STA 380 Homework2: Li_Peng_Wang

Daniel Peng, Anying Li, Jiaqiu Wang

August 15th, 2016

Q1: Flights at ABIA

Create a figure, or set of related figures, that tell an interesting story about flights into and out of Austin. You can annotate the figure and briefly describe it, but strive to make it as stand-alone as possible. It shouldn't need many, many paragraphs to convey its meaning. Rather, the figure should speak for itself as far as possible. For example, you might consider one of the following questions:

What is the best time of day to fly to minimize delays? What is the best time of year to fly to minimize delays? How do patterns of flights to different destinations or parts of the country change over the course of the year? What are the bad airports to fly to?

But anything interesting will fly.

```
library(ggplot2)
library(plyr)
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:plyr':
##
##     here

## The following object is masked from 'package:base':
##
##     date
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

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

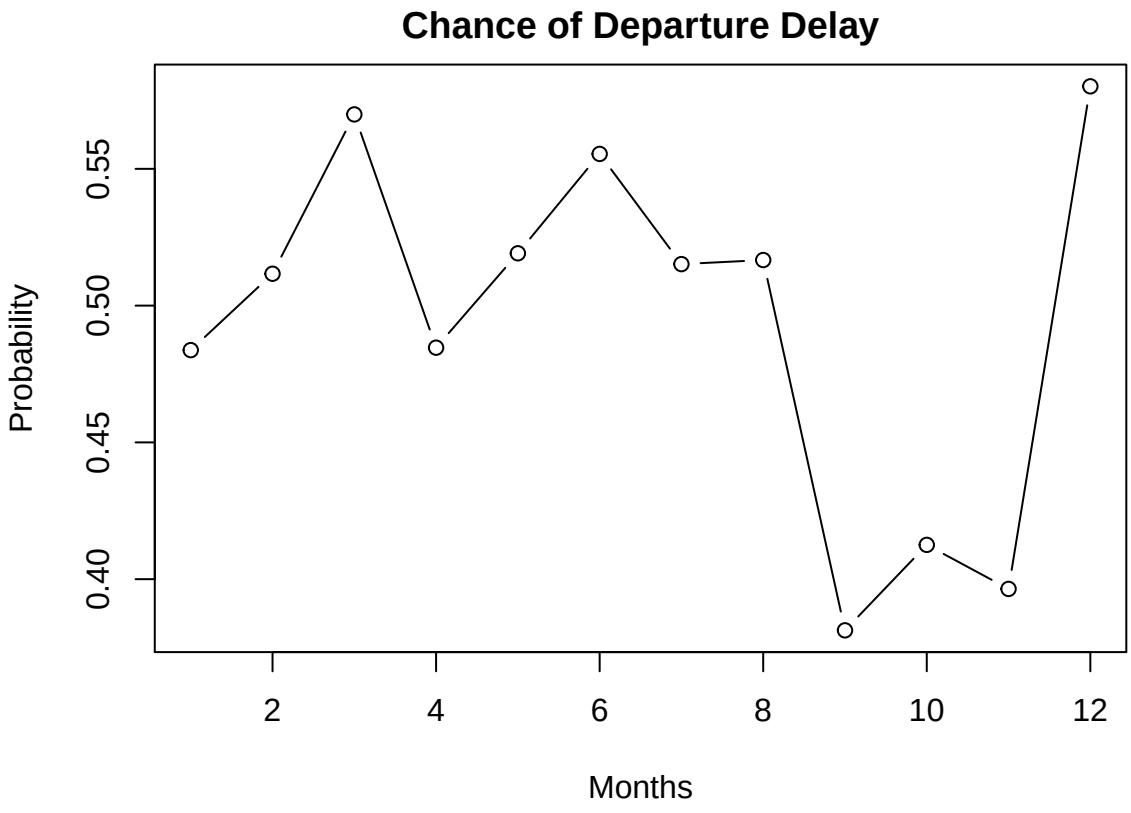
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
ABIA = read.csv("https://raw.githubusercontent.com/jgscott/STA380/master/data/ABIA.csv",header=T)
```

read in the files and load in related library.

what is the best month to fly to minimize delay?

```
flight.count = ddply(ABIA, ~Month, summarise, total_flight = length(FlightNum))

delayed = ABIA[which(ABIA$DepDelay >= 0),]
delayed.count = ddply(delayed, ~Month, summarise, delayed_flight = length(FlightNum))

delay.chance = delayed.count$delayed_flight /flight.count$total_flight

par(mar=c(5,4,2,3)+.1)
plot(delay.chance, xlab = "Months", ylab = 'Probability', type = 'b', main = 'Chance of Departure Delay
```

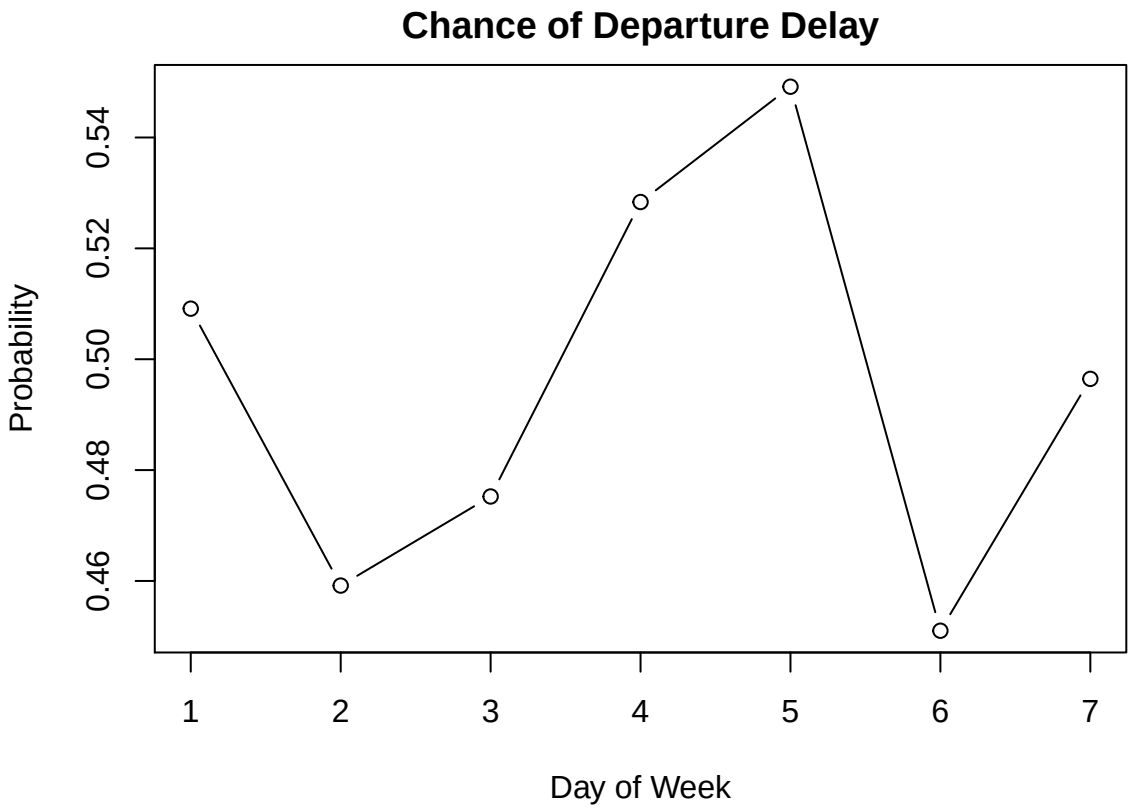


After having a few scatterplot, we found that there are lots of outliers for the delay data, and we cannot simply use mean as a measure for delay. We decide to look at the probability of delay–although this cannot reflect the length of the delay, this can be compensated by our later analysis.

According to the plot, flights departuring in September to November have the lowest chance of being delayed while the flights in December have the highest. One explaination to this phenomenon is that people tend to travel less before the holiday season to save vacation days. Thus the airport is less busy. So This is the best time to fly from Austin if one is time sensitive.

what is the best day of a week to fly to minimize delay?

```
flight.count_2 = ddply(ABIA, ~DayOfWeek, summarise, total_flight = length(FlightNum))
delayed.count_2 = ddply(delayed, ~DayOfWeek, summarise, delayed_flight = length(FlightNum))
delay.chance_2 = delayed.count_2$delayed_flight /flight.count_2$total_flight

par(mar=c(5,4,2,3)+.1)
plot(delay.chance_2, xlab = "Day of Week", ylab = 'Probability', type = 'b', main = 'Chance of Departure
```

