ashishsa_hw5_p1

We are going to use User Based Collaborative Filtering to filter the Users in the Movie Lense Data which is available in the Recommender Package.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(pander)
```

```
## Warning: package 'pander' was built under R version 3.6.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
library(recommenderlab)
```

```
## Warning: package 'recommenderlab' was built under R version 3.6.3
```

```
## Loading required package: Matrix
```

```
## Loading required package: arules
```

```
## Warning: package 'arules' was built under R version 3.6.3
```

```
##
## Attaching package: 'arules'
```

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

```
## Loading required package: proxy
```

```
## Warning: package 'proxy' was built under R version 3.6.3
```

```
##
## Attaching package: 'proxy'
```

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

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

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

```
## Loading required package: registry
```

```
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```
##
## Attaching package: 'recommenderlab'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
data("MovieLense")
d <- MovieLense
```

We now check the number of rows in the data.

```
nrow(d)
```

```
## [1] 943
```

We now check the number of columns in the data.

```
ncol(d)
```

```
## [1] 1664
```

We now take a look at the list of the names of the Movies in the data.

```
v1 <- as.vector(names(colCounts(MovieLense)))
head(v1)
```

```
## [1] "Toy Story (1995)"
## [2] "GoldenEye (1995)"
## [3] "Four Rooms (1995)"
## [4] "Get Shorty (1995)"
## [5] "Copycat (1995)"
## [6] "Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)"
```

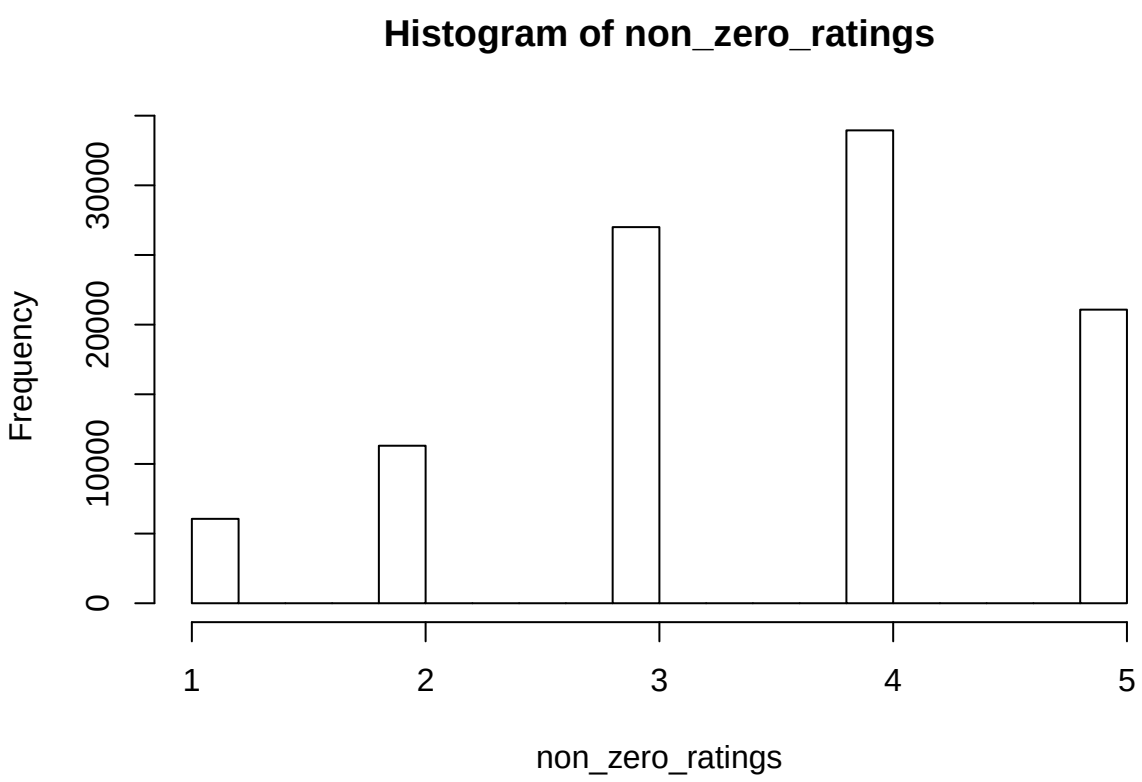
We now make a list and frequency of the ratings to analyze the number of ratings. Since there are 1664 movies and 943 users there should be a maximum of 1,569,152 ratings.

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

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

We observe from the above table that there are 1469760 zero ratings which means that there is only 37 Percent of the Ratings available. While there is 63 Percent of the Ratings Missing. We plot an Histogram to check the frequency and the number of ratings.

