Assignment 2

Hung-Yen Chen, Lu-Hao Kuo, Jihyun Lee and Matthew Zlotnik

2018.08.18

Flights at ABIA

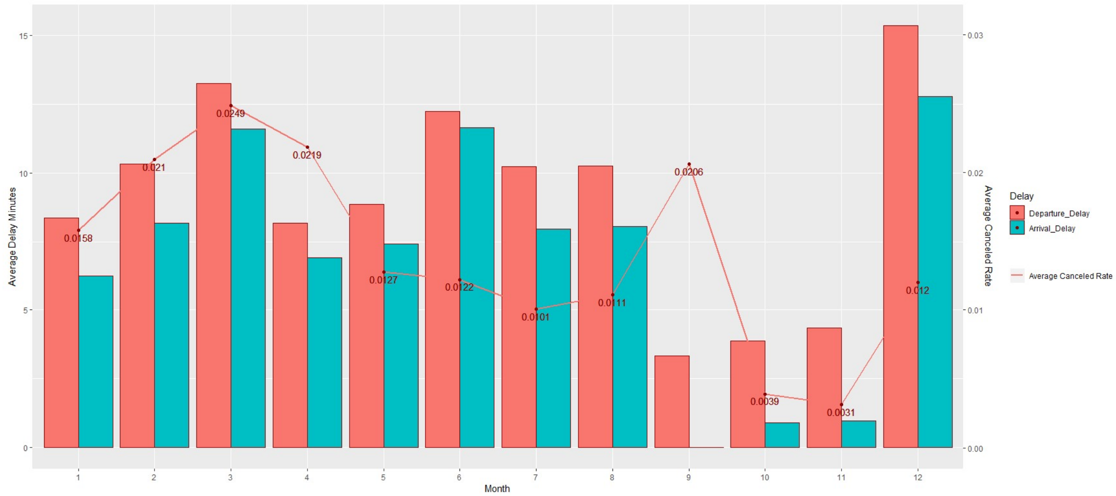
```
library(ggplot2)
library(reshape2)
air = read.csv('../data/ABIA.csv',stringsAsFactors = FALSE)
```

First, we wanted to know which month is the best time to take a flight in Austin Airport. We took departure and arrival delay time and cancellation in to consideration.

Month vs Delay and Cancellation

```
month_mean_depdelay = aggregate(DepDelay~factor(Month), air, mean)
month_mean_arrdelay = aggregate(ArrDelay~factor(Month), air, mean)
month_cancel = aggregate(Cancelled~factor(Month),air,mean)
month_delay_cancel <-
cbind(month_cancel,month_mean_depdelay[,2],month_mean_arrdelay[,2])
names(month_delay_cancel) <-
c("Month","Cancelled","Departure_Delay","Arrival_Delay")
month_delay_cancel <- melt(month_delay_cancel,id.vars = c(1,2))

update_geom_defaults("bar",   list(colour = "red4"))
ggplot(month_delay_cancel, aes(x=Month,y=value,fill=variable)) +
  ylab("Average Delay Minutes")+
  geom_bar(stat="identity",position = position_dodge())+
  geom_line(aes(y=Cancelled*500,group = 1,color="Average Canceled
Rate"),size=1,stat = "identity")+
  geom_point(aes(y = Cancelled*500,group = 1),color="red4")+
  geom_text(aes(y =Cancelled*500,group = 1, label = round(Cancelled, 4)),
vjust = 1.4, color = "red4", size = 4)+
  scale_y_continuous(sec.axis = sec_axis(~./500,name="Average Canceled
Rate"))+
  scale_fill_discrete(name = "Delay")+
  scale_color_discrete(name = " ")
```

