CS 699 Assignment 5

Katherine Rein

```r
# Import libraries
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(recommenderlab)
```

```
## Loading required package: proxy

##
## 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

## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```r
library(dplyr)
```

```
##
## 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
```

```r
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack
```

```r
library(rsample)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.3.0 --

## v broom        1.0.8     v recipes      1.3.1
## v dials        1.4.0     v tibble       3.2.1
## v ggplot2      3.5.2     v tune         1.3.0
## v infer        1.0.8     v workflows    1.2.0
## v modeldata    1.4.0     v workflowsets 1.1.1
## v parsnip      1.3.2     v yardstick    1.3.2
## v purrr        1.0.4

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()      masks scales::discard()
## x recipes::discretize() masks arules::discretize()
## x tidyr::expand()       masks Matrix::expand()
## x dplyr::filter()       masks stats::filter()
## x dplyr::lag()          masks stats::lag()
## x tidyr::pack()         masks Matrix::pack()
## x dplyr::recode()       masks arules::recode()
## x recipes::step()       masks stats::step()
## x tidyr::unpack()       masks Matrix::unpack()
## x recipes::update()     masks Matrix::update(), stats::update()
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

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

```r
library(ranger)
```

```
##
## Attaching package: 'ranger'

## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift

## The following objects are masked from 'package:recommenderlab':
##
##     MAE, RMSE
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.4     v stringr   1.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor()     masks scales::col_factor()
## x randomForest::combine() masks dplyr::combine()
## x purrr::discard()        masks scales::discard()
## x tidyr::expand()         masks Matrix::expand()
## x dplyr::filter()         masks stats::filter()
## x stringr::fixed()        masks recipes::fixed()
## x dplyr::lag()            masks stats::lag()
## x caret::lift()           masks purrr::lift()
## x randomForest::margin()  masks ggplot2::margin()
## x tidyr::pack()           masks Matrix::pack()
## x dplyr::recode()         masks arules::recode()
## x readr::spec()           masks yardstick::spec()
## x tidyr::unpack()         masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

Problem 1

```
Work in written document.
Part 1 Answer: {2345}, {2356}
Part 2 Answer:
  {234} -> {5}
  {245} -> {3}
  {345} -> {2}
  {24} -> {35}
  {34} -> {25}
```

Problem 2

(1). Load the data, discard the transaction id feature, and convert the data frame from strings containing "True" and "False" to a matrix of logical values (TRUE and FALSE). Then convert this matrix to a transactions object. Then use the an apriori rule miner with a minimum support of 15% and a minimum confidence of 50%. How many rules are mined? (2). Determine which rule has the highest confidence. What is this rule? Capture the portion of the output that states the rule, as well as the support, confidence, coverage, and lift. (3). Return to the matrix of logical values. For the rule you identified in (2), find the number of transactions with both the antecedent and consequent itemsets, the number of transactions with the antecedent itemset, and the number of transactions with the consequent itemset. Use these values to compute the coverage (LHS- support), support, confidence, and lift for the rule. Provide any code you use to do these calculations. They should match the results displayed in step (2).

```
# Load data
data = read.csv('basketanalysis.csv')

# Discard transaction ID
data = subset(data, select = -X)

# Convert True False strings to Booleans
for (col_name in colnames(data)) {
  data[[col_name]] = as.logical(data[[col_name]])
}

# Convert to transactions object
trans = as(data, "transactions")

# Mine strong rules
rules = apriori(trans, parameter = list(supp = 0.15, conf = 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.15      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 149
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[16 item(s), 999 transaction(s)] done [0.00s].
## sorting and recoding items ... [16 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [5 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
num_rules = length(rules)

# Find highest confidence
rules_sorted = sort(rules, by = "confidence", decreasing = TRUE)
inspect(rules_sorted[1])
```

```
##     lhs       rhs         support   confidence coverage  lift     count
```

```
## [1] {Milk} => {chocolate} 0.2112112 0.5209877  0.4054054 1.236263 211
```

```r
quality(rules_sorted)[1, ]
```

```
##     support confidence  coverage     lift count
## 3 0.2112112  0.5209877 0.4054054 1.236263   211
```

```r
# Calculate LHS support
lhs_support = mean(data[, 'Milk'] == 1)
print(lhs_support)
```

```
## [1] 0.4054054
```

```r
# Calculate support
support = mean((data[, 'Milk'] == 1) & (data[, 'chocolate'] == 1))
print(support)
```

```
## [1] 0.2112112
```

```r
# Calculate confidence
confidence = support / lhs_support
print(confidence)
```

```
## [1] 0.5209877
```

```r
# Calculate lift
rhs_support = mean(data[, 'chocolate'] == 1)
lift = confidence / rhs_support
print(lift)
```

```
## [1] 1.236263
```

```
1.  There are 5 strong rules mined.
2. The highest confidence rule is {milk} -> {chocolate}.This has a support of
0.211, a confidence of 0.521, a coverage of 0.405, and a lift of 1.236.
3. The answers are the same with a support of 0.211, a confidence of 0.521, a
coverage of 0.405, and a lift of 1.236.
```

