AC / Machine Learning
Rita P. Ribeiro
Nuno Moniz

2021/2022
DCC - FCUP
DEI - FEUP

2 Hands On: Web Usage Mining

Read the file log.csv, containing information on the web pages visited by a set of users, into a data frame in R.

```
library(dplyr)
library(readr)
d <- read_csv("log.csv")
```

2.1 Simple Recommendation Strategies

Most Visited Pages

1. Recommend the 3 most visited pages.
For that purpose:

(a) inspect how many times each page was visited;

```
d %>% group_by(PAGE) %>% tally()

## # A tibble: 7 x 2
##   PAGE      n
##   <chr> <int>
## 1 A         3
## 2 B         3
## 3 C         4
## 4 F         2
## 5 G         2
## 6 I         2
## 7 J         1
```

(b) sort the pages by decreasing number of visits;

```
d %>% group_by(PAGE) %>% tally(sort=TRUE)

## # A tibble: 7 x 2
##   PAGE      n
##   <chr> <int>
## 1 C         4
## 2 A         3
## 3 B         3
## 4 F         2
## 5 G         2
## 6 I         2
## 7 J         1
```

(c) obtain the top 3 pages for recommendation.

```
d %>% group_by(PAGE) %>% tally(sort=TRUE) %>% top_n(3) %>% pull(PAGE)

## [1] "C" "A" "B"
```

Using Clustering Results

2. Suppose we want to form two clusters of users, according to the pages they have visited. For that purpose:

(a) start by transforming the log access data into a matrix that has on each row a user and for each user the information on his visits to each page; this can be obtained with the `table()` function;

```
dat <-table(d$USER,d$PAGE)
```

```
##
##      A B C F G I J
##   u1 1 1 1 0 0 0 0
##   u2 1 0 1 0 0 0 0
##   u3 0 1 0 1 1 1 0
##   u4 0 1 1 0 0 0 0
##   u5 0 0 0 1 1 1 1
##   u6 1 0 1 0 0 0 0
```

(b) use the function `dist()` to obtain a distance matrix with the Euclidean distance between the users;

```
library(proxy)
dm <-dist(dat) # euclidean
summary(pr_DB)

## * Similarity measures:
## Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice, eDice,
## eJaccard, Fager, Faith, Gower, Hamman, Jaccard, Kulczynski1,
## Kulczynski2, Michael, Mountford, Mozley, Ochiai, Pearson, Phi,
## Phi-squared, Russel, simple matching, Simpson, Stiles, Tanimoto,
## Tschuprow, Yule, Yule2
##
## * Distance measures:
## Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean, fJaccard,
## Geodesic, Hellinger, Kullback, Levenshtein, Mahalanobis, Manhattan,
## Minkowski, Podani, Soergel, supremum, Wave, Whittaker

dm <- dist(dat,method="jaccard")
```

```
##           u1        u2        u3        u4        u5
## u2 0.3333333
## u3 0.8333333 1.0000000
## u4 0.3333333 0.6666667 0.8000000
## u5 1.0000000 1.0000000 0.4000000 1.0000000
## u6 0.3333333 0.0000000 1.0000000 0.6666667 1.0000000
```

(c) check for alternatives in the help page of `dist()`;

(d) use the function `hclust()` with the distance matrix to obtain an agglomerative clustering model of this data;

```
c1 <- hclust(dm)
```

(e) visualize the obtained dendogram with function `plot()`;

(f) visualize again the dendogram, but now with option `hang=-0.1`.

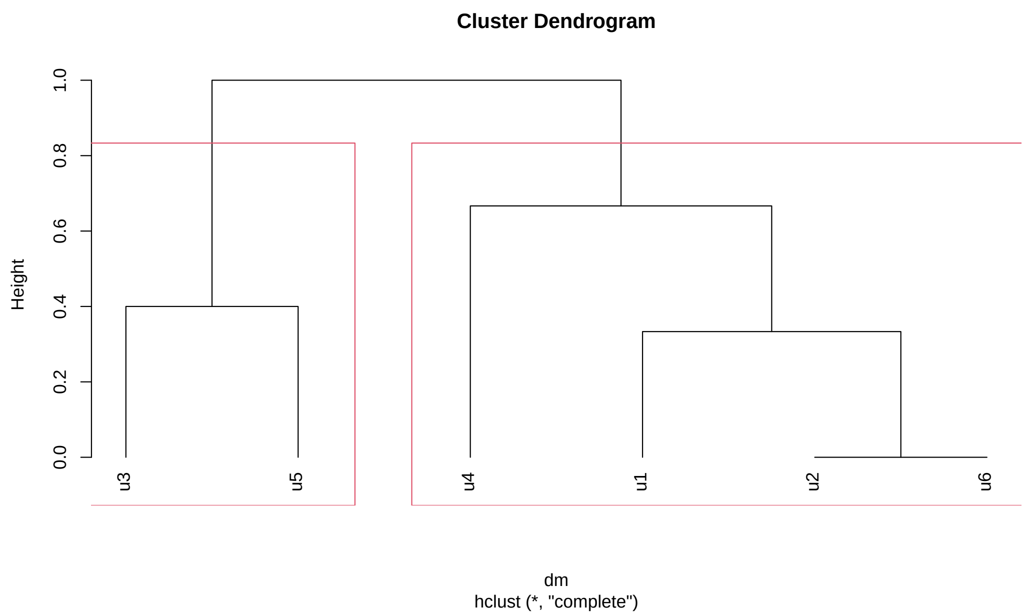
(g) use the function `cutree()` to "cut"the hierarchical clustering in just two clusters; inspect the cluster membership of each user;

```
cm <- cutree(c1,2)
```

```
## u1 u2 u3 u4 u5 u6
##  1  1  2  1  2  1
```