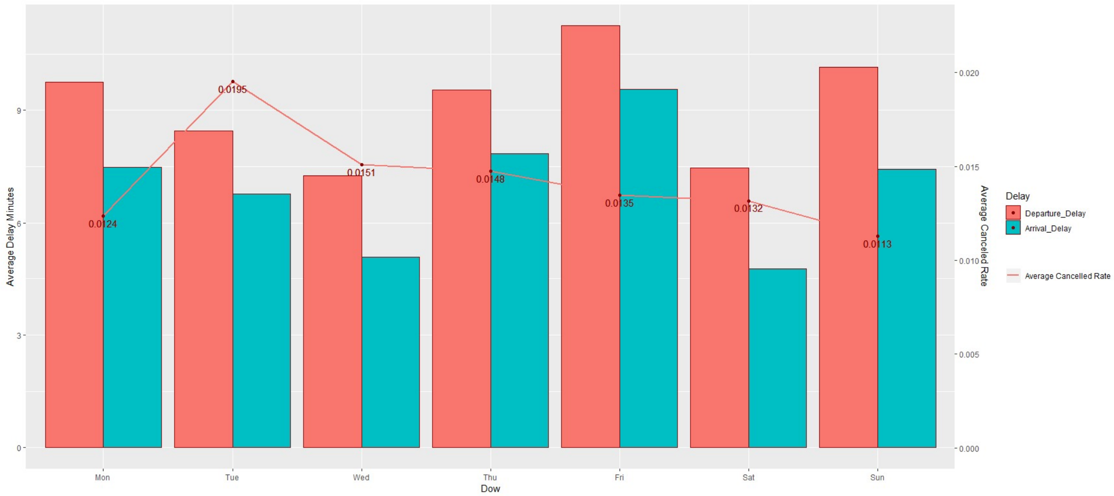


As we can see, Fall is the best season to take a flight, because September, October and November are top 3 months with least average delay time in minutes. These months have only 5 minutes departure delay and the passengers usually can arrived at the destination on time. However, September has a pretty high average cancellation rate. After searching Sep 2008 flights records, we found that the flights were cancelled because of Hurricane Ike. We can also see that December has the most delay minutes. For the next part, we wanted to find out which day of the week is the best day to catch a flight.

Weekday vs Delay and Cancellation

```
week_mean_depdelay = aggregate(DepDelay~factor(DayOfWeek), air, mean)
week_mean_arrdelay = aggregate(ArrDelay~factor(DayOfWeek), air, mean)
week_cancel = aggregate(Cancelled~factor(DayOfWeek),air,mean)
week_delay_cancel <-
cbind(week_cancel,week_mean_depdelay[,2],week_mean_arrdelay[,2])
names(week_delay_cancel) <-
c("Dow","Cancelled","Departure_Delay","Arrival_Delay")
week_delay_cancel[1]=c("Mon","Tue","Wed","Thu","Fri","Sat","Sun")
week_delay_cancel <- melt(week_delay_cancel,id.vars = c(1,2))

ggplot(week_delay_cancel, aes(x=Dow,y=value,fill=variable)) +
  ylab("Average Delay Minutes")+
  geom_bar(stat="identity",position = position_dodge())+
  geom_line(aes(y=Cancelled*500,group = 1,color="Average Cancelled
Rate"),size=1,stat = "identity")+
  geom_point(aes(y = Cancelled*500,group = 1),color="red4")+
  geom_text(aes(y =Cancelled*500,group = 1, label = round(Cancelled, 4)),
vjust = 1.4, color = "red4", size = 4)+
  scale_y_continuous(sec.axis = sec_axis(~./500,name="Average Canceled
Rate"))+
  scale_x_discrete(limits=c("Mon","Tue","Wed","Thu","Fri","Sat","Sun"))+
  scale_fill_discrete(name = "Delay")+
  scale_color_discrete(name = " ")
```

