Project2

Cheryl Bowersox

Sunday, February 26, 2017

In this project I will develop two models to provide tasty beer recommendations for users. I am continuing to use the data from beer advocate that I began exploring in Project 1, but will build out more complex models.

Data Source: https://data.world/socialmediadata/beer (https://data.world/socialmediadata/beer) advocate

Data Exploration

```
# import libraries
library(recommenderlab)
```

```
## Warning: package 'recommenderlab' was built under R version 3.1.3
```

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

```
## Warning: package 'Matrix' was built under R version 3.1.3
```

```
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:base':
##
##     crossprod, tcrossprod
##
## Loading required package: registry
```

```
## Warning: package 'registry' was built under R version 3.1.3
```

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

```
## Warning: package 'arules' was built under R version 3.1.3
```

```
##
## Attaching package: 'arules'
##
## The following objects are masked from 'package:base':
##
##     %in%, write
##
## Loading required package: proxy
```

```
## Warning: package 'proxy' was built under R version 3.1.3
```

```
##
## 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
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.1.3
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:arules':
##
##     intersect, 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
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
##
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
library(reshape2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
library(pROC)
```

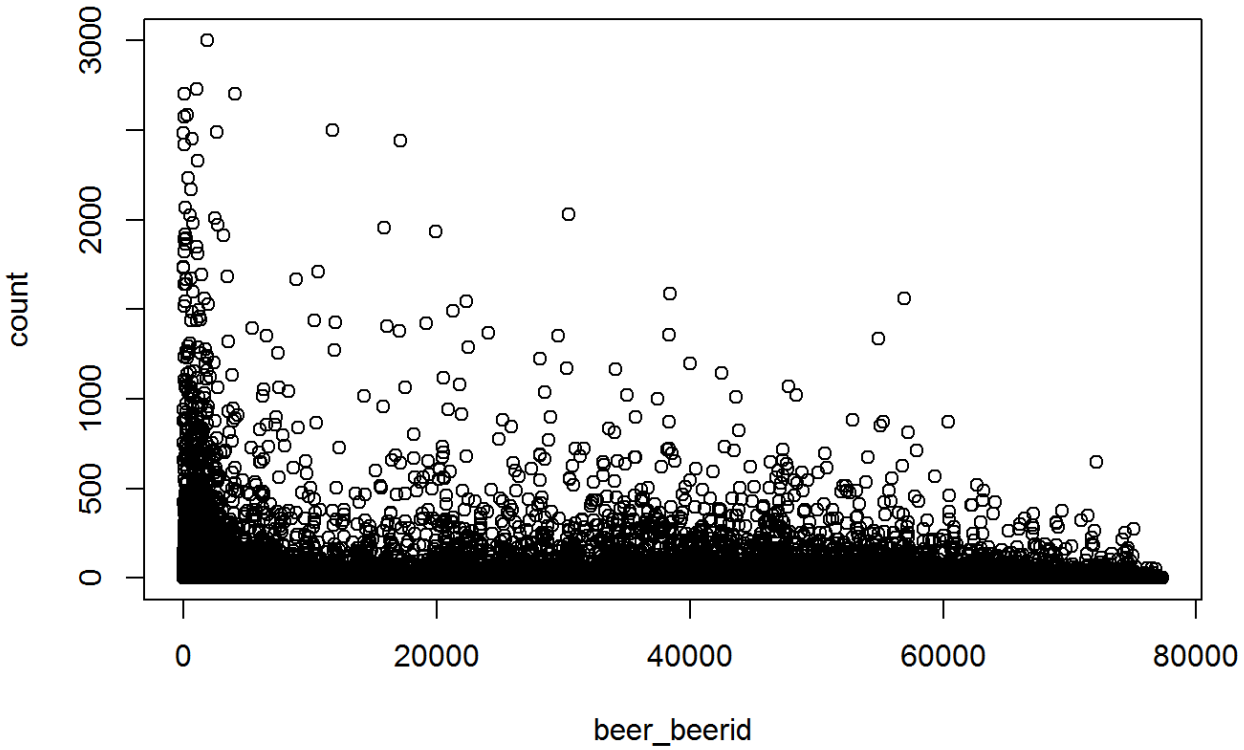
```
## Warning: package 'pROC' was built under R version 3.1.3
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)

#read data
beerdata <- read.csv("~/GitHub/IS643/beerdata.csv")

# examine some properties of the data

# group overall ratings by by beer styles
beertypes <- beerdata%>%select(beer_style, review_overall)%>%
  group_by(beer_style)%>%
  summarise(avgrev= mean(review_overall), count=n())

# remove ratings less than 1% of data points
sparseratings <- beerdata%>%select(beer_beerid, review_overall)%>%
  group_by(beer_beerid)%>%
  summarise(count=n())

# remove ratings less than 1% of data points
sparseusers <- beerdata%>%select(beer_beerid, review_profilename)%>%
  group_by(review_profilename)%>%
  summarise(count=n())
```

Plotting the number of ratings vs. the beer ID shows more ratings for smaller ID numbers. This looks like just an artifact of the numbering system as it makes sense the early (lower) beer ID's have been in the system longest, and have received more ratings.

