DATA 612 Project 2: Content Base and Collaborative Filtering

Albert Gilharry

June 12, 2019

```r
library(dplyr)
library(ggplot2)
library(recommenderlab)
```

Data

I opted to use the Jester data set 1 from [http://eigentaste.berkeley.edu/dataset/] for this project. The data set contains anonymous ratings from 24,983 users who have rated 36 or more jokes. The Ratings are real values ranging from -10.00 to +10.00.

```r
# set seed for reproducibility
set.seed(100)
# load data
data <- read.csv("data/jester-data-1.csv", header = FALSE)

# label items and ignore the first column
item_names <- paste0("Joke", seq(1,100))
ratings <-  dplyr::select(data, -1)
names(ratings ) <- item_names

# represent missing values as NA
ratings[ratings == 99] <- NA
ratings_matrix <- as.matrix(ratings)

# create realRatingMatrix for input to recommenderlab
sparse_smatrix <- as(ratings_matrix, "sparseMatrix")
real_matrix <- as(sparse_smatrix, "realRatingMatrix")
real_matrix
```

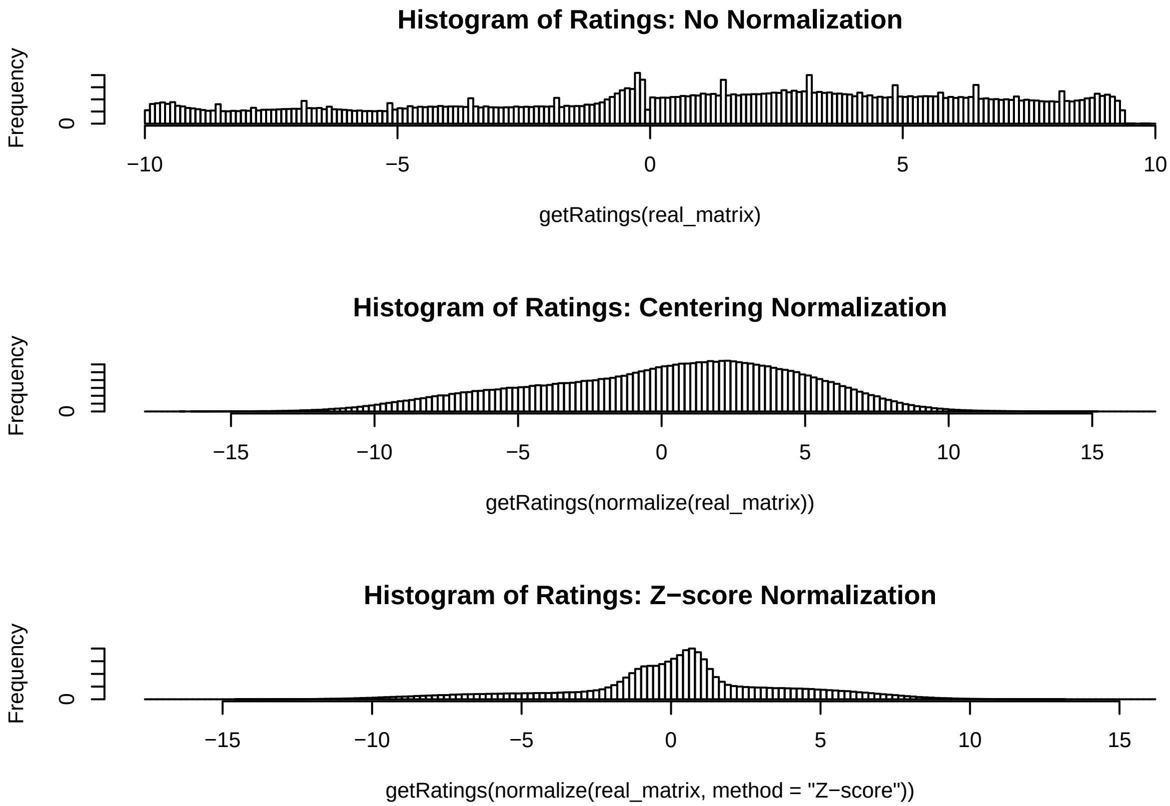
```
## 24983 x 100 rating matrix of class 'realRatingMatrix' with 2492917 ratings.
```

The output from the code below shows that data has almost 2.5 million ratings. Let's explore these ratings.

