Assignment01

Mohamed Elmoudni, Shazia Khan, Senthil Dhanapal

February 12, 2017

Project 1 :

Requirements:

The goal of this assignment is to help you build your intuition about recommender systems, with a basic soup to nuts implementation coded "from scratch."

Your task is to build a very basic recommender system, first by writing your own functions, then by replacing those functions with those provided in an R Package or a Python library (such as scikitlearn).

- You should very briefly first describe the recommender system that you're going to build out from a business perspective, e.g. "This system recommends movies to users."
- You can find a dataset, or build out your own toy dataset and load into (for example) an R or pandas dataframe, a Python dictionary or list of lists, (or other data structure of your choosing).
- You can use either collaborative filtering, or a hybrid of content management and collaborative filtering.
- You are encouraged to hand code at least your similarity function.
- After you have built out your own code base, create an alternate version using packages or libraries. Compare the results and performance.
- You are also encouraged to think about how to best handle missing data.
- Your code should be turned in an RMarkdown file or a Jupyter notebook, and posted to Github. You may work in a small group (2 or 3 people) on this assignment. While you're never discouraged from adding features or advanced capabilities such as regularization and matrix factorization methods, it is not expected at this point in the course.

Description

Our recommender system recommends movies based on user based collaborative filtering (UBCF) and item based collaborative filtering (IBCF) recommendation system. It uses a m(x)n matrix with users as rows and movies as columns. Our function predicts the missing ratings for a movie based on the similar movies (IBCF) and user ratings (UBCF). Similarity of movies and users are calculated based on cosine, Jaccard, center-cosine (Pearson) methods.

We used a toy dataset to train and test our functions

We have also tested using Recommenderlab, which is one of the R packages. Even though our values were not matching exactly, they are similar.

```
library(recommenderlab)
library(reshape2)
library(ggplot2)
library(knitr)


# Read training file along with the header
dt<-read.csv("https://raw.githubusercontent.com/simonnyc/IS-643/master/Assignment01_data.csv", header=T
```

Data Exploration

```
# Just look at first few lines of this file
kable(head(dt))
```

User	Items	Ratings
1	item_1	1
1	item_10	3
1	item_11	1
1	item_12	4
1	item_2	2
1	item_3	3

```
# Summary
kable(summary(dt), caption ="Data Summary")
```

Table 2: Data Summary

User	Items	Ratings
Min. : 1.00	item_11:10	Min. :1.000
1st Qu.: 3.00	item_12:10	1st Qu.:1.000
Median : 5.00	item_3 :10	Median :2.000
Mean : 5.39	item_1 : 9	Mean :2.295
3rd Qu.: 8.00	item_5 : 9	3rd Qu.:3.000
Max. :10.00	item_9 : 9	Max. :5.000
NA	(Other):48	NA