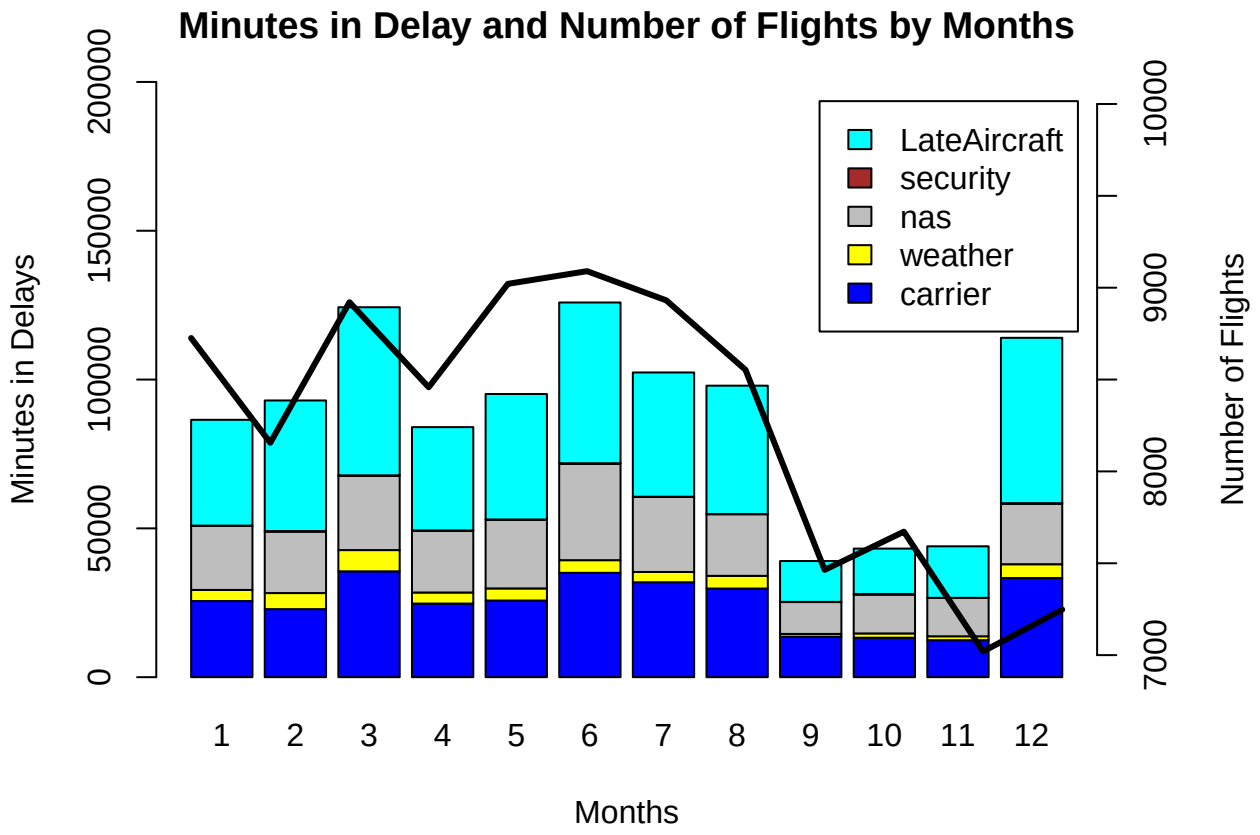


The probability of delay peaks on Friday and dips on Saturday. There is also a pickup on Sunday and Monday. This is because many business people travel to work on Sundays or Mondays, and go home on Fridays. Many people also choose to fly on Fridays and go home on Sundays for weekend trips.

what cause the delay?

```
a = ABIA[,c('Month','WeatherDelay','NASDelay','CarrierDelay','SecurityDelay','LateAircraftDelay')]
a[is.na(a)] <- 0

delay.reason = ddply(a,.(Month),summarise,carrier =sum(CarrierDelay), weather=sum(WeatherDelay), nas=sum

delay.reason = t(delay.reason)

par(mar=c(4,4,3,4))
barplot(as.matrix(delay.reason)[2:6,], col = c('blue', 'yellow', 'grey', 'brown', 'cyan'), main = "Minu
```

```r
par(new = T)
with(flight.count, plot(flight.count$Month, flight.count$total_flight, type = "l", axes=F, xlab='Months
axis(side = 4)
mtext(side = 4, line = 3, 'Number of Flights')
```

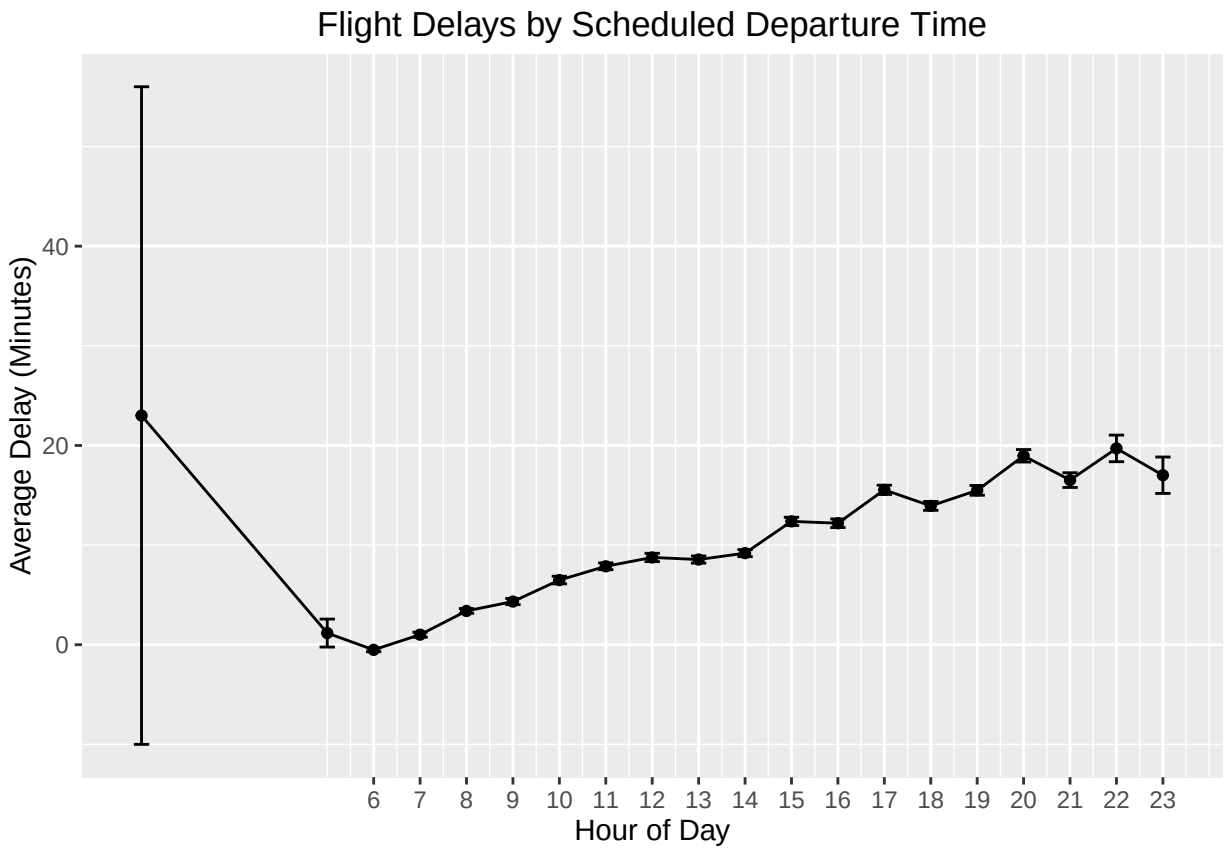


Late Aircraft, carrier and NAS are the main reasons for delays. The trend in total delay minutes follows the trend in the number of flights throughout the year, with the exception of December. In Decenmber, the airport experiences very high delay time despite low flight volume. Majority of the delays is caused by late aircrafts. This can be caused by many reasons, such as severe weather at the origin or staff shortage at the previous airport since it's the holiday season, etc.

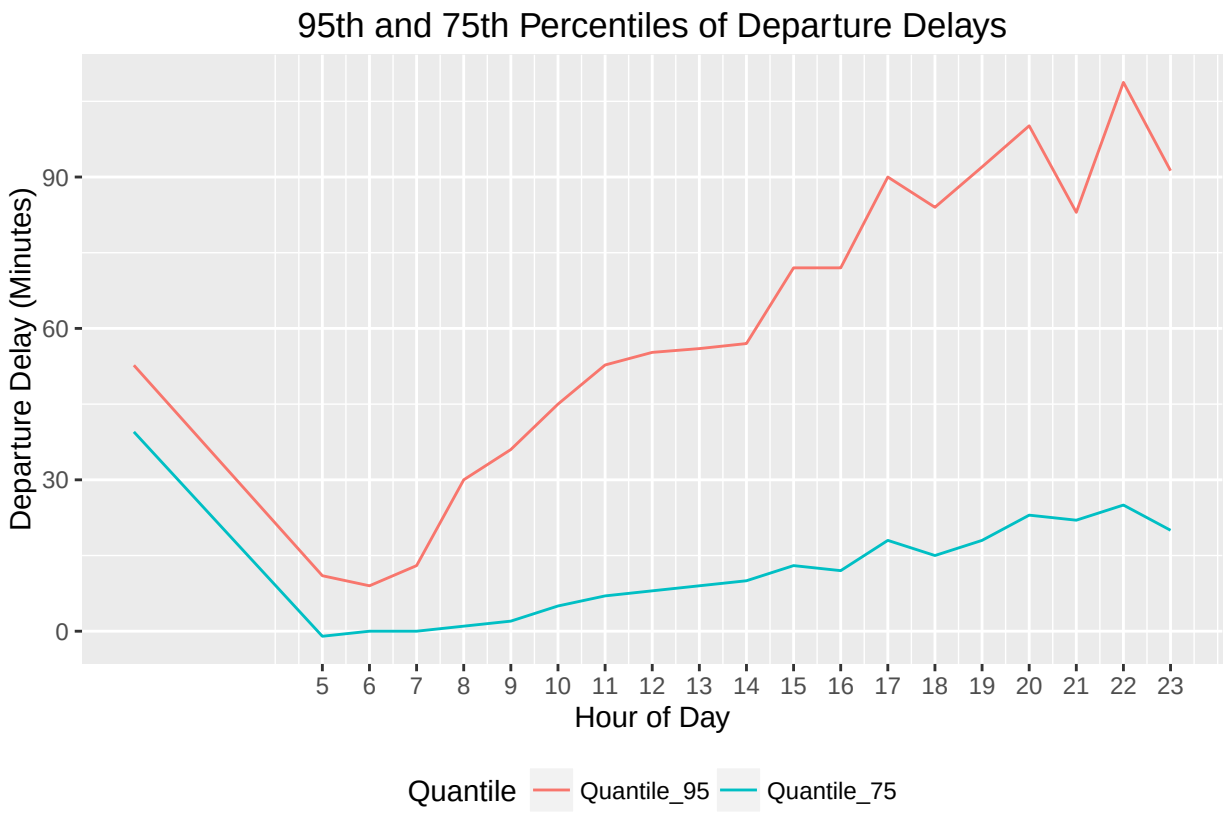
when is the best time to fly during a day to minimize delay?

```r
#departure delays as fun of scheduled departure time
ABIA$CRSDepTime = round(ABIA$CRSDepTime/100,0)

plot_data = ABIA %>%
  gather(DelayType,newdelay,DepDelay) %>%
  group_by(CRSDepTime) %>%
  dplyr::summarise(mu=mean(newdelay,na.rm=TRUE),
                   se=sqrt(var(newdelay,na.rm=TRUE)/length(na.omit(newdelay))),
                   obs=length(na.omit(newdelay)))

p=ggplot(plot_data,aes(x=CRSDepTime,y=mu,min=mu-se,max=mu+se)) +
  geom_line() +
```

```
  geom_point() +
  geom_errorbar(width=.33) +
  scale_x_continuous(breaks=seq(6,23)) +
  labs(x="Hour of Day",y="Average Delay (Minutes)",title="Flight Delays by Scheduled Departure Time") +
  theme(legend.position="bottom")
p
```




```
#Just the 95% and 75% quantiles
plot_data = ABIA %>%
  group_by(CRSDepTime) %>%
  dplyr::summarise(Quantile_95=quantile(DepDelay,.95,na.rm=TRUE),
                   Quantile_75=quantile(DepDelay,.75,na.rm=TRUE),
                   obs=length(na.omit(DepDelay)))
plot_data2 = plot_data %>%
  gather(variable, value, Quantile_75:Quantile_95) %>%
  mutate(variable=factor(variable,levels=c("Quantile_95","Quantile_75")))

p=ggplot(plot_data2,aes(x=CRSDepTime,y=value,group=variable,color=variable)) +
  geom_line() +
  scale_x_continuous(breaks=seq(5,23)) +
  labs(x="Hour of Day",y="Departure Delay (Minutes)",title="95th and 75th Percentiles of Departure Delay
  scale_color_discrete(name="Quantile") +
  theme(legend.position="bottom")
p
```