(h) use the function `rect.hclust()` to draw the previous solution in the dendogram.

```
plot(c1,hang=-0.1)
rect.hclust(c1,2)
```



3. Recommend the top 2 pages for users of cluster 1.
For that purpose:

(a) inspect what were the pages visited by users in cluster 1;

```
dd <- mutate(d, Cluster=cm[d$USER]) # adding the cluster of each user
dd

## # A tibble: 17 x 3
##    USER  PAGE  Cluster
##    <chr> <chr>   <int>
##  1 u1    A           1
##  2 u1    B           1
##  3 u1    C           1
##  4 u2    A           1
##  5 u2    C           1
##  6 u3    B           2
##  7 u3    G           2
##  8 u3    F           2
##  9 u3    I           2
## 10 u4    B           1
## 11 u4    C           1
## 12 u5    G           2
## 13 u5    F           2
## 14 u5    I           2
## 15 u5    J           2
## 16 u6    A           1
## 17 u6    C           1

filter(dd,Cluster==1) %>% pull(PAGE) # the answer (the pages)

## [1] "A" "B" "C" "A" "C" "B" "C" "A" "C"
```

(b) inspect how many times each of these pages were visited;

```
filter(dd,Cluster==1) %>% group_by(PAGE) %>% tally()
```

```
## # A tibble: 3 x 2
##   PAGE      n
##   <chr> <int>
## 1 A         3
## 2 B         2
## 3 C         4
```

(c) sort the pages by decreasing order of visits;

```
filter(dd,Cluster==1) %>% group_by(PAGE) %>% tally(sort=TRUE)
```

```
## # A tibble: 3 x 2
##   PAGE      n
##   <chr> <int>
## 1 C         4
## 2 A         3
## 3 B         2
```

(d) obtain the top 2 pages for recommendation.

```
filter(dd,Cluster==1) %>%
  group_by(PAGE) %>% tally(sort=TRUE) %>% top_n(2) %>% pull(PAGE)
```

```
## [1] "C" "A"
```

4. Recommend the top 2 pages for users of cluster 2.

```
filter(dd,Cluster==1) %>%
  group_by(PAGE) %>% tally(sort=TRUE) %>% top_n(2) %>% pull(PAGE)
```

```
## [1] "C" "A"
```

5. Using the same clustering results, recommend the top 3 pages for user u2.
From that top pages you should remove the pages that the user has already visited.

```
cluster.u2 <- dd %>% filter(USER == "u2") %>% select(Cluster) %>% head(1) %>% pull()

rec.u2 <- filter(dd,Cluster==cluster.u2) %>%
  group_by(PAGE) %>% tally(sort=TRUE) %>% top_n(3) %>% select(PAGE)

seen.u2 <- dd %>% filter(USER == "u2") %>% select(PAGE)

anti_join(rec.u2,seen.u2)
```

```
## # A tibble: 1 x 1
##   PAGE
##   <chr>
## 1 B
```

Load the package `recommenderlab`.

2.2 Recommendation using Association Rules

6. Obtain a recommendation model using association rules with the first 6 users. For that purpose:

(a) start by coercing the data frame with user-page access information from the `log1.csv` file to a `binaryRatingMatrix` (`brm`);

```r
log <- read_csv("log1.csv",col_types = list(col_factor(),col_factor()))
brm <- as(as.data.frame(log),"binaryRatingMatrix")
```

(b) select the information on the first 6 users to be used as training offline data and save it to a new variable (e.g `brm_offline`);

```r
brm_offline <- brm[1:6,]
```

