2

## [1] "Interview with the Vampire: The Vampire Chronicles (1994)"  
## [2] "Natural Born Killers (1994)"   
## [3] "Cliffhanger (1993)"   
## [4] "Beauty and the Beast (1991)"   
## [5] "Breakfast Club, The (1985)"   
## [6] "Ghost (1990)"   
## [7] "There's Something About Mary (1998)"   
## [8] "Sleepless in Seattle (1993)"   
## [9] "Monsters, Inc. (2001)"   
## [10] "Spider-Man 2 (2004)"

Recommder\_matrix\_IBCF <- sapply(predicted\_model\_IBCF@items,   
 function(x){ as.integer(colnames(movies\_ratings)[x]) }) # create matrix with the recommendations for each user  
dim(Recommder\_matrix\_IBCF) # check dimension

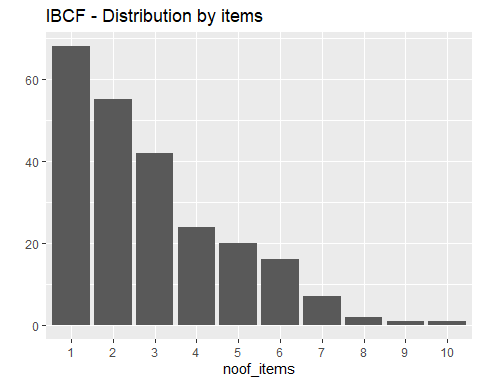
## [1] 10 68

Recommder\_matrix\_IBCF[,1:4] # 1-4 user, column = user and row = movies items (1-10 movies)

## [,1] [,2] [,3] [,4]  
## [1,] 253 2797 1246 1265  
## [2,] 288 235 588 1073  
## [3,] 434 5816 594 339  
## [4,] 595 2291 266 2174  
## [5,] 1968 4896 3751 1288  
## [6,] 587 111 1259 1259  
## [7,] 1923 1278 380 35836  
## [8,] 539 1721 1968 2987  
## [9,] 4886 648 2081 6863  
## [10,] 8636 919 35836 3948

# Recommended movies based on distribution of 1-10 movies items

noof\_items <- factor(table(Recommder\_matrix\_IBCF))  
char\_title <- "IBCF - Distribution by items"  
qplot(noof\_items) + ggtitle(char\_title) # present in graph



noof\_items\_sorted <- sort(noof\_items, decreasing = TRUE) # sorting in desc  
noof\_items\_4top <- head(noof\_items\_sorted, n = 4) # top 4 assigned in another variable  
table\_4top <- data.frame(as.integer(names(noof\_items\_4top)),  
 noof\_items\_4top) # convert into dataframe  
table\_4top # show top 4 values

## as.integer.names.noof\_items\_4top.. noof\_items\_4top  
## 919 919 10  
## 253 253 9  
## 440 440 8  
## 2542 2542 8

for (i in 1:4){  
 table\_4top[i,1] <- as.character(subset(movies\_dfc,   
 movies\_dfc$movieId == table\_4top[i,1])$title)  
} # pick movie title from movie\_dfc data set  
colnames(table\_4top) <- c("title", "# items") # assign colum names  
head(table\_4top) # check rows for 4 top list

## title # items  
## 919 Wizard of Oz, The (1939) 10  
## 253 Interview with the Vampire: The Vampire Chronicles (1994) 9  
## 440 Dave (1993) 8  
## 2542 Lock, Stock & Two Smoking Barrels (1998) 8

# User Based Collaborative Filtering system —-

# similar user identified and then item rated by similar user is recommended  
# similiar is measure through cosine and correlation  
# k- nearest-neighbours identify  
Model\_Recommendation\_UBCF <- Recommender(data = Train\_data, method = "UBCF",  
 parameter = list(method="Cosine", nn = 25)) # UBCF recommendation model  
Model\_Recommendation\_UBCF # check recommendation information

## Recommender of type 'UBCF' for 'realRatingMatrix'   
## learned using 266 users.

class(Model\_Recommendation\_UBCF) # check class of recommendation model

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

model\_info\_UBCF <- getModel(Model\_Recommendation\_UBCF)   
model\_info\_UBCF$description # model description