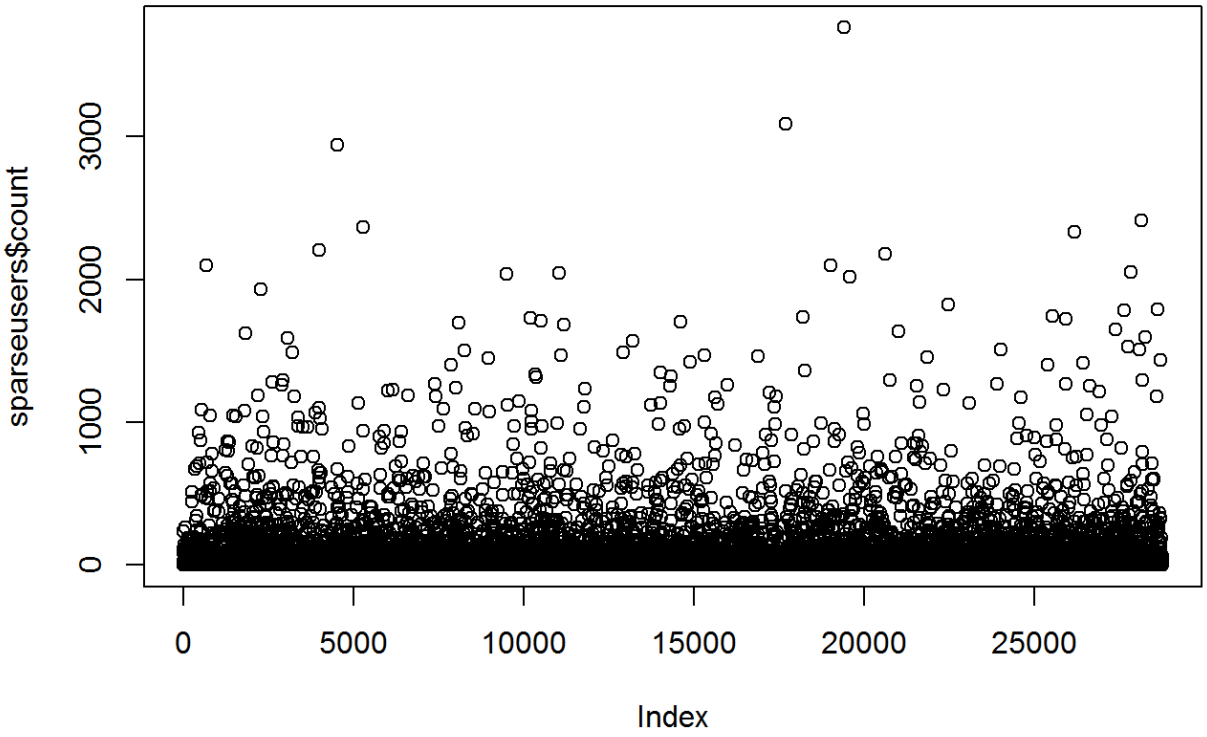
```
plot(sparseratings)
```



The total number of distinct beers is 42719 and the beers cover a wide range of number of ratings, from as few as 1 to 3000. The beer data is sparse, as there are 15240 beers that have received only 1 rating. This accounts for around 35% of the total number of items.

Plotting the number of ratings per user shows a fairly uniform distribution across the 28762 distinct users, with the number of ratings ranging from 1 to 3763. There are 12931 users that have rated less than 3 items. This accounts for around 45% of the total number of user

```
plot(sparseusers$count)
```



Our raw data contains 1048575 data points, and a ratings matrix for this data would be 28762 X 42719. By removing beers that have received only one rating, and users that have rated less than 3 beers we can reduce the size significantly.

The new data set containing only users that have rated 3 or more beers, and only beers with more than 1 rating has been reduced to 1016827 rows, with 25003 items and 15831 users. By removing these items we have reduced the ratings matrix to a more manageable size of 15831 X 27479.

Now that we have a reduced data set, we can split it into training and test data, with 80% of the data used as training, and 20% for testing the models.

```
train_amt <- floor(0.8 * nrow(dfbeerrm))
set.seed(408)
ind <- sample(seq_len(nrow(dfbeerrm)), size = train_amt)

train_beer <- dfbeerrm[ind, ]
test_beer <- dfbeerrm[-ind, ]
```

Model1: User/Item profile model

This model will compare users attributes with item attributes to find good matches for recommendations. The goal will be to predict if a beer will receive a positive rating from a given user.

First we define a statistical model for each user based on their rating history. For each beer we will develop a 'beer profile' based on mean ratings for attributes of that beer, and use these as inputs to the user profile model to estimate the probable rating for that user for that beer. The highest rated outputs will be used as the recommended beers.

```
# for user i which review metric is most important. That is, which score for ar
oma, appearance, palate or taste best predicts a high overall rating?
# model, select top 2 predictors

#model beers in all data (train + test) so this will not need to be repeated
dfbeerprof <- dfbeerrm%>%select(beer_beerid, beer_name, beer_abv, review_overal
l,review_aroma,review_appearance, review_palate, review_taste)%>%group_by(beer_
beerid,beer_name,beer_abv)%>%summarise(review_aroma=mean(review_aroma), review_
appearance = mean(review_appearance), review_palate = mean(review_palate), revi
ew_taste = mean(review_taste), count=n())

#replace any NA with 0, so these variables will not impact the model
dfbeerprof[is.na(dfbeerprof)] <- 0
```