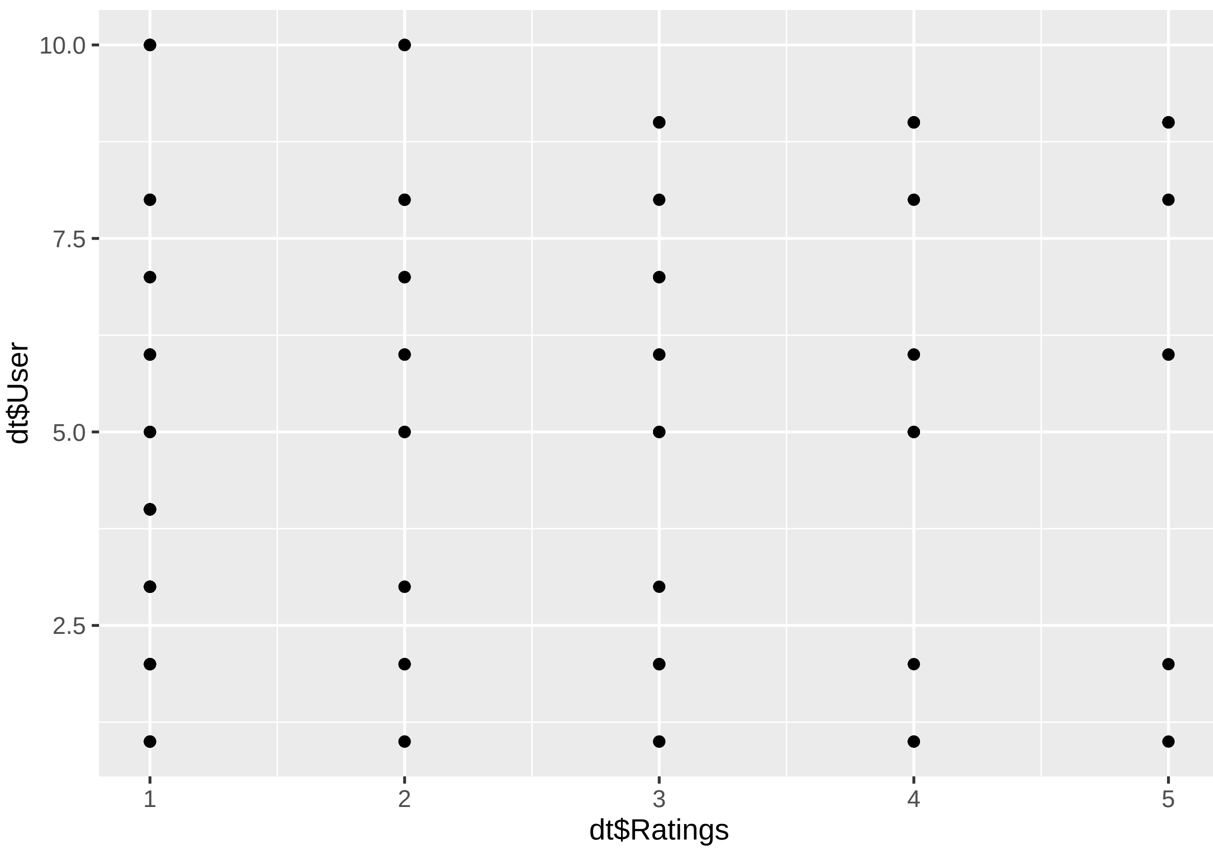
```
#Frequency table

kable(as.data.frame(table(dt$Ratings)), caption = "Ratings Frequency Table")
```