## [1] "UBCF-Real data: contains full or sample of data set"

model\_info\_UBCF$nn # nn = 30

## [1] 25

dim(model\_info\_UBCF$data) # check the dimension of similarity matrix

## [1] 266 283

no\_recommendedMovies <- 10 # max 10 movies recommend to each user  
predicted\_model\_UBCF <- predict(object = Model\_Recommendation\_UBCF, newdata = Test\_data,   
 n = no\_recommendedMovies) # apply predication  
predicted\_model\_UBCF # check predication values

## Recommendations as 'topNList' with n = 10 for 68 users.

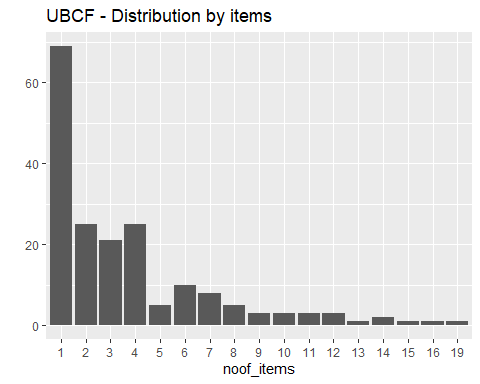
Recommder\_matrix\_UBCF <- sapply(predicted\_model\_UBCF@items,   
 function(x){ as.integer(colnames(movies\_ratings)[x]) }) # create matrix with the recommendations for each user  
dim(Recommder\_matrix\_UBCF) # check dimension

## [1] 10 68

Recommder\_matrix\_UBCF[,1:4] # 1-4 user, column = user and row = movies items (1-10 movies)

## [,1] [,2] [,3] [,4]  
## [1,] 912 91529 1288 2797  
## [2,] 904 1193 3481 1573  
## [3,] 1079 49272 4034 2700  
## [4,] 3481 778 1246 2396  
## [5,] 778 1288 508 750  
## [6,] 2396 1246 1101 1288  
## [7,] 1101 68157 3147 1193  
## [8,] 1201 919 185 1258  
## [9,] 1225 1213 339 91529  
## [10,] 3751 4011 356 1225

noof\_items <- factor(table(Recommder\_matrix\_UBCF))  
char\_title <- "UBCF - Distribution by items"  
qplot(noof\_items) + ggtitle(char\_title) # present in graph



noof\_items\_sorted <- sort(noof\_items, decreasing = TRUE) # sorting in desc  
noof\_items\_4top <- head(noof\_items\_sorted, n = 4) # top 4 assigned in another variable  
table\_4top <- data.frame(as.integer(names(noof\_items\_4top)),  
 noof\_items\_4top) # convert into dataframe  
table\_4top # show top 4 values

## as.integer.names.noof\_items\_4top.. noof\_items\_4top  
## 2700 2700 19  
## 49272 49272 16  
## 337 337 15  
## 750 750 14

for (i in 1:4){  
 table\_4top[i,1] <- as.character(subset(movies\_dfc,   
 movies\_dfc$movieId == table\_4top[i,1])$title)  
} # pick movie title from movie\_dfc data set  
colnames(table\_4top) <- c("title", "# items") # assign colum names  
head(table\_4top) # check rows for 4 top list

## title  
## 2700 South Park: Bigger, Longer and Uncut (1999)  
## 49272 Casino Royale (2006)  
## 337 What's Eating Gilbert Grape (1993)  
## 750 Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)  
## # items  
## 2700 19  
## 49272 16  
## 337 15  
## 750 14

# While comparing both IBCF and UBCF, we found that distribution of movie count is higher in UBCF (Average 17 movie viewed) as  
# compared to IBCF (Average 11 movie viewed)

# Recommended System Evaluation —-

# three way to evaluate the model. Choose best performing model among them and then optimize parameter if needed  
# 1) split the data into training (80%) and testing (20%)  
# 2) bootstrapping  
# 3) k-fold   
  
# 1) splitting data set  
min(rowCounts(movies\_ratings)) # find minimum number of items rated by user to ensure item is rated