```
non_zero_ratings <- ratings[ratings!=0]
hist(non_zero_ratings)
```



We observe that most of the users have rated the movie between 3 and 5. With the maximum users giving a rating of 4 to the movie.

We now create a Ratings Matrix from the MovieLense data by the following way.

```
R <- as(MovieLense@data, "realRatingMatrix")
l <- (getRatingMatrix(R)[1:10, 1:10])
l
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 10 column names 'Toy Story (1995)', 'GoldenEye (1995)', 'Four Rooms (1995)' ... ]]
##
## 1  5 3 4 3 3 5 4 1 5 3
## 2  4 . . . . . . . . 2
## 3  . . . . . . . . . .
## 4  . . . . . . . . . .
## 5  4 3 . . . . . . . .
## 6  4 . . . . . 2 4 4 .
## 7  . . . 5 . . 5 5 5 4
## 8  . . . . . . 3 . . .
## 9  . . . . . 5 4 . . .
## 10 4 . . 4 . . 4 . 4 .
```

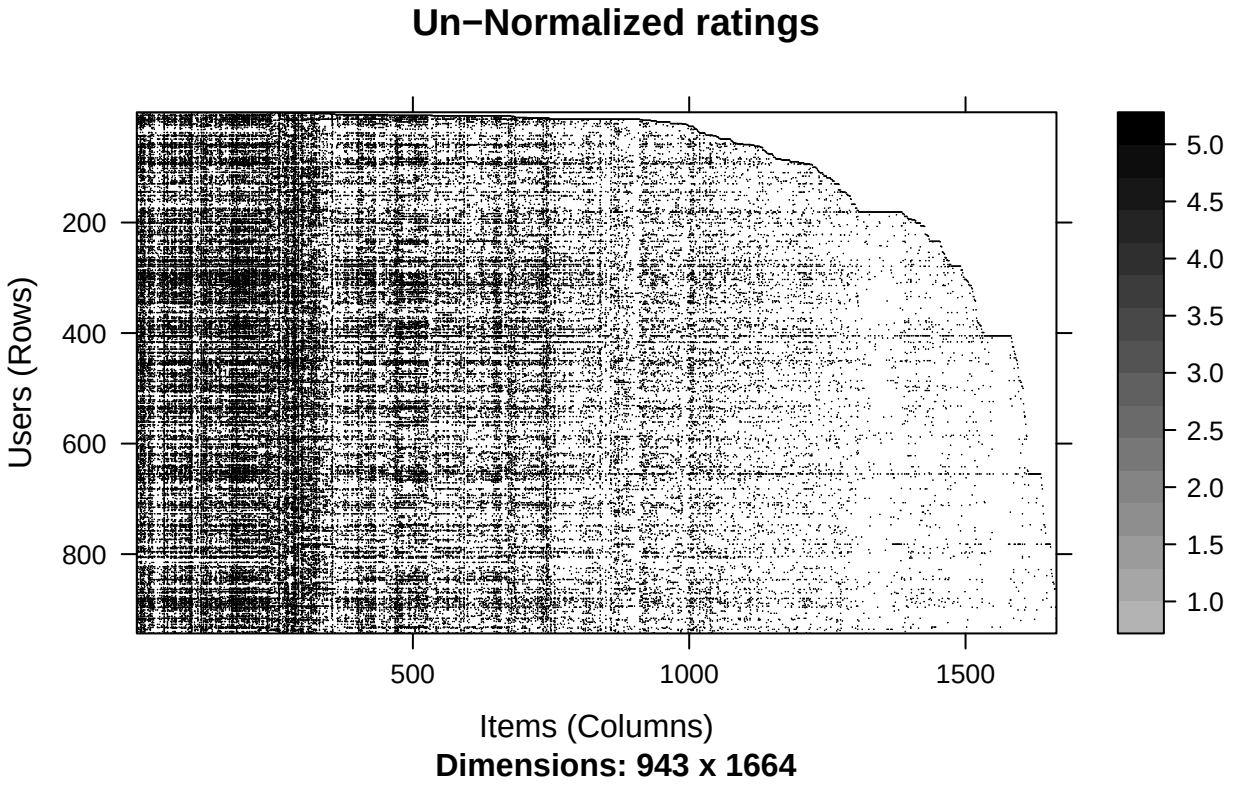
We now Normalize the Rating to remove the user bias. Since every user might overrateor underrate a movie based on his/her preferance.