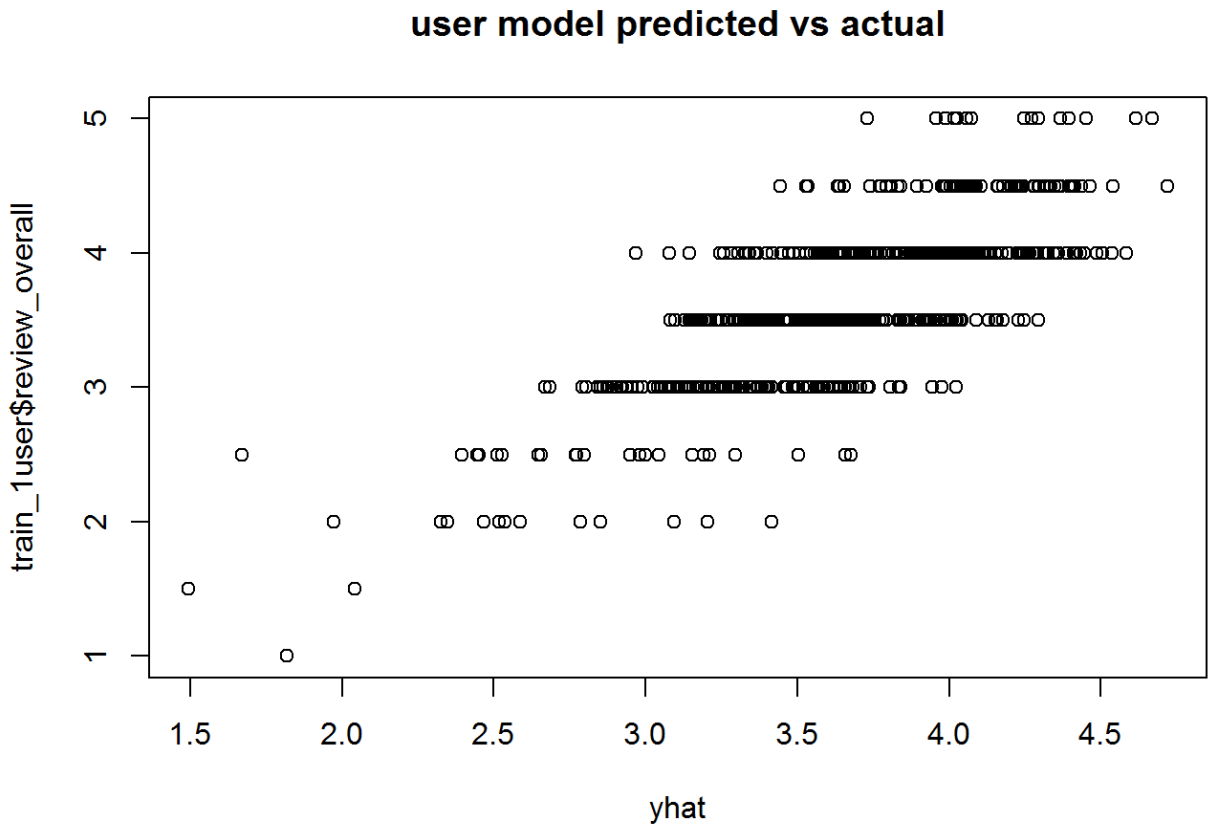
Create recommendation for a user

```
## Start:  AIC=-1539.67
## review_overall ~ review_aroma + review_appearance + review_palate +
##     review_taste + beer_abv
##
##                      Df Sum of Sq    RSS     AIC
## <none>                             106.70 -1539.7
## - review_appearance  1     0.390 107.08 -1538.8
## - review_aroma       1     1.010 107.70 -1534.3
## - beer_abv           1     5.564 112.26 -1502.0
## - review_palate      1     6.497 113.19 -1495.6
## - review_taste       1    36.537 143.23 -1312.0
```

```
## Warning: 'MASS' namespace cannot be unloaded:
##   namespace 'MASS' is imported by 'pbkrtest', 'lme4', 'car' so cannot be unl
oaded
```

```
## Start:  AIC=-1539.67
## review_overall ~ review_aroma + review_appearance + review_palate +
##     review_taste + beer_abv
##
##                      Df Sum of Sq    RSS     AIC
## <none>                            106.70 -1539.7
## - review_appearance  1     0.390 107.08 -1538.8
## - review_aroma       1     1.010 107.70 -1534.3
## - beer_abv           1     5.564 112.26 -1502.0
## - review_palate      1     6.497 113.19 -1495.6
## - review_taste       1    36.537 143.23 -1312.0
```

```
## Warning: 'MASS' namespace cannot be unloaded:
##   namespace 'MASS' is imported by 'pbkrtest', 'lme4', 'car' so cannot be unl
oaded
```



Running model1 for the user emmasdad and SwiftyMcBeer results in the following recommendations:

```
model1_user1[[3]]$beer_name
```

