CYO-OTI Class Recommender

Robert Lewis

June 17, 2019

Project Introduction

The purpose of this project (CYO) is to create recommending of classes for past users based on courses they have previously completed. I will be using a subset of modified data to protect the actual users infomation. I will use the recommenderlab for making my recommendation and use different models to identify the best model with the lowest error accuracy. The idea is that given Student Course Completion data by many students for many classes and grades, one can recommend other classes not known to her or him from similiar grades (see, e.g., Goldberg, Nichols, Oki, and Terry 1992)

Load data

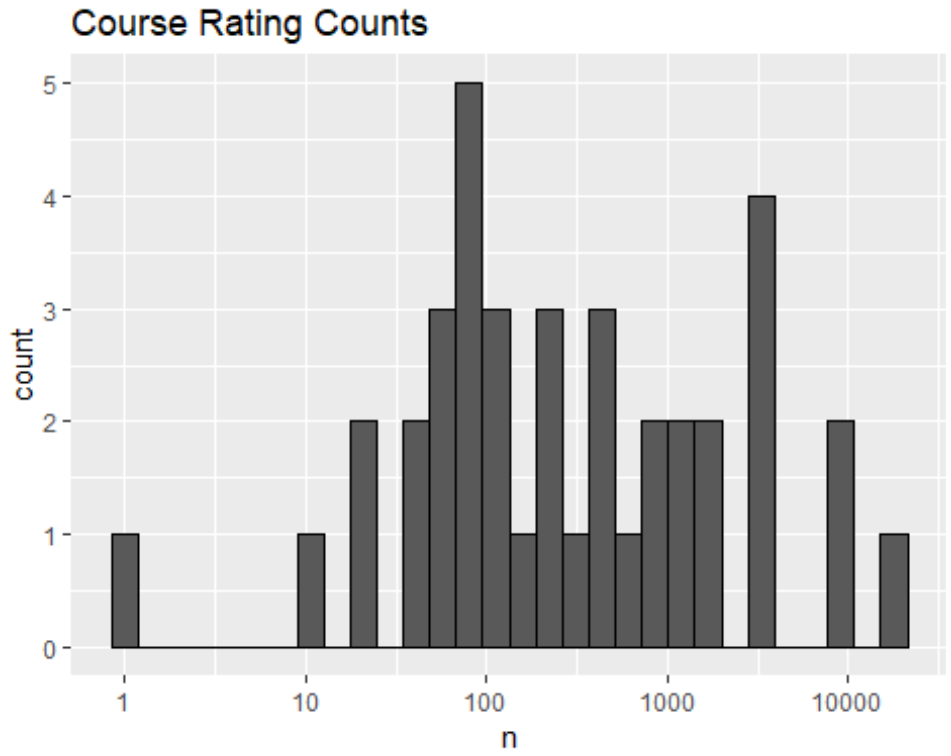
Review structure and details of dataset

```
## Total Number of Courses Completed: 58483

## Total number of Students: 36436

## Total number of Courses: 39

## Total number of columns: 6

## Column Names: StudentID CourseID CourseName X X.1 Rating

## Structure of data:

## 'data.frame':    58483 obs. of  6 variables:
##  $ StudentID : int  50000 50000 50000 50001 50001 50001 50003 50004 50005
50007 ...
##  $ CourseID  : int  168 66 84 166 168 88 5 166 165 165 ...
##  $ CourseName: Factor w/ 38 levels "10 Hour Construction Industry",..: 4 8
33 2 4 14 13 2 1 1 ...
##  $ X         : logi  NA NA NA NA NA NA ...
##  $ X.1       : logi  NA NA NA NA NA NA ...
##  $ Rating    : num  89 71 74 74 91.5 73 85.5 77 96 72 ...

## Summary of data:

##    StudentID        CourseID
##  Min.   :50000   Min.   :  5.0
##  1st Qu.:62491   1st Qu.: 83.0
##  Median :75036   Median :165.0
##  Mean   :75038   Mean   :128.8
```

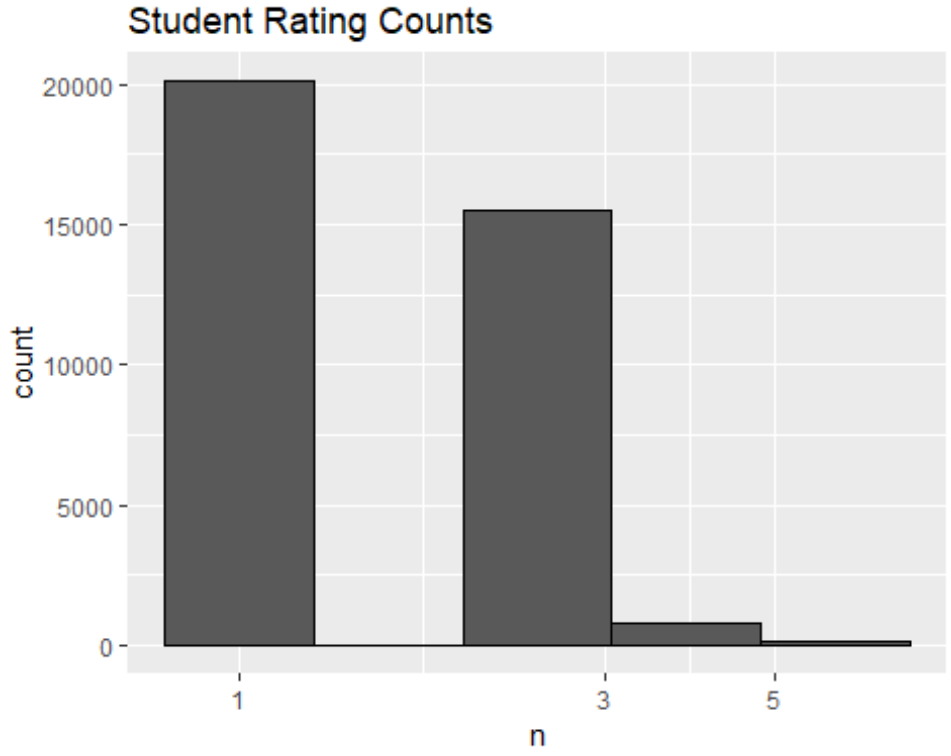
```
##  3rd Qu.:87576   3rd Qu.:166.0
##  Max.   :99999   Max.   :168.0
##
##                                                  CourseName       X
##  10 Hour Construction Industry                        :17929   Mode:logical
##  10 Hour General Industry                             : 8387   NA's:58483
##  30 Hour Construction Industry                        : 8055
##  Hazardous Waste Operations and Emergency Response: 4567
##  OSHA Fall Protection Course                          : 3140
##  30 Hour  General Industry                            : 3092
##  (Other)                                              :13313
##    X.1              Rating
##  Mode:logical   Min.   : 71.00
##  NA's:58483     1st Qu.: 78.00
##                 Median : 85.50
##                 Mean   : 85.51
##                 3rd Qu.: 93.00
##                 Max.   :100.00
##

## Sample of Student Data

##       StudentID  5  6 48 65 66 68 69 71 72 73 74
##  [1,]     50000 NA NA NA NA  1 NA NA NA NA NA NA
##  [2,]     50001 NA NA NA NA NA NA NA NA NA NA NA
##  [3,]     50003  1 NA NA NA NA NA NA NA NA NA NA
##  [4,]     50004 NA NA NA NA NA NA NA NA NA NA NA
##  [5,]     50005 NA NA NA NA NA NA NA NA NA NA NA
##  [6,]     50007 NA NA NA NA NA NA NA NA NA NA NA
##  [7,]     50012 NA NA NA  1 NA NA NA NA NA NA NA
##  [8,]     50013 NA NA NA NA NA NA NA NA NA NA NA
##  [9,]     50014 NA NA NA NA NA NA NA NA NA NA NA
## [10,]     50015 NA NA NA NA NA NA NA NA NA NA NA
## [11,]     50016 NA NA NA NA  1 NA NA NA NA NA NA
## [12,]     50017 NA NA NA NA NA NA NA NA NA NA  1
```

Note: Rating only range from 70 to 99, this may restrict be restrictive of my results.