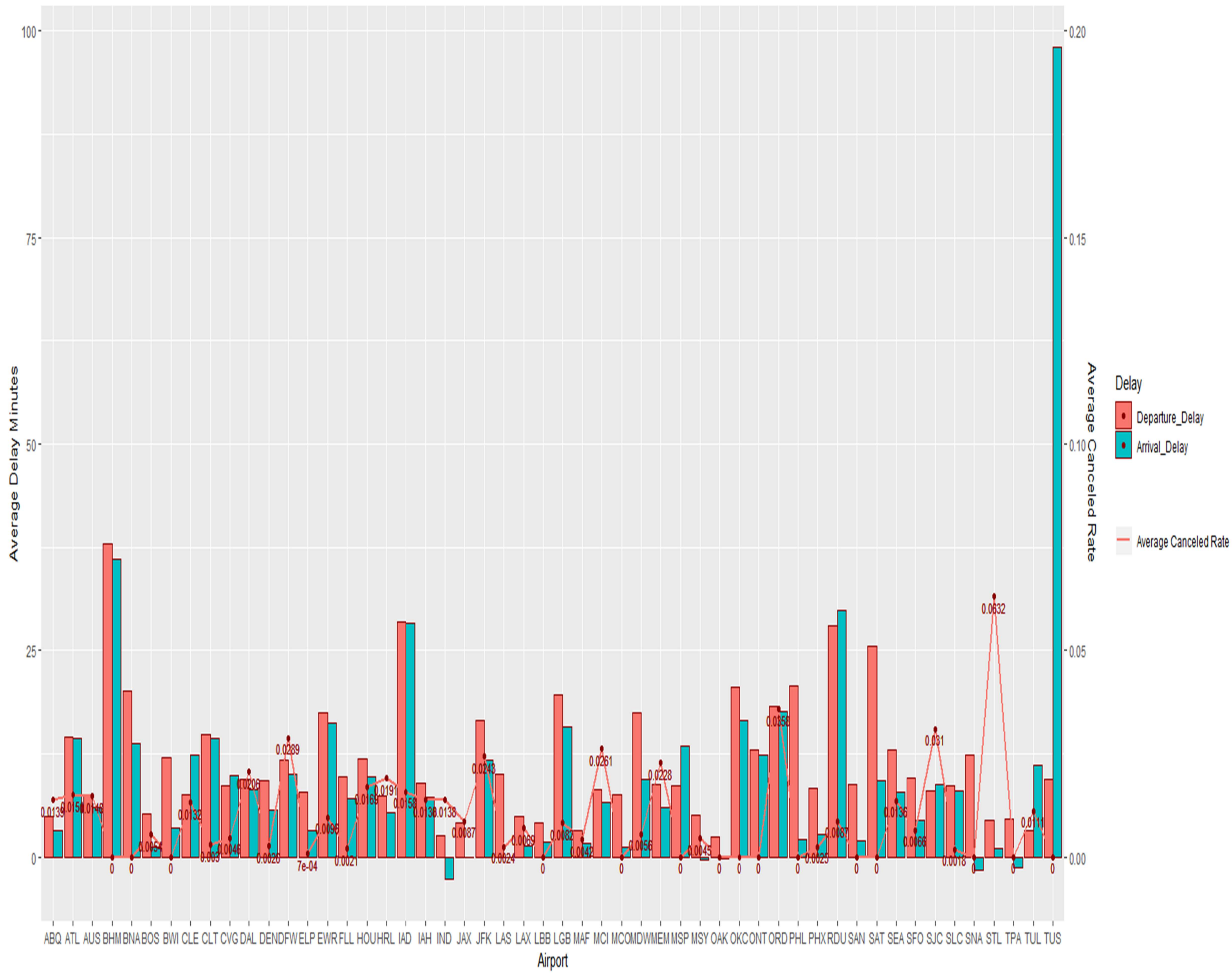


We can see that Wednesday and Saturday have the least delay time, but there is not a really significant difference between these days. One thing that is noticeable is that Tuesday has the highest cancellation rate. Next, we wanted to explore which airport has the highest delay time and cancellation rate.

Airport vs Delay and Cancellation

```
airport_mean_depdelay = aggregate(DepDelay~factor(Origin), air, mean)
airport_mean_arrdelay = aggregate(ArrDelay~factor(Origin), air, mean)
airport_mean_depdelay = airport_mean_depdelay[1:52,]
airport_cancel = aggregate(Cancelled~factor(Origin),air,mean)
airport_cancel = airport_cancel[1:52,]
airport_delay_cancel <-
cbind(airport_cancel,airport_mean_depdelay[,2],airport_mean_arrdelay[,2])
names(airport_delay_cancel) <-
c("Airport","Cancelled","Departure_Delay","Arrival_Delay")
airport_delay_cancel <- melt(airport_delay_cancel,id.vars = c(1,2))

ggplot(airport_delay_cancel, aes(x=Airport,y=value,fill=variable)) +
  ylab("Average Delay Minutes")+
  geom_bar(stat="identity",position = position_dodge())+
  geom_line(aes(y=Cancelled*500,group = 1,color="Average Canceled
Rate"),size=1,stat = "identity")+
  geom_point(aes(y = Cancelled*500,group = 1),color="red4")+
  geom_text(aes(y =Cancelled*500,group = 1, label = round(Cancelled, 4)),
vjust = 1.4, color = "red4", size = 3)+
  scale_y_continuous(sec.axis = sec_axis(~./500,name="Average Canceled
Rate"))+
  scale_fill_discrete(name = "Delay")+
  scale_color_discrete(name = " ")
```