From the plot we can see that the later you leave, the greater the average delay you will face. Planes even leave earlier than schedule departure time when the airport just begin to operate in the early morning. It makes sense that delays increase as the day goes on, as we showed in the earlier graph, the primary cause of delays is waiting for the plane to arrive from another airport. The first flights out in the morning don't have this problem.

By looking at the 75% and 95% percentile chart, we can get an idea that if one's flight is scheduled at night after 7pm, there is a 25% chance that the plane will be delayed for about 15-20 minutes, and there is 5% chance that the plane will be delayed over 90 minutes.

Q2 Author attribution

Model 1: TF-IDF with Principal Component Analysis

Set up the environment:

```
rm(list=ls())
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

Read the training data:

```
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }

author_dirs = Sys.glob('/Users/leeanthea/STA380-master/data/ReutersC50/C50train/*')
file_list = NULL
labels = NULL
for(author in author_dirs) {
  author_name = substring(author, first=57)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  labels = append(labels, rep(author_name, length(files_to_add)))
}

all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list
my_corpus = tm_map(my_corpus, content_transformer(tolower))
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers))
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation))
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace))
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords("SMART"))

unique_author  = unique(labels)
```

Calculate the TFIDF matrix of the training dataset:

```
idf.weight <- function(x) {
  doc.freq <- colSums(x>0)
  doc.freq[doc.freq == 0] <- 1
  w <- log(nrow(x)/doc.freq)
  return(scale.cols(x,w))
}

scale.cols <- function(x,s) {
  return(t(apply(x,1,function(x){x*s})))
}

my_cosine = function(v1, v2) {
  result= NULL
  for(u in (1:50)){
    result[u] = sum(v1 %*% v2[,u]) / {sqrt(sum(v1^2)) * sqrt(sum(v2[,u]^2))}
  }
  return(result)
}

DTM = DocumentTermMatrix(my_corpus)
DTM = removeSparseTerms(DTM, 0.99)
X = as.matrix(DTM)
```

```
row.names(X)=labels
DTM_TF = X / rowSums(X)
DTM_TFIDF = idf.weight(DTM_TF)
```

Read the testing data:

```
test_dirs = Sys.glob('/Users/leeanthea/STA380-master/data/ReutersC50/C50test/*')
test_list = NULL
labels_test = NULL
for(author in test_dirs) {
  author_name = substring(author, first=56)
  test_to_add = Sys.glob(paste0(author, '/*.txt'))
  test_list = append(test_list, test_to_add)
  labels_test = append(labels_test, rep(author_name, length(test_to_add)))
}


test_docs = lapply(test_list, readerPlain)
names(test_docs) = test_list
names(test_docs) = sub('.txt', '', names(test_docs))

test_corpus = Corpus(VectorSource(test_docs))
names(test_corpus) = test_list

test_corpus = tm_map(test_corpus, content_transformer(tolower))
test_corpus = tm_map(test_corpus, content_transformer(removeNumbers))
test_corpus = tm_map(test_corpus, content_transformer(removePunctuation))
test_corpus = tm_map(test_corpus, content_transformer(stripWhitespace))
test_corpus = tm_map(test_corpus, content_transformer(removeWords), stopwords("SMART"))
```

Comparing the doc-term matrices of train set and test set, we can see that some words in the test set were never seen in the training set.

```
DTM_test = DocumentTermMatrix(test_corpus)
DTM_test = removeSparseTerms(DTM_test, 0.99)
DTM_test
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3122)>>
## Non-/sparse entries: 318288/7486712
## Sparsity           : 96%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
DTM
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3076)>>
## Non-/sparse entries: 313587/7376413
## Sparsity           : 96%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

Select the intersect of train set and test set (words contained in both sets) and calculate the TFIDF matrix of the test set using only these words

```
X_test = DTM_test[,c(intersect(colnames(DTM_TFIDF), colnames(DTM_test)))]
X_test = as.matrix(X_test)
row.names(X_test)=labels_test
DTM_TF_test = X_test / rowSums(X_test)
DTM_TFIDF_test = idf.weight(DTM_TF_test)
```

Run principal component analysis on the train set to find out the latent features of documents

```
lsi = prcomp(DTM_TFIDF[,intersect(colnames(X_test), colnames(DTM_TFIDF))], scale.=FALSE)
```

Construct a query matrix with 1000 principal components and aggregate the loadings of articles written by the same author

```
query_vec = t(lsi$x[,1:1000])
colnames(query_vec)=labels
query_vec = t(rowsum(t(query_vec), group = rownames(t(query_vec))))
```

Project the TFIDF matrix of test data on the 1000 dimensions (principal components)

```
trans = DTM_TFIDF_test %*% lsi$rotation[,1:1000]
```

Find out the query vector that produces the largest inner product (cosine) with one specific document vector. Attribute the document to the author who corresponds to that query vector.

```
count = 0
predict <- list()
for(i in (1:2500)){
  temp = my_cosine(trans[i,],query_vec)
  predict[[i]] = unique_author[which.max(temp)]
  if(predict[[i]]==labels[i]){
    count = count+1
  }
}
predict = do.call(rbind, predict)
row.names(predict) = labels_test
head(predict)
```

```
##               [,1]
## AaronPressman "AaronPressman"
## AaronPressman "AaronPressman"
## AaronPressman "AaronPressman"
## AaronPressman "AaronPressman"
## AaronPressman "AaronPressman"
## AaronPressman "AaronPressman"
```