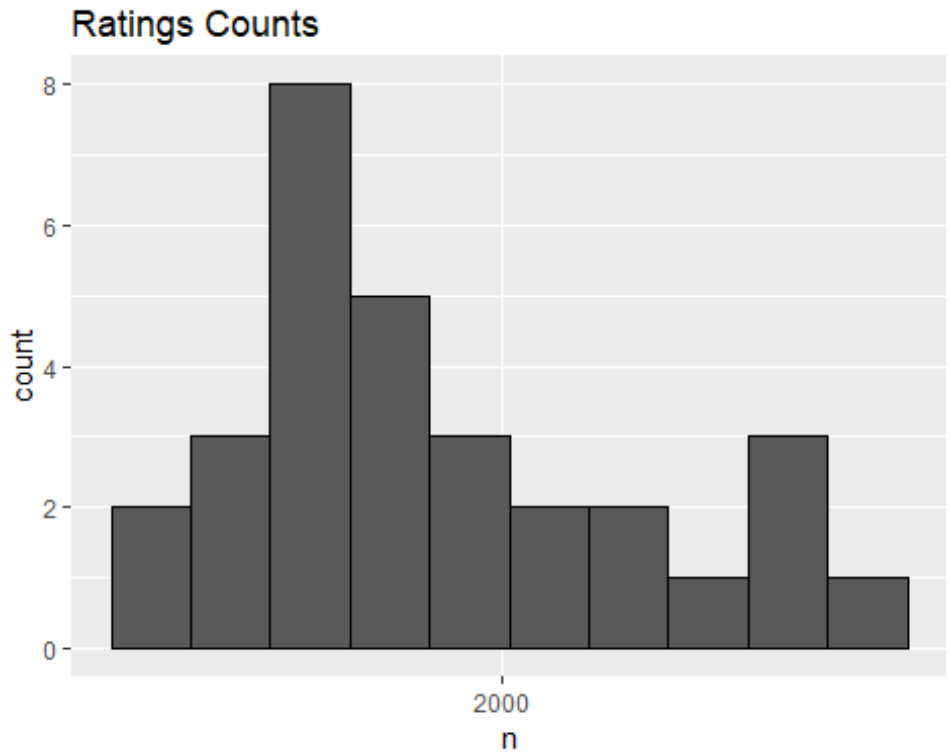
Visualize Initial data



*Note It appears in my dataset that less than 100 students have taken 5 classes



*Note: 20000+ students have taken 1 class, followed closely 3 then very few have taken 2, 4 and 5.



Preparing Data:

```
# Due to error in discovered in creating evaluation schemes i.e.: Some observations have size<given!
# I will remove the students with only one course completed.
one_class_students <- Student_data %>%
  dplyr::count(StudentID) %>%
  filter(n <= 1)

cat("Total Number of 1 course students to remove:\t")

## Total Number of 1 course students to remove:

count(one_class_students)

## # A tibble: 1 x 1
##       n
##   <int>
## 1 20104

cat("\n\n")

keep_StudentID <- Student_data %>%
  dplyr::count(StudentID) %>%
  filter(n > 1) %>%
```

```
  pull(StudentID)

Student_data1 <- Student_data %>% filter(StudentID %in% keep_StudentID)

summary(Student_data1)
```

```
##    StudentID        CourseID
##  Min.   :50000   Min.   :  5.0
##  1st Qu.:62433   1st Qu.: 83.0
##  Median :74986   Median :165.0
##  Mean   :75050   Mean   :125.8
##  3rd Qu.:87630   3rd Qu.:166.0
##  Max.   :99999   Max.   :168.0
##
##                                                 CourseName       X
##  10 Hour Construction Industry                       :10144   Mode:logical
##  10 Hour General Industry                            : 5611   NA's:38379
##  30 Hour Construction Industry                       : 5479
##  Hazardous Waste Operations and Emergency Response: 3233
##  OSHA Fall Protection Course                         : 2199
##  30 Hour  General Industry                           : 2165
##  (Other)                                             : 9548
##    X.1              Rating
##  Mode:logical   Min.   : 71.0
##  NA's:38379     1st Qu.: 78.0
##                 Median : 85.5
##                 Mean   : 85.5
##                 3rd Qu.: 93.0
##                 Max.   :100.0
##
```

Convert to realRatingMatrix

1. I used dcast.data.table to cast the data.frame as a table

2. I used sprintf to convert MovieId's, and UserId's to chr

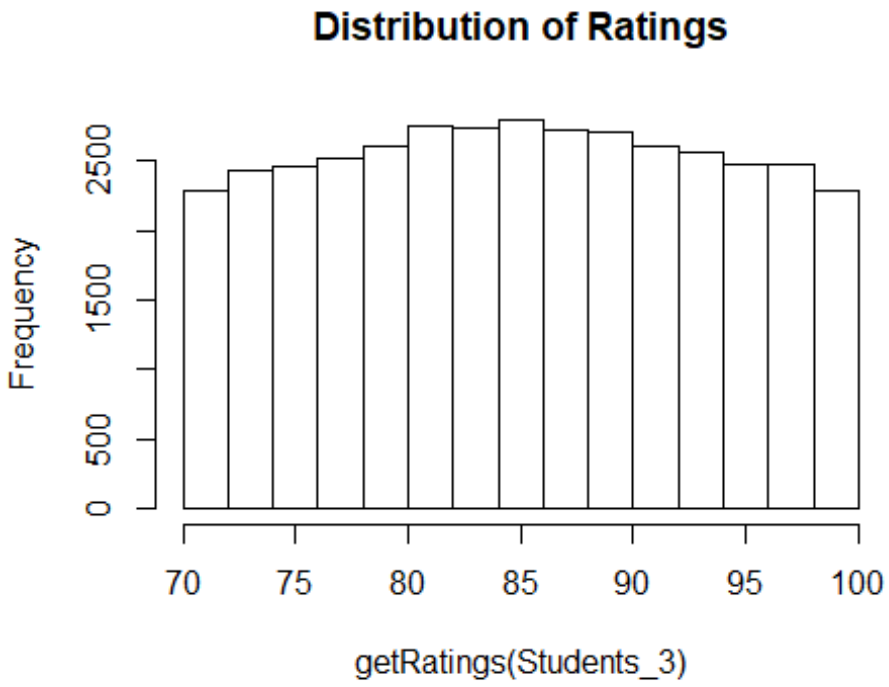
3. I used corner to view a small sample of the data to verify conversion