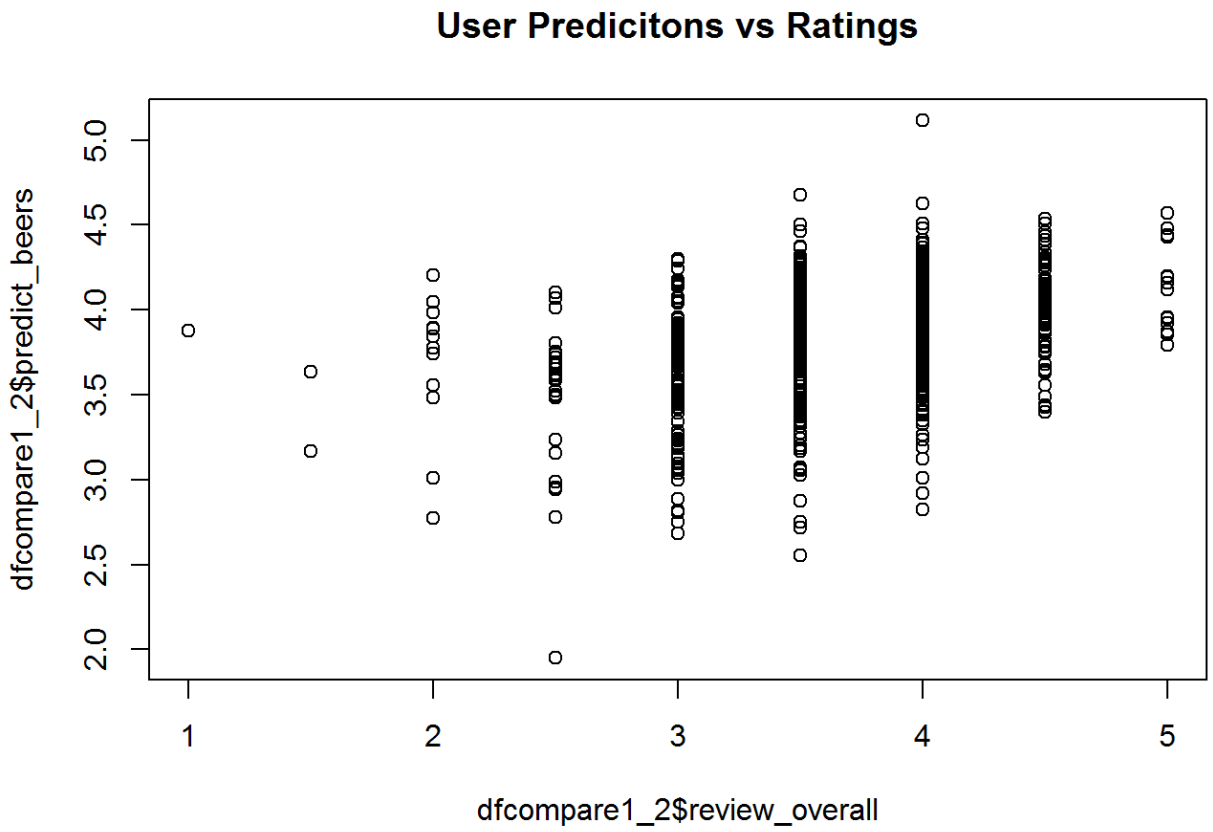
```
##  [1] Date Night With Jumbo Love
##  [2] Bourbon Barrel Vermont Smoked Porter
##  [3] Highland Oak Aged Cold Mountain Winter
##  [4] Diamond Knot IPA (Dry Hopped With Simcoe)
##  [5] Star Of India IPA
##  [6] Funky Barrel
##  [7] '99 Wee Heavy Scotch Ale
##  [8] Old Boilermaker
##  [9] Old Birdbrain
## [10] Rare D.O.S.
## 37770 Levels: '71 Pale Ale ... ZZ Lager
```

```
model1_user2[[3]]$beer_name
```

```
##  [1] Date Night With Jumbo Love
##  [2] Bourbon Barrel Vermont Smoked Porter
##  [3] Highland Oak Aged Cold Mountain Winter
##  [4] Diamond Knot IPA (Dry Hopped With Simcoe)
##  [5] Star Of India IPA
##  [6] Funky Barrel
##  [7] '99 Wee Heavy Scotch Ale
##  [8] Old Boilermaker
##  [9] Old Birdbrain
## [10] Rare D.O.S.
## 37770 Levels: '71 Pale Ale ... ZZ Lager
```

Looking at the model's predictions compared to the user's actual ratings gives the following comparison, showing the positive actual ratings are generally associated with more positive predictions, but with more variability in the lower ratings, and tends to predict a rating a little lower than the user actually predicted.

```
dfcompare1_2 <- rbind(model1_user1[[2]],model1_user2[[2]])

plot(dfcompare1_2$review_overall,dfcompare1_2$predict_beers, main = "User Predi
citons vs Ratings")
```