Authors whose articles seem difficult to distinguish from one another are as follow:

```
most_frequent = list()
for(i in (1:50)){
  begin=(i-1)*50+1
  end = i*50
  temp = table(predict[begin:end])
  most_frequent[[i]] = names(which.max(temp))
  if(most_frequent[[i]]!=unique_author[i]){
    print(paste0('author is ',unique_author[i],', model predicts ',most_frequent[[i]]))
  }
}
```

```
## [1] "author is AlanCrosby, model predicts JoeOrtiz"
## [1] "author is AlexanderSmith, model predicts JoWinterbottom"
## [1] "author is DarrenSchuettler, model predicts HeatherScoffield"
## [1] "author is DavidLawder, model predicts ToddNissen"
## [1] "author is HeatherScoffield, model predicts DarrenSchuettler"
## [1] "author is JanLopatka, model predicts JoeOrtiz"
## [1] "author is JaneMacartney, model predicts ScottHillis"
## [1] "author is JoWinterbottom, model predicts JonathanBirt"
## [1] "author is JoeOrtiz, model predicts JoWinterbottom"
## [1] "author is JohnMastrini, model predicts JoeOrtiz"
## [1] "author is JonathanBirt, model predicts JohnMastrini"
## [1] "author is PeterHumphrey, model predicts TanEeLyn"
## [1] "author is PierreTran, model predicts MarcelMichelson"
## [1] "author is SarahDavison, model predicts TanEeLyn"
## [1] "author is ScottHillis, model predicts MureDickie"
```

```
most_frequent = do.call(rbind, most_frequent)
row.names(most_frequent) = unique_author
most_frequent
```

```
##                   [,1]
## AaronPressman     "AaronPressman"
## AlanCrosby        "JoeOrtiz"
## AlexanderSmith    "JoWinterbottom"
## BenjaminKangLim   "BenjaminKangLim"
## BernardHickey     "BernardHickey"
## BradDorfman       "BradDorfman"
## DarrenSchuettler  "HeatherScoffield"
## DavidLawder       "ToddNissen"
## EdnaFernandes     "EdnaFernandes"
## EricAuchard       "EricAuchard"
## FumikoFujisaki    "FumikoFujisaki"
## GrahamEarnshaw    "GrahamEarnshaw"
## HeatherScoffield  "DarrenSchuettler"
## JanLopatka        "JoeOrtiz"
## JaneMacartney     "ScottHillis"
## JimGilchrist      "JimGilchrist"
## JoWinterbottom    "JonathanBirt"
## JoeOrtiz          "JoWinterbottom"
## JohnMastrini      "JoeOrtiz"
## JonathanBirt      "JohnMastrini"
## KarlPenhaul       "KarlPenhaul"
```

```
## KeithWeir          "KeithWeir"
## KevinDrawbaugh     "KevinDrawbaugh"
## KevinMorrison      "KevinMorrison"
## KirstinRidley      "KirstinRidley"
## KouroshKarimkhany "KouroshKarimkhany"
## LydiaZajc          "LydiaZajc"
## LynneO'Donnell     "LynneO'Donnell"
## LynnleyBrowning    "LynnleyBrowning"
## MarcelMichelson    "MarcelMichelson"
## MarkBendeich       "MarkBendeich"
## MartinWolk         "MartinWolk"
## MatthewBunce       "MatthewBunce"
## MichaelConnor      "MichaelConnor"
## MureDickie         "MureDickie"
## NickLouth          "NickLouth"
## PatriciaCommins    "PatriciaCommins"
## PeterHumphrey      "TanEeLyn"
## PierreTran         "MarcelMichelson"
## RobinSidel         "RobinSidel"
## RogerFillion       "RogerFillion"
## SamuelPerry        "SamuelPerry"
## SarahDavison       "TanEeLyn"
## ScottHillis        "MureDickie"
## SimonCowell        "SimonCowell"
## TanEeLyn           "TanEeLyn"
## TheresePoletti     "TheresePoletti"
## TimFarrand         "TimFarrand"
## ToddNissen         "ToddNissen"
## WilliamKazer       "WilliamKazer"
```

The accuracy of this model is:

```
accuracy_tfidf = count/nrow(X_test)
accuracy_tfidf
```

```
## [1] 0.5088
```

Model 2: Naive Bayes

Construct the multinomial probability matrix of the train set

```
prob <- list()
for(i in (1:50)){
  begin=(i-1)*50+1
  end = i*50
  w = colSums(X[begin:end,])
  prob[[i]] = w
}
prob = do.call(rbind, prob)
row.names(prob) <- unique_author
```

Reshape the multinomial probability matrix of the train set and add a smooth count for unseen words

```
smooth_count = 1/nrow(X)
prob = prob[,intersect(colnames(DTM), colnames(DTM_test))]
for(i in setdiff(colnames(DTM_test), colnames(DTM))){
  prob = cbind(prob,i=smooth_count)
}
prob = prob/rowSums(prob)
prob.T = t(prob)
P = as.matrix(prob.T)
```

The predictions and accuracy of the Naive Bayes model are as follow:

```
count = 0
predict <- list()
X_test = DTM_test[,c(intersect(colnames(DTM), colnames(DTM_test)),setdiff(colnames(DTM_test),colnames(D
X_test = as.matrix(X_test)
for(i in (1:2500)){
  temp = X_test[i,] %*% P
  temp = data.frame(temp)
  predict[[i]] = names(which.max(temp))
  if(predict[[i]]==labels[i]){
    count = count+1
  }
}
predict = do.call(rbind, predict)
row.names(predict) = labels_test
head(predict)
```

```
##                [,1]
## AaronPressman "AaronPressman"
## AaronPressman "DarrenSchuettler"
## AaronPressman "TheresePoletti"
## AaronPressman "AaronPressman"
## AaronPressman "AaronPressman"
## AaronPressman "FumikoFujisaki"
```

```
accuracy_naive_bayes = count/nrow(X_test)
accuracy_naive_bayes
```

```
## [1] 0.4132
```

Authors whose articles seem difficult to distinguish from one another are as follow:

```
most_frequent = list()
for(i in (1:50)){
  begin=(i-1)*50+1
  end = i*50
  temp = table(predict[begin:end])
  most_frequent[[i]] = names(which.max(temp))
  if(most_frequent[[i]]!=unique_author[i]){
    print(paste0('author is ',unique_author[i],', model predicts ',most_frequent[[i]]))
  }
}
```

```
## [1] "author is AlexanderSmith, model predicts JoeOrtiz"
## [1] "author is BernardHickey, model predicts TimFarrand"
## [1] "author is BradDorfman, model predicts TimFarrand"
## [1] "author is DarrenSchuettler, model predicts HeatherScoffield"
## [1] "author is DavidLawder, model predicts ToddNissen"
## [1] "author is EdnaFernandes, model predicts TimFarrand"
## [1] "author is EricAuchard, model predicts KouroshKarimkhany"
## [1] "author is JanLopatka, model predicts JohnMastrini"
## [1] "author is JaneMacartney, model predicts BenjaminKangLim"
## [1] "author is JoeOrtiz, model predicts TimFarrand"
## [1] "author is JonathanBirt, model predicts TimFarrand"
## [1] "author is KeithWeir, model predicts TimFarrand"
## [1] "author is KevinDrawbaugh, model predicts TimFarrand"
## [1] "author is KevinMorrison, model predicts TimFarrand"
## [1] "author is LynneO'Donnell, model predicts LynneO.Donnell"
## [1] "author is MarkBendeich, model predicts TimFarrand"
## [1] "author is PatriciaCommins, model predicts TimFarrand"
## [1] "author is PierreTran, model predicts MarcelMichelson"
## [1] "author is SamuelPerry, model predicts KouroshKarimkhany"
## [1] "author is SarahDavison, model predicts PeterHumphrey"
## [1] "author is ScottHillis, model predicts MureDickie"
## [1] "author is SimonCowell, model predicts TimFarrand"
## [1] "author is TanEeLyn, model predicts PeterHumphrey"
## [1] "author is ToddNissen, model predicts DavidLawder"
## [1] "author is WilliamKazer, model predicts BenjaminKangLim"
```

With the help of PCA, TFIDF model gets an accuracy rate slightly higher than Naive Bayes. More importantly, PCA significantly reduced the running time required for the former model. The TFIDF model is preferred considering the accuracy and efficiency aspects of the two models.

Q3: Practice with association rule mining

Revisit the notes on association rule mining, and walk through the R example on music playlists: playlists.R and playlists.csv. Then use the data on grocery purchases in groceries.txt and find some interesting association rules for these shopping baskets. The data file is a list of baskets: one row per basket, with multiple items per row separated by commas – you'll have to cobble together a few utilities for processing this into the format expected by the "arules" package. Pick your own thresholds for lift and confidence; just be clear what these thresholds are and how you picked them. Do your discovered item sets make sense? Present your discoveries in an interesting and concise way.

```
num_col <- max(count.fields("https://raw.githubusercontent.com/jgscott/STA380/master/data/groceries.txt
raw <- read.table("https://raw.githubusercontent.com/jgscott/STA380/master/data/groceries.txt",sep=",",
raw <- cbind(id = rownames(raw), raw)
dim(raw)
```

```
## [1] 9835   33
```

```
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

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

## The following object is masked from 'package:lubridate':
##
##     stamp

## The following objects are masked from 'package:tidyr':
##
##     expand, smiths

## The following objects are masked from 'package:plyr':
##
##     rename, round_any
```

```
raw = cbind(raw[1], stack(lapply(raw[2:33], as.character)))
raw = raw[order(raw$id),]
raw = na.omit(raw)
raw = raw[,1:2]

library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:reshape':
##
##     expand

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

##
## Attaching package: 'arules'

## The following object is masked from 'package:tm':
##
##     inspect

## The following object is masked from 'package:dplyr':
##
##     recode

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

```
str(raw)
```

```
## 'data.frame':    43367 obs. of  2 variables:
##  $ id    : Factor w/ 9835 levels "1","10","100",..: 1 1 1 1 2 2 3 3 4 4 ...
##  $ values: chr  "citrus fruit" "semi-finished bread" "margarine" "ready soups" ...
```

```
raw$id <- factor(raw$id)
post <- split(x=raw$values, f=raw$id)
post <- lapply(post, unique)
trans <- as(post, "transactions")
rules <- apriori(trans,
                 parameter=list(support=.001, confidence=.2, maxlen=4))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.2    0.1    1 none FALSE            TRUE   0.001      1      4
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [19782 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
arules::inspect(subset(rules,subset=lift > 10))
```

```
##    lhs                          rhs                        support confidence     lift
## 1  {softener}                => {detergent}              0.001118454  0.2037037 10.60014
## 2  {Instant food products} => {hamburger meat}         0.003050330  0.3797468 11.42144
## 3  {liquor,
##     red/blush wine}         => {bottled beer}           0.001931876  0.9047619 11.23527
## 4  {bottled beer,
##     liquor}                 => {red/blush wine}         0.001931876  0.4130435 21.49356
## 5  {bottled beer,
##     red/blush wine}         => {liquor}                 0.001931876  0.3958333 35.71579
## 6  {popcorn,
##     soda}                   => {salty snack}            0.001220132  0.6315789 16.69779
## 7  {Instant food products,
##     soda}                   => {hamburger meat}         0.001220132  0.6315789 18.99565
## 8  {hamburger meat,
##     soda}                   => {Instant food products} 0.001220132  0.2105263 26.20919
```