## [1] 9

training\_Percentage <- 0.8  
items\_to\_recommend <- 6 # # of items to be recommended  
  
ratings\_threshold <- 3.5 # set minimum Rating threshold consider to be good  
noof\_evaluation <- 1 # set 1 time to run evaulation  
  
evaluation\_Matrix <- evaluationScheme(data = movies\_ratings,method = "split",   
 train = training\_Percentage, given = items\_to\_recommend,  
 goodRating = ratings\_threshold, k = noof\_evaluation) # system evaluation  
  
evaluation\_Matrix # show evaluation parameter

## Evaluation scheme with 6 items given  
## Method: 'split' with 1 run(s).  
## Training set proportion: 0.800  
## Good ratings: >=3.500000  
## Data set: 334 x 283 rating matrix of class 'realRatingMatrix' with 26739 ratings.

getData(evaluation\_Matrix, "train") # training matrix information

## 267 x 283 rating matrix of class 'realRatingMatrix' with 21834 ratings.

getData(evaluation\_Matrix, "known") # set with the items used to build the recommendations

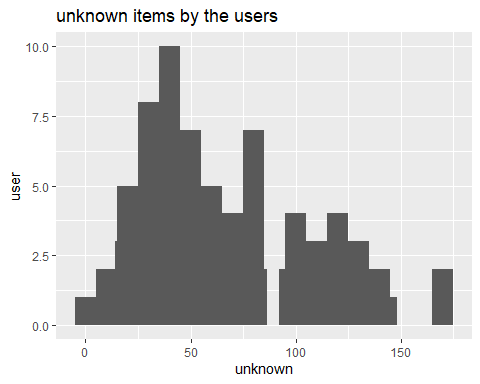
## 67 x 283 rating matrix of class 'realRatingMatrix' with 402 ratings.

getData(evaluation\_Matrix, "unknown") # set with the items used to test the recommendations

## 67 x 283 rating matrix of class 'realRatingMatrix' with 4503 ratings.

qplot(rowCounts(getData(evaluation\_Matrix, "unknown")), xlab="unknown", ylab="user") +   
 geom\_histogram(binwidth = 10) +   
 ggtitle("unknown items by the users") # present in graph shows about majority of unknown items by user

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# 2) bootstrapping the data

# In this splitting method, possibility is that same user may be a sample user in test set based on the same size of training data set and other parameters  
  
evaluation\_Matrix <- evaluationScheme(data = movies\_ratings, method = "bootstrap",   
 train = training\_Percentage, given = items\_to\_recommend,  
 goodRating = ratings\_threshold, k = noof\_evaluation) # system evaluation  
  
evaluation\_Matrix # show evaluation parameter

## Evaluation scheme with 6 items given  
## Method: 'bootstrap' with 1 run(s).  
## Training set proportion: 0.800  
## Good ratings: >=3.500000  
## Data set: 334 x 283 rating matrix of class 'realRatingMatrix' with 26739 ratings.

getData(evaluation\_Matrix, "train") # training matrix information

## 267 x 283 rating matrix of class 'realRatingMatrix' with 20512 ratings.

getData(evaluation\_Matrix, "known") # set with the items used to build the recommendations

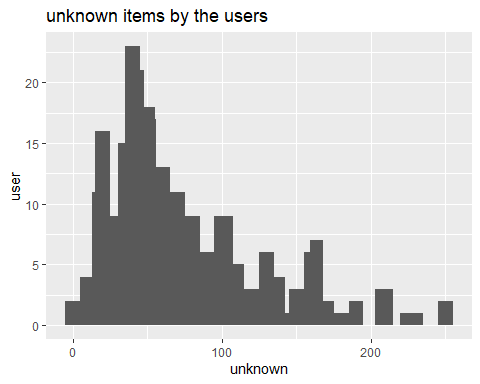
## 155 x 283 rating matrix of class 'realRatingMatrix' with 930 ratings.

getData(evaluation\_Matrix, "unknown") # set with the items used to test the recommendations