Problem 3

(1). Load the data and force the data.frame to a realRatingMatrix object. Fit a user- based collaborative filtering model to all but 3 of the users (you can choose which three to hold out). (2). Use the model to make predictions for the three users you held back.

```r
# Load data
data = read.csv('bookratings-small.csv')

# Force into a realRatingMatrix
ratings_wide <- data %>%
  pivot_wider(names_from = ISBN, values_from = Book.Rating,
              values_fn = mean, values_fill = NA_real_) %>%
  arrange(User.ID)

row_ids = ratings_wide$User.ID
rating_mat = as.matrix(ratings_wide[, -1])
rownames(rating_mat) = row_ids
rating_matrix = as(rating_mat, "realRatingMatrix")
```

```r
# Fit collaborative filtering model (all but 3 users)
holdout = 1:3
recommender = Recommender(rating_matrix[-holdout], method = "UBCF")

# Predict
pred_ratings = predict(recommender, rating_matrix[holdout], type = "ratings")
pred_list = as(pred_ratings, "list")
actual_ids = ratings_wide$User.ID[holdout]

top3_list <- Map(function(uid, v) {
  v <- sort(v, decreasing = TRUE)
  head(
    data.frame(user = uid, ISBN = names(v), rating = v, row.names = NULL), 3
    )
  }, actual_ids, pred_list)

top3_df = do.call(rbind, top3_list)
print(top3_df)
```

```
##   user       ISBN   rating
## 1  183 840142321X 10.60825
## 2  183 8473861906 10.60825
## 3  183 0141439475 10.60825
## 4 1733 0446606189 12.70757
## 5 1733 0451132378 12.70757
## 6 1733 0671042262 12.70757
## 7 2033 0451152301 11.71970
## 8 2033 0553254030 11.71970
## 9 2033 0886777844 11.71970
```

Problem 4

Consider the following A/B testing scenario for a manufacturing plant which is having a problem with a customer refusing to buy their product because too many of the items produced have a manufacturing defect. The plant makes some changes to the production process and uses A/B testing to determine whether they have made a significant reduction of the defect rate. Before the changes, the plant manufactured 200 products for the customer, but 20 of them had defects. After making their process improvements, they made another 250 products, and only 10 of them had defects. It appears the defect rate has decreased. Use a statistical test to determine whether the reduction is statistically significant.

```r
# Before improvement
nA = 200
defectA = 20
successA = nA - defectA

# After improvement
nB = 250
defectB = 10
successB = nB - defectB

# Test
prop.test(c(defectA, defectB), c(nA, nB), alt = "greater", correct = FALSE)
```

```
##
```

```
##  2-sample test for equality of proportions without continuity correction
##
## data:  c(defectA, defectB) out of c(nA, nB)
## X-squared = 6.4286, df = 1, p-value = 0.005615
## alternative hypothesis: greater
## 95 percent confidence interval:
##  0.01958879 1.00000000
## sample estimates:
## prop 1 prop 2
##   0.10   0.04
```

Because the p value is less than 0.05 we can reject the null hypothesis. This means that the reduction is statistically significant.