```r
R_Normalize <- normalize(R)
getRatingMatrix(R_Normalize)[1:10,1:10]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 10 column names 'Toy Story (1995)', 'GoldenEye (1995)', 'Four Rooms (1995)' ... ]]
##
## 1   1.3948339 -0.6051661 0.3948339 -0.6051661 -0.6051661 1.3948339  0.3948339
## 2   0.2950820  .         .         .          .         .          .
## 3   .          .         .         .          .         .          .
## 4   .          .         .         .          .         .          .
## 5   1.1257143  0.1257143 .         .          .         .          .
## 6   0.3605769  .         .         .          .         .         -1.6394231
## 7   .          .         .         1.0350000  .         .          1.0350000
## 8   .          .         .         .          .         .         -0.7966102
## 9   .          .         .         .          .         0.7272727 -0.2727273
## 10 -0.2065217  .         .        -0.2065217  .         .         -0.2065217
##
## 1  -2.6051661  1.3948339 -0.6051661
## 2   .          .         -1.7049180
## 3   .          .          .
## 4   .          .          .
## 5   .          .          .
## 6   0.3605769  0.3605769  .
## 7   1.0350000  1.0350000  0.0350000
## 8   .          .          .
## 9   .          .          .
## 10  .         -0.2065217  .
```

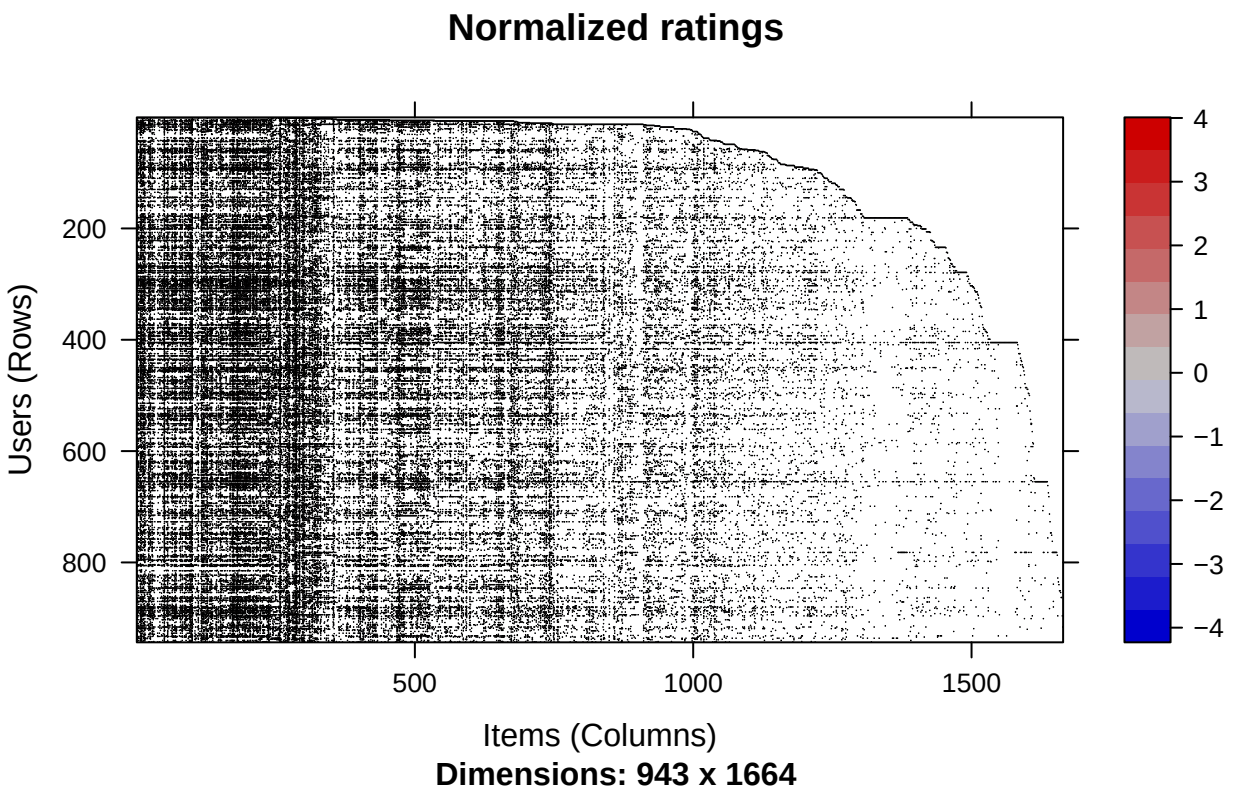
We now plot the image to observe the Un-Normalized ratings data.