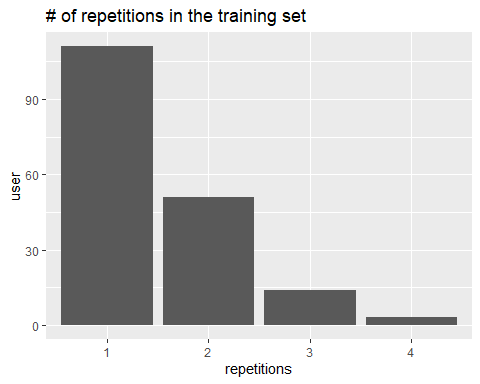
## 155 x 283 rating matrix of class 'realRatingMatrix' with 11729 ratings.

qplot(rowCounts(getData(evaluation\_Matrix, "unknown")), xlab="unknown", ylab="user") +   
 geom\_histogram(binwidth = 10) +   
 ggtitle("unknown items by the users") # present in graph shows about majority of unknown items by user

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



train\_table <- table(evaluation\_Matrix@runsTrain[[1]]) # generate training table  
  
noof\_repetitions <- factor(as.vector(train\_table))  
qplot(noof\_repetitions, xlab="repetitions", ylab="user") +   
 ggtitle("# of repetitions in the training set") # mostly user are repeated in training set as a sample



# 3) k-fold (cross-validation)

# more accurate approach using average (weight) accuracy.  
# Data is split into different chunk. take out testing data from one chunk  
# and then evaluate the accuracy. same thing for another chunk. finally compute   
# all chunk accuracy by using average (weight) method.  
  
  
noof\_fold <- 5 # set the K fold parameter  
  
evaluation\_Matrix <- evaluationScheme(data = movies\_ratings, method = "cross-validation",   
 given = items\_to\_recommend,  
 goodRating = ratings\_threshold,  
 k = noof\_fold) # system evaluation  
  
evaluation\_Matrix # show evaluation parameter

## Evaluation scheme with 6 items given  
## Method: 'cross-validation' with 5 run(s).  
## Good ratings: >=3.500000  
## Data set: 334 x 283 rating matrix of class 'realRatingMatrix' with 26739 ratings.

getData(evaluation\_Matrix, "train") # training matrix information

## 264 x 283 rating matrix of class 'realRatingMatrix' with 21201 ratings.

getData(evaluation\_Matrix, "known") # set with the items used to build the recommendations

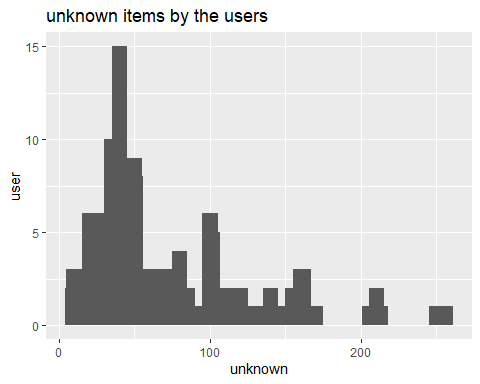
## 70 x 283 rating matrix of class 'realRatingMatrix' with 420 ratings.

getData(evaluation\_Matrix, "unknown") # set with the items used to test the recommendations

## 70 x 283 rating matrix of class 'realRatingMatrix' with 5118 ratings.

qplot(rowCounts(getData(evaluation\_Matrix, "unknown")), xlab="unknown", ylab="user") +   
 geom\_histogram(binwidth = 10) +   
 ggtitle("unknown items by the users") # present in graph shows about majority of unknown i

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



size\_sets <- sapply(evaluation\_Matrix@runsTrain, length)  
size\_sets # using 5-fold method, we get 5 sets of the same size

## [1] 264 264 264 264 264

# Ratings Evaluation

# As K-fold approach is accurate then we build IBCF (Item\_based) model and find prediction ratings  
  
evaluation\_Matrix <- evaluationScheme(data = movies\_ratings,  
 method = "cross-validation",   
 given = items\_to\_recommend,  
 goodRating = ratings\_threshold,  
 k = noof\_fold) # system evaluation  
  
evaluate\_model <- "IBCF" # item-based calloborative filtering  
parameters\_model <- NULL  
  