We can also use a confusion matrix to evaluating this prediction model for this user In this case, we will assume any rating greater than 3 is a positive rating, indicated as '1' and any rating lower as negative, indicated as '0'.

```
  #evaluate the prediction for this user using confusion matrix using beers alr
eady rated
  #where rating above 3 is considered positive, and 3 or below is negative

  ### evalute output, model


  target <- ifelse(dfcompare1_2$review_overall > 2.9, 1, 0) #reprsents positiv
e rating
  predbeer <- ifelse(dfcompare1_2$predict_beers >2.9, 1,0)

  (cm1<-confusionMatrix(data=predbeer,
  reference=target))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0    6   20
##          1   78 1554
##
##                Accuracy : 0.9409
##                  95% CI : (0.9284, 0.9518)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 0.9451
##
##                   Kappa : 0.0872
##  Mcnemar's Test P-Value : 8.518e-09
##
##             Sensitivity : 0.071429
##             Specificity : 0.987294
##          Pos Pred Value : 0.230769
##          Neg Pred Value : 0.952206
##              Prevalence : 0.050663
##          Detection Rate : 0.003619
##    Detection Prevalence : 0.015682
##       Balanced Accuracy : 0.529361
##
##        'Positive' Class : 0
##
```

```
accuracy1<-cm1$overall["Accuracy"]
recall1 <- cm1$byClass['Sensitivity']
specificity1 <- cm1$byClass['Specificity']
precision1 <- cm1$byClass['Pos Pred Value']
f_measure1 <- 2 * ((precision1 * recall1) / (precision1 + recall1))
rocModel1 <- roc(target,predbeer)
```

From the confusion matrix we can see that it correctly classified 0.9408926 of the ratings as either of positive or negative. More interestingly, the positive prediction value, or precision, is 0.2307692, meaning this model tends to err on the side of ‘false positives’. This could be possibly be corrected by adjusting the threshold for what is considered a ‘positive’ rating.

To fully test the accuracy it will be necessary to run this model against the training data for all users, but this seems a promising, if calculation -intensive approach.

model 2: item based collaborative filtering method

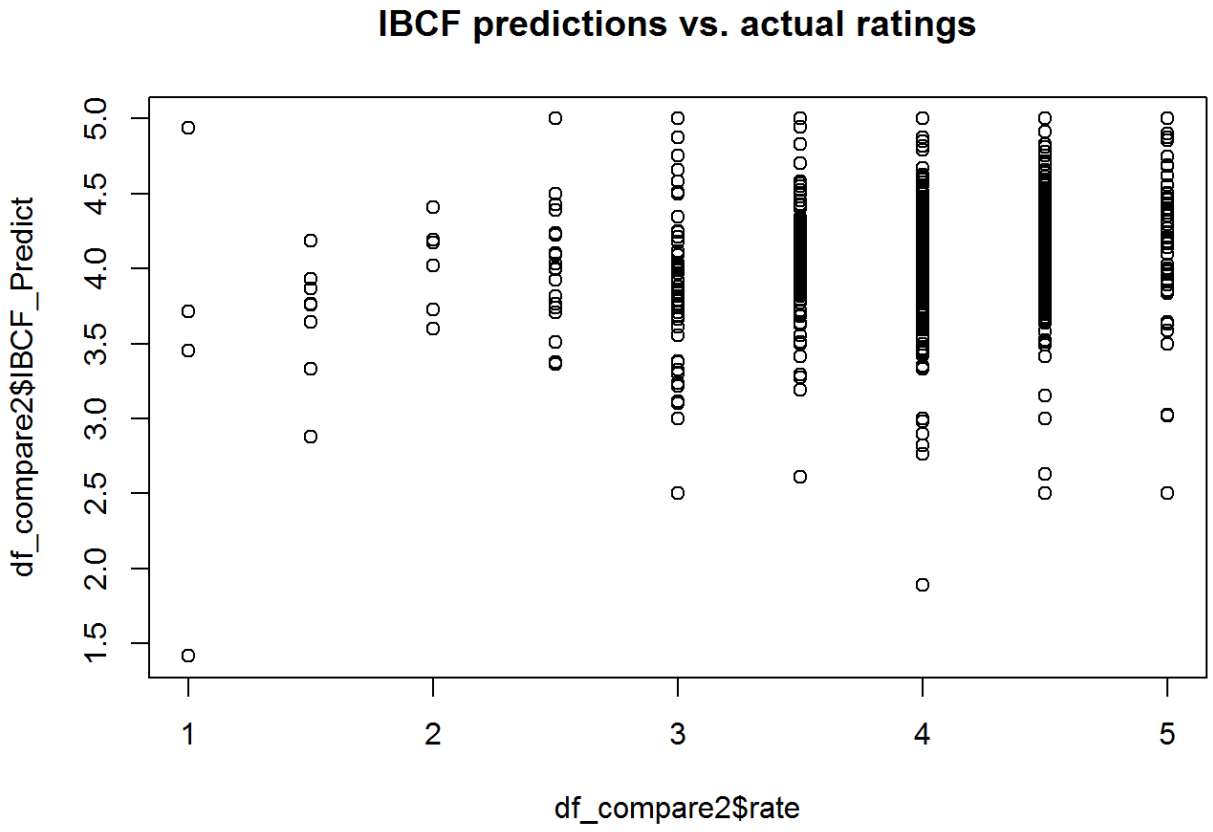
For the item-based collaborative filtering method I am using the recoomenderlab package, and using the ICBF algorithm, normalizing the data to remove user-rating bias, and using the cosine distance method. I am selecting a subset of the training data (80% of total) to run these models. I am choosing to evaluate it for a sample of 502 users, and the 100 items with the most ratings for that sample of users.

```
#get data in correct format
usersample <- c(as.character(sample(users$review_profilename,500)),user1,user2)
itemsample <- head(train_beer%>%filter(review_profilename %in% usersample)%>%
  group_by(beer_name)%>%summarise(count =n())%>%arrange(-count),100)$beer_name


df <- train_beer %>%
  select(user = review_profilename, beer = beer_name, rate = review_overall)%>%
  filter(user %in% usersample, beer %in% itemsample)

user_n<- df%>%group_by(user)%>%summarise(count=n())

#user hold in reserve to evaluate on test data:
df_test <- test_beer %>%
  select(user = review_profilename, beer = beer_name, rate = review_overall)%>%
  filter(user %in% usersample, beer %in% itemsample)


ratemat <- as(df,"realRatingMatrix")
#ratemat_test <- as(df_test, "realRatingMatrix")

#train models
recIBCF <- Recommender(ratemat,method="IBCF",
      param=list(normalize = "Z-score",method="Cosine", minRating=1))

#recSVD <- Recommender(ratemat,method="SVD",
#    param=list(maxiter=50, normalize = "center"))



# compare predictions
pred_IBCF <- predict(recIBCF, ratemat, type ="ratingMatrix")
M_pred_IBCF <- as(pred_IBCF, "matrix")
dfpred_IBCF<- melt(M_pred_IBCF)

#merge test data for IBCF with predictions
nms <- c("user","beer","IBCF_Predict")
colnames(dfpred_IBCF) = nms
dfpred_IBCF<- na.omit(dfpred_IBCF)

#predict top 10 for our two example users

model2user1 <- head(dfpred_IBCF%>%filter(user == user1)%>%arrange(-IBCF_Predic
t),10)
model2user2 <- head(dfpred_IBCF%>%filter(user == user1)%>%arrange(-IBCF_Predic
t),10)

df_compare2 <- merge(x = df_test, y = dfpred_IBCF, by = c("user","beer"))
```