Data Exploration

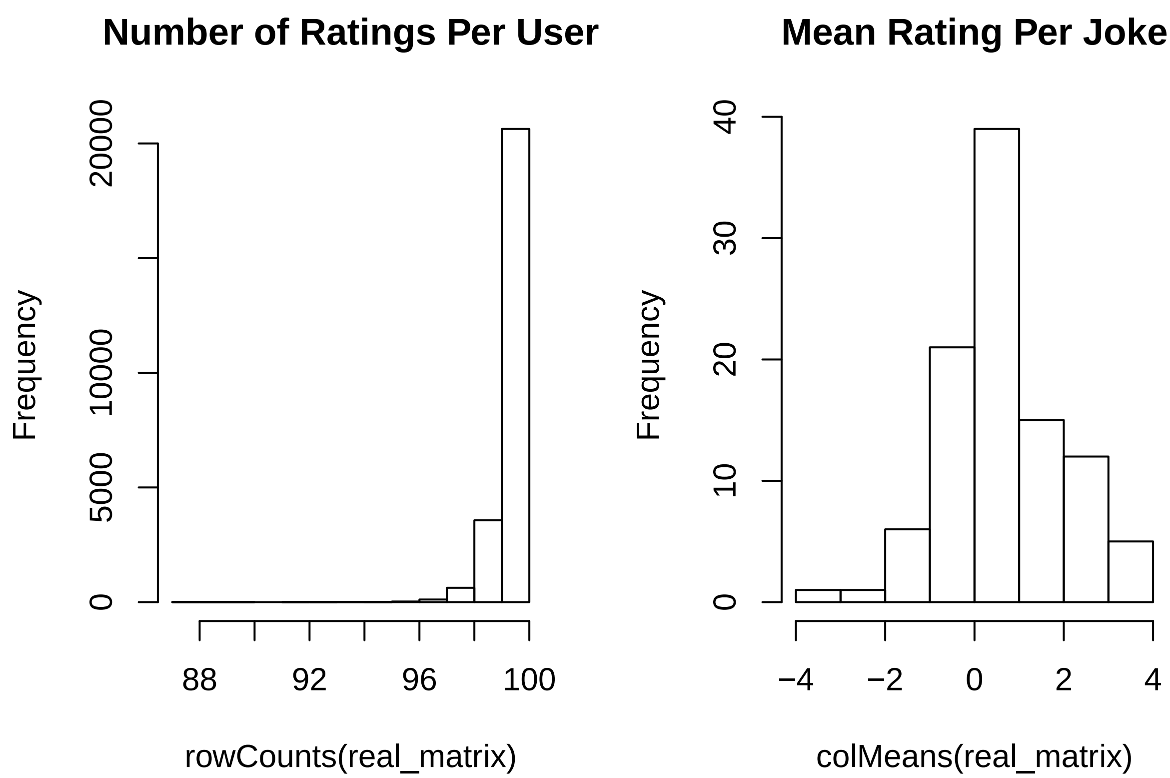
The histogram of the un-normalized ratings shows a near uniform distribution when compared to its normalized versions in the histograms below that both resemble a near normal distribution. This means that the data may benefit from normalization before used for modeling. The `recommenderlab` package has parameters for normalization that will be utilized when training the models in this project.

```r
par(mfrow=c(3,1))
hist(getRatings(real_matrix), breaks=200, main="Histogram of Ratings: No Normalization")
hist(getRatings(normalize(real_matrix)), breaks=200, main="Histogram of Ratings: Centering Normalizatio
hist(getRatings(normalize(real_matrix, method="Z-score")), breaks=200, main="Histogram of Ratings: Z-sc
```