  
Model\_Recommendation\_Evaluation <- Recommender(data = getData(evaluation\_Matrix, "train"),  
 method = evaluate\_model,  
 parameter = parameters\_model) # Model recommendation based on K-fold  
Model\_Recommendation\_Evaluation # check recommendation information

## Recommender of type 'IBCF' for 'realRatingMatrix'   
## learned using 264 users.

class(Model\_Recommendation\_Evaluation) # check class of recommendation model

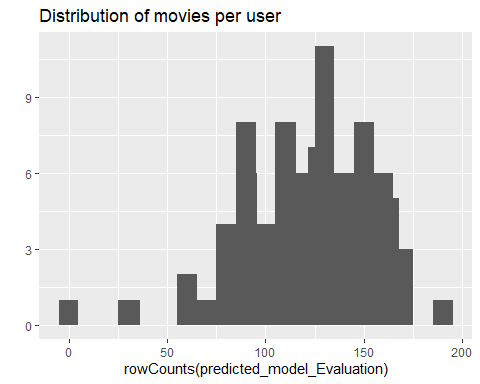
## [1] "Recommender"  
## attr(,"package")  
## [1] "recommenderlab"

no\_recommendedMovies <- 10 # max 10 movies recommend to each user  
predicted\_model\_Evaluation <- predict(object = Model\_Recommendation\_Evaluation,  
 newdata = getData(evaluation\_Matrix, "known"),  
 type = "ratings",  
 n = no\_recommendedMovies) # apply predication  
predicted\_model\_Evaluation # check predication values

## 70 x 283 rating matrix of class 'realRatingMatrix' with 8482 ratings.

qplot(rowCounts(predicted\_model\_Evaluation)) +  
 geom\_histogram(binwidth = 10) +   
 ggtitle("Distribution of movies per user") # movie per user in predicted ratings

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



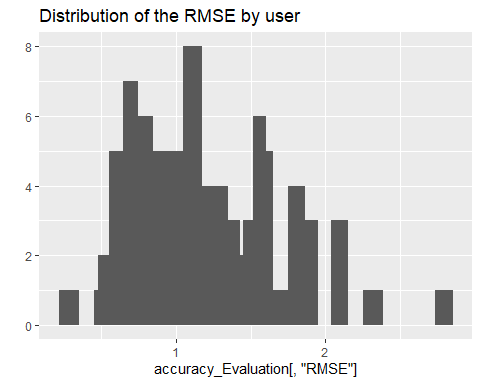
accuracy\_Evaluation <- calcPredictionAccuracy(x = predicted\_model\_Evaluation,   
 data = getData(evaluation\_Matrix, "unknown"),   
 byUser = TRUE) # apply rating accuracy for each user, RMSE should be lower which mean that model is better fit towards prediction values  
  
head(accuracy\_Evaluation) # check the row

## RMSE MSE MAE  
## [1,] 2.0514223 4.2083333 1.4166667  
## [2,] 1.1244004 1.2642764 0.9224619  
## [3,] 1.5752807 2.4815092 1.2098766  
## [4,] 0.7693730 0.5919348 0.6440562  
## [5,] 0.7471777 0.5582745 0.5389679  
## [6,] 2.1181781 4.4866787 1.8374144

qplot(accuracy\_Evaluation[, "RMSE"]) +   
 geom\_histogram(binwidth = 0.1) +  
 ggtitle("Distribution of the RMSE by user") # Root mean square erros

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 1 rows containing non-finite values (stat\_bin).  
  
## Warning: Removed 1 rows containing non-finite values (stat\_bin).



accuracy\_Evaluation <- calcPredictionAccuracy(x = predicted\_model\_Evaluation,   
 data = getData(evaluation\_Matrix, "unknown"),   
 byUser = FALSE) # apply rating accuracy for each user, RMSE should be lower which mean that model is better fit towards prediction values  
head(accuracy\_Evaluation) # check accuracy of whole model

## RMSE MSE MAE   
## 1.3005508 1.6914324 0.9683621

# Evaluating the recommendations

# another way to measure the accuracy is positive rating.  
# and set the N (# of items) in parameters to evaluate performance  
  
Model\_results <- evaluate(x = evaluation\_Matrix,   
 method = evaluate\_model,   
 n = seq(10, 100, 10))