Problem 5

(1). Generate training and holdout partitions on the data set. Use 1/3 of the data in the holdout. (2). Fit a random forest model, applying a training grid to tune the parameters. Use the random forest model to make probability-metric predictions on two versions of the holdout data: one with the offer (treatment) feature set to Discount and the other with the offer feature set to No Offer. Then compute the uplift for the treatment. Obtain Q1, the median, and Q3 for the predicted uplift. Make conclusions.

```r
# Load data
data = read.csv('uplift-small.csv')

# Train test split
set.seed(42)
split <- initial_split(data, prop = 2/3, strata = conversion)
train <- training(split)
test <- testing(split)

# Fit random forest model
train_control <- trainControl(method="cv")
rf.grid = expand.grid(mtry = c(3, 4, 5))
model <- train(conversion ~ ., data = train,
                      trControl = train_control,
                      method = "rf", tuneGrid = rf.grid)

# Probability Metric Predictions

# No Offer
uplift_df = test
uplift_df$offer <- 'No Offer'
predTreatment <- predict(model, newdata = uplift_df, type = "prob")
predTreatment <- tibble::rowid_to_column(predTreatment, "CID")
head(predTreatment)
```

```
##   CID    No   Yes
## 1   1 1.000 0.000
## 2   2 0.988 0.012
## 3   3 0.952 0.048
## 4   4 0.884 0.116
## 5   5 0.890 0.110
## 6   6 0.974 0.026
```

```r
# Discount
uplift_df = test
uplift_df$offer <- 'Discount'
predControl <- predict(model, newdata = uplift_df, type = "prob")
predControl <- tibble::rowid_to_column(predControl, "CID")
head(predControl)
```

```
##   CID    No   Yes
## 1   1 0.956 0.044
## 2   2 0.828 0.172
## 3   3 0.966 0.034
## 4   4 0.934 0.066
## 5   5 0.976 0.024
## 6   6 0.838 0.162
```

```r
# Calculate uplift
upliftResult <- data.frame(CID = predTreatment$CID,
  probYesPromotion = predTreatment[, 3],
  probNoPromotion = predControl[, 3],
  uplift = predTreatment[, 3] - predControl[, 3]
)
head(upliftResult)
```

```
##   CID probYesPromotion probNoPromotion uplift
## 1   1            0.000           0.044 -0.044
## 2   2            0.012           0.172 -0.160
## 3   3            0.048           0.034  0.014
## 4   4            0.116           0.066  0.050
## 5   5            0.110           0.024  0.086
## 6   6            0.026           0.162 -0.136
```

```r
# select uplift > 5%
uplift_0.05 <- upliftResult[upliftResult$uplift > 0.05, ]
# sort
sorted_uplift <- uplift_0.05[order(-uplift_0.05$uplift), ]
sorted_uplift
```

```
##       CID probYesPromotion probNoPromotion uplift
## 8920 8920            0.370           0.116  0.254
## 7213 7213            0.266           0.042  0.224
## 14     14            0.328           0.120  0.208
## 4284 4284            0.274           0.068  0.206
## 5539 5539            0.216           0.014  0.202
## 6700 6700            0.276           0.082  0.194
## 8311 8311            0.330           0.140  0.190
## 2690 2690            0.304           0.118  0.186
## 8755 8755            0.270           0.084  0.186
## 1792 1792            0.210           0.026  0.184
## 4294 4294            0.198           0.016  0.182
## 6119 6119            0.272           0.094  0.178
## 7314 7314            0.238           0.060  0.178
## 8586 8586            0.254           0.076  0.178
## 2615 2615            0.200           0.024  0.176
## 6166 6166            0.240           0.064  0.176
## 890   890            0.222           0.052  0.170
```

```
## 3251 3251              0.258            0.088  0.170
## 1206 1206              0.262            0.094  0.168
## 3343 3343              0.246            0.078  0.168
## 8422 8422              0.212            0.044  0.168
## 7544 7544              0.280            0.116  0.164
## 4751 4751              0.322            0.158  0.164
## 6880 6880              0.176            0.012  0.164
## 6501 6501              0.198            0.038  0.160
## 1918 1918              0.180            0.022  0.158
## 5318 5318              0.172            0.018  0.154
## 6075 6075              0.202            0.050  0.152
## 8841 8841              0.216            0.066  0.150
## 289   289              0.204            0.056  0.148
## 1725 1725              0.192            0.048  0.144
## 4794 4794              0.154            0.010  0.144
## 7916 7916              0.212            0.068  0.144
## 4365 4365              0.326            0.184  0.142
## 1573 1573              0.150            0.008  0.142
## 5484 5484              0.246            0.106  0.140
## 5361 5361              