4. I then convert the data ta a matrix and then a realRatingMatrix

```
## [1] 16332    39

## [1] "data.frame"

##   StudentID  5  6 65 66
## 1     50000 NA NA NA 71
## 2     50001 NA NA NA NA
## 3     50007 NA NA NA NA
## 4     50012 NA NA 84 NA
## 5     50014 NA NA NA NA
```

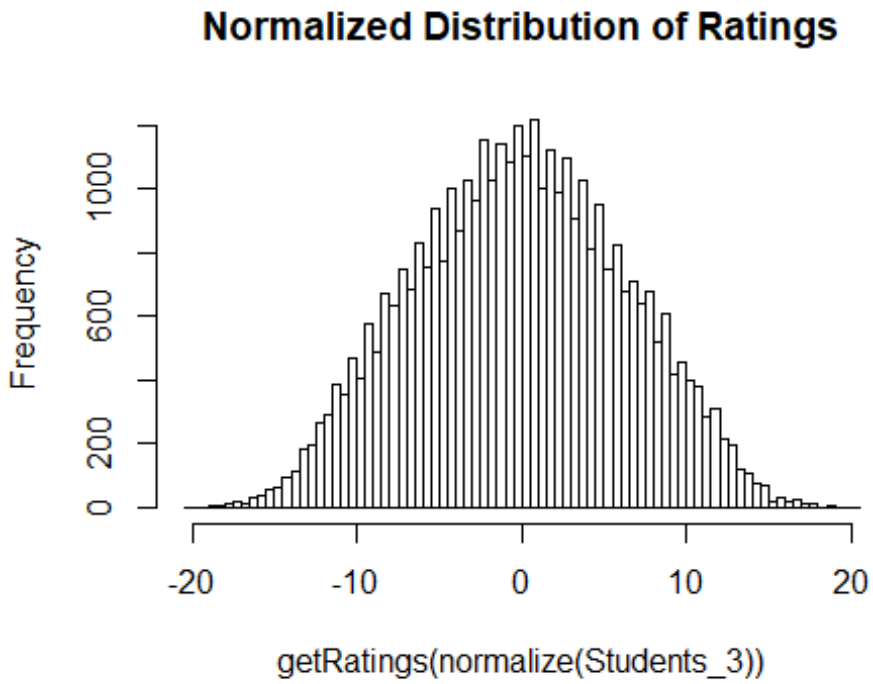
```
##                5  6 65 66 68
## Student_50000 NA NA NA 71 NA
## Student_50001 NA NA NA NA NA
## Student_50007 NA NA NA NA NA
## Student_50012 NA NA 84 NA NA
## Student_50014 NA NA NA NA NA

##               Course_5 Course_6 Course_65 Course_66 Course_68
## Student_50000       NA       NA        NA        71        NA
## Student_50001       NA       NA        NA        NA        NA
## Student_50007       NA       NA        NA        NA        NA
## Student_50012       NA       NA        84        NA        NA
## Student_50014       NA       NA        NA        NA        NA

## [1] 16332    38

## [1] "matrix"

## [1] "realRatingMatrix"
## attr(,"package")
## [1] "recommenderlab"

## Formal class 'realRatingMatrix' [package "recommenderlab"] with 2 slots
##   ..@ data     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. .. ..@ i       : int [1:38379] 8 23 24 31 50 56 61 67 68 77 ...
##   .. .. ..@ p       : int [1:39] 0 2117 3343 4091 6576 6634 6643 6686 6703
6748 ...
##   .. .. ..@ Dim     : int [1:2] 16332 38
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : chr [1:16332] "Student_50000" "Student_50001"
"Student_50007" "Student_50012" ...
##   .. .. .. ..$ : chr [1:38] "Course_5" "Course_6" "Course_65" "Course_66"
...
##   .. .. ..@ x       : num [1:38379] 71 91 73 83 78 96 74 71 88 81 ...
##   .. .. ..@ factors : list()
##   ..@ normalize: NULL
```

Review the prepared dataset

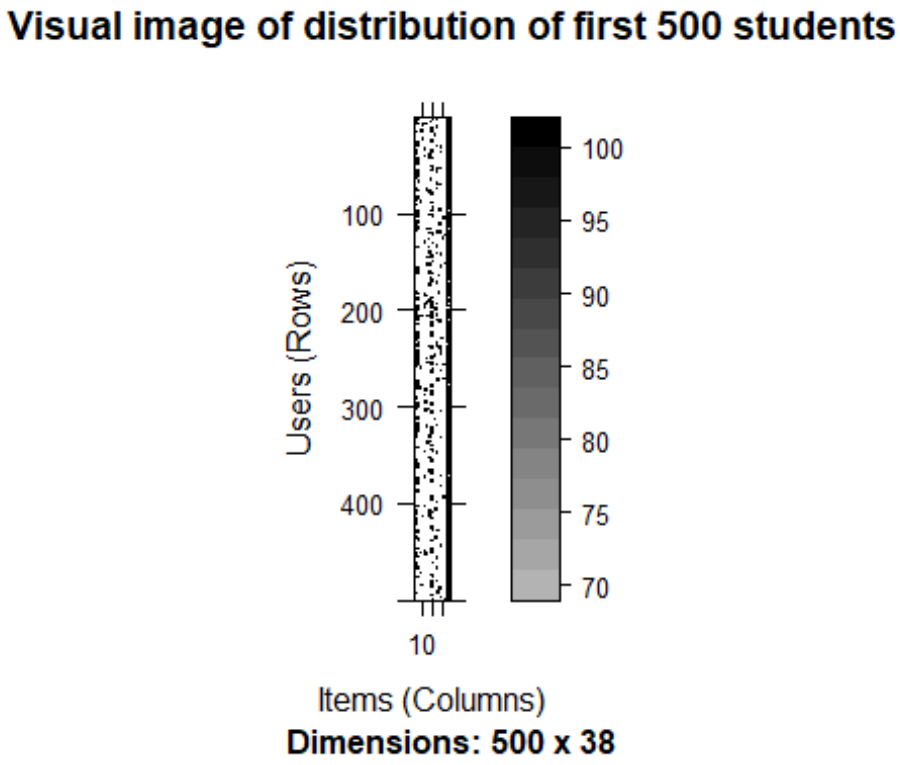


The Distribution of Ratings shows we have different possible ratings from 70 to 100. This data is based on the students who completed courses which reqires a score of 70% or higher to complete, note the frequencies are all fair similiar.