## IBCF run fold/sample [model time/prediction time]  
## 1 [0.16sec/0.01sec]   
## 2 [0.18sec/0.04sec]   
## 3 [0.11sec/0.01sec]   
## 4 [0.17sec/0.05sec]   
## 5 [0.18sec/0.04sec]

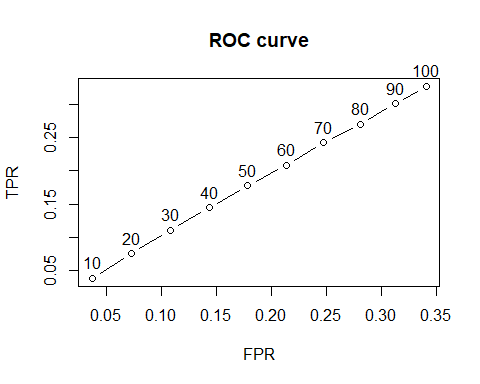
head(getConfusionMatrix(Model\_results)[[1]]) # get confusion Matrix

## TP FP FN TN precision recall TPR  
## 10 2.185714 7.671429 57.60000 209.5429 0.2217391 0.04009824 0.04009824  
## 20 3.957143 15.757143 55.82857 201.4571 0.2007246 0.06982816 0.06982816  
## 30 6.214286 23.357143 53.57143 193.8571 0.2101449 0.10704157 0.10704157  
## 40 8.357143 30.928571 51.42857 186.2857 0.2138889 0.14353582 0.14353582  
## 50 10.414286 38.585714 49.37143 178.6286 0.2143961 0.17680596 0.17680596  
## 60 12.028571 46.657143 47.75714 170.5571 0.2072797 0.20218788 0.20218788  
## FPR  
## 10 0.03575769  
## 20 0.07355627  
## 30 0.10827639  
## 40 0.14317014  
## 50 0.17813916  
## 60 0.21539312

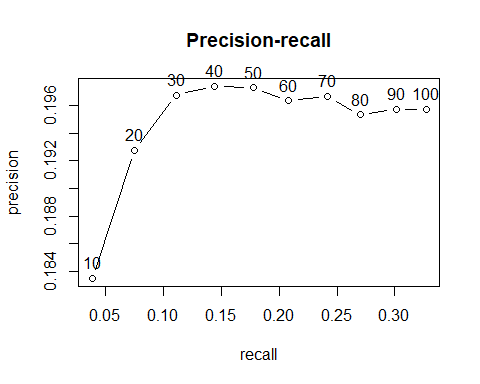
columns\_to\_sum <- c("TP", "FP", "FN", "TN") # column name assigned  
indices\_summed <- Reduce("+", getConfusionMatrix(Model\_results))[, columns\_to\_sum]  
head(indices\_summed)

## TP FP FN TN  
## 10 9.014286 40.12857 270.1571 1065.7000  
## 20 18.942857 79.34286 260.2286 1026.4857  
## 30 28.985714 118.32857 250.1857 987.5000  
## 40 38.657143 157.35714 240.5143 948.4714  
## 50 48.214286 196.34286 230.9571 909.4857  
## 60 57.400000 235.11429 221.7714 870.7143

plot(Model\_results, annotate = TRUE, main = "ROC curve") # Plot ROC graph



plot(Model\_results, "prec/rec", annotate = TRUE, main = "Precision-recall") # checking precision /recall curves [small % of movie recommended then precision is lower and higher % of rated movies recommended than recall is higher]



# Comparing all recommendation models —-

# pick all recommendation model and then set N = # of recommend movies in sequence (10, 100, 10)  
evaluate\_model\_Complex <- list(  
IBCF\_cos = list(name = "IBCF",   
 param = list(method = "cosine")), # using the Cosine as the distance function  
IBCF\_Pearson = list(name = "IBCF",   
 param = list(method = "pearson")), # using the Pearson correlation as the distance functio  
UBCF\_cos = list(name = "UBCF",   
 param = list(method = "cosine")), # using the Cosine as the distance function  
UBCF\_Person = list(name = "UBCF",   
 param = list(method = "pearson")), # using the Pearson correlation as the distance functio  
random = list(name = "RANDOM", param=NULL) #Random recommendations to have a base line  
)  
  
  
noof\_recommendations <- c(1, 5, seq(10, 100, 10))  
  
list\_results <- evaluate(x = evaluation\_Matrix,   
 method = evaluate\_model\_Complex,   
 n = noof\_recommendations) # compare all model

