Build a recommender system with the given data using UBCF.

Description of the data

In this dataset have users on the rows rated the jokes in the columns. The data is formated as an excel file representing a 66336 x 151 matrix with rows as users and columns as jokes.

Each rating is from (-10.00 to +10.00) and 99 corresponds to a null rating (user did not rate that joke).

Note that the ratings are real values ranging from -10.00 to +10.00.

##Recommendtaion system for the jokes rating

##importing the data set

joke <- read\_excel(file.choose())

summary(joke)

#The data set consists of the ratings from the people on the jokes scaling from -10 to 10 and 99 null rating the user did not rate the joke

str(joke)

##varibles are no.of the jokes and observation aere the rating given by the people

dim(joke)

## the data set as 151 variables and 50691 observation in it some minimizing the data set to 100 observation

trows <- sample(nrow(joke),100)

jokenew <- joke[trows,]

##removing the first column in the data set

jokenew <-jokenew[,-1]

jokenew[,][jokenew[,]==99] <- NA##Here we are giving the 99 value to na because the implies that people did not rate the joke

min(jokenew[][],na.rm = TRUE)## minimum rating is -10 in the data set

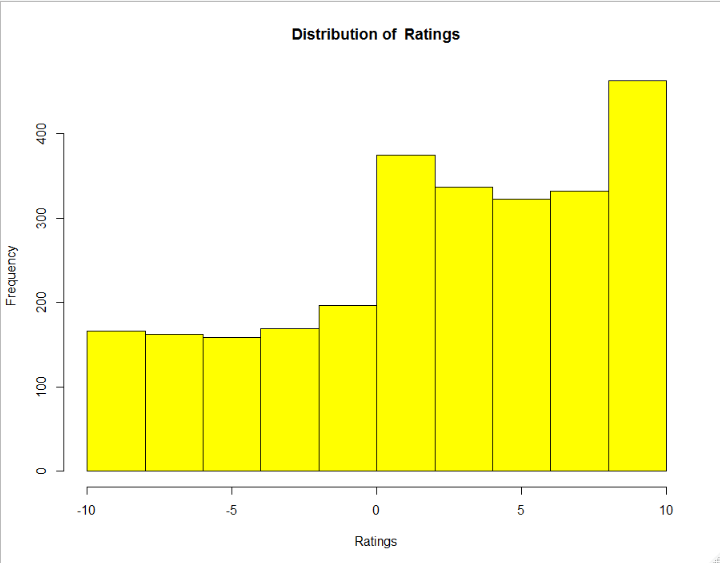
max(jokenew[][],na.rm = TRUE)## maximum rating is 10 in the data set

##vizualization

hist(as.vector(as.matrix(jokenew)), main = "Distribution of Ratings",

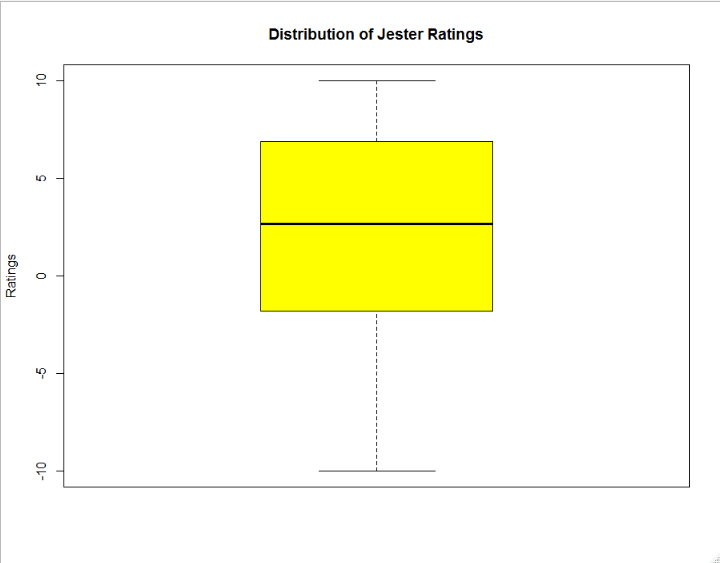
col = "yellow", xlab = "Ratings")