The histogram to the left below shows the number of ratings per user. It reveals that most users rated majority of the jokes making this data set almost complete and not very sparse. The histogram to the right shows that on average users are neutral regarding these jokes but they do not provide as much negative ratings as positive ones.

```
par(mfrow=c(1,2))
hist(rowCounts(real_matrix), main = "Number of Ratings Per User")
hist(colMeans(real_matrix), main = "Mean Rating Per Joke")
```



Model Evaluation

Models will be created using both item-based collaborative filtering and user-based collaborative filtering using the `recommenderlab` package. 10-Fold cross-validation will used to train and evaluation the models. 85% of the data will be used for trainig the models and the remaining 15% will be used as the testing set on which the RMSE will be calculated. The main performance metric will be the RMSE but the MAE will also be looked at. A rating of 5 will be deemed as a good rating for the purposes of this project. For each test set user, all but 20 randomly selected ratings will be withheld for downstream evaluation.

```
# initialize the evaluation scheme
eval_scheme <- evaluationScheme(real_matrix, method="cross-validation", train=0.85, k=10, goodRating=5,
```

Item Based Collaborative Filtering

Item-based collaborative filtering will be implemented in this section. The main idea behind item based collaborative filtering is that the user is presented with similar items that they have previously liked. Various parameters of the algorithm will be explored to find a suitable combinatio as determined by the RMSE. The main similarity functions that will be explored are **Cosine**, **pearson**, and **jaccard**. The normalization techniques that will be considered are **center** and **Z-score**. The number of similar items will be either **5**,**10**,**20**, or **30**. Values of `alpha` between **0.1** and **0.9** will be explored for the each model, resulting in a total of 36 models.

```
# initialize to list to store models
icbf_models = list()
```

```r
# initialize dataframe to store results
icbf_df = data.frame(Method=character(),
                     K=integer(),
                     Normalize=character(),
                     Alpha=double(),
                     RMSE=double(),
                     MAE=double(),
                     stringsAsFactors = FALSE)

# try different values of  alpha for each model
for(i in 1:9){
  # permutation 1
  icbf_model <- Recommender(data = getData(eval_scheme, "train"),
                            method = "IBCF",
                            parameter = list(k = 30,
                                             method = 'Cosine',
                                             normalize = 'center',
                                             alpha = i/10,
                                             normalize_sim_matrix = TRUE ) )
  # calculate and store RMSE
  prediction <- predict(icbf_model, getData(eval_scheme, "known"),type="ratings")
  err <- calcPredictionAccuracy(prediction, getData(eval_scheme, "unknown"))
  icbf_df[nrow(icbf_df) + 1,] = list(Method="Cosine", K=30,
                                     Normalize="center",
                                     Alpha=i/10,
                                     RMSE=as.numeric(err["RMSE"]),
                                     MAE=as.numeric(err["MAE"]))
  # add to list of models
  append(icbf_models, icbf_model)


  # permutation 2
  icbf_model <- Recommender(data = getData(eval_scheme, "train"),
                            method = "IBCF",
                            parameter = list(k = 20,
                                             method = 'pearson',
                                             normalize = 'Z-score',
                                             alpha = i/10,
                                             normalize_sim_matrix = TRUE  ) )
  # calculate and store RMSE
  prediction <- predict(icbf_model, getData(eval_scheme, "known"), type="ratings")
              err <- calcPredictionAccuracy(prediction, getData(eval_scheme, "unknown"))
              icbf_df[nrow(icbf_df) + 1,] = list(Method="pearson", K=100,
                                                 Normalize="Z-score",
                                                 Alpha=i/10,
                                                 RMSE=as.numeric(err["RMSE"]),