```r
image(R, main = "Un-Normalized ratings")
```



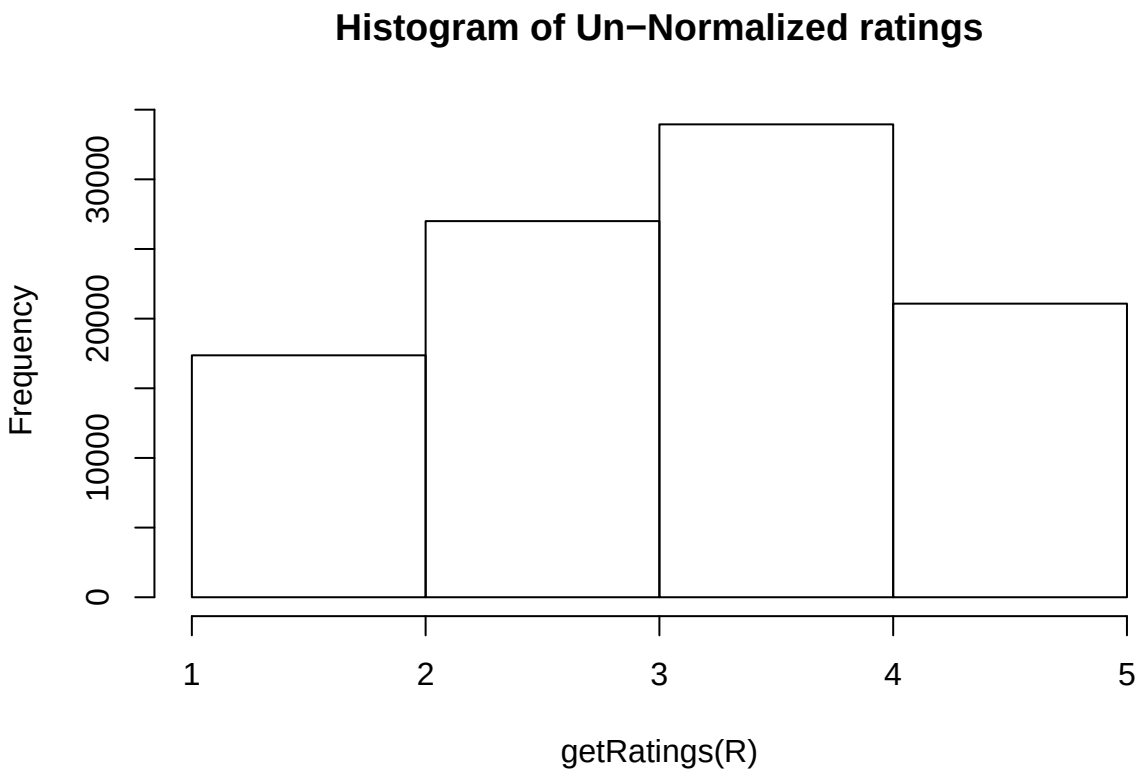
The user ratings lie between 1-5. We now plot the image to observe the Normalized ratings data.