0.262            0.124  0.138
## 5447 5447              0.196            0.058  0.138
## 8692 8692              0.200            0.062  0.138
## 9209 9209              0.322            0.184  0.138
## 3586 3586              0.144            0.010  0.134
## 1306 1306              0.156            0.026  0.130
## 6536 6536              0.194            0.064  0.130
## 9374 9374              0.176            0.046  0.130
## 2578 2578              0.248            0.120  0.128
## 8759 8759              0.164            0.036  0.128
## 1205 1205              0.148            0.022  0.126
## 1900 1900              0.248            0.122  0.126
## 2018 2018              0.130            0.004  0.126
## 2887 2887              0.174            0.048  0.126
## 6794 6794              0.314            0.188  0.126
## 8698 8698              0.160            0.036  0.124
## 397   397              0.174            0.052  0.122
## 5189 5189              0.156            0.034  0.122
## 2333 2333              0.206            0.084  0.122
## 1704 1704              0.168            0.048  0.120
## 5257 5257              0.186            0.066  0.120
## 6837 6837              0.186            0.066  0.120
## 7757 7757              0.248            0.128  0.120
## 7563 7563              0.170            0.052  0.118
## 5567 5567              0.134            0.016  0.118
## 606   606              0.210            0.092  0.118
## 1005 1005              0.172            0.054  0.118
## 2550 2550              0.158            0.042  0.116
## 1583 1583              0.170            0.058  0.112
## 5581 5581              0.156            0.044  0.112
## 9316 9316              0.162            0.050  0.112
## 1503 1503              0.152            0.040  0.112
## 7990 7990              0.146            0.036  0.110
## 8369 8369              0.130            0.022  0.108
## 3717 3717              0.118            0.010  0.108
```

```
## 3356 3356            0.234           0.128  0.106
## 7829 7829            0.136           0.030  0.106
## 9102 9102            0.116           0.010  0.106
## 7062 7062            0.156           0.050  0.106
## 7378 7378            0.210           0.104  0.106
## 5689 5689            0.164           0.060  0.104
## 738   738            0.104           0.000  0.104
## 5363 5363            0.104           0.000  0.104
## 6390 6390            0.104           0.000  0.104
## 8399 8399            0.104           0.000  0.104
## 8656 8656            0.182           0.078  0.104
## 7743 7743            0.104           0.002  0.102
## 6370 6370            0.140           0.040  0.100
## 7876 7876            0.112           0.012  0.100
## 9161 9161            0.268           0.168  0.100
## 3573 3573            0.106           0.006  0.100
## 3887 3887            0.102           0.002  0.100
## 8386 8386            0.102           0.002  0.100
## 1062 1062            0.100           0.002  0.098
## 1078 1078            0.128           0.030  0.098
## 5152 5152            0.116           0.018  0.098
## 8785 8785            0.158           0.060  0.098
## 6111 6111            0.102           0.004  0.098
## 3941 3941            0.134           0.038  0.096
## 4781 4781            0.132           0.036  0.096
## 7746 7746            0.120           0.024  0.096
## 8202 8202            0.226           0.130  0.096
## 8840 8840            0.150           0.054  0.096
## 8944 8944            0.108           0.012  0.096
## 9163 9163            0.222           0.126  0.096
## 171   171            0.098           0.004  0.094
## 2054 2054            0.150           0.056  0.094
## 6492 6492            0.118           0.024  0.094
## 7665 7665            0.100           0.006  0.094
## 8066 8066            0.192           0.098  0.094
## 8911 8911            0.174           0.080  0.094
## 1251 1251            0.098           0.006  0.092
## 3463 3463            0.162           0.070  0.092
## 7755 7755            0.104           0.012  0.092
## 9339 9339            0.112           0.020  0.092
## 3144 3144            0.232           0.142  0.090
## 8040 8040            0.274           0.184  0.090
## 4348 4348            0.134           0.044  0.090
## 268   268            0.090           0.000  0.090
## 2990 2990            0.106           0.016  0.090
## 7922 7922            0.092           0.002  0.090
## 100   100            0.116           0.028  0.088
## 292   292            0.116           0.028  0.088
## 2911 2911            0.114           0.026  0.088
## 4637 4637            0.098           0.010  0.088
## 1094 1094            0.156           0.068  0.088
## 3174 3174            0.090           0.002  0.088
## 4372 4372            0.104           0.016  0.088
## 5155 5155            0.110           0.022  0.088
```