(c) inspect the content of `brm_offline`; use the function `getRatingMatrix` and `getData.frame`;

```r
getData.frame(brm_offline)
```

```
##    user item rating
## 1    u1    A      1
## 4    u1    B      1
## 7    u1    C      1
## 2    u2    A      1
## 8    u2    C      1
## 5    u3    B      1
## 11   u3    G      1
## 13   u3    F      1
## 15   u3    I      1
## 6    u4    B      1
## 9    u4    C      1
## 12   u5    G      1
## 14   u5    F      1
## 16   u5    I      1
## 17   u5    J      1
## 3    u6    A      1
## 10   u6    C      1
```

```r
getRatingMatrix(brm_offline)
```

```
## itemMatrix in sparse format with
##  6 rows (elements/transactions) and
##  7 columns (items)
```

```r
inspect(getRatingMatrix(brm_offline))
```

```
##     items
## [1] {A,B,C}
## [2] {A,C}
## [3] {B,G,F,I}
## [4] {B,C}
## [5] {G,F,I,J}
## [6] {A,C}
```

(d) apply the functions `rowCounts` and `colCounts` to `brm_offline`; what information does it give you?

```r
rowCounts(brm_offline)
```

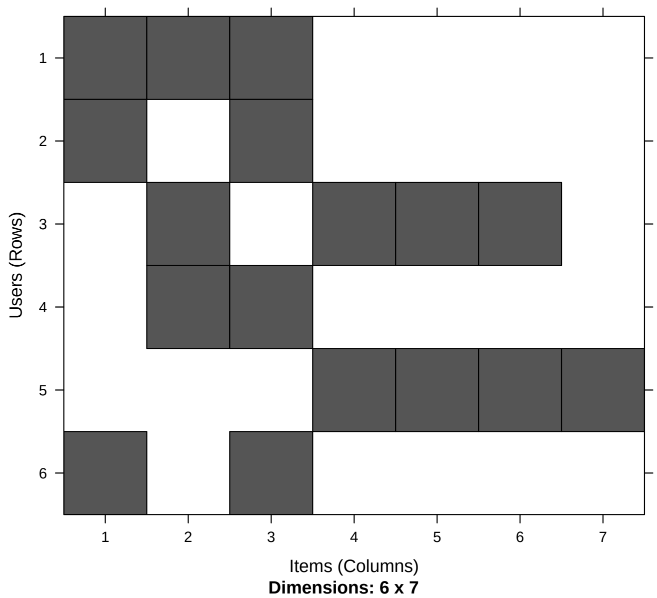
```
## u1 u2 u3 u4 u5 u6
##  3  2  4  2  4  2
```

```r
colCounts(brm_offline)
```

```
## A B C G F I J
## 3 3 4 2 2 2 1
```

(e) apply the function `image` to `brm_offline`;

```r
image(brm_offline)
```



(f) obtain the recommender model based on association rules with the instruction `modelAR <- Recommender(brm_offline,"AR")`

```
model <- Recommender(brm_offline, "AR")
```

(g) apply the function `getModel` to the obtained model and then inspect the association rules that compose the model.

```
getModel(model)

## $description
## [1] "AR: rule base"
##
## $rule_base
## set of 26 rules
##
## $support
## [1] 0.1
##
## $confidence
## [1] 0.8
##
## $maxlen
## [1] 3
##
## $sort_measure
## [1] "confidence"
##
## $sort_decreasing
## [1] TRUE
##
## $apriori_control
## $apriori_control$verbose
## [1] FALSE
##
##
## $verbose
## [1] FALSE

rules <- getModel(model)$rule_base
inspect(rules)
```

```
##      lhs      rhs support   confidence lift count
## [1]  {J}   => {G} 0.1666667 1          3.0  1
## [2]  {J}   => {F} 0.1666667 1          3.0  1
## [3]  {J}   => {I} 0.1666667 1          3.0  1
## [4]  {A}   => {C} 0.5000000 1          1.5  3
## [5]  {G}   => {F} 0.3333333 1          3.0  2
## [6]  {F}   => {G} 0.3333333 1          3.0  2
## [7]  {G}   => {I} 0.3333333 1          3.0  2
## [8]  {I}   => {G} 0.3333333 1          3.0  2
## [9]  {F}   => {I} 0.3333333 1          3.0  2
## [10] {I}   => {F} 0.3333333 1          3.0  2
## [11] {G,J} => {F} 0.1666667 1          3.0  1
## [12] {F,J} => {G} 0.1666667 1          3.0  1
## [13] {G,J} => {I} 0.1666667 1          3.0  1
## [14] {I,J} => {G} 0.1666667 1          3.0  1
## [15] {F,J} => {I} 0.1666667 1          3.0  1
## [16] {I,J} => {F} 0.1666667 1          3.0  1
## [17] {A,B} => {C} 0.1666667 1          1.5  1
## [18] {G,F} => {I} 0.3333333 1          3.0  2
## [19] {G,I} => {F} 0.3333333 1          3.0  2
## [20] {F,I} => {G} 0.3333333 1          3.0  2
## [21] {B,G} => {F} 0.1666667 1          3.0  1
## [22] {B,F} => {G} 0.1666667 1          3.0  1
## [23] {B,G} => {I} 0.1666667 1          3.0  1
## [24] {B,I} => {G} 0.1666667 1          3.0  1
## [25] {B,F} => {I} 0.1666667 1          3.0  1
## [26] {B,I} => {F} 0.1666667 1          3.0  1
```

7. Suppose that `u7` enters the system and becomes an active user. Deploy the recommendation model for him/her.

For that purpose:

(a) apply the `predict` function with the model and the rating matrix of the user, such that only the top 2 recommendations are given as output;

(b) apply the function `getList` to the obtained predictions to inspect the actual recommendations; which are they?