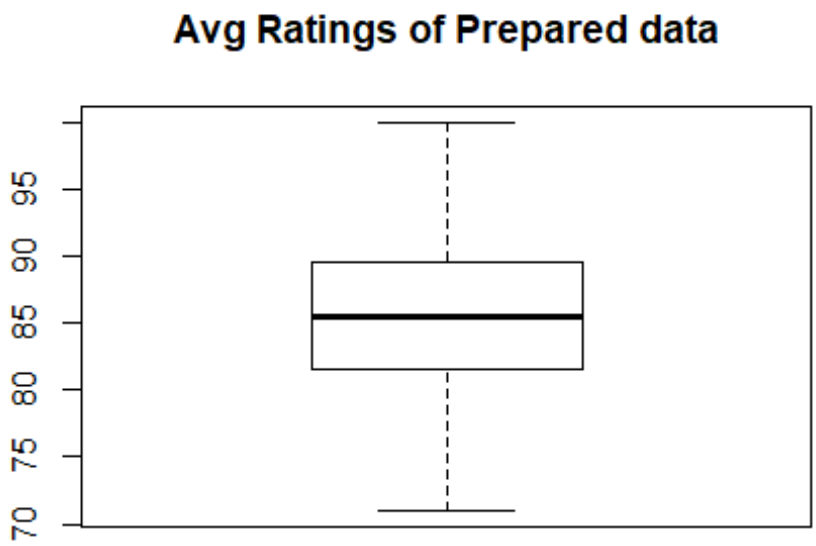
```
hist(getRatings(normalize(Students_3)),breaks = 100,main = "Normalized
Distribution of Ratings")
```



Data appears to be normalized

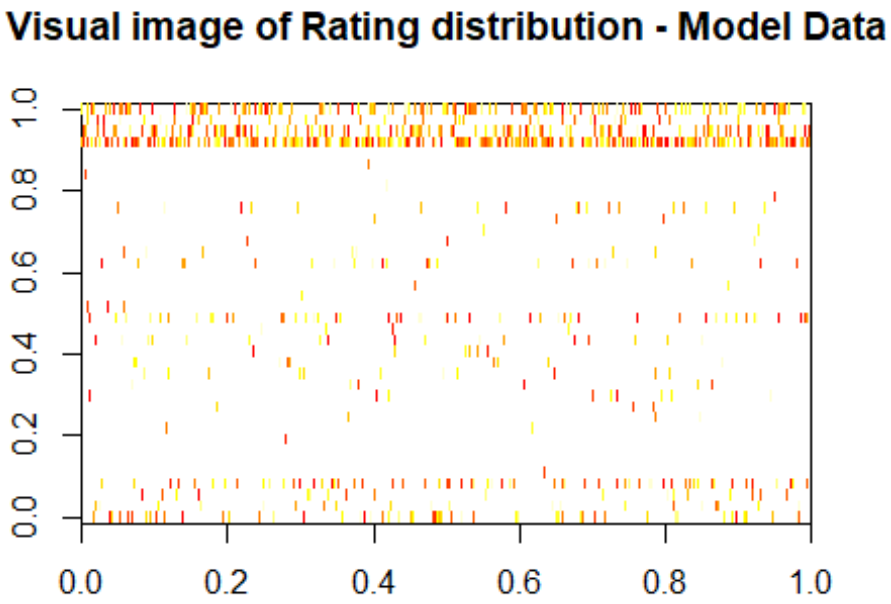


It appears the scores are evenly disriputed



No outliers appear to be found that may skew the results

```
## [1] 16332    38

## [1] 38379

image(as(model_data,"matrix"), main = "Visual image of Rating distribution -
Model Data")
```



Note Prepared Model ratings data appears slightly skewed.

Analyze number of course ratings per student:

```
##
##     2     3     4     5     6
## 11619  3843   758    92    20
```

Note that 11619 have rated 2 classes, we have a few Outlier who have rated 6 classes, will keep for now performance ihas not been a issue

Create Modeling datasets

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE

## [1] 12992    38

## [1] 3340   38
```

Build Models Options

I will use various models then compare prediction accuracies to determine the best algorithim. The available models I will first use a user-based collaborative filtering algorithm (UBCF), then item-based collaborative filtering (IBCF) and item popularity algorithms (POPULAR).

User Based Collborative Filtering (UBCF) Model

Collaborative filtering uses algorithims to filter users ratings to make personalized recommendations from similiar users (definition from whatis.techtarget.com/definition/collaborative-filtering).

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 12992 users.
```

Using the UBCF recommendations

List of recommendation courses for test set users 7 thru 10:

```
#Show recommender couseID
ubcf_list[1:5]

## [1] "Course_165" "Course_68"  "Course_69"  "Course_71"  "Course_72"

#get integer to match with course title to display
ubcf_list_Courses <- as.integer(sub(".*_", "", ubcf_list[1:5]))
#Get list of Courses by ID and name
courses <- Student_data %>% select("CourseID","CourseName")
courses_recommended <- courses %>% group_by(CourseName) %>%filter(CourseID
%in%ubcf_list_Courses)
unique(courses_recommended)

## # A tibble: 5 x 2
## # Groups:   CourseName [5]
##   CourseID CourseName
##      <int> <fct>
## 1      165 10 Hour Construction Industry
## 2       68 Introduction to OSHA
## 3       71 OSHA Standards and Inspection Procedures
## 4       72 OSHA Recordkeeping
## 5       69 HAZWOPER 8-Hour Refresher

cat("Recommended Courses Names for test user:\n\n")

## Recommended Courses Names for test user:

#corner(unique(courses_recommended$CourseName))
```

Total number of recommendations by users in test set

```
## number_of_items
##     1
## 16700
```

Create an evaluators scheme:

```
## Evaluation scheme with 2 items given
## Method: 'cross-validation' with 5 run(s).
## Good ratings: >=50.000000
```

```
## Data set: 16332 x 38 rating matrix of class 'realRatingMatrix' with 38379
ratings.
```

Create evaluation datasets using cross-validation method, keeping **2** items and **5** folds with rating threshold of **50** using the recommenderLab evaluationScheme function.

```
size_sets <- sapply(eval_sets@runsTrain, length)
cat("Sizes of Evaluation Sets:\t", size_sets, "\n")

## Sizes of Evaluation Sets:     13064 13064 13064 13064 13064

getData(eval_sets, "train")

## 13064 x 38 rating matrix of class 'realRatingMatrix' with 30657 ratings.
```

3 sets will be used: train = training set
known = test set used to build recommendations
unknown = test set to test the recommendations

Create UBCF Recommender

```
model_to_evaluate <-"UBCF"
model_paramenter <- NULL
eval_recommender <- Recommender(data = getData(eval_sets, "train"), method =
model_to_evaluate, parameter = model_paramenter)
eval_recommender

## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 13064 users.
```

Calculate the UBCF predictions for known test set:

```
## 3268 x 38 rating matrix of class 'realRatingMatrix' with 117648 ratings.
```

Calculate the prediction accuracy for each user in unknown test set:

```
eval_accuracy <- calcPredictionAccuracy(x = eval_prediction, data =
getData(eval_sets, "unknown"), byUser = TRUE)
head(eval_accuracy)

##                    RMSE      MSE      MAE
## Student_50007       NaN      NaN      NaN
## Student_50012  9.377302  87.9338  9.37000
## Student_50024       NaN      NaN      NaN
## Student_50059       NaN      NaN      NaN
## Student_50070       NaN      NaN      NaN
## Student_50077 16.606667 275.7814 16.60667
```

note: being of a small dataset many student did not get any recommendation due to unmatch similiar students courses and ratings

Calculate the overall avgerages in unknown test set:

```
## RMSE  MSE  MAE
##  NaN  NaN  NaN
```

Calculate the overall accuracy given in unknown test set:

```
##       RMSE        MSE        MAE
##  10.430619 108.797811   8.616915
```

Note the overall RMSE and the accuracy are good.