                                                MAE=as.numeric(err["MAE"]))
  # add to list of models
  append(icbf_models, icbf_model)


  # permutation 3
  icbf_model <- Recommender(data = getData(eval_scheme, "train"),
```

```r
                              method = "IBCF",
                              parameter = list(k = 5,
                                               method = 'jaccard',
                                               normalize = 'Z-score',
                                               alpha = i/10,
                                               normalize_sim_matrix = TRUE  ) )
  # calculate and store RMSE
  prediction <- predict(icbf_model, getData(eval_scheme, "known"), type="ratings")
                err <- calcPredictionAccuracy(prediction, getData(eval_scheme, "unknown"))
                icbf_df[nrow(icbf_df) + 1,] = list(Method="jaccard", K=150,
                                                   Normalize="Z-score",
                                                   Alpha=i/10,
                                                   RMSE=as.numeric(err["RMSE"]),
                                                   MAE=as.numeric(err["MAE"]))
   # add to list of models
  append(icbf_models, icbf_model)


  # permutation 4
  icbf_model <- Recommender(data = getData(eval_scheme, "train"),
                              method = "IBCF",
                              parameter = list(k = 10,
                                               method = 'Cosine',
                                               normalize = 'center',
                                               alpha = i/10,
                                               normalize_sim_matrix = TRUE  ) )
   # calculate and store RMSE
  prediction <- predict(icbf_model, getData(eval_scheme, "known"), type="ratings")
                err <- calcPredictionAccuracy(prediction, getData(eval_scheme, "unknown"))
                icbf_df[nrow(icbf_df) + 1,] = list(Method="Cosine", K=200,
                                                   Normalize="center",
                                                   Alpha=i/10,
                                                   RMSE=as.numeric(err["RMSE"]),
                                                   MAE=as.numeric(err["MAE"]))
  # add to list of models
  append(icbf_models, icbf_model)

}

# results
dplyr::arrange(icbf_df, RMSE)
```

```
##     Method   K Normalize Alpha     RMSE      MAE
## 1  pearson 100   Z-score   0.1 4.586878 3.459432
## 2  pearson 100   Z-score   0.2 4.586878 3.459432
## 3  pearson 100   Z-score   0.3 4.586878 3.459432
## 4  pearson 100   Z-score   0.4 4.586878 3.459432
## 5  pearson 100   Z-score   0.5 4.586878 3.459432
## 6  pearson 100   Z-score   0.6 4.586878 3.459432
## 7  pearson 100   Z-score   0.7 4.586878 3.459432
## 8  pearson 100   Z-score   0.8 4.586878 3.459432
## 9  pearson 100   Z-score   0.9 4.586878 3.459432
## 10  Cosine  30    center   0.1 4.917976 3.853578
## 11  Cosine  30    center   0.2 4.917976 3.853578
```

```
## 12  Cosine  30    center   0.3 4.917976 3.853578
## 13  Cosine  30    center   0.4 4.917976 3.853578
## 14  Cosine  30    center   0.5 4.917976 3.853578
## 15  Cosine  30    center   0.6 4.917976 3.853578
## 16  Cosine  30    center   0.7 4.917976 3.853578
## 17  Cosine  30    center   0.8 4.917976 3.853578
## 18  Cosine  30    center   0.9 4.917976 3.853578
## 19  Cosine 200    center   0.1 5.517030 4.138754
## 20  Cosine 200    center   0.2 5.517030 4.138754
## 21  Cosine 200    center   0.3 5.517030 4.138754
## 22  Cosine 200    center   0.4 5.517030 4.138754
## 23  Cosine 200    center   0.5 5.517030 4.138754
## 24  Cosine 200    center   0.6 5.517030 4.138754
## 25  Cosine 200    center   0.7 5.517030 4.138754
## 26  Cosine 200    center   0.8 5.517030 4.138754
## 27  Cosine 200    center   0.9 5.517030 4.138754
## 28 jaccard 150   Z-score   0.1 5.994644 4.548931
## 29 jaccard 150   Z-score   0.2 5.994644 4.548931
## 30 jaccard 150   Z-score   0.3 5.994644 4.548931
## 31 jaccard 150   Z-score   0.4 5.994644 4.548931
## 32 jaccard 150   Z-score   0.5 5.994644 4.548931
## 33 jaccard 150   Z-score   0.6 5.994644 4.548931
## 34 jaccard 150   Z-score   0.7 5.994644 4.548931
## 35 jaccard 150   Z-score   0.8 5.994644 4.548931
## 36 jaccard 150   Z-score   0.9 5.994644 4.548931
```