```
## 6024 6024              0.130            0.042  0.088
## 6057 6057              0.108            0.020  0.088
## 7691 7691              0.110            0.022  0.088
## 8863 8863              0.108            0.020  0.088
## 4037 4037              0.096            0.010  0.086
## 4219 4219              0.112            0.026  0.086
## 5018 5018              0.114            0.028  0.086
## 8349 8349              0.132            0.046  0.086
## 8709 8709              0.130            0.044  0.086
## 9108 9108              0.130            0.044  0.086
## 5       5              0.110            0.024  0.086
## 3424 3424              0.126            0.040  0.086
## 7183 7183              0.142            0.056  0.086
## 7265 7265              0.148            0.062  0.086
## 8392 8392              0.092            0.006  0.086
## 8238 8238              0.228            0.144  0.084
## 2936 2936              0.156            0.072  0.084
## 5423 5423              0.130            0.046  0.084
## 3242 3242              0.138            0.056  0.082
## 270   270              0.084            0.002  0.082
## 3960 3960              0.160            0.078  0.082
## 9100 9100              0.130            0.048  0.082
## 1390 1390              0.102            0.020  0.082
## 3925 3925              0.242            0.160  0.082
## 5530 5530              0.176            0.094  0.082
## 5835 5835              0.258            0.178  0.080
## 893   893              0.098            0.018  0.080
## 6173 6173              0.098            0.018  0.080
## 7672 7672              0.080            0.000  0.080
## 7736 7736              0.090            0.010  0.080
## 2639 2639              0.086            0.006  0.080
## 8194 8194              0.148            0.068  0.080
## 3774 3774              0.138            0.060  0.078
## 4700 4700              0.200            0.122  0.078
## 6584 6584              0.140            0.062  0.078
## 8549 8549              0.100            0.022  0.078
## 106   106              0.084            0.006  0.078
## 1917 1917              0.126            0.048  0.078
## 3456 3456              0.110            0.032  0.078
## 3757 3757              0.096            0.018  0.078
## 7986 7986              0.108            0.030  0.078
## 9010 9010              0.110            0.032  0.078
## 8949 8949              0.086            0.008  0.078
## 9159 9159              0.146            0.068  0.078
## 4495 4495              0.084            0.008  0.076
## 8685 8685              0.084            0.008  0.076
## 9363 9363              0.130            0.054  0.076
## 1340 1340              0.090            0.014  0.076
## 1605 1605              0.180            0.104  0.076
## 7494 7494              0.088            0.012  0.076
## 510   510              0.116            0.042  0.074
## 916   916              0.116            0.042  0.074
## 5588 5588              0.134            0.060  0.074
## 7012 7012              0.082            0.008  0.074
```

```
## 1126 1126              0.092             0.018  0.074
## 4444 4444              0.102             0.028  0.074
## 4448 4448              0.092             0.018  0.074
## 5358 5358              0.092             0.018  0.074
## 5735 5735              0.094             0.020  0.074
## 6450 6450              0.122             0.048  0.074
## 3584 3584              0.208             0.134  0.074
## 300   300              0.128             0.056  0.072
## 1450 1450              0.224             0.152  0.072
## 2305 2305              0.130             0.058  0.072
## 5891 5891              0.094             0.022  0.072
## 5898 5898              0.112             0.040  0.072
## 6272 6272              0.080             0.008  0.072
## 6529 6529              0.166             0.094  0.072
## 31     31              0.118             0.046  0.072
## 588   588              0.120             0.048  0.072
## 6296 6296              0.152             0.080  0.072
## 7798 7798              0.090             0.018  0.072
## 8102 8102              0.174             0.102  0.072
## 8659 8659              0.076             0.004  0.072
## 7560 7560              0.078             0.008  0.070
## 7562 7562              0.080             0.010  0.070
## 7818 7818              0.084             