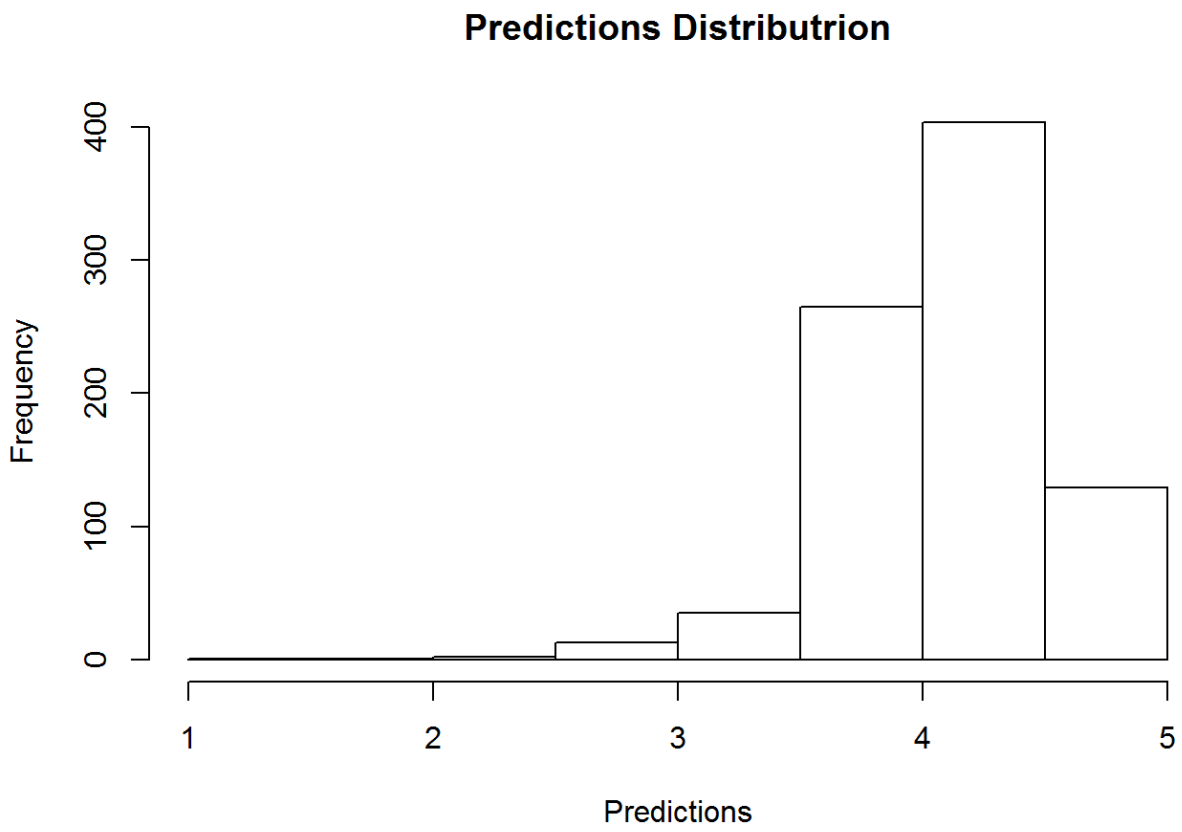
```
#clean up data
df_compare2$IBCF_Predict <- ifelse(df_compare2$IBCF_Predict < 1, 1, df_compare2
$IBCF_Predict)
df_compare2$IBCF_Predict <- ifelse(df_compare2$IBCF_Predict > 5, 5, df_compare2
$IBCF_Predict)
```

A subset of the test data (20% of total data) was created containing just users that are included in our training subset. When this is merged with the predictions created by the IBCF model we can compare the two. The predictions seem skewed when compared to the actual ratings, which will lead in errors on the side of too many positive ratings.

