Assignment01

Mohamed Elmoudni

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.

```
library(recommenderlab)
```

```
## Warning: package 'recommenderlab' was built under R version 3.3.2
```

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

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

```
## Warning: package 'arules' was built under R version 3.3.2
```

```
##
## 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 3.3.1

##
## 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

## Warning: package 'registry' was built under R version 3.3.2
```

```
library(reshape2)
library(ggplot2)
library(knitr)
#######################################

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

#############
# 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

Data Exploration

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

Table 2: Data Summary

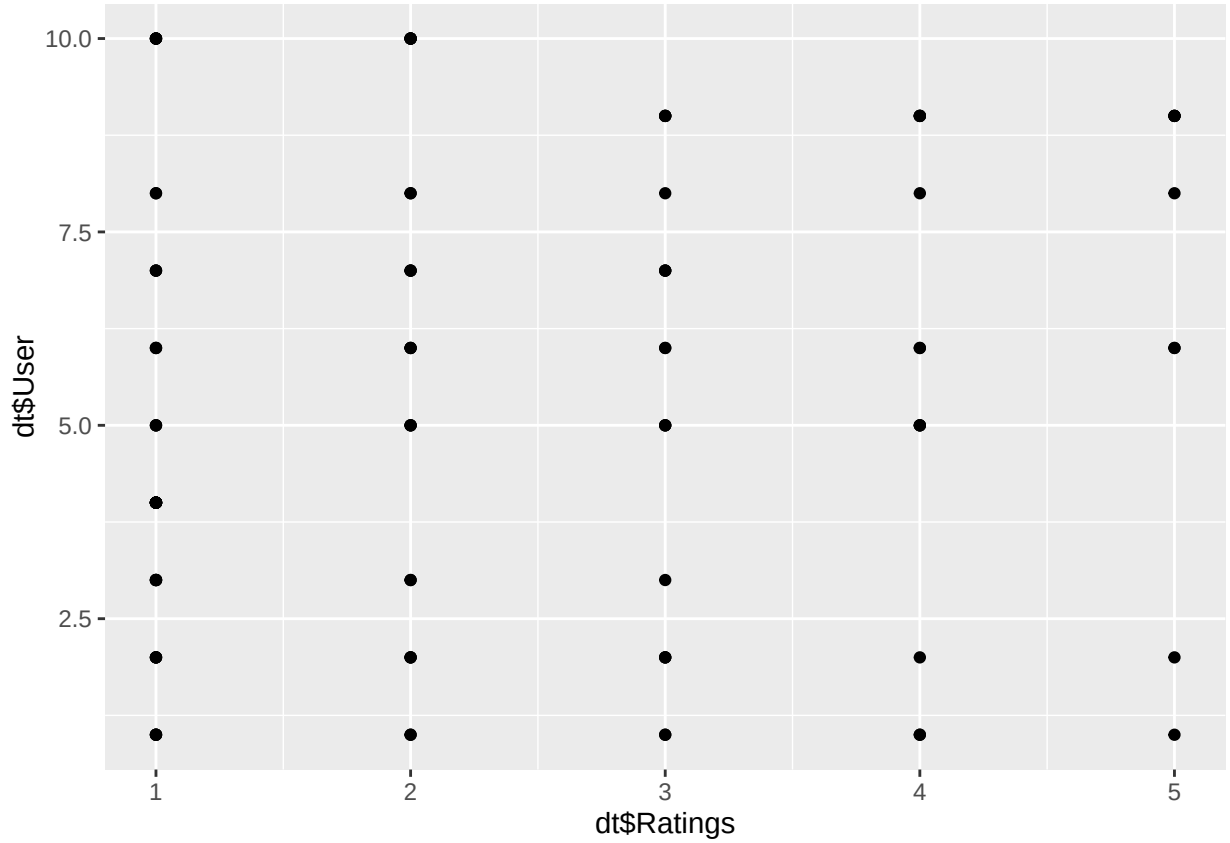
User	Items	Ratings
Min. : 1.00	item_11:10	Min. :1.000
1st Qu.: 3.00	item_12:10	1st Qu.:1.000
Median : 5.00	item_3 :10	Median :2.000
Mean : 5.39	item_1 : 9	Mean :2.295
3rd Qu.: 8.00	item_5 : 9	3rd Qu.:3.000
Max. :10.00	item_9 : 9	Max. :5.000
NA	(Other):48	NA