0.014  0.070
## 8252 8252              0.130             0.060  0.070
## 8253 8253              0.114             0.044  0.070
## 8860 8860              0.106             0.036  0.070
## 5523 5523              0.074             0.004  0.070
## 6903 6903              0.118             0.048  0.070
## 7300 7300              0.142             0.072  0.070
## 8028 8028              0.142             0.072  0.070
## 355   355              0.078             0.010  0.068
## 1422 1422              0.190             0.122  0.068
## 2316 2316              0.070             0.002  0.068
## 2572 2572              0.096             0.028  0.068
## 2646 2646              0.098             0.030  0.068
## 4793 4793              0.080             0.012  0.068
## 7423 7423              0.126             0.058  0.068
## 8207 8207              0.126             0.058  0.068
## 8894 8894              0.078             0.010  0.068
## 8834 8834              0.118             0.050  0.068
## 2857 2857              0.068             0.002  0.066
## 3143 3143              0.074             0.008  0.066
## 3659 3659              0.074             0.008  0.066
## 4848 4848              0.082             0.016  0.066
## 5142 5142              0.094             0.028  0.066
## 6708 6708              0.070             0.004  0.066
## 1408 1408              0.144             0.078  0.066
## 8604 8604              0.174             0.108  0.066
## 486   486              0.096             0.032  0.064
## 593   593              0.074             0.010  0.064
## 2499 2499              0.066             0.002  0.064
## 2914 2914              0.080             0.016  0.064
## 3460 3460              0.080             0.016  0.064
## 3907 3907              0.080             0.016  0.064
```

```
## 4533 4533              0.086            0.022  0.064
## 5935 5935              0.068            0.004  0.064
## 6459 6459              0.156            0.092  0.064
## 6473 6473              0.070            0.006  0.064
## 7705 7705              0.074            0.010  0.064
## 7735 7735              0.116            0.052  0.064
## 7796 7796              0.064            0.000  0.064
## 7827 7827              0.112            0.048  0.064
## 7984 7984              0.080            0.016  0.064
## 8279 8279              0.120            0.056  0.064
## 8446 8446              0.086            0.022  0.064
## 8486 8486              0.100            0.036  0.064
## 1945 1945              0.136            0.074  0.062
## 8532 8532              0.136            0.074  0.062
## 537   537              0.070            0.008  0.062
## 676   676              0.100            0.038  0.062
## 2230 2230              0.070            0.008  0.062
## 7793 7793              0.068            0.006  0.062
## 7834 7834              0.084            0.022  0.062
## 47     47              0.122            0.060  0.062
## 1273 1273              0.078            0.016  0.062
## 5386 5386              0.078            0.016  0.062
## 6391 6391              0.110            0.048  0.062
## 7305 7305              0.092            0.030  0.062
## 7998 7998              0.082            0.020  0.062
## 639   639              0.104            0.042  0.062
## 3265 3265              0.104            0.042  0.062
## 3459 3459              0.120            0.058  0.062
## 6691 6691              0.118            0.056  0.062
## 2157 2157              0.146            0.084  0.062
## 812   812              0.114            0.054  0.060
## 5500 5500              0.116            0.056  0.060
## 8762 8762              0.066            0.006  0.060
## 1246 1246              0.086            0.026  0.060
## 1348 1348              0.092            0.032  0.060
## 1972 1972              0.194            0.134  0.060
## 2178 2178              0.104            0.044  0.060
## 2221 2221              0.072            0.012  0.060
## 4462 4462              0.080            0.020  0.060
## 5982 5982              0.090            0.030  0.060
## 7064 7064              0.072            0.012  0.060
## 7236 7236              0.060            0.000  0.060
## 8782 8782              0.150            0.090  0.060
## 9249 9249              0.074            0.014  0.060
## 4523 4523              0.144            0.084  0.060
## 326   326              0.070            0.012  0.058
## 1301 1301              0.070            0.012  0.058
## 6269 6269              0.084            0.026  0.058
## 5067 5067              0.068            0.010  0.058
## 5642 5642              0.068            0.010  0.058
## 6824 6824              0.112            0.054  0.058
## 7527 7527              0.066            0.008  0.058
## 2671 2671              0.120            0.062  0.058
## 2767 2767              0.106            0.048  0.058
```