(c) to comprove the obtained recommendations, filter the rules which have been triggered for this active user.

```
brm_u7 <- brm[7,]
recsAR <- predict(model, brm_u7, n=2)
recsAR

## Recommendations as 'topNList' with n = 2 for 1 users.

getList(recsAR)

## $u7
## [1] "G" "I"

r <- subset(rules,lhs %in% c("C","F"))
inspect(r)

##     lhs      rhs support   confidence lift count
## [1] {F}   => {G} 0.3333333 1          3    2
## [2] {F}   => {I} 0.3333333 1          3    2
## [3] {F,J} => {G} 0.1666667 1          3    1
## [4] {F,J} => {I} 0.1666667 1          3    1
## [5] {G,F} => {I} 0.3333333 1          3    2
## [6] {F,I} => {G} 0.3333333 1          3    2
## [7] {B,F} => {G} 0.1666667 1          3    1
## [8] {B,F} => {I} 0.1666667 1          3    1
```

8. Now suppose that `u8` enters the system and becomes an active user. Deploy the recommendation model for him/her. Be critical regarding the results.

```
brm_u8 <- brm[8,]
recsAR <- predict(model, brm_u8, n=2)
recsAR

## Recommendations as 'topNList' with n = 2 for 1 users.

getList(recsAR)

## $u8
## character(0)

r <- subset(rules,lhs %in% c("C"))
inspect(r)
```

9. Explore the types of recommendation models available for binary rating matrices.

`recommenderRegistry$get_entries(dataType ="binaryRatingMatrix")`

10. Make the top 2 recommendations to u7 and u8 using the popularity of the pages, instead of association rules. Try to understand the obtained recommendations.

```
modelPop <- Recommender(brm_offline,"POPULAR")
recsPop <- predict(modelPop, brm[7:8,], n=2)
recsPop

## Recommendations as 'topNList' with n = 2 for 2 users.

getList(recsPop)

## $u7
## [1] "B" "A"
##
## $u8
## [1] "B" "A"
```

2.3 Recommendation using Collaborative Filtering

Binary Rating Data

Considering the same binary rating matrix of the previous exercise `brm_offline`, build a recommendation model based on collaborative filtering.

11. Start by using the function `similarity` to build the similarity cosine matrix for:

(a) an user-based approach;

```
simCos_users<- similarity(brm_offline,method="cosine")
simCos_users

##           u1        u2        u3        u4        u5
## u2 0.8164966
## u3 0.2886751 0.0000000
## u4 0.8164966 0.5000000 0.3535534
## u5 0.0000000 0.0000000 0.7500000 0.0000000
## u6 0.8164966 1.0000000 0.0000000 0.5000000 0.0000000
```

(b) an item-based approach.

```
simCos_items <- similarity(brm_offline,method="cosine",which="items")
simCos_items

##          A         B         C         G         F         I
## B 0.3333333
## C 0.8660254 0.5773503
## G 0.0000000 0.4082483 0.0000000
## F 0.0000000 0.4082483 0.0000000 1.0000000
## I 0.0000000 0.4082483 0.0000000 1.0000000 1.0000000
## J 0.0000000 0.0000000 0.0000000 0.7071068 0.7071068 0.7071068
```

12. Obtain the top 2 recommendations with user-based CF and item-based CF methods using the cosine similarity with a neighborhood of size 3, for:

```
modelUBCF <- Recommender(brm_offline, "UBCF",parameter=list(method="cosine",nn=3)) #weighted=FALSE
getModel(modelUBCF)

## $description
## [1] "UBCF-Binary Data: contains full or sample of data set"
##
## $data
## 6 x 7 rating matrix of class 'binaryRatingMatrix' with 17 ratings.
##
## $method
## [1] "cosine"
##
## $nn
## [1] 3
##
## $weighted
## [1] TRUE
##
## $sample
## [1] FALSE
##
## $verbose
## [1] FALSE

modelIBCF <- Recommender(brm_offline, "IBCF",parameter=list(method="cosine",k=3))
getModel(modelIBCF)

## $description
## [1] "IBCF: Reduced similarity matrix"
##
## $sim
## 7 x 7 sparse Matrix of class "dgCMatrix"
##           A         B         C         G         F         I         J
## A .         0.3333333 0.8660254 .         .         .         .
## B .         .         0.5773503 .         0.4082483 0.4082483 .
## C 0.8660254 0.5773503 .         .         .         .         .
## G .         .         .         .         1.0000000 1.0000000 0.7071068
## F .         .         .         1.0000000 .         1.0000000 0.7071068
## I .         .         .         1.0000000 1.0000000 .         0.7071068
## J .         .         .         0.7071068 0.7071068 0.7071068 .
##
## $k
## [1] 3
##
## $method
## [1] "cosine"
##
## $normalize_sim_matrix
## [1] FALSE
##
## $alpha
## [1] 0.5
##
## $verbose
## [1] FALSE
```