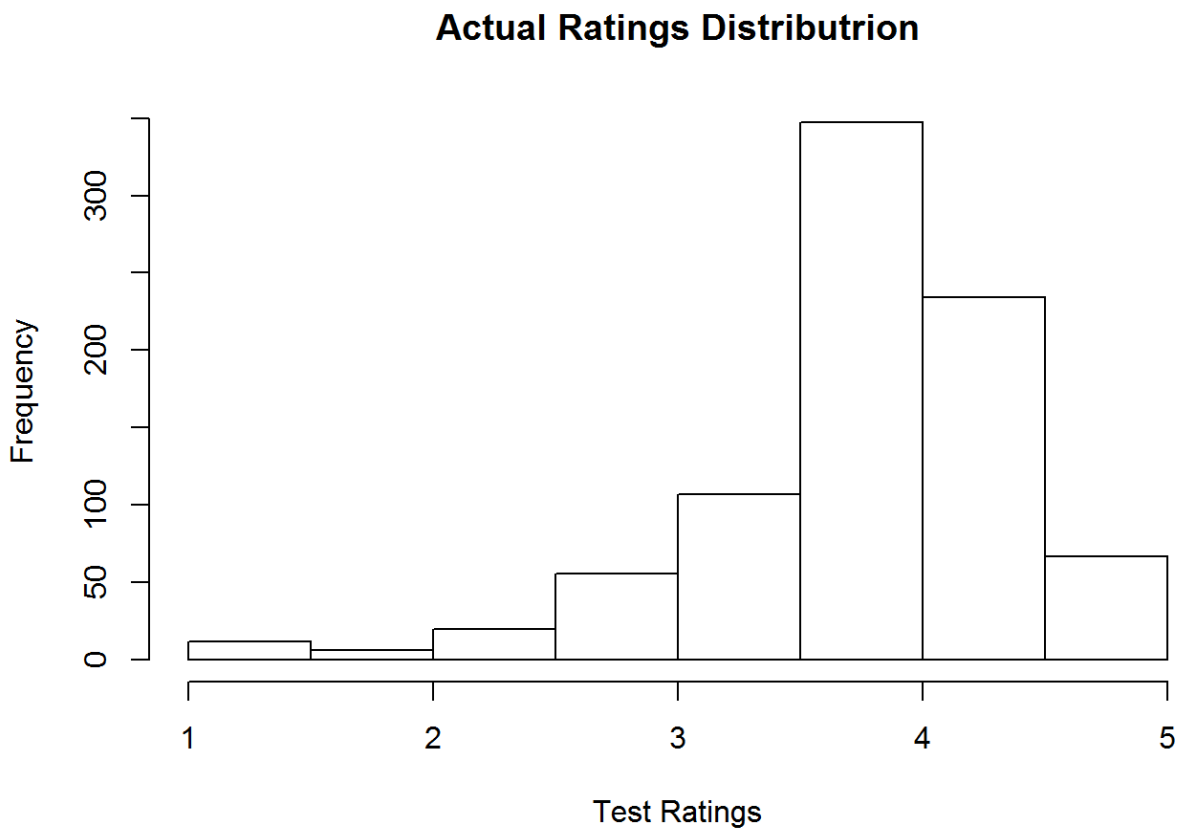
```
plot(x = df_compare2$rate, y=df_compare2$IBCF_Predict, main = "IBCF predictions
vs. actual ratings")
```




```
hist(df_compare2$IBCF_Predict,main="Predictions Distributrion",xlab="Prediction
s")
```




```
hist(df_compare2$rate,main="Actual Ratings Distributrion",xlab="Test Ratings")
```



confusion matrix for IBCF model

A confusion matrix for the IBCF model results in a high accuracy, but this may be misleading when the majority of ratings in the test data are above 3. There is little distinction between a prediction of 4.9 and an actual rating of 3.1 in this case.

```
target <- ifelse(df_compare2$rate > 2.9, 1, 0) #reprsents positive rating
predbeer <- ifelse(df_compare2$IBCF_Predict > 2.9, 1,0)

(cm2<-confusionMatrix(data=predbeer,
reference=target))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   2   9
##          1  36 802
##
##                Accuracy : 0.947
##                  95% CI : (0.9297, 0.9611)
##     No Information Rate : 0.9552
##     P-Value [Acc > NIR] : 0.8912401
##
##                   Kappa : 0.0628
##  Mcnemar's Test P-Value : 0.0001063
##
##             Sensitivity : 0.052632
##             Specificity : 0.988903
##          Pos Pred Value : 0.181818
##          Neg Pred Value : 0.957041
##              Prevalence : 0.044759
##          Detection Rate : 0.002356
##    Detection Prevalence : 0.012956
##       Balanced Accuracy : 0.520767
##
##        'Positive' Class : 0
##
```

```
accuracy2<-cm2$overall["Accuracy"]
recall2 <- cm2$byClass['Sensitivity']
specificity2 <- cm2$byClass['Specificity']
precision2 <- cm2$byClass['Pos Pred Value']
f_measure2 <- 2 * ((precision2 * recall2) / (precision2 + recall2))
rocModel2 <- roc(target,predbeer)
```