```
## 3437 3437              0.072              0.014  0.058
## 3555 3555              0.196              0.138  0.058
## 5577 5577              0.062              0.004  0.058
## 7516 7516              0.102              0.044  0.058
## 8398 8398              0.104              0.046  0.058
## 9036 9036              0.102              0.044  0.058
## 9136 9136              0.074              0.016  0.058
## 5477 5477              0.166              0.110  0.056
## 6455 6455              0.100              0.044  0.056
## 7152 7152              0.100              0.044  0.056
## 9248 9248              0.082              0.026  0.056
## 1326 1326              0.056              0.000  0.056
## 1475 1475              0.110              0.054  0.056
## 1915 1915              0.056              0.000  0.056
## 3465 3465              0.096              0.040  0.056
## 4095 4095              0.058              0.002  0.056
## 4647 4647              0.078              0.022  0.056
## 7774 7774              0.108              0.052  0.056
## 8407 8407              0.062              0.006  0.056
## 424   424              0.150              0.094  0.056
## 3275 3275              0.118              0.062  0.056
## 4274 4274              0.076              0.020  0.056
## 4393 4393              0.104              0.048  0.056
## 4682 4682              0.060              0.004  0.056
## 5026 5026              0.122              0.066  0.056
## 6284 6284              0.072              0.016  0.056
## 7181 7181              0.060              0.004  0.056
## 7249 7249              0.076              0.020  0.056
## 8006 8006              0.104              0.048  0.056
## 317   317              0.100              0.046  0.054
## 1445 1445              0.156              0.102  0.054
## 3044 3044              0.066              0.012  0.054
## 3208 3208              0.098              0.044  0.054
## 3691 3691              0.116              0.062  0.054
## 4978 4978              0.066              0.012  0.054
## 5006 5006              0.066              0.012  0.054
## 6025 6025              0.082              0.028  0.054
## 6754 6754              0.058              0.004  0.054
## 674   674              0.064              0.010  0.054
## 2234 2234              0.054              0.000  0.054
## 2467 2467              0.060              0.006  0.054
## 5445 5445              0.060              0.006  0.054
## 5710 5710              0.056              0.002  0.054
## 5739 5739              0.078              0.024  0.054
## 5797 5797              0.054              0.000  0.054
## 7191 7191              0.054              0.000  0.054
## 8153 8153              0.062              0.008  0.054
## 8416 8416              0.096              0.042  0.054
## 9325 9325              0.078              0.024  0.054
## 2032 2032              0.122              0.068  0.054
## 4275 4275              0.074              0.020  0.054
## 4317 4317              0.072              0.018  0.054
## 1295 1295              0.056              0.004  0.052
## 2611 2611              0.100              0.048  0.052
```

```
## 2867 2867            0.064          0.012  0.052
## 3367 3367            0.064          0.012  0.052
## 4174 4174            0.124          0.072  0.052
## 4732 4732            0.068          0.016  0.052
## 8654 8654            0.064          0.012  0.052
## 747   747            0.060          0.008  0.052
## 1352 1352            0.062          0.010  0.052
## 1485 1485            0.104          0.052  0.052
## 3564 3564            0.110          0.058  0.052
## 3692 3692            0.060          0.008  0.052
## 4101 4101            0.110          0.058  0.052
## 4192 4192            0.110          0.058  0.052
## 4206 4206            0.104          0.052  0.052
```

```r
# Q1, the median, and Q3 for the predicted uplift
summary_stats = quantile(upliftResult$uplift, probs = c(0.25, 0.5, 0.75))
print(summary_stats)
```

```
##    25%    50%    75%
## -0.036 -0.010  0.002
```

Since the median uplift is less than 0, this means the discount has an overall negative effect. Additionally the third quartile is 0.002 which is very close to 0. This means that if the discount isn't working it simply has no effect so we would be wasting resources.