(a) active user u8;

```
recsUBCF <- predict(modelUBCF, brm_u8, n=2)
recsUBCF

## Recommendations as 'topNList' with n = 2 for 1 users.

getList(recsUBCF)

## $u8
## [1] "A" "B"

similarity(brm_offline,brm[8,],method="cosine")
```

```
##           u8
## u1 0.5773503
## u2 0.7071068
## u3 0.0000000
## u4 0.7071068
## u5 0.0000000
## u6 0.7071068
## attr(,"method")
## [1] "cosine"
## attr(,"type")
## [1] "simil"
```

```
recsIBCF <- predict(modelIBCF, brm_u8, n=2)
recsIBCF
```

```
## Recommendations as 'topNList' with n = 2 for 1 users.
```

```
getList(recsIBCF)
```

```
## $u8
## [1] "A" "B"
```

(b) active user u7.

```
recsUBCF <- predict(modelUBCF, brm_u7, n=2)
getList(recsUBCF)
```

```
## $u7
## [1] "A" "B"
```

```
similarity(brm_offline,brm[7,],method="cosine")
```

```
##           u7
## u1 0.4082483
## u2 0.5000000
## u3 0.3535534
## u4 0.5000000
## u5 0.3535534
## u6 0.5000000
## attr(,"method")
## [1] "cosine"
## attr(,"type")
## [1] "simil"
```

```
recsIBCF <- predict(modelIBCF, brm_u7, n=2)
recsIBCF
```

```
## Recommendations as 'topNList' with n = 2 for 1 users.
```

```
getList(recsIBCF)
```

```
## $u7
## [1] "G" "I"
```

```
similarity(brm[1:7,],method="cosine",which="items")
```

```
##           A         B         C         G         F         I
## B 0.3333333
## C 0.7745967 0.5163978
## G 0.0000000 0.4082483 0.0000000
## F 0.0000000 0.3333333 0.2581989 0.8164966
## I 0.0000000 0.4082483 0.0000000 1.0000000 0.8164966
## J 0.0000000 0.0000000 0.0000000 0.7071068 0.5773503 0.7071068
```

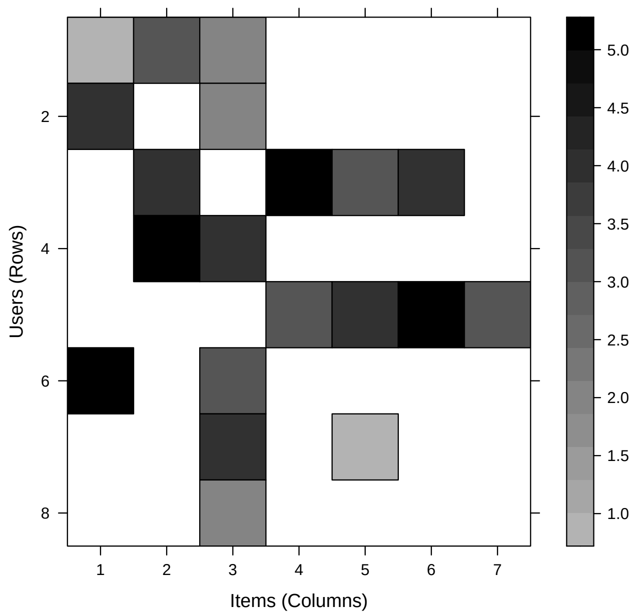
Non-Binary Rating Data

13. Explore the types of recommendation models available for real rating matrices.

```
recommenderRegistry$get_entries(dataType ="realRatingMatrix")
```

