Prediction & Evaluation for the Dataset 1

Saaya Yasuda

Prediction & Evaluation for the Dataset 1

project: 4
group: 6
author: Saaya Yasuda

Preparation

Load source

```
#rm(list=ls())
source('./lib/evaluation_ranked_scoring.r')
source('./lib/z_score.R')
```

Define helper functions

Helper 1: assign_rownames(dataset)
Most datasets contain usernames in the 1st col. Assigning them as rownames.

```
#######################################
# Helper 1: assign_rownames(dataset)
# Most datasets contain usernames in the 1st col. Assigning them as rownames.
#######################################
assign_rownames = function(dataset){
  rownames(dataset) = dataset[,1]
  dataset = dataset[,-1]
  return(dataset)
}
```

Helper 2: evaluate(train, test, weight, neighbor) + *Input:* 4 matrices: train data, test data, weights, and neighbor matrix.
+ *Output:* ranked score.

```
#######################################
# Helper 2: evaluate(train, test, weight, neighbor)
# Input: 4 matrices: train data, test data, weights, and neighbor matrix
# Output: ranked score
#######################################
evaluate = function(train, test, weight, neighbor){
  weight = assign_rownames(weight)
  neighbor = assign_rownames(neighbor)

  #prediction
  pred = z_score(train,weight,neighbor)
  row.names(pred) = row.names(train)

  #for alpha values in ranked scoring formula:
```

```r
  avg_half_life = mean(rowSums(test)) # average number of sites visited by user.
  avg_half_life = round(avg_half_life, 3)
  alpha = c(avg_half_life, 5, 10) #paper 1 used 5 & 10 for alpha in ranked scoring.

  score_vec = vector()
  for (a in alpha){
    score_vec = c(score_vec, randked_scoring(pred, test, a))
  }
  names(score_vec) = alpha
  return (score_vec)
}
```

Load dataset

```r
train1 <- read.csv("./output/dataset1_train.csv",header=T)
test1 <- read.csv("./output/dataset1_test.csv",header=T)

train1 = assign_rownames(train1)
test1 = assign_rownames(test1)
```

Load correlation matrices & neighbor matrices for prediction.

Note: When you pull the csv files in the output folder in Git, please use "git lfs pull." The files bigger than 100mb are uploaded using git large file storage.

```r
Spearman_SW = read.csv("./output/spearman_train1.csv",header=T)
Spearman_thresh = read.csv("./output/neighbor_thresh_spearman.csv",header=T)
Spearman_bnn = read.csv("./output/neighbor_bnn_spearman.csv",header=T)
Spearman_combo = read.csv("./output/neighbor_comb_spearman.csv",header=T)

Spearman_VW = read.csv("./output/spearman_vm_train1.csv",header=T)
Spearman_VW_thresh = read.csv("./output/neighbor_thresh_spearman_wv.csv",header=T)
Spearman_VW_bnn = read.csv("./output/neighbor_bnn_spearman_vw.csv",header=T)

VecSim_SW = read.csv("./output/vectorsimilarity_train1.csv",header=T)
VecSim_thresh = read.csv("./output/neighbor_thresh_vecsim.csv",header=T)
VecSim_bnn = read.csv("./output/neighbor_bnn_vecsim.csv",header=T)

VecSim_VW = read.csv("./output/vectorsimilarity_vm_train1.csv",header=T)
VecSim_VW_thresh = read.csv("./output/neighbor_thresh_vecsim_vm.csv",header=T)
VecSim_VW_bnn = read.csv("./output/neighbor_bnn_vecsim_vm.csv",header=T)
```

Step 1: Compare Similarity Weighting (SW)

Fix variance weighting (VW) & neighbor selection (NS) to be no VW and Weight Threshold.
Then compare the scores between the SW algorithms.

```r
SCORE_Spearman_SW = evaluate(train1, test1, Spearman_SW, Spearman_thresh)
SCORE_VecSim_SW = evaluate(train1, test1, VecSim_SW, VecSim_thresh)

SW_score = rbind(SCORE_Spearman_SW, SCORE_VecSim_SW)
```

```
rownames(SW_score) = c("Spearman with NS thresh & No VW","VecSim with NS thresh & no VW")
SW_score
```

```
##                                     3.087        5       10
## Spearman with NS thresh & No VW 33.57993 41.65699 53.24078
## VecSim with NS thresh & no VW   34.46454 42.45547 53.89660
```

Vector Similarity does better.