```
image(R_Normalize, main = "Normalized ratings")
```



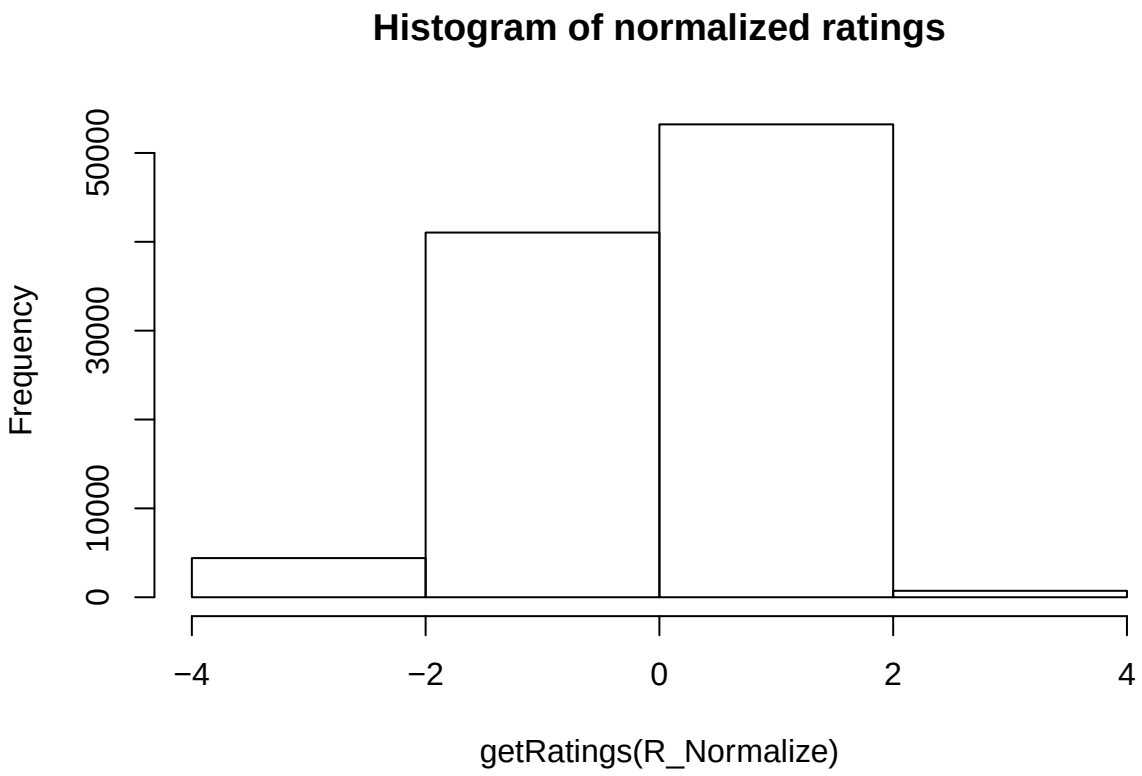
After Normalizing the ratings now lie between -4 to 4. This is done to remove any Bias in the data. We now plot the histogram to observe the Un-Normalized ratings data.