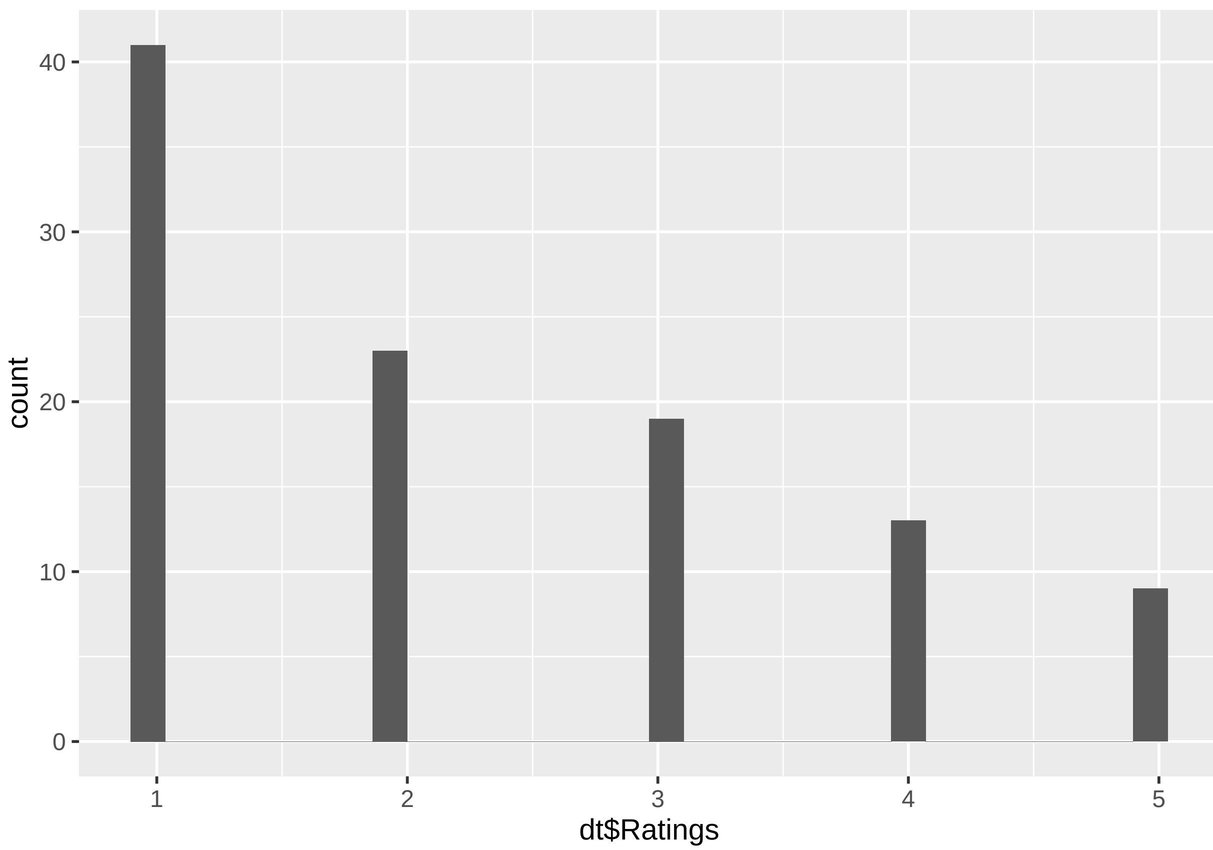
Table 3: Ratings Frequency Table

Var1	Freq
1	41
2	23
3	19
4	13
5	9

```
qplot(dt$Ratings, dt$User)
```




```
qplot(dt$Ratings)
```




```
#[head(sort(dt$Ratings,decreasing=TRUE), n = 5)]

#head(dt[sort(dt$Ratings, decreasing=TRUE), ], 100)
```

Data Preparation

```
#step 1: item-similarity calculation co-rated items are considered and similarity between two items
#are calculated using cosine similarity

g<-acast(dt, User ~ Items, value.var = 'Ratings')
#g[is.na(g)] = 0
df<- data.frame(g)
df$userno<- as.numeric(rownames(df))

#re-arrange the columns
library(dplyr)
df <- df %>%
  select(item_10:item_12, everything())

df <- df %>%
  select(item_1:item_9, everything())

df <- df %>%
  select(userno, everything())
```

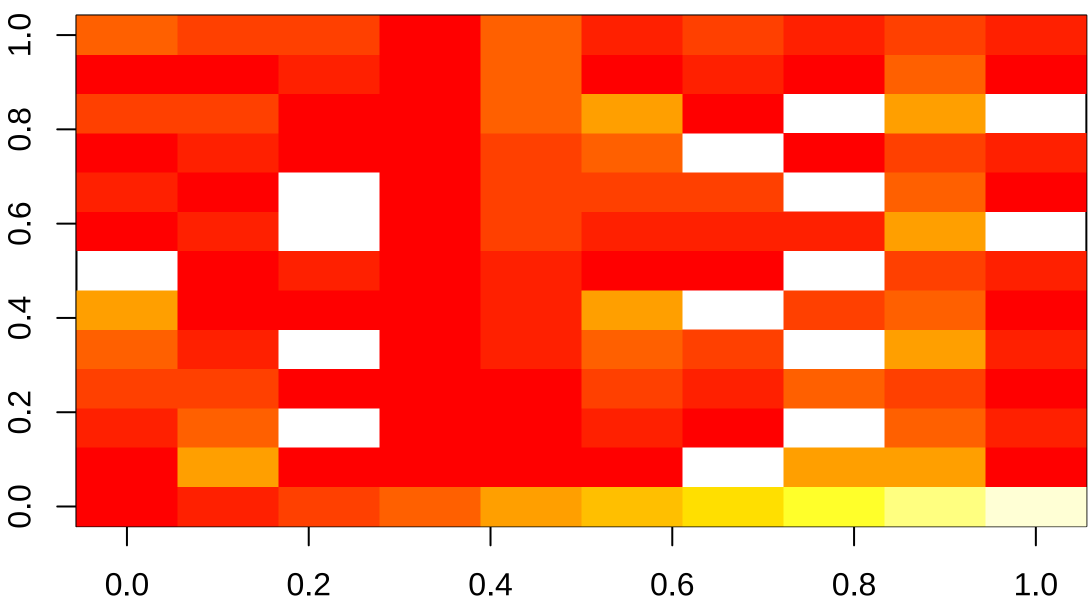
Input Matrix

```
#### Input Matrix
kable(df, caption= "Input Rating Matrix")
```