## IBCF run fold/sample [model time/prediction time]  
## 1 [0.17sec/0.02sec]   
## 2 [0.17sec/0.05sec]   
## 3 [0.11sec/0.01sec]   
## 4 [0.1sec/0.02sec]   
## 5 [0.11sec/0.03sec]   
## IBCF run fold/sample [model time/prediction time]  
## 1 [0.14sec/0.03sec]   
## 2 [0.14sec/0.02sec]   
## 3 [0.14sec/0.01sec]   
## 4 [0.14sec/0.05sec]   
## 5 [0.23sec/0.03sec]   
## UBCF run fold/sample [model time/prediction time]  
## 1 [0sec/0.1sec]   
## 2 [0sec/0.08sec]   
## 3 [0sec/0.17sec]   
## 4 [0.02sec/0.17sec]   
## 5 [0sec/0.2sec]   
## UBCF run fold/sample [model time/prediction time]  
## 1

## Timing stopped at: 0.05 0 0.05

## Error in neighbors[, x] : incorrect number of dimensions  
## RANDOM run fold/sample [model time/prediction time]  
## 1 [0sec/0.04sec]   
## 2 [0sec/0.03sec]   
## 3 [0sec/0.03sec]   
## 4 [0sec/0.05sec]   
## 5 [0sec/0.03sec]

## Warning in .local(x, method, ...):   
## Recommender 'UBCF\_Person' has failed and has been removed from the results!

sapply(list\_results, class) == "evaluationResults"

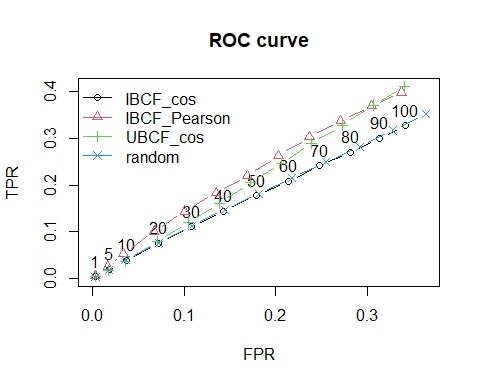
## IBCF\_cos IBCF\_Pearson UBCF\_cos random   
## TRUE TRUE TRUE TRUE

avg\_matrices <- lapply(list\_results, avg)  
head(avg\_matrices$IBCF\_cos[, 5:8])

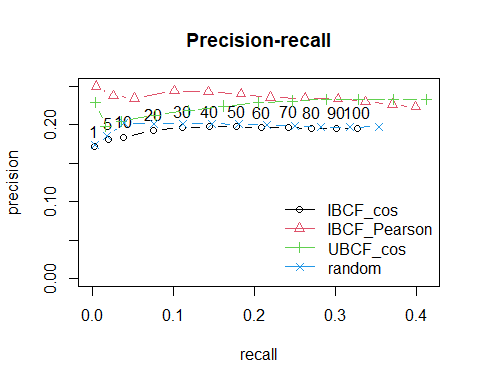
## precision recall TPR FPR  
## 1 0.1716234 0.003523115 0.003523115 0.003737755  
## 5 0.1808910 0.019946014 0.019946014 0.018644773  
## 10 0.1834784 0.038384059 0.038384059 0.036994671  
## 20 0.1927600 0.075296582 0.075296582 0.072754739  
## 30 0.1967815 0.111026078 0.111026078 0.107972428  
## 40 0.1974086 0.144504387 0.144504387 0.143338987

# Identifying the most suitable model —-

# compare all model graph by displaying ROC and Precision/Recall curves  
# chart shows highest ROC is IBCF with Pearson technique.   
  
plot(list\_results, annotate = 1, legend = "topleft")   
title("ROC curve")



plot(list\_results, "prec/rec", annotate = 1, legend = "bottomright")  
title("Precision-recall")