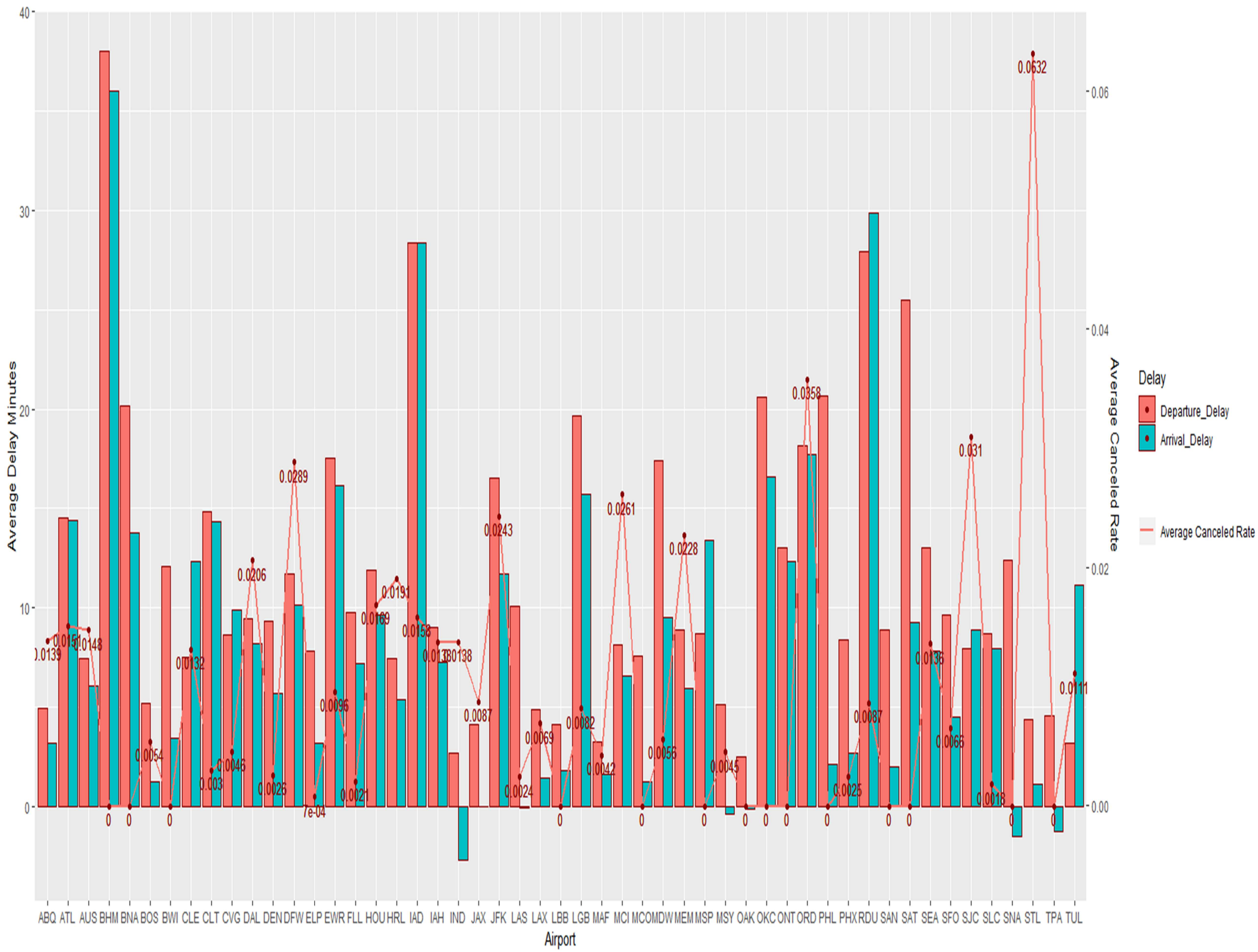

Average Delay Minutes
Airport
Average Canceled Rate
Delay
Departure_Delay
Arrival_Delay
Average Canceled Rate

In this bar chart, we plotted both delay time for origin airport of the flight. As a result, we can figure out if the passengers is in specific airport and have a flight to Austin, how much delay time they should expect. We can see that for TUS airport, the mean arrival delay is almost 100 minutes. That is because that the flights from TUS to Austin are all from XE, which is a private jet company. Because TUS airport is an outlier for our analysis, we tried to create the other plot excluding TUS airport.