14. Read the file `log1Ratings.csv`, containing information on the ratings given to web pages by a set of users, into a data frame in R. Build an deploy the following colaborative filtering recommendation models using, again, the first 6 users for training:

```
logR <- read_csv("log1Ratings.csv",col_types = list(col_factor(),col_factor(),col_integer()))
rrm <- as(as.data.frame(logR),"realRatingMatrix")
image(rrm)
```




```
rrm_offline <- rrm[1:6,]

getRatingMatrix(rrm_offline)

## 6 x 7 sparse Matrix of class "dgCMatrix"
##    A B C G F I J
## u1 1 3 2 . . . .
## u2 4 . 2 . . . .
## u3 . 4 . 5 3 4 .
## u4 . 5 4 . . . .
## u5 . . . 3 4 5 3
## u6 5 . 3 . . . .

similarity(rrm_offline,method="cosine")

##           u1        u2        u3        u4        u5
## u2 0.8000000
## u3 1.0000000        NA
## u4 0.9962406 1.0000000 1.0000000
## u5        NA        NA 0.9400000        NA
## u6 0.8436615 0.9970545        NA 1.0000000        NA

similarity(rrm_offline,method="cosine",which="items")

##           A         B         C         G         F         I
## B 1.0000000
## C 0.9356015 0.9970545
## G        NA 1.0000000        NA
## F        NA 1.0000000        NA 0.9260924
## I        NA 1.0000000        NA 0.9374253 0.9995121
## J        NA        NA        NA 1.0000000 1.0000000 1.0000000
```

(a) an user-based CF approach with two neighbours to predict the ratings of users u7 and u8;

```
modelUBCF_R <- Recommender(rrm_offline, "UBCF",parameter=list(nn=2))
getModel(modelUBCF_R)
```

```
## $description
## [1] "UBCF-Real data: contains full or sample of data set"
##
## $data
## 6 x 7 rating matrix of class 'realRatingMatrix' with 17 ratings.
## Normalized using center on rows.
##
## $method
## [1] "cosine"
##
## $nn
## [1] 2
##
## $sample
## [1] FALSE
##
## $normalize
## [1] "center"
##
## $verbose
## [1] FALSE
```

```
# user u8
recsUBCF_R <- predict(modelUBCF_R,rrm[8,],type="ratings")
getList(recsUBCF_R)
```

```
## $u8
## A B G F I J
## 1 3 2 2 2 2
```

```
# user u7
recsUBCF_R <- predict(modelUBCF_R,rrm[7,],type="ratings")
getList(recsUBCF_R)
```

```
## $u7
##   A   B   G   I   J
## 3.0 2.5 3.0 2.5 2.5
```

(b) an item-based CF approach with two neighbours to predict the ratings of users u7 and u8.

```
modelIBCF_R <- Recommender(rrm_offline, "IBCF",parameter=list(k=2))
getModel(modelIBCF_R)
```

```
## $description
## [1] "IBCF: Reduced similarity matrix"
##
## $sim
## 7 x 7 sparse Matrix of class "dgCMatrix"
##           A         B         C         G         F I J
## A .         1.0000000 0.8164966 .         .         . .
## B 1.0000000 .         .         .         .         1 .
## C 0.8164966 0.4472136 .         .         .         . .
## G .         .         .         .         0.9216354 . 1
## F .         .         .         0.9216354 .         . 1
## I .         1.0000000 .         .         .         . 1
## J .         .         .         .         1.0000000 1 .
##
## $k
## [1] 2
##
## $method
## [1] "Cosine"
##
## $normalize
## [1] "center"
##
## $normalize_sim_matrix
## [1] FALSE
```

```
##
## $alpha
## [1] 0.5
##
## $na_as_zero
## [1] FALSE
##
## $verbose
## [1] FALSE
```

```
# user u7
recsIBCF_R <- predict(modelIBCF_R,rrm[7,],type="ratings")
getList(recsIBCF_R)
```

```
## $u7
## A G J
## 4 1 1
```

```
# user u8
recsIBCF_R <- predict(modelIBCF_R,rrm[8,],type="ratings")
getList(recsIBCF_R)
```

```
## $u8
## named numeric(0)
```

15. Considering the log1 binary data, evaluate different recommendation strategies.

(a) Set the seed to 2021. Use the function `evaluationScheme` to define an evaluation scheme that splits the data into train and test set (80%-20% proportion) and establishes that 2 items of test cases are already known. In case that one or more users do not comply with this setting, you can disregard them.

```
log <- read_csv("log1.csv",col_types = list(col_factor(),col_factor()))
brm <- as(as.data.frame(log),"binaryRatingMatrix")

set.seed(2021) # for replication of results
e <- evaluationScheme(brm, method="split", train=0.8, given = 2)
```

```
## Error in .local(data, ...):  Some observations have size<given!
```

```
brm <- brm[rowCounts(brm)>=2,]
e <- evaluationScheme(brm, method="split", train=0.8, given = 2)
e
```

```
## Evaluation scheme with 2 items given
## Method: 'split' with 1 run(s).
## Training set proportion: 0.800
## Good ratings: NA
## Data set: 7 x 7 rating matrix of class 'binaryRatingMatrix' with 19 ratings.
```