Table 4: Input Rating Matrix

userno	item_1	item_2	item_3	item_4	item_5	item_6	item_7	item_8	item_9	item_10	item_11	ite
1	1	2	3	4	5	NA	1	2	1	3	1	
2	5	4	3	2	1	1	2	1	2	3	1	
3	1	NA	1	NA	1	2	NA	NA	1	1	2	
4	1	1	1	1	1	1	1	1	1	1	1	
5	1	1	1	2	2	2	3	3	3	4	4	
6	1	2	3	4	5	1	2	3	4	5	1	
7	NA	1	2	3	NA	1	2	3	NA	1	2	
8	5	NA	4	NA	3	NA	2	NA	1	NA	1	
9	5	4	3	5	4	3	5	4	3	5	4	
10	1	2	1	2	1	2	NA	1	2	NA	1	

```
image(as.matrix(df))
```



IBCF Implementation

Implementing Item based recommender systems, like user based collaborative filtering, requires two steps:

1. Calculating Item similarities
2. Predicting the targeted item rating for the targeted User.

Step1: Calculating Item Similarity: we calculate the similarity between co-rated items. We use cosine similarity compute the similarity between items.

The output for step is similarity matrix between Items.

```
#install.packages("lsa")
library(lsa)
x = df[,2:ncol(df)]
x[is.na(x)] = 0
itemSimil = cosine(as.matrix(x))

kable(itemSimil)
```

	item_1	item_2	item_3	item_4	item_5	item_6	item_7	item_8	item_9
item_1	1.0000000	0.7779466	0.8606630	0.6000469	0.6707816	0.6222222	0.8012336	0.5499719	0.6880625
item_2	0.7779466	1.0000000	0.7909058	0.8861991	0.7204843	0.7584980	0.8091134	0.8045086	0.8172515
item_3	0.8606630	0.7909058	1.0000000	0.8133901	0.8785713	0.6196773	0.8235321	0.7485542	0.7804217
item_4	0.6000469	0.8861991	0.8133901	1.0000000	0.8521116	0.7425580	0.8425167	0.9546687	0.8294258
item_5	0.6707816	0.7204843	0.8785713	0.8521116	1.0000000	0.5927270	0.7610782	0.7761505	0.8415606
item_6	0.6222222	0.7584980	0.6196773	0.7425580	0.5927270	1.0000000	0.7765803	0.7919596	0.8256750
item_7	0.8012336	0.8091134	0.8235321	0.8425167	0.7610782	0.7765803	1.0000000	0.9021342	0.8178608
item_8	0.5499719	0.8045086	0.7485542	0.9546687	0.7761505	0.7919596	0.9021342	1.0000000	0.8340577
item_9	0.6880625	0.8172515	0.7804217	0.8294258	0.8415606	0.8256750	0.8178608	0.8340577	1.0000000
item_10	0.6432675	0.8444719	0.7750911	0.9046656	0.8590614	0.7504788	0.8771840	0.9097177	0.9168313
item_11	0.6552976	0.7097184	0.7042830	0.7796603	0.6797220	0.9141401	0.8996469	0.8757605	0.8043478
item_12	0.6913580	0.7779466	0.8606630	0.8375654	0.7927419	0.8000000	0.8012336	0.8328147	0.8027395