Excluding TUS from Airport Graph

```
airport_mean_depdelay_notus = airport_mean_depdelay[1:51,]
airport_mean_arrdelay_notus = airport_mean_arrdelay[1:51,]
airport_cancel_notus = airport_cancel[1:51,]
airport_delay_cancel_notus =
cbind(airport_cancel_notus,airport_mean_depdelay_notus[,2],airport_mean_arrde
lay_notus[,2])
names(airport_delay_cancel_notus) <-
c("Airport","Cancelled","Departure_Delay","Arrival_Delay")
airport_delay_cancel_notus <- melt(airport_delay_cancel_notus,id.vars =
c(1,2))

ggplot(airport_delay_cancel_notus, aes(x=Airport,y=value,fill=variable)) +
  ylab("Average Delay Minutes")+
  geom_bar(stat="identity",position = position_dodge())+
  geom_line(aes(y=Cancelled*600,group = 1,color="Average Canceled
Rate"),size=1,stat = "identity")+
  geom_point(aes(y = Cancelled*600,group = 1),color="red4")+
  geom_text(aes(y =Cancelled*600,group = 1, label = round(Cancelled, 4)),
vjust = 1.4, color = "red4", size = 3.5)+
  scale_y_continuous(sec.axis = sec_axis(~./600,name="Average Canceled
Rate"))+
  scale_fill_discrete(name = "Delay")+
  scale_color_discrete(name = " ")
```


Average Delay Minutes
Average Canceled Rate
Airport
ABQ ATL AUS BHM BNA BOS BWI CLE CLT CVG DAL DEN DFW ELP EWR FLL HOU HRL IAD IAH IND JAX JFK LAS LAX LBB LGB MAF MCI MCO MDW MEM MSP MSY OAK OKC ONT ORD PHL PHX RDU SAN SAT SEA SFO SJC SLC SNA STL TPA TUL
Delay
Departure_Delay
Arrival_Delay
Average Canceled Rate

Now we can see that among all the flights to Austin airport, BHM airport will have the highest average delay time in both departure delay and arrival delay. Although the cancellation rate is 0, the passengers have to wait 38 minutes more for departure. Besides, we found that STL airport has the highest cancellation rate. Despite having low average delay time, STL airport have an average of 6.3% cancellation rate.

Author attribution

In this part, I will clean the text data, transform as TF-IDF, and construct two models with the first 100 principle components.

Import and cleaning:

When importing the document, doc_list is a list of authors. Please note that saving any other file under directory of C50train can lead to an importing bug.

```
library(tm)

## Loading required package: NLP

##
## Attaching package: 'NLP'

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

library(magrittr)
library(slam)
library(proxy)

##
## Attaching package: 'proxy'

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
```

```
            id=fname, language='en') }
setwd("../data/ReutersC50/C50train")
doc_list = Sys.glob('*')
file_list = Sys.glob(paste0(doc_list, '/*.txt'))

file_list = Sys.glob(paste0(doc_list, '/*.txt'))
temp = lapply(file_list, readerPlain)


mynames = file_list %>%
{ strsplit(., '/', fixed=TRUE) } %>%
{ lapply(., tail, n=2) } %>%
{ lapply(., paste0, collapse = '') } %>%
  unlist
names(temp) = mynames


documents_raw = VCorpus(VectorSource(temp))

my_documents = documents_raw
my_documents = tm_map(my_documents, content_transformer(tolower))
my_documents = tm_map(my_documents, content_transformer(removeNumbers))
my_documents = tm_map(my_documents, content_transformer(removePunctuation))
my_documents = tm_map(my_documents, content_transformer(stripWhitespace))

DTM = DocumentTermMatrix(my_documents)

DTM = removeSparseTerms(DTM, 0.95)
#Orignal Sparsity is over 90%. To decrease, remove terms that never show up
in 95% or more articles.

# construct TF IDF weights
tfidf = weightTfIdf(DTM)
```

For the document cleaning, I DID NOT exclude any stopwords. As IDF will take term frequency accross articles into account, a common but meaningless word tends to have lower TF-IDF. Thus, I'll let the TF-IDF calculation to do the work.