(b) Check how the data was splitted according to the previous evaluation scheme, using the function `getData` on the evaluation scheme with the arguments `"train"`, `"known"` and `"unknown"`.

```
inspect(getRatingMatrix(brm))
```

```
##     items
## [1] {A,B,C}
## [2] {A,C}
## [3] {B,G,F,I}
## [4] {B,C}
## [5] {G,F,I,J}
## [6] {A,C}
## [7] {C,F}
```

```
inspect(getRatingMatrix(getData(e,"train")))
```

```
##     items
## [1] {C,F}
## [2] {A,C}
## [3] {A,C}
## [4] {B,G,F,I}
## [5] {G,F,I,J}
```

```
as(getRatingMatrix(getData(e,"train")),"matrix")
```

```
##        A     B     C     G     F     I     J
## u7 FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
## u6  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE
## u2  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE
## u3 FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## u5 FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

```
inspect(getRatingMatrix(getData(e,"known")))
```

```
##     items
## [1] {A,B}
## [2] {B,C}
```

```
inspect(getRatingMatrix(getData(e,"unknown")))
```

```
##     items
## [1] {C}
## [2] {}
```

(c) Define the list of methods that will be used to obtain the top N recommendations, as follows:

```
methods <- list(
   "popular" = list(name="POPULAR", param = NULL),
   "user-based CF" = list(name="UBCF", param = NULL)
   "item-based CF" = list(name="IBCF", param = NULL)
)
```

(d) Use the function `evaluate` with the previously defined evaluation scheme, methods and considering top 1, 3 and 5 recommendations for each of the models.

```
methods <- list(
  "popular" = list(name="POPULAR", param = NULL),
  "user-based CF" = list(name="UBCF", param = NULL),
  "item-based CF" = list(name="IBCF", param = NULL)
)

results <- evaluate(e, methods, type="topNList", n=c(1,3,5))
```

```
## POPULAR run fold/sample [model time/prediction time]
##   1  [0.002sec/0.005sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0sec/0.008sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [0.001sec/0.007sec]
```

(e) Explore the obtained object.

```
results
```

```
## List of evaluation results for 3 recommenders:
## Evaluation results for 1 folds/samples using method 'POPULAR'.
## Evaluation results for 1 folds/samples using method 'UBCF'.
## Evaluation results for 1 folds/samples using method 'IBCF'.
```

```
class(results)
```

```
## [1] "evaluationResultList"
## attr(,"package")
## [1] "recommenderlab"
```

```
avg(results) # just one run
```

```
## $popular
##    TP  FP  FN  TN precision recall TPR   FPR
## 1 0.0 1.0 0.5 3.5 0.0000000      0   0 0.225
## 3 0.5 2.5 0.0 2.0 0.1666667      1   1 0.550
## 5 0.5 4.5 0.0 0.0 0.1000000      1   1 1.000
##
## $`user-based CF`
##    TP  FP FN TN precision recall TPR  FPR
## 1 0.5 0.5  0  4 0.5000000      1   1 0.10
## 3 0.5 2.5  0  2 0.1666667      1   1 0.55
## 5 0.5 4.5  0  0 0.1000000      1   1 1.00
##
## $`item-based CF`
##    TP  FP FN TN precision recall TPR   FPR
## 1 0.5 0.5  0  4 0.5000000      1   1 0.100
## 3 0.5 2.5  0  2 0.1666667      1   1 0.550
## 5 0.5 3.5  0  1 0.1250000      1   1 0.775
```

```
names(results)
```

```
## [1] "popular"       "user-based CF" "item-based CF"
```

```
results[["popular"]]
```

```
## Evaluation results for 1 folds/samples using method 'POPULAR'.
```

(f) Use the function `getConfusionMatrix` on one of the methods to obtain the corresponding confusion matrices. Be critical regarding the values that are shown.

```
getConfusionMatrix(results[["popular"]])
```

```
## [[1]]
##    TP  FP  FN  TN precision recall TPR   FPR
## 1 0.0 1.0 0.5 3.5 0.0000000      0   0 0.225
## 3 0.5 2.5 0.0 2.0 0.1666667      1   1 0.550
## 5 0.5 4.5 0.0 0.0 0.1000000      1   1 1.000
```

```
model1 <- Recommender(getData(e,"train"), "POPULAR")
preds1 <- predict(model1,getData(e,"known"),n=3)
getList(preds1)
```

```
## $u1
## [1] "C" "F" "I"
##
## $u4
## [1] "F" "A" "I"
```

```
getConfusionMatrix(results[["user-based CF"]])
```

```
## [[1]]
##    TP  FP FN TN precision recall TPR  FPR
## 1 0.5 0.5  0  4 0.5000000      1   1 0.10
## 3 0.5 2.5  0  2 0.1666667      1   1 0.55
## 5 0.5 4.5  0  0 0.1000000      1   1 1.00
```

```
model2 <- Recommender(getData(e,"train"), "UBCF")
preds2 <- predict(model2,getData(e,"known"),n=2)
getList(preds2)
```