```
hist(getRatings(R), breaks = 5, main = "Histogram of Un-Normalized ratings")
```



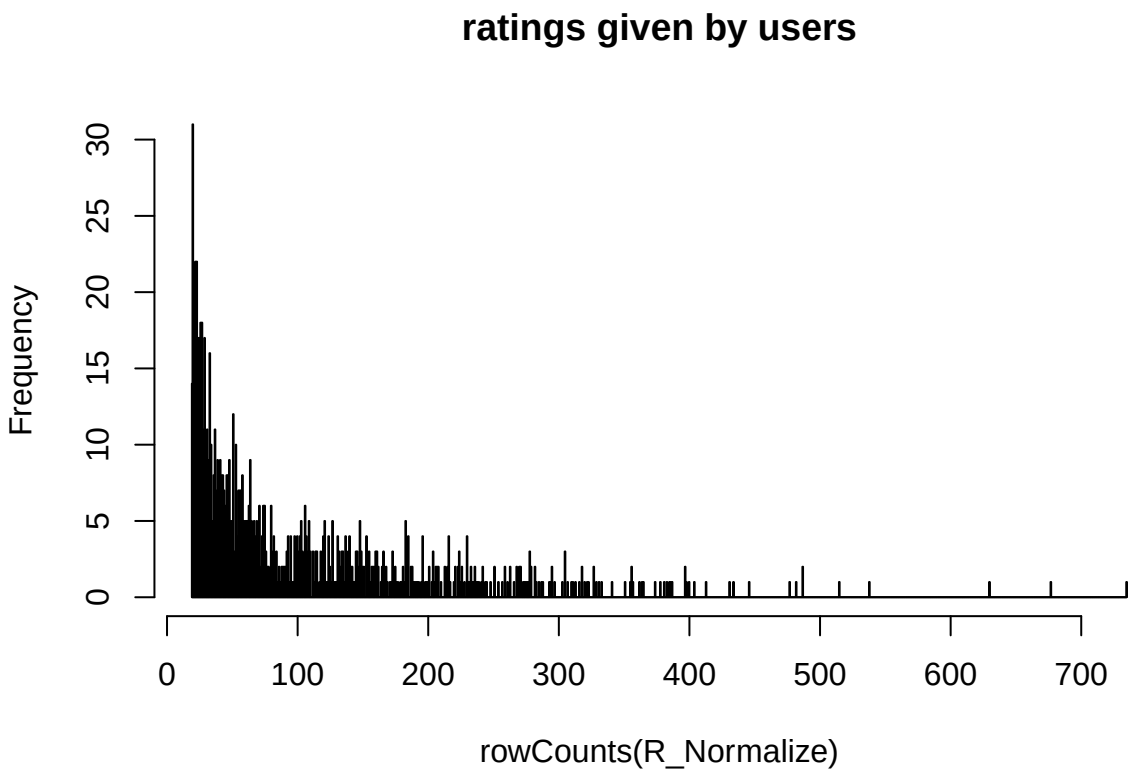
We now plot the histogram to observe the Normalized ratings data.

```
hist(getRatings(R_Normalize), breaks = 5, main = "Histogram of normalized ratings")
```



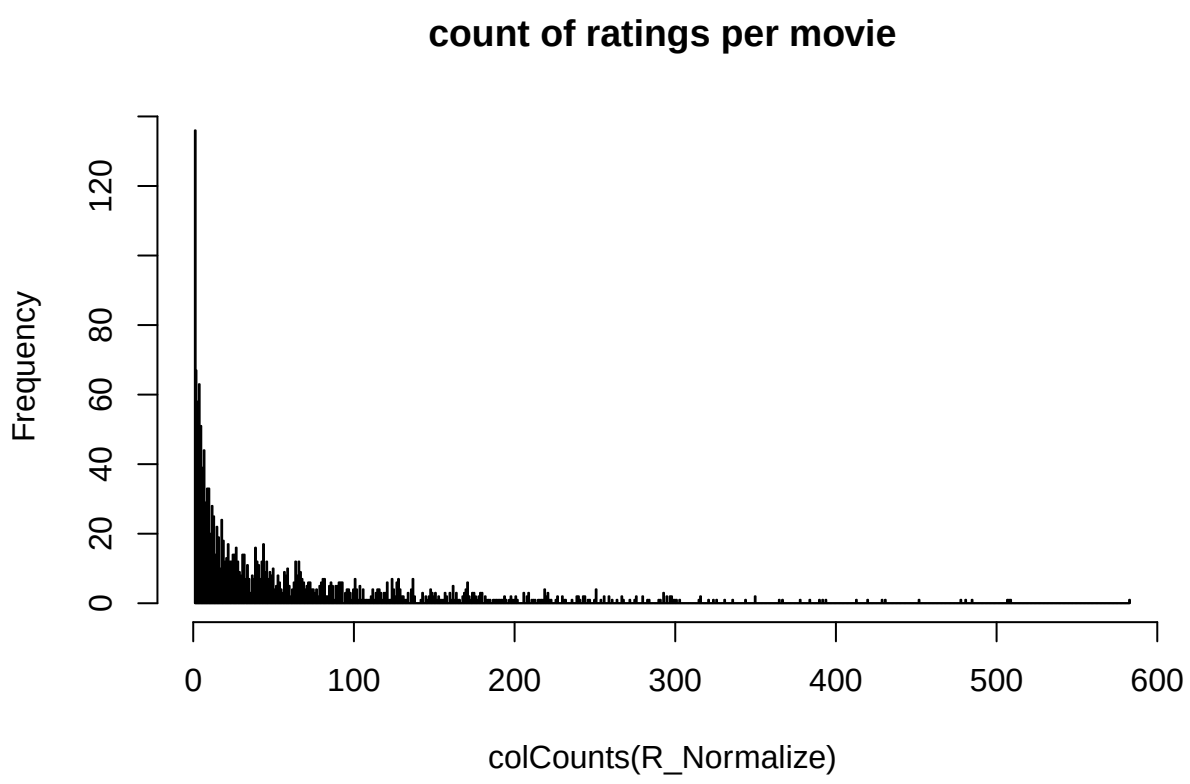
After Normalizing the data the Ratings that existed between 0-5 now lie in the range -4 to +4. Mostof the ratings are in the range -2 to +2.