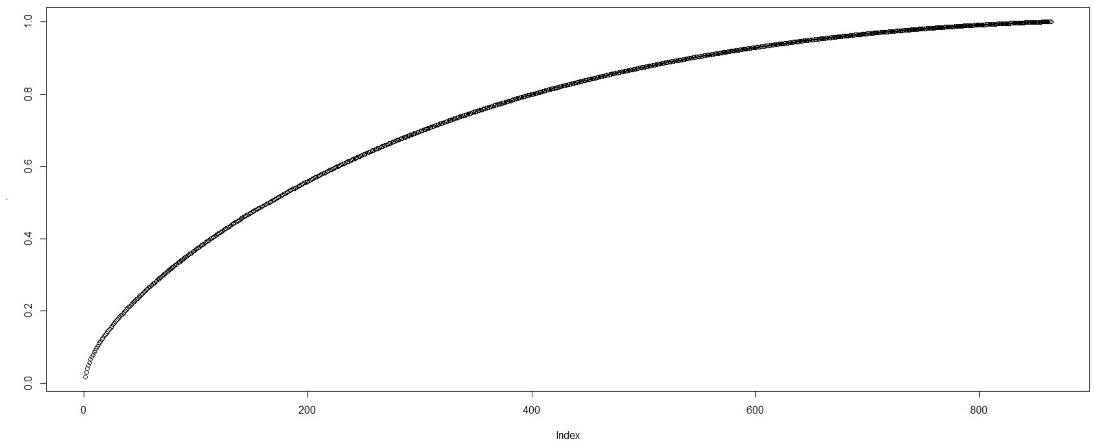
PCA of the training data

```
# Now PCA on tfidf
X = as.matrix(tfidf)
summary(colSums(X))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.212   4.356   4.899   5.891  20.616

scrub_cols = which(colSums(X) == 0)
X = X[,-scrub_cols]
```

```
pca= prcomp(X, scale=TRUE)
summary(pca)$importance[3,]%>%plot()
```




```
#independent variables: X
X = pca$x[,1:100]

all  <- vector()
i = 1
for (auther in doc_list){
  all[i] = auther
  i = i+1
}

#Model dependent variable: Y
Y = vector()
for( i in 1:2500){
  Y[i] = all[ceiling(i/50)]
}
```

The increasing rate of cumulative explained variance decreases gradually with the number of PCs. Considering a balance between variable number and explained variance, I pick up the top 100 PCs, with which 37% variance are explained, to build the model.

Test set import and clean

likewisely to the process of train set.

```
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }
setwd("../data/ReutersC50/C50test")
doc_list = Sys.glob('*')
file_list = Sys.glob(paste0(doc_list, '/*.txt'))
```

```
file_list = Sys.glob(paste0(doc_list, '/*.txt'))
cp2 = lapply(file_list, readerPlain)


mynames = file_list %>%
{ strsplit(., '/', fixed=TRUE) } %>%
{ lapply(., tail, n=2) } %>%
{ lapply(., paste0, collapse = '') } %>%
  unlist
names(cp2) = mynames


documents_raw_1 = VCorpus(VectorSource(cp2))

my_documents1 = documents_raw_1
my_documents1 = tm_map(my_documents1, content_transformer(tolower)) # make
everything lowercase
my_documents1 = tm_map(my_documents1, content_transformer(removeNumbers)) #
remove numbers
my_documents1 = tm_map(my_documents1, content_transformer(removePunctuation))
# remove punctuation
my_documents1 = tm_map(my_documents1, content_transformer(stripWhitespace))
## remove excess white-space

DTM_test = DocumentTermMatrix(my_documents1,control =
list(dictionary=Terms(DTM)))
DTM_test = removeSparseTerms(DTM_test, 0.95)

tfidf_test = weightTfIdf(DTM_test)

X_test = as.matrix(tfidf_test)
scrub_cols = which(colSums(X_test) == 0)
X_test = X_test[,-scrub_cols]
```

Now, X_test is our TF-IDF of the test file. As our model will be based on the 100 PCs of training set, we need to calculate the linear combination of the test set TF-IDF with the loadings of 100 PCs of trainings set before making prediction.

```
####Matching the column name of test TFIDF to the Train TFIDF
train_pre_pc = as.matrix(tfidf)
scrub_cols = which(colSums(train_pre_pc) == 0)
train_pre_pc = train_pre_pc[,-scrub_cols]

train_name = colnames(train_pre_pc)
test_name = colnames(X_test)
sup = setdiff(train_name, test_name)

temp_x = data.frame(X_test)
for (colname_ in sup){
```

```
  temp_x[,colname_] = 0
}

##somehow there is still difference
#This can be identified using:
#setdiff(colnames(t), train_name)

#hereby I manually fix them
colnames(temp_x)[colnames(temp_x)=="for."] <- "for"
colnames(temp_x)[colnames(temp_x)=="next."] <- "next"
colnames(temp_x)[colnames(temp_x)=="while."] <- "while"
t = data.matrix(temp_x)
t <- t[, order(colnames(t))]
#####

#transform the test set to the principal component spaces of the training set
test.data <- predict(pca, newdata =t)
test.data <- as.data.frame(test.data)
test.data <- test.data[,1:100]
```

test.data is the X of test_set for our model to predict.