```
## $u1
## [1] "C" "G"
##
## $u4
## [1] "A" "F"
```

```r
getConfusionMatrix(results[["item-based CF"]])
```

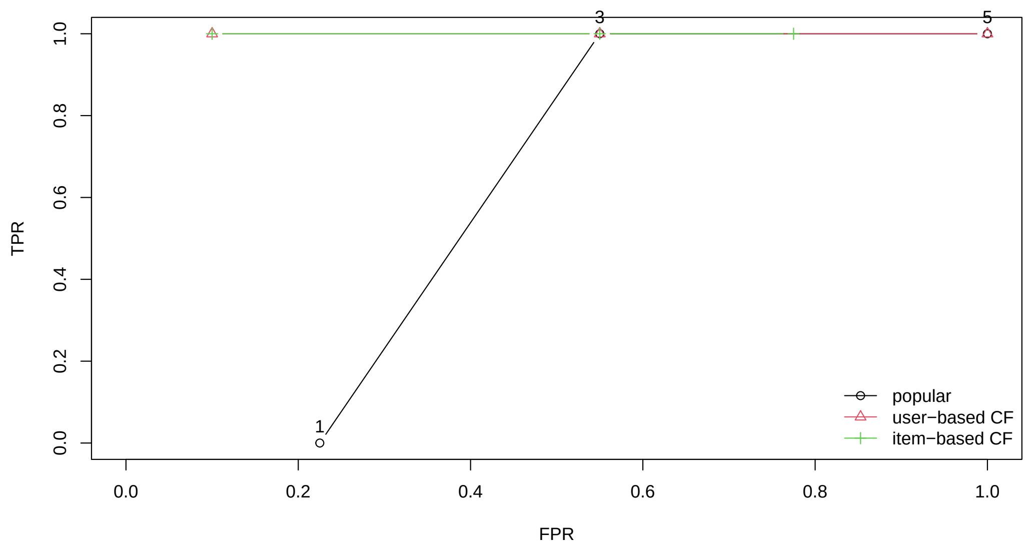
```
## [[1]]
##    TP  FP FN TN precision recall TPR   FPR
## 1 0.5 0.5  0  4 0.5000000      1   1 0.100
## 3 0.5 2.5  0  2 0.1666667      1   1 0.550
## 5 0.5 3.5  0  1 0.1250000      1   1 0.775
```

```r
model3 <- Recommender(getData(e,"train"), "IBCF")
preds3 <- predict(model3,getData(e,"known"),n=2)
getList(preds3)
```

```
## $u1
## [1] "C" "G"
##
## $u4
## [1] "A" "G"
```

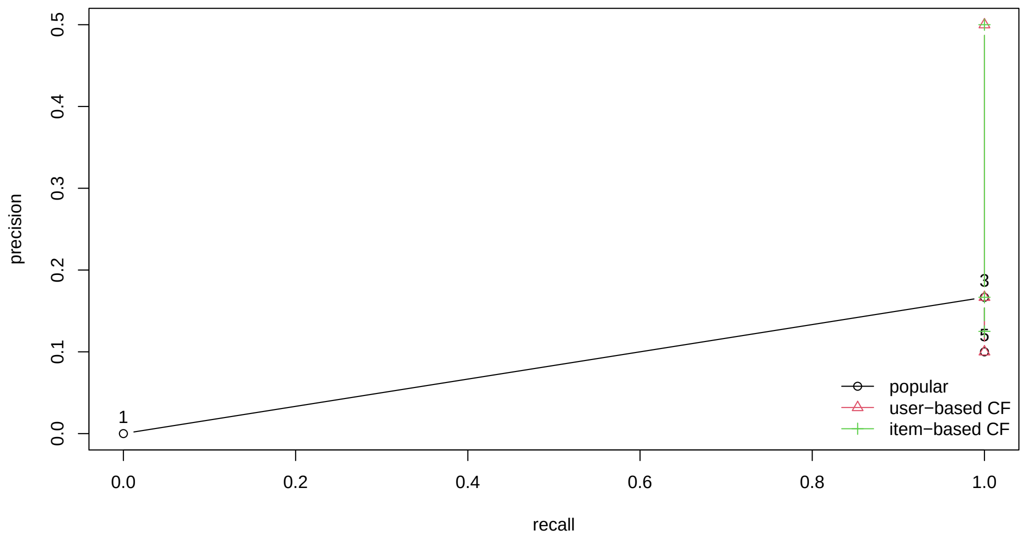
(g) Plot the ROC curves for each of the methods and different values of N. What can you conclude?

```r
plot(results,annotate=TRUE)
```



(h) Plot the precision/recall curves for each of the methods and different values of N. What can you conclude?

```r
plot(results, "prec/rec", annotate=TRUE)
```


precision
0.0
0.1
0.2
0.3
0.4
0.5
recall
0.0
0.2
0.4
0.6
0.8
1.0
1
3
5
popular
user−based CF
item−based CF