If we shift the threshold for what is considered 'positive, to 4 and above, the model is significantly less accurate

```
target <- ifelse(df_compare2$rate > 4, 1, 0) #reprsents positive rating
predbeer <- ifelse(df_compare2$IBCF_Predict >4, 1,0)

(cm2<-confusionMatrix(data=predbeer,
reference=target))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 232  85
##          1 316 216
##
##                Accuracy : 0.5277
##                  95% CI : (0.4935, 0.5617)
##     No Information Rate : 0.6455
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1202
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.4234
##             Specificity : 0.7176
##          Pos Pred Value : 0.7319
##          Neg Pred Value : 0.4060
##              Prevalence : 0.6455
##          Detection Rate : 0.2733
##    Detection Prevalence : 0.3734
##       Balanced Accuracy : 0.5705
##
##        'Positive' Class : 0
##
```

Model1 vs. Model 2

Comparing recommendations for model1 vs. model2 we can see very recommendations for the two users selected for model1 Model1 produces the following for emmasdad

```
#User1
model1_user1[[3]]
```

```
## Source: local data frame [10 x 3]
##
##    beer_beerid                                beer_name predict_beers
## 1        70781                Date Night With Jumbo Love      5.320555
## 2        14252      Bourbon Barrel Vermont Smoked Porter      5.229938
## 3        64484    Highland Oak Aged Cold Mountain Winter      5.229938
## 4        58131 Diamond Knot IPA (Dry Hopped With Simcoe)      5.145110
## 5         6046                          Star Of India IPA      5.124168
## 6        39560                               Funky Barrel      5.124168
## 7         3676                  '99 Wee Heavy Scotch Ale      5.114895
## 8        60293                            Old Boilermaker      5.113747
## 9        56622                              Old Birdbrain      5.105591
## 10       63649                                Rare D.O.S.      5.099654
```

```
model1_user2[[3]]
```

```
## Source: local data frame [10 x 3]
##
##    beer_beerid                                beer_name predict_beers
## 1        70781                Date Night With Jumbo Love      5.320555
## 2        14252      Bourbon Barrel Vermont Smoked Porter      5.229938
## 3        64484    Highland Oak Aged Cold Mountain Winter      5.229938
## 4        58131 Diamond Knot IPA (Dry Hopped With Simcoe)      5.145110
## 5         6046                          Star Of India IPA      5.124168
## 6        39560                               Funky Barrel      5.124168
## 7         3676                  '99 Wee Heavy Scotch Ale      5.114895
## 8        60293                            Old Boilermaker      5.113747
## 9        56622                              Old Birdbrain      5.105591
## 10       63649                                Rare D.O.S.      5.099654
```

Model1, however, predicts the following

```
model2user1
```

```
##        user                   beer IBCF_Predict
## 1  emmasdad          Imperial Stout     8.326043
## 2  emmasdad                Pale Ale     7.389353
## 3  emmasdad                     IPA     7.204364
## 4  emmasdad   India Pale Ale (IPA)     6.622041
## 5  emmasdad                  Porter     6.081376
## 6  emmasdad Racer 5 India Pale Ale     5.208426
## 7  emmasdad   Sierra Nevada Porter     5.203530
## 8  emmasdad          India Pale Ale     5.188668
## 9  emmasdad   IPA (India Pale Ale)     5.087321
## 10 emmasdad     Fat Tire Amber Ale     5.029184
```

```
model2user2
```

```
##        user                    beer IBCF_Predict
## 1  emmasdad         Imperial Stout     8.326043
## 2  emmasdad               Pale Ale     7.389353
## 3  emmasdad                    IPA     7.204364
## 4  emmasdad   India Pale Ale (IPA)     6.622041
## 5  emmasdad                 Porter     6.081376
## 6  emmasdad Racer 5 India Pale Ale     5.208426
## 7  emmasdad   Sierra Nevada Porter     5.203530
## 8  emmasdad         India Pale Ale     5.188668
## 9  emmasdad   IPA (India Pale Ale)     5.087321
## 10 emmasdad     Fat Tire Amber Ale     5.029184
```

When comparing the different model's power to predict a positive rating, we can look at the confusion matrix statistics for each. They seem very similar in accuracy although the model 1 has greater sensitivity and higher F1 Score.

```
##                Model 1    Model 2
## Accuracy    0.94089264 0.94699647
## Precision   0.23076923 0.18181818
## Sensitivity 0.07142857 0.05263158
## Specificity 0.98729352 0.98890259
## F1 Score    0.10909091 0.08163265
## AUC         0.52936105 0.52076708
```

While model 1 may seem to predict better when comparing per-user errors, it is not very scale-able, as it is creating a model for each user and then applying the all item's averaged profile. A new model must be generated for each user recommendation, wheres with the IBCF method you may have greater error in specific ratings, but the recommendations are computed all at once for all users, making it more efficient. If the goal of this recommendation is to provide a list of beers that are likely to get a positive rating, Model 2 would be a better choice. If the goal, however, is to provide a very customized list for each user of beers they are likely to rate very highly, Model 1 would be a better choice, provided we have enough data from the user in question.