Using a precicion recall plot to predict accuracy with confusion matrix for known test set:

Evaluate the result with confusion matrix

```
head(getConfusionMatrix(results)[[1]])
```

```
##           TP        FP        FN       TN  precision    recall       TPR
## 1 0.04865361 0.9513464 0.3142595 34.68574 0.04865361 0.1366667 0.1366667
## 2 0.07037944 1.9296206 0.2925337 33.70747 0.03518972 0.1976068 0.1976068
## 3 0.09210526 2.9078947 0.2708078 32.72919 0.03070175 0.2565812 0.2565812
## 4 0.11260710 3.8873929 0.2503060 31.74969 0.02815177 0.3122222 0.3122222
## 5 0.12209302 4.8779070 0.2408201 30.75918 0.02441860 0.3393162 0.3393162
## 6 0.12882497 5.8711750 0.2340881 29.76591 0.02147083 0.3569231 0.3569231
##          FPR
## 1 0.02666320
## 2 0.05410522
## 3 0.08154508
## 4 0.10902073
## 5 0.13681607
## 6 0.16469039
```

Sum up the UBCF TP, FP, FN, TN indexes and plot:

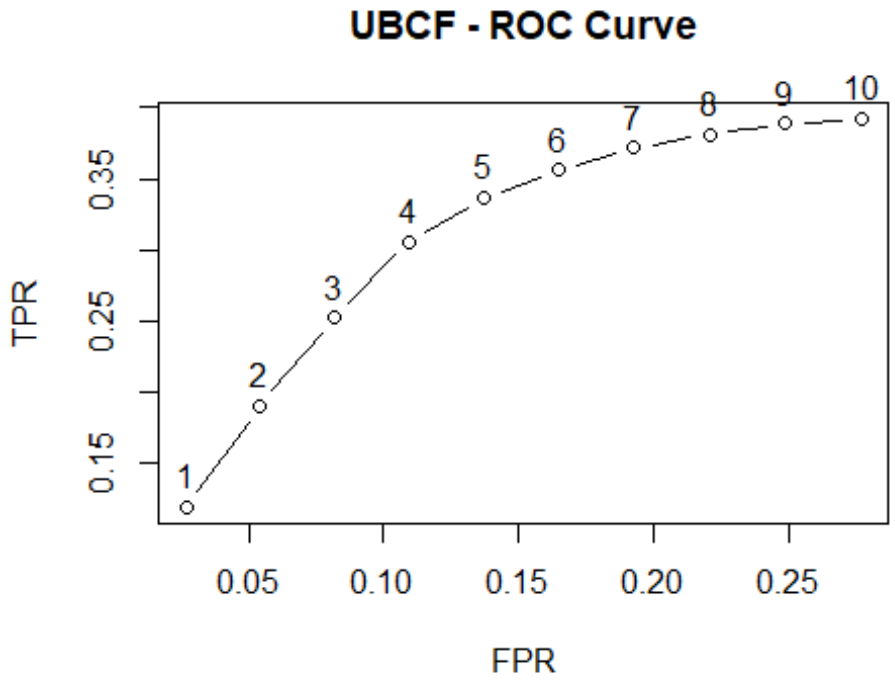
```
##           TP        FP       FN       TN  precision    recall       TPR
## 1  0.2093023  4.790698 1.541922 173.4581 0.20930233 0.5964639 0.5964639
## 2  0.3356793  9.664321 1.415545 168.5845 0.16783966 0.9527252 0.9527252
## 3  0.4452264 14.554774 1.305998 163.6940 0.14840881 1.2615592 1.2615592
## 4  0.5382497 19.461750 1.212974 158.7870 0.13456242 1.5296476 1.5296476
## 5  0.5917993 24.408201 1.159425 153.8406 0.11835985 1.6833203 1.6833203
## 6  0.6257650 29.374235 1.125459 148.8745 0.10429417 1.7808254 1.7808254
## 7  0.6542228 34.345777 1.097001 143.9030 0.09346040 1.8574575 1.8574575
## 8  0.6722766 39.327723 1.078947 138.9211 0.08403458 1.9049858 1.9049858
## 9  0.6878825 44.312118 1.063341 133.9367 0.07643139 1.9430047 1.9430047
## 10 0.6940024 49.305998 1.057222 128.9428 0.06940024 1.9609156 1.9609156
##          FPR
## 1  0.1342431
## 2  0.2708826
## 3  0.4080079
## 4  0.5456148
## 5  0.6843620
## 6  0.8236755
## 7  0.9631451
```

```
## 8  1.1029161
## 9  1.2427566
## 10 1.3828754
```

Note: it is difficult to visulize the data provided unless the results are plotted.

Create UBCF Receiver operating characteristic (ROC) plot

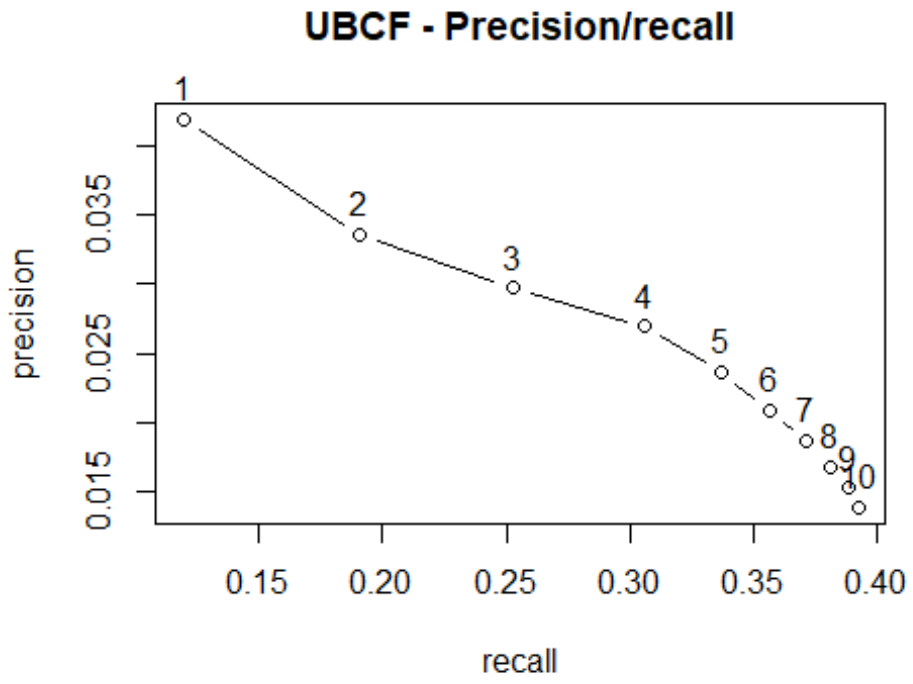
```
plot(results, annotate = TRUE, main = "UBCF - ROC Curve")
```



ROC curves are used in clinical biochemistry to choose the most appropriate cut-off for a test. The best cut-off has the highest true positive rate together with the lowest false positive rate. (Ekelund, 2011). Looking at the ratio of area above the curve compared to the area below the curve it appears model has almost failed.