```
## 9  {Instant food products,
##     rolls/buns}           => {hamburger meat}       0.001016777  0.4347826 13.07672
## 10 {Instant food products,
##     whole milk}           => {hamburger meat}       0.001525165  0.5000000 15.03823
## 11 {ham,
##     processed cheese}     => {white bread}          0.001931876  0.6333333 15.04549
## 12 {processed cheese,
##     white bread}          => {ham}                  0.001931876  0.4634146 17.80345
## 13 {ham,
##     white bread}          => {processed cheese}     0.001931876  0.3800000 22.92822
## 14 {fruit/vegetable juice,
##     processed cheese}     => {ham}                  0.001118454  0.3793103 14.57233
## 15 {fruit/vegetable juice,
##     ham}                  => {processed cheese}     0.001118454  0.2894737 17.46610
## 16 {ham,
##     soda}                 => {processed cheese}     0.001016777  0.2040816 12.31376
## 17 {domestic eggs,
##     processed cheese}     => {white bread}          0.001118454  0.5238095 12.44364
## 18 {pip fruit,
##     processed cheese}     => {white bread}          0.001016777  0.4347826 10.32871
## 19 {rolls/buns,
##     white bread}          => {processed cheese}     0.001321810  0.2031250 12.25604
## 20 {baking powder,
##     flour}                => {sugar}                0.001016777  0.5555556 16.40807
## 21 {baking powder,
##     sugar}                => {flour}                0.001016777  0.3125000 17.97332
## 22 {flour,
##     sugar}                => {baking powder}        0.001016777  0.2040816 11.53530
## 23 {baking powder,
##     margarine}            => {sugar}                0.001118454  0.3666667 10.82933
## 24 {margarine,
##     sugar}                => {baking powder}        0.001118454  0.2037037 11.51394
## 25 {domestic eggs,
##     sugar}                => {baking powder}        0.001016777  0.2040816 11.53530
## 26 {sugar,
##     whipped/sour cream}   => {baking powder}        0.001321810  0.2708333 15.30831
## 27 {curd,
##     flour}                => {sugar}                0.001118454  0.3548387 10.48000
## 28 {curd,
##     sugar}                => {flour}                0.001118454  0.3235294 18.60767
## 29 {flour,
##     margarine}            => {sugar}                0.001626843  0.4324324 12.77169
## 30 {margarine,
##     sugar}                => {flour}                0.001626843  0.2962963 17.04137
## 31 {sugar,
##     whipped/sour cream}   => {flour}                0.001016777  0.2083333 11.98221
## 32 {citrus fruit,
##     sugar}                => {flour}                0.001016777  0.2127660 12.23715
## 33 {root vegetables,
##     sugar}                => {flour}                0.001423488  0.2222222 12.78103
## 34 {flour,
##     soda}                 => {sugar}                0.001118454  0.3928571 11.60285
## 35 {dessert,
##     pip fruit}            => {butter milk}          0.001423488  0.2857143 10.21818
```

```
## 36 {sliced cheese,
##     whipped/sour cream}    => {ham}                  0.001016777  0.2631579 10.10999
## 37 {ham,
##     pip fruit}             => {sliced cheese}        0.001016777  0.2564103 10.46388
## 38 {fruit/vegetable juice,
##     ham}                   => {white bread}          0.001626843  0.4210526 10.00254
## 39 {soda,
##     white bread,
##     whole milk}            => {processed cheese}     0.001016777  0.2500000 15.08436
## 40 {flour,
##     root vegetables,
##     whole milk}            => {sugar}                0.001016777  0.3448276 10.18432
## 41 {root vegetables,
##     sugar,
##     whole milk}            => {flour}                0.001016777  0.2941176 16.91606
## 42 {citrus fruit,
##     fruit/vegetable juice,
##     tropical fruit}        => {grapes}               0.001118454  0.2820513 12.60897
## 43 {hard cheese,
##     whipped/sour cream,
##     yogurt}                => {butter}               0.001016777  0.5882353 10.61522
## 44 {butter,
##     whipped/sour cream,
##     yogurt}                => {hard cheese}          0.001016777  0.2631579 10.73924
## 45 {chocolate,
##     rolls/buns,
##     soda}                  => {candy}                0.001220132  0.3000000 10.03571
## 46 {pip fruit,
##     sausage,
##     yogurt}                => {sliced cheese}        0.001220132  0.3076923 12.55665
## 47 {coffee,
##     other vegetables,
##     yogurt}                => {oil}                  0.001016777  0.2857143 10.18116
## 48 {citrus fruit,
##     fruit/vegetable juice,
##     root vegetables}       => {oil}                  0.001016777  0.2941176 10.48061
## 49 {hamburger meat,
##     whipped/sour cream,
##     yogurt}                => {butter}               0.001016777  0.6250000 11.27867
```

```
arules::inspect(subset(rules,subset=confidence > 0.8))
```

```
##     lhs                          rhs                    support confidence      lift
## 1   {liquor,
##      red/blush wine}          => {bottled beer}     0.001931876  0.9047619 11.235269
## 2   {cereals,
##      curd}                    => {whole milk}       0.001016777  0.9090909  3.557863
## 3   {cereals,
##      yogurt}                  => {whole milk}       0.001728521  0.8095238  3.168192
## 4   {butter,
##      jam}                     => {whole milk}       0.001016777  0.8333333  3.261374
## 5   {bottled beer,
##      soups}                   => {whole milk}       0.001118454  0.9166667  3.587512
## 6   {house keeping products,
```

```
##      napkins}                 => {whole milk}       0.001321810  0.8125000  3.179840
## 7   {house keeping products,
##      whipped/sour cream}      => {whole milk}       0.001220132  0.9230769  3.612599
## 8   {pastry,
##      sweet spreads}           => {whole milk}       0.001016777  0.9090909  3.557863
## 9   {rice,
##      sugar}                   => {whole milk}       0.001220132  1.0000000  3.913649
## 10  {butter,
##      rice}                    => {whole milk}       0.001525165  0.8333333  3.261374
## 11  {domestic eggs,
##      rice}                    => {whole milk}       0.001118454  0.8461538  3.311549
## 12  {bottled water,
##      rice}                    => {whole milk}       0.001220132  0.9230769  3.612599
## 13  {rice,
##      yogurt}                  => {other vegetables} 0.001931876  0.8260870  4.269346
## 14  {mustard,
##      oil}                     => {whole milk}       0.001220132  0.8571429  3.354556
## 15  {canned fish,
##      hygiene articles}        => {whole milk}       0.001118454  1.0000000  3.913649
## 16  {herbs,
##      shopping bags}           => {other vegetables} 0.001931876  0.8260870  4.269346
## 17  {herbs,
##      tropical fruit}          => {whole milk}       0.002338587  0.8214286  3.214783
## 18  {chocolate,
##      pickled vegetables}      => {whole milk}       0.001220132  0.8571429  3.354556
## 19  {grapes,
##      onions}                  => {other vegetables} 0.001118454  0.9166667  4.737476
## 20  {margarine,
##      meat}                    => {other vegetables} 0.001728521  0.8500000  4.392932
## 21  {hard cheese,
##      oil}                     => {other vegetables} 0.001118454  0.9166667  4.737476
## 22  {butter milk,
##      onions}                  => {other vegetables} 0.001321810  0.8125000  4.199126
## 23  {butter milk,
##      pork}                    => {other vegetables} 0.001830198  0.8571429  4.429848
## 24  {curd,
##      hamburger meat}          => {whole milk}       0.002541942  0.8064516  3.156169
## 25  {bottled beer,
##      hamburger meat}          => {whole milk}       0.001728521  0.8095238  3.168192
## 26  {other vegetables,
##      specialty cheese,
##      yogurt}                  => {whole milk}       0.001321810  0.8125000  3.179840
## 27  {butter,
##      rice,
##      root vegetables}         => {whole milk}       0.001016777  1.0000000  3.913649
## 28  {other vegetables,
##      rice,
##      tropical fruit}          => {whole milk}       0.001016777  0.8333333  3.261374
## 29  {rice,
##      root vegetables,
##      yogurt}                  => {other vegetables} 0.001423488  0.8750000  4.522136
## 30  {rice,
##      root vegetables,
##      yogurt}                  => {whole milk}       0.001423488  0.8750000  3.424443
```

```
## 31  {other vegetables,
##      rice,
##      root vegetables}          => {whole milk}       0.001830198  0.8181818  3.202076
## 32  {rice,
##      whole milk,
##      yogurt}                   => {other vegetables} 0.001525165  0.8333333  4.306796
## 33  {frozen fish,
##      pip fruit,
##      tropical fruit}           => {other vegetables} 0.001016777  0.8333333  4.306796
## 34  {frozen fish,
##      pip fruit,
##      whole milk}               => {other vegetables} 0.001118454  0.8461538  4.373055
## 35  {frozen fish,
##      root vegetables,
##      yogurt}                   => {whole milk}       0.001220132  0.8571429  3.354556
## 36  {frozen fish,
##      other vegetables,
##      yogurt}                   => {whole milk}       0.001220132  0.8571429  3.354556
## 37  {curd,
##      herbs,
##      root vegetables}          => {whole milk}       0.001220132  0.8571429  3.354556
## 38  {domestic eggs,
##      herbs,
##      other vegetables}         => {whole milk}       0.001016777  0.8333333  3.261374
## 39  {fruit/vegetable juice,
##      herbs,
##      other vegetables}         => {whole milk}       0.001016777  0.8333333  3.261374
## 40  {fruit/vegetable juice,
##      herbs,
##      whole milk}               => {other vegetables} 0.001016777  0.9090909  4.698323
## 41  {citrus fruit,
##      herbs,
##      tropical fruit}           => {other vegetables} 0.001016777  0.8333333  4.306796
## 42  {citrus fruit,
##      herbs,
##      tropical fruit}           => {whole milk}       0.001118454  0.9166667  3.587512
## 43  {citrus fruit,
##      herbs,
##      root vegetables}          => {other vegetables} 0.001321810  0.8125000  4.199126
## 44  {citrus fruit,
##      herbs,
##      root vegetables}          => {whole milk}       0.001321810  0.8125000  3.179840
## 45  {herbs,
##      root vegetables,
##      shopping bags}            => {other vegetables} 0.001118454  0.8461538  4.373055
## 46  {herbs,
##      root vegetables,
##      tropical fruit}           => {whole milk}       0.001525165  0.8823529  3.453220
## 47  {herbs,
##      tropical fruit,
##      yogurt}                   => {whole milk}       0.001016777  0.8333333  3.261374
## 48  {herbs,
##      other vegetables,
##      tropical fruit}           => {whole milk}       0.001321810  0.8125000  3.179840
```