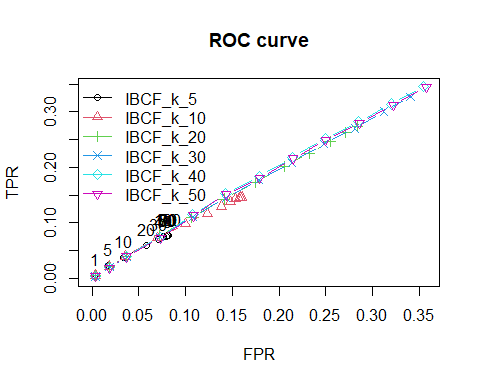


# Optimizing a numeric parameter —-

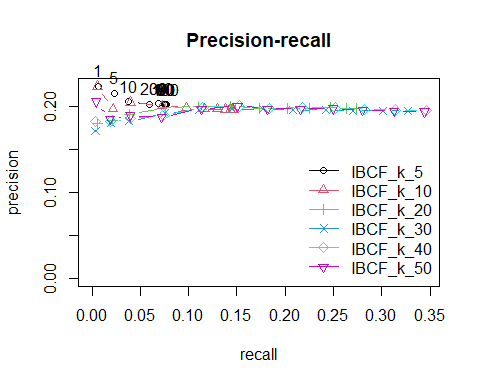
# for optimizing values, set the K-closes items range from 5 and 50  
  
K\_closestItems <- c(5, 10, 20, 30, 40, 50)  
  
models\_to\_evaluate <- lapply(K\_closestItems, function(k){  
 list(name = "IBCF",  
 param = list(method = "cosine", k = k))  
})  
names(models\_to\_evaluate) <- paste0("IBCF\_k\_", K\_closestItems)  
  
  
  
noof\_recommendations <- c(1, 5, seq(10, 100, 10))  
  
list\_results <- evaluate(x = evaluation\_Matrix,   
 method = models\_to\_evaluate,   
 n = noof\_recommendations) # compare all model

## IBCF run fold/sample [model time/prediction time]  
## 1 [0.21sec/0.03sec]   
## 2 [0.14sec/0.01sec]   
## 3 [0.18sec/0.03sec]   
## 4 [0.09sec/0.02sec]   
## 5 [0.15sec/0.03sec]   
## IBCF run fold/sample [model time/prediction time]  
## 1 [0.14sec/0.01sec]   
## 2 [0.17sec/0.01sec]   
## 3 [0.12sec/0.02sec]   
## 4 [0.11sec/0.01sec]   
## 5 [0.11sec/0.01sec]   
## IBCF run fold/sample [model time/prediction time]  
## 1 [0.14sec/0.03sec]   
## 2 [0.19sec/0.04sec]   
## 3 [0.18sec/0.03sec]   
## 4 [0.18sec/0.04sec]   
## 5 [0.15sec/0.02sec]   
## IBCF run fold/sample [model time/prediction time]  
## 1 [0.17sec/0.03sec]   
## 2 [0.17sec/0.03sec]   
## 3 [0.17sec/0.03sec]   
## 4 [0.19sec/0.03sec]   
## 5 [0.19sec/0.03sec]   
## IBCF run fold/sample [model time/prediction time]  
## 1 [0.19sec/0.03sec]   
## 2 [0.16sec/0.03sec]   
## 3 [0.14sec/0.03sec]   
## 4 [0.17sec/0.03sec]   
## 5 [0.19sec/0.03sec]   
## IBCF run fold/sample [model time/prediction time]  
## 1 [0.19sec/0.03sec]   
## 2 [0.17sec/0.05sec]   
## 3 [0.19sec/0.03sec]   
## 4 [0.21sec/0.03sec]   
## 5 [0.19sec/0.01sec]

plot(list\_results, annotate = 1, legend = "topleft")   
title("ROC curve")



plot(list\_results, "prec/rec", annotate = 1, legend = "bottomright")  
title("Precision-recall")



# Based on the ROC curve's plot, the k having the biggest AUC is 50  
#items which recomend to user as compared to precision/recalls same 50 items are so high for recalls  
 #precision/recall