At approx #6 the TPR/FPR is at its best

Plot UBCF Precision/recall to verify accuracy



*Note the precision/recall also indicates being bad

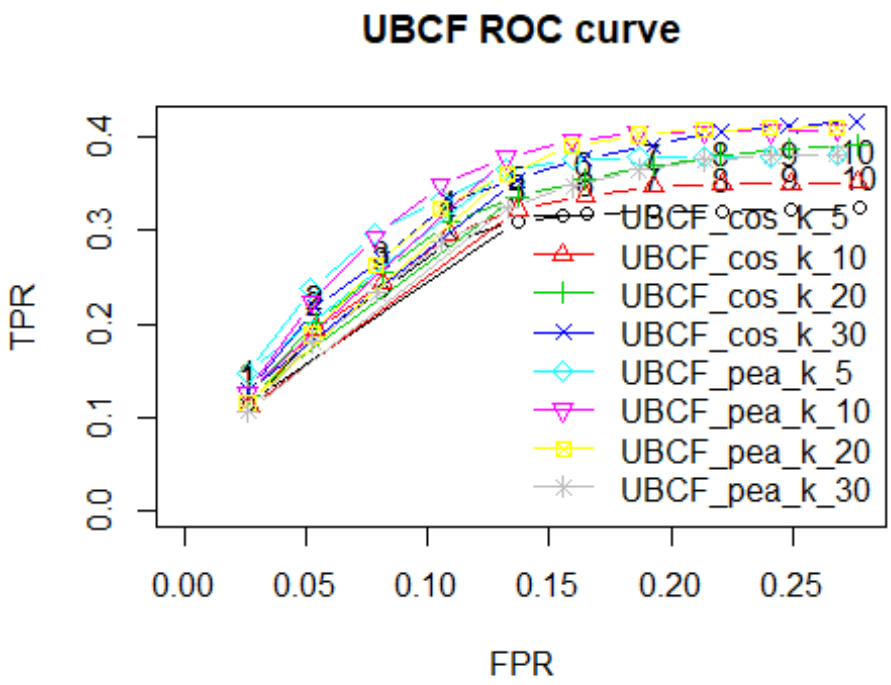
Fine Tuning of the UBCF models to get better results

Lets try different factors to see if we can get a better ROC and Precision Recall result. Create UBCF Models with varing vector_nn and different methods i.e.: cosine and pearson.

Determine the best UBCF results based on number of recommendations

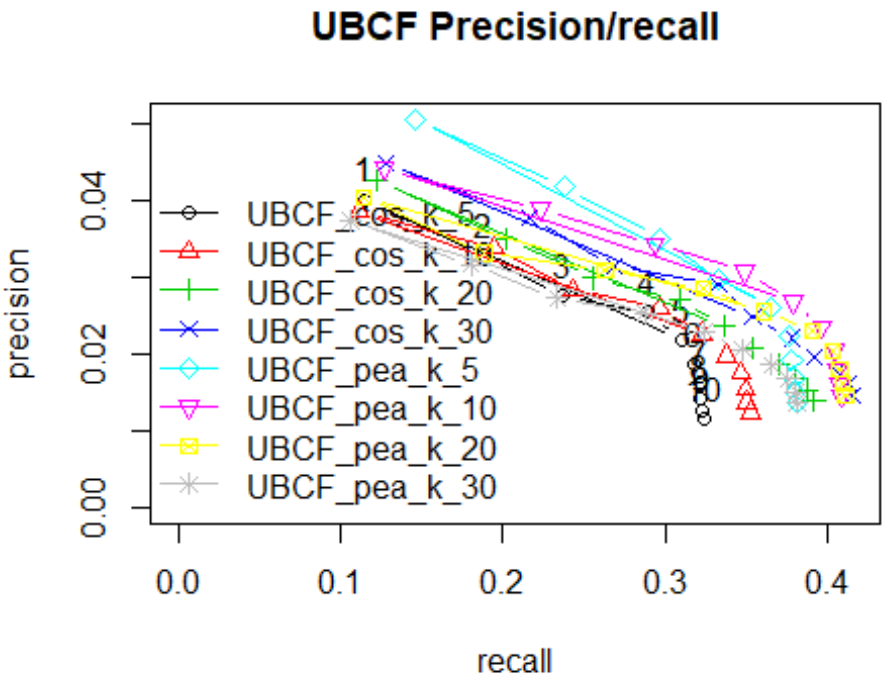
Plot UBCF Models with varing vector_nn and different methods results

```
plot(list_results, annotate = c (1,2), legend = "bottomright")
title("UBCF ROC curve")
```



Note: UBCF_pea_k_30 appears to be the best UBCF model with TPR closes to 0.7 and FPR less than 0.4

```
plot(list_results, "prec/rec",annotate = 1, legend= "bottomleft")
title("UBCF Precision/recall")
```



Note: The precision/recall support UBCF_pea_k_10 appears to be the best UBCF model

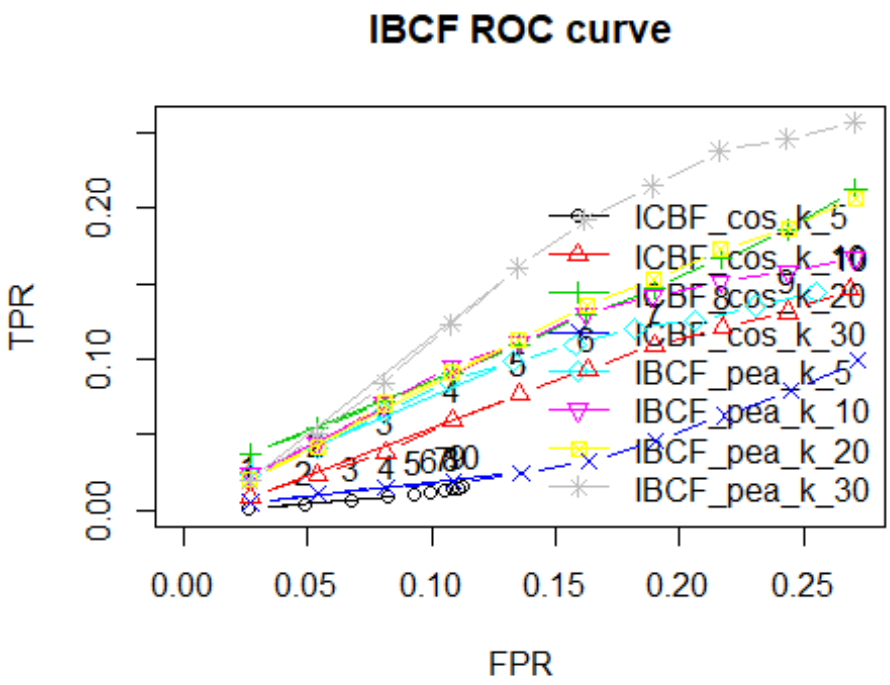
Create IBCF Model

Create IBCF Models with varing vector_kn and different methods i.e.: cosine and pearson