```
## 49  {herbs,
##      rolls/buns,
##      root vegetables}          => {whole milk}       0.001525165  0.8333333  3.261374
## 50  {semi-finished bread,
##      tropical fruit,
##      yogurt}                   => {other vegetables} 0.001321810  0.8125000  4.199126
## 51  {detergent,
##      other vegetables,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.8333333  3.261374
## 52  {baking powder,
##      tropical fruit,
##      yogurt}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 53  {flour,
##      other vegetables,
##      sugar}                    => {whole milk}       0.001220132  0.8571429  3.354556
## 54  {flour,
##      root vegetables,
##      whipped/sour cream}       => {whole milk}       0.001728521  1.0000000  3.913649
## 55  {flour,
##      rolls/buns,
##      root vegetables}          => {other vegetables} 0.001016777  0.8333333  4.306796
## 56  {butter,
##      domestic eggs,
##      soft cheese}              => {whole milk}       0.001016777  1.0000000  3.913649
## 57  {domestic eggs,
##      root vegetables,
##      soft cheese}              => {other vegetables} 0.001016777  0.8333333  4.306796
## 58  {domestic eggs,
##      root vegetables,
##      soft cheese}              => {whole milk}       0.001016777  0.8333333  3.261374
## 59  {soft cheese,
##      tropical fruit,
##      whipped/sour cream}       => {other vegetables} 0.001220132  0.9230769  4.770605
## 60  {root vegetables,
##      soft cheese,
##      whipped/sour cream}       => {whole milk}       0.001220132  0.9230769  3.612599
## 61  {citrus fruit,
##      root vegetables,
##      soft cheese}              => {other vegetables} 0.001016777  1.0000000  5.168156
## 62  {grapes,
##      pork,
##      whole milk}               => {other vegetables} 0.001016777  0.8333333  4.306796
## 63  {citrus fruit,
##      fruit/vegetable juice,
##      grapes}                   => {tropical fruit}   0.001118454  0.8461538  8.063879
## 64  {grapes,
##      tropical fruit,
##      yogurt}                   => {other vegetables} 0.001423488  0.8235294  4.256128
## 65  {meat,
##      tropical fruit,
##      whole milk}               => {other vegetables} 0.001016777  0.8333333  4.306796
## 66  {meat,
##      root vegetables,
##      yogurt}                   => {other vegetables} 0.001220132  0.8571429  4.429848
```

```
## 67  {curd,
##      frozen meals,
##      yogurt}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 68  {frankfurter,
##      frozen meals,
##      tropical fruit}           => {other vegetables} 0.001016777  0.9090909  4.698323
## 69  {frankfurter,
##      frozen meals,
##      tropical fruit}           => {whole milk}       0.001016777  0.9090909  3.557863
## 70  {butter,
##      frozen meals,
##      tropical fruit}           => {whole milk}       0.001016777  0.9090909  3.557863
## 71  {frozen meals,
##      root vegetables,
##      tropical fruit}           => {whole milk}       0.001118454  0.8461538  3.311549
## 72  {butter,
##      hard cheese,
##      yogurt}                   => {whole milk}       0.001321810  0.8125000  3.179840
## 73  {hard cheese,
##      tropical fruit,
##      whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909  4.698323
## 74  {hard cheese,
##      root vegetables,
##      whipped/sour cream}       => {other vegetables} 0.001321810  0.8125000  4.199126
## 75  {hard cheese,
##      tropical fruit,
##      yogurt}                   => {whole milk}       0.001423488  0.8235294  3.223005
## 76  {butter milk,
##      dessert,
##      yogurt}                   => {whole milk}       0.001321810  0.8125000  3.179840
## 77  {butter milk,
##      pork,
##      whole milk}               => {other vegetables} 0.001016777  0.9090909  4.698323
## 78  {butter milk,
##      fruit/vegetable juice,
##      pip fruit}                => {other vegetables} 0.001016777  0.9090909  4.698323
## 79  {butter milk,
##      pip fruit,
##      root vegetables}          => {other vegetables} 0.001220132  0.8571429  4.429848
## 80  {butter milk,
##      sausage,
##      yogurt}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 81  {butter milk,
##      root vegetables,
##      yogurt}                   => {whole milk}       0.001525165  0.8823529  3.453220
## 82  {candy,
##      rolls/buns,
##      root vegetables}          => {other vegetables} 0.001016777  0.8333333  4.306796
## 83  {frozen vegetables,
##      ham,
##      whole milk}               => {other vegetables} 0.001016777  0.8333333  4.306796
## 84  {ham,
##      pip fruit,
##      tropical fruit}           => {other vegetables} 0.001626843  0.8888889  4.593916
```

```
## 85  {frankfurter,
##      root vegetables,
##      sliced cheese}            => {whole milk}       0.001016777  0.9090909  3.557863
## 86  {frankfurter,
##      sliced cheese,
##      yogurt}                   => {whole milk}       0.001016777  0.8333333  3.261374
## 87  {butter,
##      sliced cheese,
##      whipped/sour cream}       => {whole milk}       0.001220132  0.9230769  3.612599
## 88  {pip fruit,
##      sausage,
##      sliced cheese}            => {yogurt}           0.001220132  0.8571429  6.144315
## 89  {coffee,
##      oil,
##      yogurt}                   => {other vegetables} 0.001016777  0.9090909  4.698323
## 90  {citrus fruit,
##      fruit/vegetable juice,
##      oil}                      => {other vegetables} 0.001118454  0.8461538  4.373055
## 91  {fruit/vegetable juice,
##      oil,
##      tropical fruit}           => {other vegetables} 0.001220132  0.8571429  4.429848
## 92  {oil,
##      root vegetables,
##      shopping bags}            => {whole milk}       0.001016777  0.8333333  3.261374
## 93  {oil,
##      root vegetables,
##      tropical fruit}           => {other vegetables} 0.001728521  0.8500000  4.392932
## 94  {frozen vegetables,
##      onions,
##      root vegetables}          => {other vegetables} 0.001321810  0.8666667  4.479068
## 95  {curd,
##      onions,
##      yogurt}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 96  {napkins,
##      onions,
##      root vegetables}          => {other vegetables} 0.001016777  0.9090909  4.698323
## 97  {butter,
##      domestic eggs,
##      onions}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 98  {bottled water,
##      butter,
##      onions}                   => {whole milk}       0.001016777  0.8333333  3.261374
## 99  {butter,
##      onions,
##      tropical fruit}           => {whole milk}       0.001220132  0.8571429  3.354556
## 100 {butter,
##      onions,
##      root vegetables}          => {whole milk}       0.001728521  0.8500000  3.326602
## 101 {butter,
##      onions,
##      yogurt}                   => {whole milk}       0.001321810  0.8125000  3.179840
## 102 {citrus fruit,
##      onions,
##      root vegetables}          => {other vegetables} 0.001423488  0.8235294  4.256128
```

```
## 103 {onions,
##      root vegetables,
##      tropical fruit}           => {other vegetables} 0.001626843  0.8888889  4.593916
## 104 {berries,
##      butter,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.8333333  3.261374
## 105 {berries,
##      butter,
##      sausage}                  => {whole milk}       0.001016777  0.9090909  3.557863
## 106 {hamburger meat,
##      tropical fruit,
##      whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909  4.698323
## 107 {hamburger meat,
##      root vegetables,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.8333333  3.261374
## 108 {butter,
##      hygiene articles,
##      napkins}                  => {whole milk}       0.001016777  0.9090909  3.557863
## 109 {hygiene articles,
##      margarine,
##      rolls/buns}               => {whole milk}       0.001016777  0.8333333  3.261374
## 110 {butter,
##      hygiene articles,
##      pip fruit}                => {whole milk}       0.001016777  1.0000000  3.913649
## 111 {butter,
##      citrus fruit,
##      hygiene articles}         => {whole milk}       0.001016777  0.8333333  3.261374
## 112 {bottled water,
##      butter,
##      hygiene articles}         => {whole milk}       0.001220132  0.8571429  3.354556
## 113 {butter,
##      hygiene articles,
##      tropical fruit}           => {whole milk}       0.001220132  0.9230769  3.612599
## 114 {butter,
##      hygiene articles,
##      root vegetables}          => {whole milk}       0.001423488  0.8235294  3.223005
## 115 {domestic eggs,
##      hygiene articles,
##      tropical fruit}           => {whole milk}       0.001220132  0.9230769  3.612599
## 116 {hygiene articles,
##      tropical fruit,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.8333333  3.261374
## 117 {hygiene articles,
##      root vegetables,
##      whipped/sour cream}       => {whole milk}       0.001016777  1.0000000  3.913649
## 118 {hygiene articles,
##      pip fruit,
##      sausage}                  => {whole milk}       0.001321810  0.8125000  3.179840
## 119 {hygiene articles,
##      pip fruit,
##      root vegetables}          => {whole milk}       0.001016777  1.0000000  3.913649
## 120 {citrus fruit,
##      hygiene articles,
##      root vegetables}          => {whole milk}       0.001220132  0.8571429  3.354556
```