```
#frequency table

table(dt$Ratings)
```

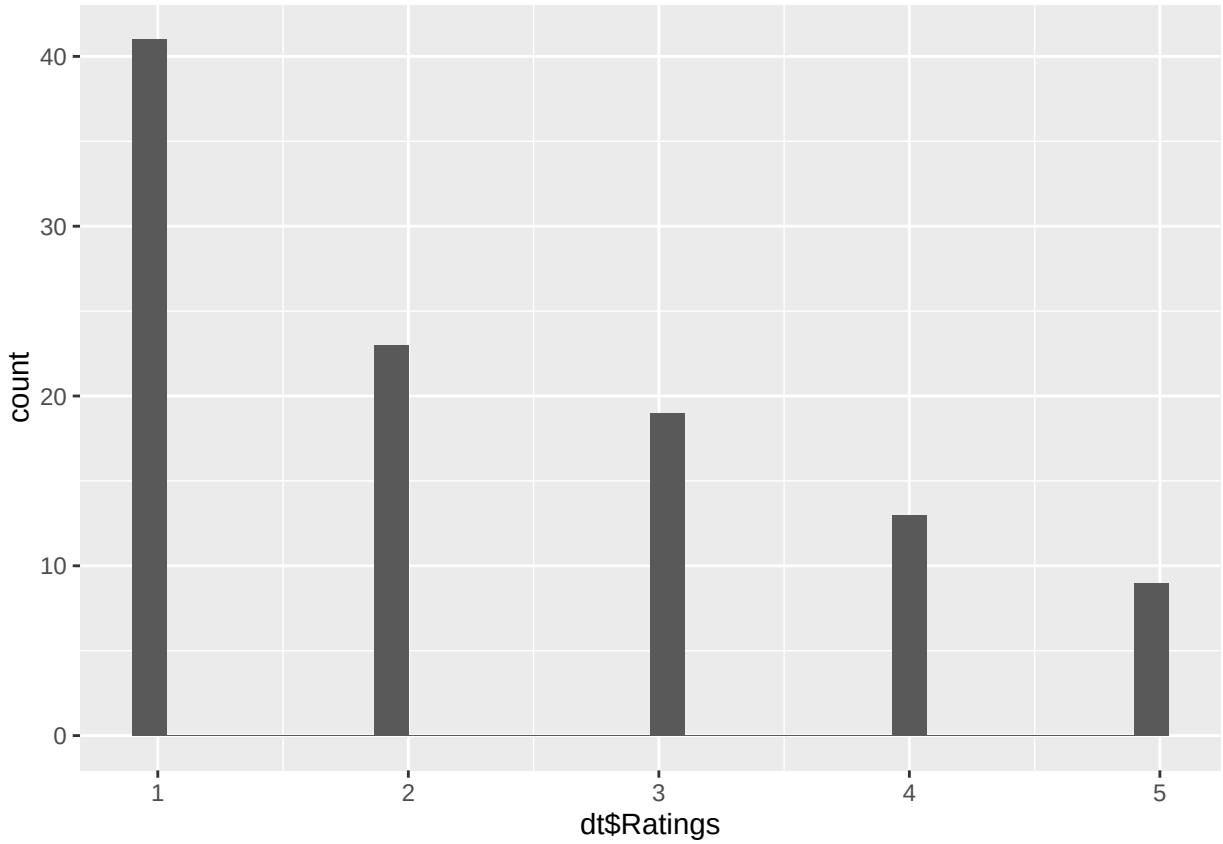
```
##
##  1  2  3  4  5
## 41 23 19 13  9
```

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




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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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

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

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 or pearson-similarity to compute the similarity between items. The output for step is similarity matrix between Items.

```
## Warning: package 'dplyr' was built under R version 3.3.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:arules':
##
##     intersect, recode, setdiff, setequal, union

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Warning: failed to assign NativeSymbolInfo for lhs since lhs is already
## defined in the 'lazyeval' namespace

## Warning: failed to assign NativeSymbolInfo for rhs since rhs is already
## defined in the 'lazyeval' namespace

## Warning: package 'lsa' was built under R version 3.3.2

## Loading required package: SnowballC

## Warning: package 'SnowballC' was built under R version 3.3.2

##
## Attaching package: 'lsa'

## The following object is masked from 'package:dplyr':
##
##     query
```

	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

Step2: Predicting the targeted item rating for the targeted User

Recommending Top N items: Once all the non rated movies are predicted we recommend top N movies to a user Code for Item based collaborative filtering in R:

```
userRecmd = function(userno)
 {
   #extract all the movies not rated by CHAN
   userRatings = df[userno,]
   non_rated_movies = list()
   rated_movies = list()
   for(i in 2:ncol(userRatings)){
     if(is.na(userRatings[,i]))
```

```
    {
      non_rated_movies = c(non_rated_movies,colnames(userRatings)[i])
    }
    else
    {
      rated_movies = c(rated_movies,colnames(userRatings)[i])
    }
  }
  non_rated_movies = unlist(non_rated_movies)
  rated_movies = unlist(rated_movies)
  #create weighted similarity for all the rated movies by user
  non_rated_pred_score = list()
  for(j in 1:length(non_rated_movies)){
    temp_sum = 0
    df2 = itemSimil[which(rownames(itemSimil)==non_rated_movies[j]),]
    for(i in 1:length(rated_movies)){
      temp_sum = temp_sum+ df2[which(names(df2)==rated_movies[i])]
       }
    weight_mat = df2*df[userno,2:7]
    non_rated_pred_score = c(non_rated_pred_score,rowSums(weight_mat,na.rm=T)/temp_sum)
    }
  pred_rat_mat = as.data.frame(non_rated_pred_score)
  names(pred_rat_mat) = non_rated_movies
  for(k in 1:ncol(pred_rat_mat)){
    df[userno,][which(names(df[userno,]) == names(pred_rat_mat)[k])] = pred_rat_mat[1,k]
  }
  return(df[userno,])
}
```

Recommend for User 10 Now we will use the above to find a rating for user 10

```
userRecmd(10)
```

```
##    userno item_1 item_2 item_3 item_4 item_5 item_6    item_7 item_8
## 10     10      1      2      1      2      1      2 0.8794568      1
##    item_9   item_10 item_11 item_12
## 10      2 0.8822832       1       2
```

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

item_1	item_2
1	2
5	4
1	NA
1	1
1	1

item_1	item_2
1	2
NA	1
5	NA
5	4
1	2

```
rec=Recommender(x[1:nrow(x)],method="UBCF", param=list(normalize = "Z-score",method="Cosine", nn=12))
rec2 <- predict(rec, x[1:nrow(x)])
(as(rec2, "matrix")['10',])
```

```
## item_1 item_2
##     NA     NA
```

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

```
## item_1 item_2
##     NA     NA
```

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

```
## item_1 item_2
##     NA     NA
```

```
##################

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

```
## item_1 item_2
##     NA     NA

##################
```