Model: Logistics LASSO

```
library(MLmetrics)

##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

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

out1 = glmnet(X, factor(Y), family="multinomial")
p1 = predict(out1, data.matrix(test.data), s=0.01, type = "response")

myPredict_for_out1 <- function(which_article){
  return(which.max(p1[which_article,,]))
}

Ya  <- vector()
```

```r
i = 1
for (auther in doc_list){
  Ya[i] = auther
  i = i+1
}

#real is the true values
real = vector()
for( i in 1:2500){
  real[i] = Ya[ceiling(i/50)]
}

#aut is our prediction
aut = vector()
for (i in 1:2500){
  aut[i] =names(myPredict_for_out1(i))
}

Accuracy(aut, real)
```

```
## [1] 0.416
```

```r
(table(aut,real)%>%confusionMatrix)$byClass[,"Balanced Accuracy"]
```

```
##     Class: AaronPressman        Class: AlanCrosby    Class: AlexanderSmith
##                0.8646939                0.6275510                0.7381633
##   Class: BenjaminKangLim     Class: BernardHickey        Class: BradDorfman
##                0.6642857                0.5804082                0.5848980
##  Class: DarrenSchuettler       Class: DavidLawder     Class: EdnaFernandes
##                0.6559184                0.5442857                0.5379592
##       Class: EricAuchard    Class: FumikoFujisaki    Class: GrahamEarnshaw
##                0.5838776                0.9234694                0.8342857
##  Class: HeatherScoffield     Class: JaneMacartney         Class: JanLopatka
##                0.6414286                0.5079592                0.6738776
##      Class: JimGilchrist          Class: JoeOrtiz      Class: JohnMastrini
##                0.8969388                0.5385714                0.6646939
##      Class: JonathanBirt    Class: JoWinterbottom        Class: KarlPenhaul
##                0.7410204                0.8328571                0.8065306
##         Class: KeithWeir    Class: KevinDrawbaugh     Class: KevinMorrison
##                0.6473469                0.6020408                0.5973469
##     Class: KirstinRidley Class: KouroshKarimkhany          Class: LydiaZajc
##                0.6871429                0.8185714                0.8087755
##    Class: LynneO'Donnell   Class: LynnleyBrowning  Class: MarcelMichelson
##                0.8826531                0.9434694                0.7746939
##      Class: MarkBendeich        Class: MartinWolk       Class: MatthewBunce
##                0.7132653                0.5461224                0.9053061
##     Class: MichaelConnor        Class: MureDickie          Class: NickLouth
##                0.7442857                0.7285714                0.8212245
##   Class: PatriciaCommins     Class: PeterHumphrey         Class: PierreTran
##                0.6161224                0.8557143                0.6955102
```

```
##         Class: RobinSidel      Class: RogerFillion       Class: SamuelPerry
##                 0.9026531                0.8065306                0.6055102
##       Class: SarahDavison       Class: ScottHillis       Class: SimonCowell
##                 0.6883673                0.5195918                0.6057143
##           Class: TanEeLyn    Class: TheresePoletti         Class: TimFarrand
##                 0.5634694                0.6202041                0.7871429
##         Class: ToddNissen      Class: WilliamKazer
##                 0.6620408                0.5089796
```

Out-of-sample overall accuracy is around 41%. The baseline in this prediction is 1/50, which is 2%. To improve the accuracy, one may cross validate on LASSO lambda. Due to limited computation ability, current lambda is only based on manual adjustment. "Balanced Accuracy = (sensitivity+specificity)/2" Though the overall accuracy is only 41%, the model performs well in term of Sensitivity and Specificity, according to the balanced accuracy(50%~90%). That is, the proportion of actual positives/negatives that are correctly identified is high.