```
## 121 {hygiene articles,
##      root vegetables,
##      yogurt}                   => {whole milk}       0.001220132  0.8571429  3.354556
## 122 {long life bakery product,
##      other vegetables,
##      salty snack}              => {whole milk}       0.001016777  0.8333333  3.261374
## 123 {pip fruit,
##      salty snack,
##      yogurt}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 124 {salty snack,
##      tropical fruit,
##      yogurt}                   => {other vegetables} 0.001321810  0.8125000  4.199126
## 125 {root vegetables,
##      salty snack,
##      yogurt}                   => {other vegetables} 0.001220132  0.8571429  4.429848
## 126 {cream cheese ,
##      domestic eggs,
##      sugar}                    => {whole milk}       0.001118454  1.0000000  3.913649
## 127 {cream cheese ,
##      other vegetables,
##      sugar}                    => {whole milk}       0.001525165  0.9375000  3.669046
## 128 {beef,
##      root vegetables,
##      sugar}                    => {other vegetables} 0.001118454  0.8461538  4.373055
## 129 {curd,
##      domestic eggs,
##      sugar}                    => {whole milk}       0.001016777  1.0000000  3.913649
## 130 {butter,
##      sugar,
##      whipped/sour cream}       => {other vegetables} 0.001016777  0.8333333  4.306796
## 131 {butter,
##      sugar,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.8333333  3.261374
## 132 {citrus fruit,
##      domestic eggs,
##      sugar}                    => {whole milk}       0.001423488  0.9333333  3.652739
## 133 {domestic eggs,
##      sugar,
##      tropical fruit}           => {whole milk}       0.001118454  0.9166667  3.587512
## 134 {domestic eggs,
##      sugar,
##      yogurt}                   => {whole milk}       0.001423488  0.9333333  3.652739
## 135 {citrus fruit,
##      sugar,
##      whipped/sour cream}       => {whole milk}       0.001118454  0.8461538  3.311549
## 136 {root vegetables,
##      sugar,
##      whipped/sour cream}       => {whole milk}       0.001220132  0.9230769  3.612599
## 137 {bottled water,
##      other vegetables,
##      sugar}                    => {whole milk}       0.001321810  0.8125000  3.179840
## 138 {pork,
##      rolls/buns,
##      waffles}                  => {whole milk}       0.001016777  0.9090909  3.557863
```

```
## 139 {rolls/buns,
##      waffles,
##      whipped/sour cream}       => {whole milk}       0.001728521  0.8095238  3.168192
## 140 {rolls/buns,
##      root vegetables,
##      waffles}                  => {whole milk}       0.001626843  0.8421053  3.295704
## 141 {long life bakery product,
##      napkins,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.9090909  3.557863
## 142 {long life bakery product,
##      napkins,
##      tropical fruit}           => {whole milk}       0.001220132  0.9230769  3.612599
## 143 {long life bakery product,
##      napkins,
##      other vegetables}         => {whole milk}       0.001220132  0.8571429  3.354556
## 144 {butter,
##      long life bakery product,
##      sausage}                  => {whole milk}       0.001016777  0.9090909  3.557863
## 145 {long life bakery product,
##      sausage,
##      whipped/sour cream}       => {whole milk}       0.001016777  0.8333333  3.261374
## 146 {long life bakery product,
##      whipped/sour cream,
##      yogurt}                   => {whole milk}       0.001728521  0.8095238  3.168192
## 147 {long life bakery product,
##      root vegetables,
##      tropical fruit}           => {other vegetables} 0.001118454  0.8461538  4.373055
## 148 {dessert,
##      sausage,
##      whipped/sour cream}       => {other vegetables} 0.001016777  0.8333333  4.306796
## 149 {dessert,
##      tropical fruit,
##      whipped/sour cream}       => {other vegetables} 0.001118454  0.9166667  4.737476
## 150 {cream cheese ,
##      curd,
##      root vegetables}          => {other vegetables} 0.001728521  0.8500000  4.392932
## 151 {cream cheese ,
##      domestic eggs,
##      napkins}                  => {whole milk}       0.001118454  1.0000000  3.913649
## 152 {cream cheese ,
##      pork,
##      yogurt}                   => {whole milk}       0.001016777  0.8333333  3.261374
## 153 {cream cheese ,
##      frankfurter,
##      yogurt}                   => {whole milk}       0.001016777  0.8333333  3.261374
## 154 {cream cheese ,
##      margarine,
##      whipped/sour cream}       => {yogurt}           0.001016777  0.8333333  5.973639
## 155 {butter,
##      cream cheese ,
##      whipped/sour cream}       => {whole milk}       0.001118454  0.8461538  3.311549
## 156 {butter,
##      cream cheese ,
##      root vegetables}          => {yogurt}           0.001016777  0.9090909  6.516698
```

```
## 157 {butter,
##      cream cheese ,
##      root vegetables}          => {whole milk}       0.001016777  0.9090909  3.557863
## 158 {cream cheese ,
##      domestic eggs,
##      whipped/sour cream}       => {whole milk}       0.001220132  0.8571429  3.354556
## 159 {cream cheese ,
##      domestic eggs,
##      pip fruit}                => {whole milk}       0.001118454  0.8461538  3.311549
## 160 {citrus fruit,
##      cream cheese ,
##      domestic eggs}            => {whole milk}       0.001626843  0.8888889  3.478799
## 161 {cream cheese ,
##      domestic eggs,
##      yogurt}                   => {whole milk}       0.001321810  0.8125000  3.179840
## 162 {cream cheese ,
##      pip fruit,
##      whipped/sour cream}       => {whole milk}       0.001321810  0.9285714  3.634103
## 163 {citrus fruit,
##      cream cheese ,
##      whipped/sour cream}       => {other vegetables} 0.001321810  0.8125000  4.199126
## 164 {cream cheese ,
##      tropical fruit,
##      whipped/sour cream}       => {other vegetables} 0.001423488  0.8750000  4.522136
## 165 {cream cheese ,
##      pip fruit,
##      sausage}                  => {whole milk}       0.001016777  0.9090909  3.557863
## 166 {citrus fruit,
##      cream cheese ,
##      root vegetables}          => {other vegetables} 0.001220132  0.9230769  4.770605
## 167 {chicken,
##      domestic eggs,
##      sausage}                  => {whole milk}       0.001220132  0.8571429  3.354556
## 168 {chicken,
##      pastry,
##      root vegetables}          => {other vegetables} 0.001016777  0.8333333  4.306796
## 169 {butter,
##      tropical fruit,
##      white bread}              => {yogurt}           0.001118454  0.8461538  6.065542
## 170 {butter,
##      tropical fruit,
##      white bread}              => {other vegetables} 0.001118454  0.8461538  4.373055
## 171 {butter,
##      root vegetables,
##      white bread}              => {other vegetables} 0.001016777  0.8333333  4.306796
## 172 {butter,
##      root vegetables,
##      white bread}              => {whole milk}       0.001118454  0.9166667  3.587512
## 173 {tropical fruit,
##      whipped/sour cream,
##      white bread}              => {other vegetables} 0.001728521  0.8500000  4.392932
## 174 {root vegetables,
##      whipped/sour cream,
##      white bread}              => {other vegetables} 0.001423488  0.8750000  4.522136
```

```
## 175 {root vegetables,
##      whipped/sour cream,
##      white bread}              => {whole milk}       0.001321810  0.8125000  3.179840
## 176 {chocolate,
##      domestic eggs,
##      sausage}                  => {whole milk}       0.001016777  0.8333333  3.261374
## 177 {bottled beer,
##      coffee,
##      yogurt}                   => {whole milk}       0.001016777  0.8333333  3.261374
## 178 {butter,
##      coffee,
##      whipped/sour cream}       => {whole milk}       0.001220132  0.9230769  3.612599
## 179 {coffee,
##      domestic eggs,
##      root vegetables}          => {whole milk}       0.001016777  0.9090909  3.557863
## 180 {citrus fruit,
##      frozen vegetables,
##      napkins}                  => {whole milk}       0.001118454  0.8461538  3.311549
## 181 {frozen vegetables,
##      napkins,
##      other vegetables}         => {whole milk}       0.001321810  0.8125000  3.179840
## 182 {frozen vegetables,
##      margarine,
##      rolls/buns}               => {whole milk}       0.001321810  0.8666667  3.391829
## 183 {citrus fruit,
##      frozen vegetables,
##      fruit/vegetable juice}    => {whole milk}       0.001626843  0.8421053  3.295704
## 184 {beef,
##      butter,
##      curd}                     => {whole milk}       0.001016777  0.8333333  3.261374
## 185 {beef,
##      butter,
##      tropical fruit}           => {yogurt}           0.001016777  0.8333333  5.973639
## 186 {beef,
##      tropical fruit,
##      whipped/sour cream}       => {other vegetables} 0.001423488  0.8750000  4.522136
## 187 {beef,
##      tropical fruit,
##      whipped/sour cream}       => {whole milk}       0.001321810  0.8125000  3.179840
## 188 {curd,
##      margarine,
##      rolls/buns}               => {whole milk}       0.001321810  0.8125000  3.179840
## 189 {butter,
##      curd,
##      domestic eggs}            => {other vegetables} 0.001016777  0.8333333  4.306796
## 190 {butter,
##      curd,
##      domestic eggs}            => {whole milk}       0.001118454  0.9166667  3.587512
## 191 {butter,
##      citrus fruit,
##      curd}                     => {whole milk}       0.001118454  0.9166667  3.587512
## 192 {curd,
##      domestic eggs,
##      other vegetables}         => {whole milk}       0.002846975  0.8235294  3.223005
```