Get IBCF model results

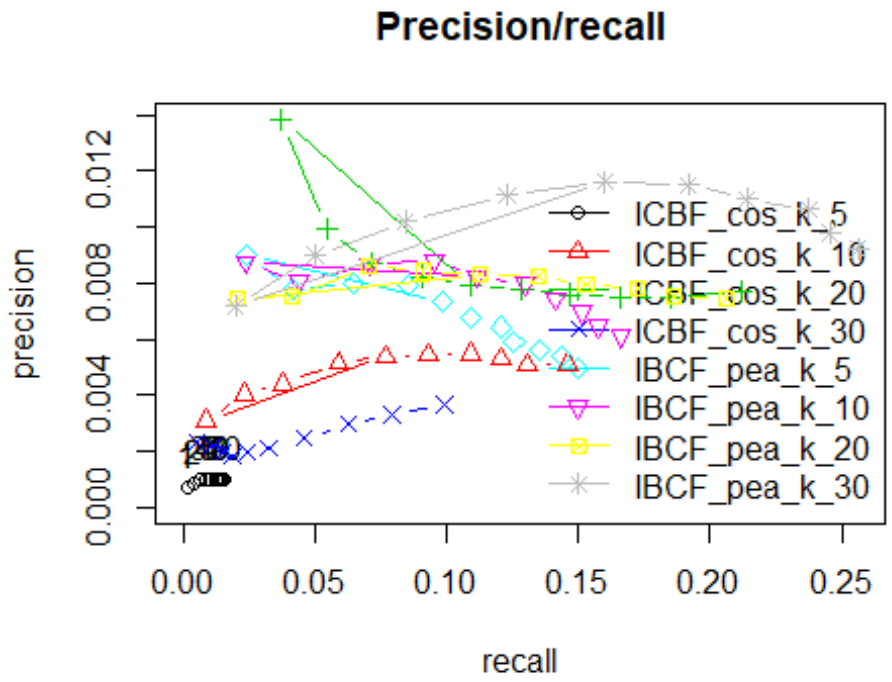
Plot IBCF with varing vector_kn and different methods results

```
plot(list_results, annotate = c (1,2), legend = "bottomright")
title("IBCF ROC curve")
```



Note ICBF_pea_k30 appears the best

```
plot(list_results, "prec/rec",annotate = 1, legend= "bottomright")
title("Precision/recall")
```

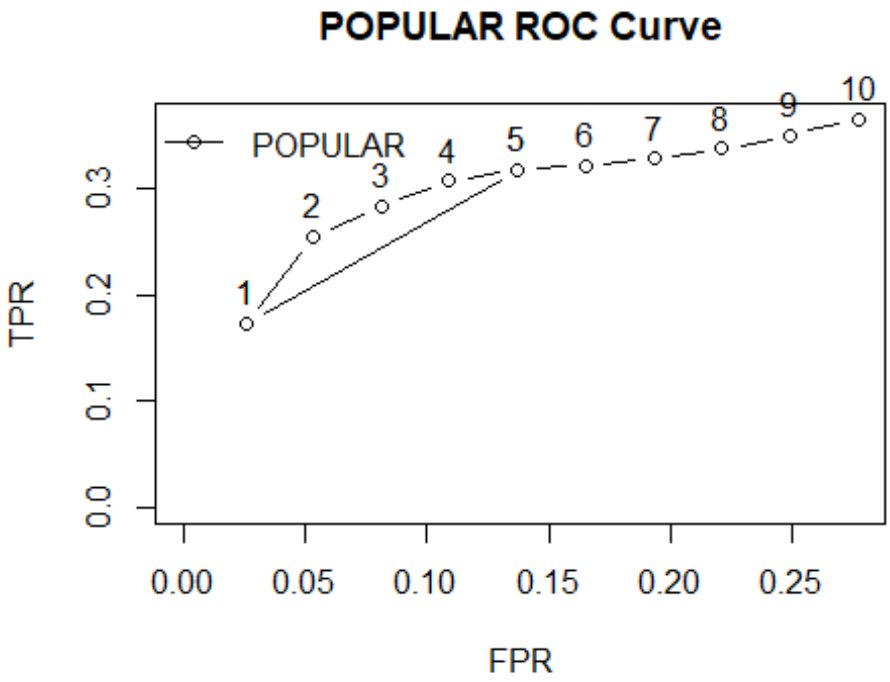


Note: The precision/recall support IBCF_pea_k_30 appears to be the best IBCF model

Create a POPULAR model

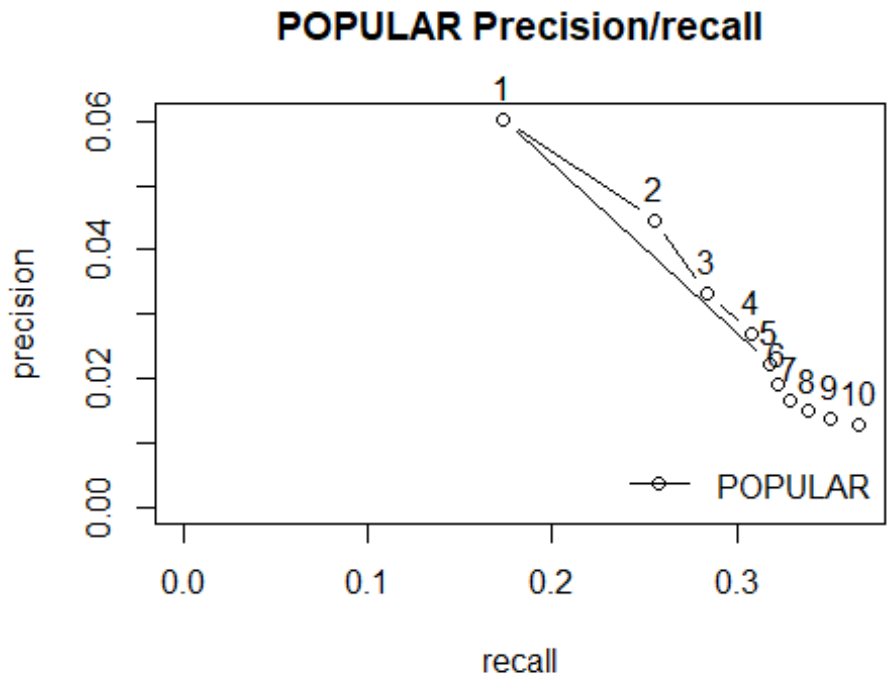
The POPULAR model is simple based on items popularity.

```
#plot and choose the optimal parameters
plot(list_results, annotate = c (1,2), legend = "topleft")
title("POPULAR ROC Curve")
```



Note The POPULAR Appears to be a good model.

```
plot(list_results, "prec/rec",annotate = 1, legend= "bottomright")
title("POPULAR Precision/recall")
```