```
hist(rowCounts(R_Normalize), breaks = 1664, main = "ratings given by users")
```



There are very few users who have given upards of 30 Ratings.

```
hist(colCounts(R_Normalize), breaks = 934, main = "count of ratings per movie")
```



We now get the Movie Lense data Metadata which divides the movies into various Genres.

```
moviemeta <- MovieLenseMeta
head(moviemeta)
```

```
##                                                 title year
## 1                                    Toy Story (1995) 1995
## 2                                    GoldenEye (1995) 1995
## 3                                   Four Rooms (1995) 1995
## 4                                   Get Shorty (1995) 1995
## 5                                      Copycat (1995) 1995
## 6 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995) 1995
##                                                          url unknown Action
## 1        http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)       0      0
## 2          http://us.imdb.com/M/title-exact?GoldenEye%20(1995)       0      1
## 3       http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)       0      0
## 4       http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)       0      1
## 5            http://us.imdb.com/M/title-exact?Copycat%20(1995)       0      0
## 6 http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)       0      0
##   Adventure Animation Children's Comedy Crime Documentary Drama Fantasy
## 1         0         1          1      1     0           0     0       0
## 2         1         0          0      0     0           0     0       0
## 3         0         0          0      0     0           0     0       0
## 4         0         0          0      1     0           0     1       0
## 5         0         0          0      0     1           0     1       0
## 6         0         0          0      0     0           0     1       0
##   Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western
```

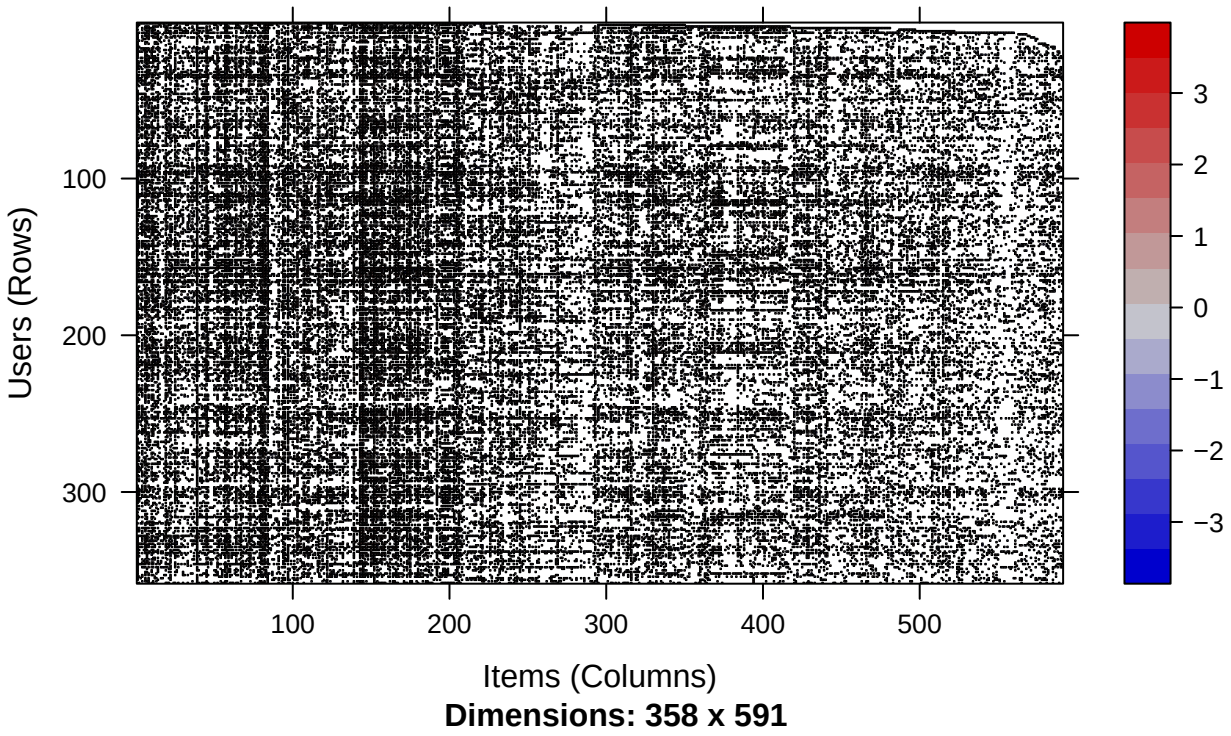
```
## 1          0       0       0       0       0       0       0    0       0
## 2          0       0       0       0       0       0       1    0       0
## 3          0       0       0       0       0       0       1    0       0
## 4          0       0       0       0       0       0       0    0       0
## 5          0       0       0       0       0       0       1    0       0
## 6          0       0       0       0       0       0       0    0       0
```

Since this is a large dataset that contains 945 Users and 1664 Movies, there might be users that have hardly rated any movies.

In order to improve the ratings performance in the User Based Filtering, we divide the dataset into a set that contains the top 50 most rated movies and users that have rated upwards of 100 movies. As the number of movies rated increases, it will increase the probability of prediction of users that have similar tastes.