Cosine similarity matrix for items Step2: Predicting the targeted item rating for the targeted User using content based system

```
ratings<- as.matrix(x)
userRecmd <- function(userno)
{

  userRatings <- ratings[userno,]
  userRatings[userRatings==0] <- NA
  #

  non_rated_items <- names(userRatings[is.na(userRatings)])
  rated_items <- names(userRatings[!is.na(userRatings)])
  m1 <- itemSimil[non_rated_items,]
  v1 <- apply(m1,1,function(x) sum((x*ratings[userno,]),na.rm = T)/(sum(x[rated_items])))
```

```
  #ratings[userno,names(v1)] <- v1
  #

  return(v1)
}
```

Recommendation for User 10 using IBCF Following line predicts the missing ratings for User 10

```
kable(userRecmd(10), caption = "user 10, IBCF")
```

Table 6: user 10, IBCF

item_7	1.49148
item_10	1.51390

UBCF Implementation

User defined function for user based collobarative filtering

```
Recmd.UIB <- function(M, user, k)
{
  #M <- ratings
  #user <- 'u10'
  M[is.na(M)] <- 0
  rowsums <- rowSums(M)
  rowcounts <- apply(M,1,function(x) length(x[x>0]))
  rowavg <- rowsums/rowcounts
  for(r in 1:nrow(M))
  {
    for(c in 1:ncol(M))
    {
      if(M[r,c] >0 )
      {
        M[r,c] <- M[r,c] - rowavg[r]
      }
    }
  }

  user.M <- M[user,]
  if(is.element(0,user.M)==F)
  {
    print('no missing ratings')
    return()
  }

  otherusers.M <- M[rownames(M)!=user,]
  sim.M <- matrix(,nrow(otherusers.M))
  rownames(sim.M) <- rownames(otherusers.M)

  for(r in 1:nrow(otherusers.M))
```

```
  {
    sim.user <- rownames(otherusers.M)[r]
    sim.M[sim.user,1] <- cosine(user.M,otherusers.M[r,])
  }
  sim.M[is.na(sim.M)] <- -1
  sim.M <- sim.M[order(-sim.M[,1]),,drop=F]

  non.rated.items <- names(user.M[user.M==0])

  #top k similar users who have rated the items not rated by the current user
  if(k > nrow(sim.M))
  {
    k <- nrow(sim.M)
  }
  sim.user.k <- sim.M[1:k,,drop=F]
  sim.user.k1 <- ratings[rownames(sim.user.k),non.rated.items,drop=F]
  sim.user.k1 <- cbind(sim.user.k1,sim=sim.user.k)
  colnames(sim.user.k1) <- c(non.rated.items,'sim')
  sim.user.k2 <- matrix(,ncol=ncol(sim.user.k1))
  for(r in 1:nrow(sim.user.k1))
  {
    v.temp <- sim.user.k1[r,,drop=F]
    if(is.element(NA,v.temp)!=T)
    {
      sim.user.k2 <- rbind(sim.user.k2,v.temp)

    }
  }
  sim.user.k2 <- sim.user.k2[2:nrow(sim.user.k2),]
  sim.user.k3 <- sim.user.k2[,1:ncol(sim.user.k2)-1]
  sim.user.k4 <- sim.user.k2[,ncol(sim.user.k2),drop=F]

  v1 <- apply(sim.user.k3,2,function(x) sum(x*sim.user.k4)/sum(sim.user.k4))
  return(v1)


}
```

Prediction for same user 10 using User based collobarative filtering (UBCF)

```
Recmd.UIB(ratings,'10',5)
```

```
##    item_7   item_10
## 0.7561378 1.9112942
```