##it is like left skewed curve most of the rating lie between the 0-5 rating



boxplot(as.vector(as.matrix(jokenew)), col = "yellow", main = "Distribution of Jester Ratings", ylab = "Ratings")

##the distribution is some what left skewed



## model building for given data set

##train UBCF cosine similarity models

# non-normalized

UBCF\_N\_C <- Recommender(getData(e, "train"), "UBCF",

param=list(normalize = NULL, method="Cosine"))

# centered

UBCF\_C\_C <- Recommender(getData(e, "train"), "UBCF",

param=list(normalize = "center",method="Cosine"))

# Z-score normalization

UBCF\_Z\_C <- Recommender(getData(e, "train"), "UBCF",

param=list(normalize = "Z-score",method="Cosine"))

##predciting the model

# compute predicted ratings

p1 <- predict(UBCF\_N\_C, getData(e, "known"), type="ratings")

p2 <- predict(UBCF\_C\_C, getData(e, "known"), type="ratings")

p3 <- predict(UBCF\_Z\_C, getData(e, "known"), type="ratings")

# set all predictions that fall outside the valid range to the boundary values

p1@data@x[p1@data@x[] < -10] <- -10

p1@data@x[p1@data@x[] > 10] <- 10

p2@data@x[p2@data@x[] < -10] <- -10

p2@data@x[p2@data@x[] > 10] <- 10

p3@data@x[p3@data@x[] < -10] <- -10

p3@data@x[p3@data@x[] > 10] <- 10

# aggregate the performance statistics

error\_UCOS <- rbind(

UBCF\_N\_C = calcPredictionAccuracy(p1, getData(e, "unknown")),

UBCF\_C\_C = calcPredictionAccuracy(p2, getData(e, "unknown")),

UBCF\_Z\_C = calcPredictionAccuracy(p3, getData(e, "unknown"))

)

kable(error\_UCOS)

| | RMSE| MSE| MAE|

|:--------|--------:|--------:|--------:|

|UBCF\_N\_C | 5.795662| 33.58969| 4.917743|

|UBCF\_C\_C | 4.377341| 19.16112| 2.945781|

|UBCF\_Z\_C | 4.442507| 19.73587| 2.893614|

## seeing the above table we can say that user base collaborative filtering with z-score normalization has small error

#train UBCF Euclidean Distance models

# non-normalized

UBCF\_N\_E <- Recommender(getData(e, "train"), "UBCF",

param=list(normalize = NULL, method="Euclidean"))

# centered

UBCF\_C\_E <- Recommender(getData(e, "train"), "UBCF",

param=list(normalize = "center",method="Euclidean"))

# Z-score normalization

UBCF\_Z\_E <- Recommender(getData(e, "train"), "UBCF",

param=list(normalize = "Z-score",method="Euclidean"))

# compute predicted ratings

p1 <- predict(UBCF\_N\_E, getData(e, "known"), type="ratings")

p2 <- predict(UBCF\_C\_E, getData(e, "known"), type="ratings")

p3 <- predict(UBCF\_Z\_E, getData(e, "known"), type="ratings")

# set all predictions that fall outside the valid range to the boundary values

p1@data@x[p1@data@x[] < -10] <- -10

p1@data@x[p1@data@x[] > 10] <- 10

p2@data@x[p2@data@x[] < -10] <- -10

p2@data@x[p2@data@x[] > 10] <- 10

p3@data@x[p3@data@x[] < -10] <- -10

p3@data@x[p3@data@x[] > 10] <- 10

# aggregate the performance statistics

error\_UEUC <- rbind(

UBCF\_N\_E = calcPredictionAccuracy(p1, getData(e, "unknown")),

UBCF\_C\_E = calcPredictionAccuracy(p2, getData(e, "unknown")),

UBCF\_Z\_E = calcPredictionAccuracy(p3, getData(e, "unknown"))

)

kable(error\_UEUC)

|  |
| --- |
| | | RMSE| MSE| MAE|  |:--------|--------:|--------:|--------:|  |UBCF\_N\_E | 5.480121| 30.03172| 4.575737|  |UBCF\_C\_E | 4.372632| 19.11991| 2.877288|  |UBCF\_Z\_E | 4.416588| 19.50625| 2.849098| |
|  |
| |  | | --- | | > | |

## From the above table we can say that UBCF with z normalization has least error score in distance method