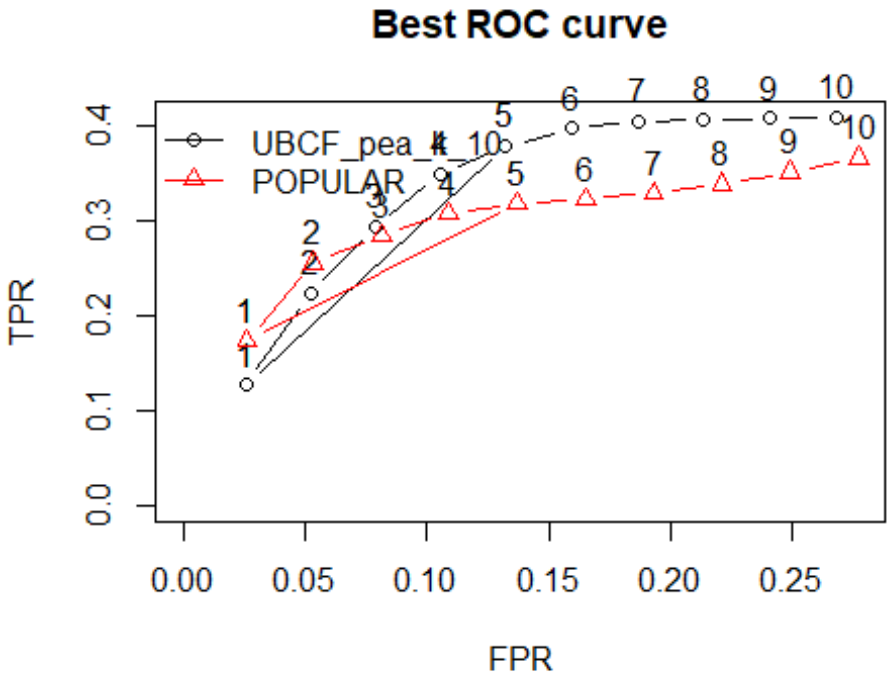


Note The POPOULAR Precision and recall precion do not total support a good model

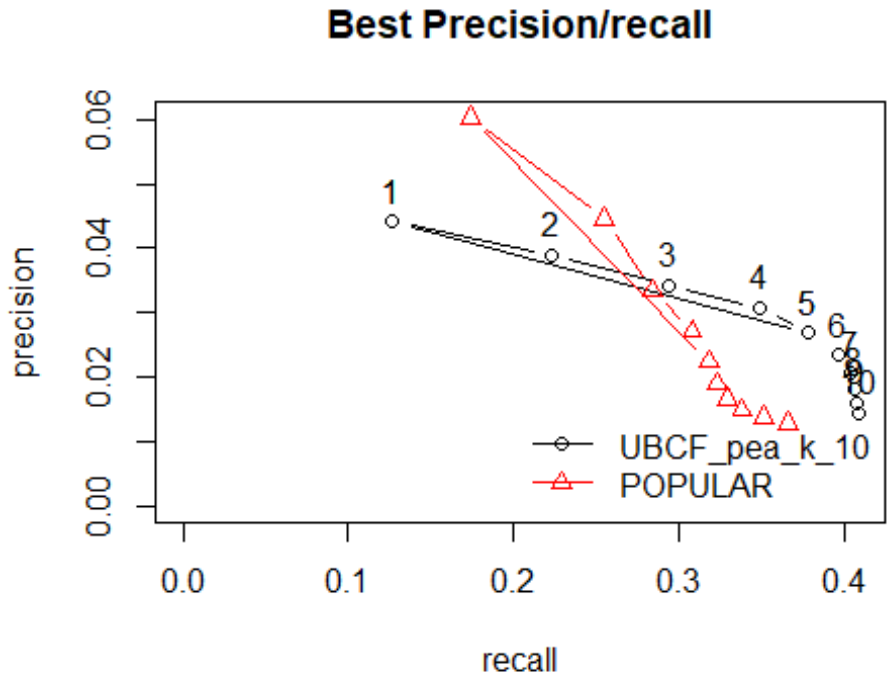
Plot Best Results

Now I going to compare the best result of UBCF and POPULAR models to determine my Final Model

```
#plot and choose the optimal parameters
plot(list_results, annotate = c (1,2), legend = "topleft")
title("Best ROC curve")
```




```
plot(list_results, "prec/rec",annotate = 1, legend= "bottomright")
title("Best Precision/recall")
```



```
UBCF_eval <- evaluate(x = eval_sets, method = "UBCF", k = best_vector_k, type
= "ratings")

## UBCF run fold/sample [model time/prediction time]
##   1  [0.02sec/10.49sec]
##   2  [0sec/11.06sec]
##   3  [0sec/11.11sec]
##   4  [0sec/10.51sec]
##   5  [0.02sec/10.98sec]

head(getConfusionMatrix(UBCF_eval)[[1]])

##         RMSE      MSE      MAE
## res 10.43062 108.7978 8.616915
```

Final IBCF Evaluation results:

```
IBCF_eval <- evaluate(x = eval_sets, method = "IBCF", k = best_vector_nn,
type = "ratings")

## IBCF run fold/sample [model time/prediction time]
##   1  [0.06sec/0.09sec]
##   2  [0.06sec/0.09sec]
##   3  [0.05sec/0.09sec]
##   4  [0.06sec/0.09sec]
##   5  [0.04sec/0.11sec]

head(getConfusionMatrix(IBCF_eval)[[1]])

##         RMSE      MSE      MAE
## res 10.44891 109.1796 8.611864
```

Final POPULAR Evaluation results:

```
POP_eval <- evaluate(x = eval_sets, method = "POPULAR", n =
n_recommendations, type = "ratings")

## POPULAR run fold/sample [model time/prediction time]
##   1  [0.01sec/0.02sec]
##   2  [0sec/0.03sec]
##   3  [0sec/0.03sec]
##   4  [0sec/0.04sec]
##   5  [0sec/0.02sec]

head(getConfusionMatrix(POP_eval)[[1]])

##         RMSE      MSE      MAE
## res 10.42442 108.6686 8.609829
```

Conclusion

The finding do show using the POPULAR model returns a lower RMSE of 10.740 than UBCF 10.4894 and IBCF 11.59713 with the dataset I used. The dataset that I used for this project

was 10th the size of the actual data I have access to but was not authoried to use for this project as originally planned. Therefore the lack of records and knowing that RecommendLab works best with large sets of data my result were also lacking in apeal.

References

Goldberg D, Nichols D, Oki BM, Terry D (1992). "Using collaborative filtering to weave an information tapestry." Communications of the ACM, 35(12), 61–70. ISSN 0001-0782. doi:http://doi.acm.org/10.1145/138859.138867.

Michael Hahsler (2019). recommenderlab: Lab for Developing and Testing Recommender Algorithms. R package version 0.2-4. https://github.com/mhahsler/recommenderlab

Suzanne Ekelund (2011), "ROC curves - what are they and how are they used?",https://acutecaretesting.org/en/articles/roc-curves-what-are-they-and-how-are-they-used