Step 2: Compare Variance Weighting (VW)

Fix neighbor selection (NS) to be Weight Threshold.
Then compare the scores between the VW algorithms.

```
SCORE_Spearman_VW = evaluate(train1, test1, Spearman_VW, Spearman_VW_thresh)
SCORE_VecSim_VW = evaluate(train1, test1, VecSim_VW, VecSim_VW_thresh)

VW_score = rbind(SCORE_Spearman_VW, SCORE_VecSim_VW)
rownames(VW_score) = c("Spearman with NS thresh & VW","VecSim with NS thresh & VW")
VW_score
```

```
##                                  3.087        5       10
## Spearman with NS thresh & VW 33.26044 41.37642 53.01921
## VecSim with NS thresh & VW   32.82186 41.32847 53.27723
```

Variance weighting didn't improve the score.

Step 3: Compare Neighbor Selection (NS)

Fix variance weighting (VW) to be no VW.
Then compare the scores between the NS algorithms.

```
SCORE_Spearman_bnn = evaluate(train1, test1, Spearman_SW, Spearman_bnn)
SCORE_Spearman_combo = evaluate(train1, test1, Spearman_SW, Spearman_combo)
SCORE_VecSim_bnn = evaluate(train1, test1, VecSim_SW, VecSim_bnn)

NS_score = rbind(SCORE_Spearman_bnn,SCORE_Spearman_combo,SCORE_VecSim_bnn)
rownames(NS_score) = c("Spearman with NS bnn & No VW",
                       "Spearman with NS combo & No VW",
                       "VecSim with NS bnn & No VW")
NS_score
```

```
##                                   3.087        5       10
## Spearman with NS bnn & No VW   23.39347 29.85521 40.55418
## Spearman with NS combo & No VW 33.57338 41.65260 53.23769
## VecSim with NS bnn & No VW     23.58450 30.01127 40.69014
```

Spearman with combo did very well. bnn doesn't seem to work well for this.

Conclusion

```
scores = rbind(SW_score,VW_score,NS_score)

print("Best score when alpha = avg number of site visits per user")
```

```
## [1] "Best score when alpha = avg number of site visits per user"
```

```r
alpha3 = scores[order(scores[,1],decreasing=T),]
alpha3
```

```
##                                    3.087        5       10
## VecSim with NS thresh & no VW   34.46454 42.45547 53.89660
## Spearman with NS thresh & No VW 33.57993 41.65699 53.24078
## Spearman with NS combo & No VW  33.57338 41.65260 53.23769
## Spearman with NS thresh & VW    33.26044 41.37642 53.01921
## VecSim with NS thresh & VW      32.82186 41.32847 53.27723
## VecSim with NS bnn & No VW      23.58450 30.01127 40.69014
## Spearman with NS bnn & No VW    23.39347 29.85521 40.55418
```

```r
write.csv(alpha3,"./output/dataset1_result_alpha3.csv")

print("Best score when alpha = 5")
```

```
## [1] "Best score when alpha = 5"
```

```r
alpha5 = scores[order(scores[,2],decreasing=T),]
alpha5
```

```
##                                    3.087        5       10
## VecSim with NS thresh & no VW   34.46454 42.45547 53.89660
## Spearman with NS thresh & No VW 33.57993 41.65699 53.24078
## Spearman with NS combo & No VW  33.57338 41.65260 53.23769
## Spearman with NS thresh & VW    33.26044 41.37642 53.01921
## VecSim with NS thresh & VW      32.82186 41.32847 53.27723
## VecSim with NS bnn & No VW      23.58450 30.01127 40.69014
## Spearman with NS bnn & No VW    23.39347 29.85521 40.55418
```

```r
write.csv(alpha5,"./output/dataset1_result_alpha5.csv")

print("Best score when alpha = 10")
```

```
## [1] "Best score when alpha = 10"
```

```r
alpha10 = scores[order(scores[,3],decreasing=T),]
alpha10
```

```
##                                    3.087        5       10
## VecSim with NS thresh & no VW   34.46454 42.45547 53.89660
## VecSim with NS thresh & VW      32.82186 41.32847 53.27723
## Spearman with NS thresh & No VW 33.57993 41.65699 53.24078
## Spearman with NS combo & No VW  33.57338 41.65260 53.23769
## Spearman with NS thresh & VW    33.26044 41.37642 53.01921
## VecSim with NS bnn & No VW      23.58450 30.01127 40.69014
## Spearman with NS bnn & No VW    23.39347 29.85521 40.55418
```

```r
write.csv(alpha10,"./output/dataset1_result_alpha10.csv")
```

"Vector Similarity + no variance weighting (VW) + Weight Threshold neighbor selection (NS)" is the best.

Changing the alpha values slightly change the ranks. For a=10, for example, "Vector Similarity + variance weighting + Weight Threshold" performed better than "Spearman + variance weighting + Weight Threshold", but that's not the case for a=3.087 and a=5.