```
## 193 {curd,
##      pip fruit,
##      whipped/sour cream}       => {whole milk}       0.001830198  0.8181818  3.202076
## 194 {butter,
##      napkins,
##      whipped/sour cream}       => {whole milk}       0.001423488  0.8235294  3.223005
## 195 {bottled water,
##      butter,
##      napkins}                  => {whole milk}       0.001118454  0.8461538  3.311549
## 196 {butter,
##      napkins,
##      yogurt}                   => {whole milk}       0.001118454  0.8461538  3.311549
## 197 {domestic eggs,
##      napkins,
##      tropical fruit}           => {whole milk}       0.001321810  0.8125000  3.179840
## 198 {bottled beer,
##      pork,
##      rolls/buns}               => {whole milk}       0.001118454  0.8461538  3.311549
## 199 {butter,
##      pork,
##      whipped/sour cream}       => {whole milk}       0.001423488  0.8750000  3.424443
## 200 {butter,
##      pork,
##      yogurt}                   => {whole milk}       0.001423488  0.8235294  3.223005
## 201 {butter,
##      other vegetables,
##      pork}                     => {whole milk}       0.002236909  0.8461538  3.311549
## 202 {fruit/vegetable juice,
##      pork,
##      tropical fruit}           => {yogurt}           0.001016777  0.8333333  5.973639
## 203 {pip fruit,
##      pork,
##      soda}                     => {other vegetables} 0.001118454  0.8461538  4.373055
## 204 {bottled beer,
##      domestic eggs,
##      margarine}                => {whole milk}       0.001016777  0.9090909  3.557863
## 205 {brown bread,
##      domestic eggs,
##      root vegetables}          => {whole milk}       0.001525165  0.8333333  3.261374
## 206 {brown bread,
##      pip fruit,
##      whipped/sour cream}       => {other vegetables} 0.001118454  1.0000000  5.168156
## 207 {brown bread,
##      sausage,
##      whipped/sour cream}       => {other vegetables} 0.001016777  0.8333333  4.306796
## 208 {brown bread,
##      pip fruit,
##      root vegetables}          => {other vegetables} 0.001321810  0.8125000  4.199126
## 209 {brown bread,
##      pip fruit,
##      root vegetables}          => {whole milk}       0.001423488  0.8750000  3.424443
## 210 {butter,
##      margarine,
##      tropical fruit}           => {yogurt}           0.001118454  0.8461538  6.065542
```

```
## 211 {domestic eggs,
##      fruit/vegetable juice,
##      margarine}               => {whole milk}       0.001118454  0.9166667  3.587512
## 212 {bottled water,
##      margarine,
##      whipped/sour cream}      => {whole milk}       0.001016777  0.8333333  3.261374
## 213 {margarine,
##      rolls/buns,
##      whipped/sour cream}      => {whole milk}       0.001626843  0.8888889  3.478799
## 214 {butter,
##      domestic eggs,
##      whipped/sour cream}      => {whole milk}       0.001626843  0.8421053  3.295704
## 215 {butter,
##      domestic eggs,
##      pip fruit}               => {whole milk}       0.001220132  0.8571429  3.354556
## 216 {butter,
##      pip fruit,
##      whipped/sour cream}      => {whole milk}       0.001830198  0.9000000  3.522284
## 217 {bottled water,
##      butter,
##      whipped/sour cream}      => {whole milk}       0.001220132  0.8571429  3.354556
## 218 {butter,
##      soda,
##      whipped/sour cream}      => {other vegetables} 0.001321810  0.9285714  4.799002
## 219 {butter,
##      pastry,
##      pip fruit}               => {other vegetables} 0.001321810  0.9285714  4.799002
## 220 {bottled water,
##      butter,
##      pip fruit}               => {whole milk}       0.001321810  0.8125000  3.179840
## 221 {butter,
##      pip fruit,
##      root vegetables}         => {whole milk}       0.001728521  0.8095238  3.168192
## 222 {citrus fruit,
##      newspapers,
##      root vegetables}         => {other vegetables} 0.001626843  0.8421053  4.352131
## 223 {domestic eggs,
##      pastry,
##      whipped/sour cream}      => {other vegetables} 0.001220132  0.8571429  4.429848
## 224 {domestic eggs,
##      tropical fruit,
##      whipped/sour cream}      => {whole milk}       0.001830198  0.9000000  3.522284
## 225 {domestic eggs,
##      pip fruit,
##      sausage}                 => {whole milk}       0.001423488  0.8235294  3.223005
## 226 {domestic eggs,
##      pastry,
##      tropical fruit}          => {whole milk}       0.001321810  0.8125000  3.179840
## 227 {domestic eggs,
##      pastry,
##      root vegetables}         => {other vegetables} 0.001220132  0.8571429  4.429848
## 228 {fruit/vegetable juice,
##      tropical fruit,
##      whipped/sour cream}      => {other vegetables} 0.001931876  0.9047619  4.675950
```

```
## 229 {citrus fruit,
##      pastry,
##      whipped/sour cream}       => {whole milk}       0.001525165  0.8823529  3.453220
## 230 {bottled water,
##      sausage,
##      whipped/sour cream}       => {other vegetables} 0.001321810  0.8125000  4.199126
## 231 {citrus fruit,
##      pastry,
##      root vegetables}          => {other vegetables} 0.001525165  0.8823529  4.560137
## 232 {pastry,
##      root vegetables,
##      shopping bags}            => {other vegetables} 0.001118454  0.8461538  4.373055
```

```
arules::inspect(subset(rules,subset=lift > 8 & confidence > 0.6))
```

```
##    lhs                          rhs                          support confidence      lift
## 1  {liquor,
##     red/blush wine}        => {bottled beer}          0.001931876  0.9047619 11.235269
## 2  {popcorn,
##     soda}                  => {salty snack}           0.001220132  0.6315789 16.697793
## 3  {Instant food products,
##     soda}                  => {hamburger meat}        0.001220132  0.6315789 18.995654
## 4  {ham,
##     processed cheese}      => {white bread}           0.001931876  0.6333333 15.045491
## 5  {frozen vegetables,
##     specialty chocolate}   => {fruit/vegetable juice} 0.001016777  0.6250000  8.645394
## 6  {frozen fish,
##     other vegetables,
##     tropical fruit}        => {pip fruit}             0.001016777  0.6666667  8.812724
## 7  {citrus fruit,
##     fruit/vegetable juice,
##     grapes}                => {tropical fruit}        0.001118454  0.8461538  8.063879
## 8  {fruit/vegetable juice,
##     grapes,
##     tropical fruit}        => {citrus fruit}          0.001118454  0.6875000  8.306588
## 9  {citrus fruit,
##     grapes,
##     tropical fruit}        => {fruit/vegetable juice} 0.001118454  0.6111111  8.453274
## 10 {butter,
##     hard cheese,
##     yogurt}                => {whipped/sour cream}    0.001016777  0.6250000  8.718972
## 11 {butter,
##     hard cheese,
##     whole milk}            => {whipped/sour cream}    0.001423488  0.6666667  9.300236
## 12 {ham,
##     other vegetables,
##     tropical fruit}        => {pip fruit}             0.001626843  0.6153846  8.134822
## 13 {hamburger meat,
##     whipped/sour cream,
##     yogurt}                => {butter}                0.001016777  0.6250000 11.278670
## 14 {curd,
##     sugar,
##     yogurt}                => {whipped/sour cream}    0.001016777  0.6250000  8.718972
## 15 {butter,
```

```
##     other vegetables,
##     sugar}                 => {whipped/sour cream}    0.001016777  0.7142857  9.964539
## 16 {domestic eggs,
##     frankfurter,
##     tropical fruit}        => {pip fruit}             0.001016777  0.6250000  8.261929
## 17 {shopping bags,
##     tropical fruit,
##     whipped/sour cream}    => {pip fruit}             0.001118454  0.6470588  8.553526
```

Here we choose to set the threshold at support level equal to 0.001 and confidence level equal to 0.2. The general rule for choosing threshhold is that the bigger the dataset is, the looser the threshold is. Here the dataset include approximately 10000 transaction records, we choose a relatively low support level because we don't want some of the interesting associations to be missed out just because the co-occurance of two items among the whole dataset is not high enough. And we set the confidence level at 0.2, because a low support and high confidence level help us extract strong relationship even for less overall co-occurrences in data. We also set maxlen = 4, because grocery shopping involves a large amount of randomness, we don't want the correlation to be over-analyzed and just want to focus on the most important/relavant associations.

After we get the rules, we take a closer look at it by first filtering out rules with a high lift (lift >10). First interesting discover is that when people buy sofener, they are 10 times more likely to purchase detergent as well. Second interesting thing is that we find there is a group of people that likes instant food products, and even when they buy meat they prefer to buy hamburger meat, the "quick meat". The third interesting thing is that when people buy alcoholic drinks, they tent to buy different alcoholic drinks together. Perhaps most of people buy alcoholic drinks for party/large event and want to provide different choice for their guests.The forth interesting thing is that we find some "sandwich maker" who like to purchase ham and processed cheese with white bread. We also find the "baker group", and discovered that they tend to but several baking materials at one time.

Then we tried to look at the association rules that have a high confidence level(confidence > 0.8). Here the result is more close to our common knowledge–when people buy fruit/vegetable/meat, they are more likely to buy whole milk and other vegetables. Most housewives go to grocery shoppings for milk, bread, vegetables and fruit which most family consume most quickly.