Following lines predict the ratings for the user 10 using built-in package :- RecommenderLab package and using Item based collobarative filtering

```
m <- ratings
m[m==0] <- NA

affinity.matrix<- as(m,"realRatingMatrix")
```

```
Rec.model<-Recommender(affinity.matrix,method="IBCF",
                       param=list(normalize = "Z-score",method="Cosine"))

u <- predict(Rec.model, affinity.matrix[10,], type="ratings")
as(u, "list")
```

```
## $`10`
##   item_7  item_10
## 1.738512 1.542999
```

Using Recommnderlab package

Following lines predict the ratings for the user 10 using built-in package :- RecommenderLab package and using User based collobarative filtering

Rating Matrix

```
x = df[,2:ncol(df)]
#x[is.na(x)] = 0
x<- as.matrix(x)
x <- as(x, "realRatingMatrix")
kable(as(x, "matrix"))
```

item_1	item_2	item_3	item_4	item_5	item_6	item_7	item_8	item_9	item_10	item_11	item_12
1	2	3	4	5	NA	1	2	1	3	1	4
5	4	3	2	1	1	2	1	2	3	1	3
1	NA	1	NA	1	2	NA	NA	1	1	2	3
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	2	2	2	3	3	3	4	4	4
1	2	3	4	5	1	2	3	4	5	1	2
NA	1	2	3	NA	1	2	3	NA	1	2	3
5	NA	4	NA	3	NA	2	NA	1	NA	1	2
5	4	3	5	4	3	5	4	3	5	4	3
1	2	1	2	1	2	NA	1	2	NA	1	2

UBCF using Cosine method

```
rec=Recommender(x[1:nrow(x)],method="UBCF", param=list(normalize = "Z-score",method="Cosine", nn=12))
rec2 <- predict(rec, x[1:nrow(x)])

kable(as(rec2, "matrix")['10',], caption = "UBCF using Cosine method")
```

Table 8: UBCF using Cosine method

item_1	NA
item_2	NA
item_3	NA
item_4	NA
item_5	NA

item_6	NA
item_7	1.385918
item_8	NA
item_9	NA
item_10	1.384313
item_11	NA
item_12	NA

UBCF using Jaccard method

```
### Using
rec=Recommender(x[1:nrow(x)],method="UBCF", param=list(normalize = "Z-score",method="Jaccard", nn=5))
rec2 <- predict(rec, x[1:nrow(x)])
kable(as(rec2, "matrix")['10',], caption = "UBCF using Jaccard method")
```

Table 9: UBCF using Jaccard method

item_1	NA
item_2	NA
item_3	NA
item_4	NA
item_5	NA
item_6	NA
item_7	1.371636
item_8	NA
item_9	NA
item_10	1.877474
item_11	NA
item_12	NA

IBCF using Cosine method

```
rec=Recommender(x[1:nrow(x)],method="IBCF", param=list(normalize = "Z-score",method="Cosine"))
rec2 <- predict(rec, x[1:nrow(x)])
kable(as(rec2, "matrix")['10',], caption = "IBCF using Cosine method")
```

Table 10: IBCF using Cosine method

item_1	NA
item_2	NA
item_3	NA
item_4	NA
item_5	NA
item_6	NA
item_7	1.738512
item_8	NA
item_9	NA
item_10	1.542999

item_11	NA
item_12	NA

Conclusion

from the below comparaison table, most of the rating values are similar using different methods. However, we observed some outliers. For item7, the outlier value 0.7561378 is produced by user-defined UBCF-Pearson. For item 10, the outlier value 1.384313 is produced by Recommender UBCF Cosine.

```
dt_conc<-read.csv("https://raw.githubusercontent.com/simonnyc/IS-643/master/Proj01conclusion.csv", head

kable(dt_conc, caption= "Recommendation for User 10")
```

Table 11: Recommendation for User 10

Method	Item.7	Item.10
user-defined IBCF-Cosine	1.4914800	1.513900
Recommenderlab IBCF Cosine	1.5429990	1.738512
user-defined UBCF-Pearson	0.7561378	1.911294
Recommenderlab UBCF Cosine	1.3859180	1.384313
Recommenderlab UBCF Jaccard	1.3716360	1.877474