The best item-based colloborative model was based on the **pearson** similarity metric, **100** similar items, normalized using **Z-score** normalization and resulted in an RMSE of **4.6190** and a MAE of **3.4867**. Interestingly, the `alpha` parameter didn't have a strong effect on the performance of the model.

User Based Collaborative Filtering

User-based collaborative filtering will be implemented in this section. The main idea behind user based collaborative filtering is that the user is presented with top rated items from similar users. As before, various parameters of the algorithm will be explored to find the best combination. The main similarity functions that will be explored are **Cosine**, **pearson**, and **jaccard**. The normalization techniques that will be considered are **center** and **Z-score**. The number of similar users will be between **10** and **100**. Ten models will be trained by randomly selecting parameter values.

```r
# initialize list to store models
ucbf_models = list()

# initialize dataframe to store results
ucbf_df = data.frame(Method=character(),
                     NN=integer(),
                     Normalize=character(),
                     RMSE=double(),
                     MAE=double(),
                     stringsAsFactors = FALSE)

for(i in 1:10){
  # candiate parameter values
  NN = sample(c(10,20,30,40,50,60,70,80,90,100), 1)
  Method = sample(c("Cosine","pearson", "jaccard"), 1)
```

```
  Normalize = sample(c('center', 'z-score'), 1)

  # train model
  ucbf_model <- Recommender(data = getData(eval_scheme, "train"),
                            method = "UBCF",
                            parameter = list(nn = NN, method = Method, normalize = Normalize) )
  # calculate and store RMSE
  prediction <- predict(ucbf_model, getData(eval_scheme, "known"), type="ratings")
  err <- calcPredictionAccuracy(prediction, getData(eval_scheme, "unknown"))
  ucbf_df[nrow(ucbf_df) + 1,] = list(Method=Method,
                                     NN=NN,
                                     Normalize=Normalize,
                                     RMSE=as.numeric(err["RMSE"]),
                                     MAE=as.numeric(err["MAE"]))
  # add to list of models
  append(ucbf_models, ucbf_model)
}

# results
dplyr::arrange(ucbf_df, RMSE)
```

```
##     Method  NN Normalize     RMSE      MAE
## 1  pearson  40    center 4.554504 3.620508
## 2  pearson  50    center 4.560977 3.631596
## 3  pearson  90    center 4.563341 3.643079
## 4  pearson 100    center 4.563696 3.645391
## 5   Cosine  10    center 4.659424 3.674907
## 6  jaccard  40   z-score 5.638329 4.364191
## 7   Cosine 100   z-score 6.638966 5.030048
## 8   Cosine  60   z-score 6.701683 5.038370
## 9  pearson 100   z-score 6.727647 5.079814
## 10 pearson  80   z-score 6.758332 5.091039
```

The best user-based colloborative model was based on the **Cosine** similarity metric, **80** similar users, used **center** normalization and resulted in an RMSE of **4.5071** and an MAE of **3.5804**.

Summary

Overall, the user-based colloborative models performed slightly better than the item-based models. The best user-based model resulted in an RMSE of **4.5071** and the best item-based model had an RMSE of **4.6190**, an improvement of over **0.1**. The user-based model did better with `center` normalization but the item-based model preferred the `Z-score` normalization for these data.