```
v2 <- R_Normalize[rowCounts(R_Normalize) > 100, colCounts(R_Normalize) > 50]

image(v2)
```



From the image we can observe that most of the users have rated the movies. Unlike the Entire Data which showed a huge Empty space attributing to users with very low or no ratings.

We now divide the dataset into train and test datasets

```
R_denormalize <- denormalize(R_Normalize)
which_set <- sample(x = 1:5, size = nrow(R_denormalize), replace = TRUE)
for(i_model in 1:5)
{
  which_train <- which_set == i_model
  data_test <- R_denormalize[which_train, ]
```

```
  data_train <- R_denormalize[!which_train, ]
}
```

We now apply the Recommender System on the subset dataset containing top 50 Most Rated Movies and a user-list containing top 100 Most Rated Users

```
recommender_popularity <- Recommender(data_train, method = "UBCF",parameter = list(method = "Cosine"))
recommender_popularity
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 763 users.
```

We now make a list of top 20 movies to most recommended users

```
number_recommendation <- 20
recommender_prediction1 <- predict(recommender_popularity,newdata=data_test,n=number_recommendation)
recommender_prediction1
```

```
## Recommendations as 'topNList' with n = 20 for 180 users.
```

```
recdfub <- data.frame(user = sort(rep(1:length(recommender_prediction1@items), recommender_prediction1@
    rating = unlist(recommender_prediction1@ratings), index = unlist(recommender_prediction1@items))
head(recdfub)
```

```
##   user   rating index
## 1    1 3.898947   311
## 2    1 3.875260   285
## 3    1 3.872910   299
## 4    1 3.807502   345
## 5    1 3.785457   308
## 6    1 3.758882   894
```

The User Based Recommendation System uses the data available in the Train and Test Dataset to recommend the ratings of the movie not seen by using the top nearest users and recommend 20 movies and rate it by using the ratings of the nearest users.