Model: Random Forest

```
library(randomForest)

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

fY = factor(Y)
dfX =data.frame(X)
XY = cbind(dfX, fY)

rffit = randomForest(fY~.,data=XY,ntree=500)
prf<- predict(rffit, newdata = test.data)
Accuracy(prf, factor(real))

## [1] 0.522

(table(prf,real)%>%confusionMatrix)$byClass[,"Balanced Accuracy"]

##     Class: AaronPressman         Class: AlanCrosby    Class: AlexanderSmith
##                 0.8581633                0.7693878                0.7342857
##   Class: BenjaminKangLim      Class: BernardHickey         Class: BradDorfman
##                 0.6530612                0.6465306                0.7626531
##  Class: DarrenSchuettler        Class: DavidLawder      Class: EdnaFernandes
##                 0.6077551                0.5377551                0.5783673
##        Class: EricAuchard    Class: FumikoFujisaki     Class: GrahamEarnshaw
##                 0.6651020                0.8967347                0.8859184
```

```
##  Class: HeatherScoffield     Class: JaneMacartney        Class: JanLopatka
##                0.6632653                0.6493878                0.7069388
##      Class: JimGilchrist          Class: JoeOrtiz      Class: JohnMastrini
##                0.9475510                0.6638776                0.7748980
##      Class: JonathanBirt    Class: JoWinterbottom        Class: KarlPenhaul
##                0.7644898                0.8277551                0.9126531
##         Class: KeithWeir    Class: KevinDrawbaugh     Class: KevinMorrison
##                0.8248980                0.7787755                0.7618367
##     Class: KirstinRidley Class: KouroshKarimkhany         Class: LydiaZajc
##                0.7963265                0.8277551                0.8100000
##    Class: LynneO'Donnell   Class: LynnleyBrowning  Class: MarcelMichelson
##                0.8883673                0.9581633                0.7271429
##      Class: MarkBendeich        Class: MartinWolk      Class: MatthewBunce
##                0.6977551                0.6585714                0.8991837
##     Class: MichaelConnor        Class: MureDickie         Class: NickLouth
##                0.7871429                0.6344898                0.8051020
##   Class: PatriciaCommins     Class: PeterHumphrey        Class: PierreTran
##                0.7067347                0.8989796                0.7655102
##        Class: RobinSidel      Class: RogerFillion       Class: SamuelPerry
##                0.8661224                0.8275510                0.7610204
##      Class: SarahDavison       Class: ScottHillis       Class: SimonCowell
##                0.7353061                0.5351020                0.7948980
##          Class: TanEeLyn    Class: TheresePoletti        Class: TimFarrand
##                0.6175510                0.7281633                0.8210204
##        Class: ToddNissen      Class: WilliamKazer
##                0.7822449                0.6038776
```

Out-of-sample overall accuracy is around 51%, with a similar balanced accuracy of each class. Specifically, both of the models have difficulies to identify ScottHillis, EdnaFernandes and WilliamKazer (balanced accuracy <60%). Acheiving a significantly higher overall accuracy, random forest is preferable to LASSO.

So, are there any sets of authors whose articles seem difficult to distinguish from one another?

To answer this, I derived the cosine distance matrix of each article, drop the reversed duplicates and sorted them. With the matrix, we are able to find a corresponding pair of authors who have similar articles. I wrote a function to calculate “the number of close article that a specific pair of authors have” under a user-defined threshold of cosine distance.

```
cosine_docs = function(dtm) {
  crossprod_simple_triplet_matrix(t(dtm))/(sqrt(col_sums(t(dtm)^2) %*%
t(col_sums(t(dtm)^2))))
}

# use the function to compute pairwise cosine similarity for all documents
cosine_mat = cosine_docs(tfidf)
```

```r
myStore = data.frame()
for(i in 1:2500){
  myStore[i,1] = as.numeric(i)
  myStore[i,2] = as.numeric(sort(cosine_mat[i,], decreasing=F)[1]%>%names)
  myStore[i,3] = sort(cosine_mat[i,], decreasing=F)[1]
}
colnames(myStore)<- c("Article_1", "Article_2", "Cosine_Distance")

#These are the articles who are very similar to each other
myrank = myStore[order(myStore$Cosine_Distance),]
#drop reversed duplicates
temp1 = apply(myrank[,1:2],1,function(x) paste(sort(x),collapse=''))
#These are the articles who are very similar, even identical, to each other
(myrank[!duplicated(gsub(" ", "", temp1, fixed = TRUE)),])[1:10,]

##      Article_1 Article_2 Cosine_Distance
## 6            6       615    0.000000e+00
## 764        764         6    0.000000e+00
## 765        765         6    0.000000e+00
## 1206      1206      1681    0.000000e+00
## 2187      2187      1681    0.000000e+00
## 800        800      2393    2.792552e-08
## 212        212       765    1.206647e-07
## 1513      1513       615    1.921827e-07
## 602        602      2393    2.364743e-07
## 780        780       615    2.912278e-07

#These are the corresponding authers
myrank1 = myrank[!duplicated(gsub(" ", "", temp1, fixed = TRUE)),]
myrank1$Article_1 = ceiling(myrank1$Article_1/50)
myrank1$Article_2 = ceiling(myrank1$Article_2/50)
myrank1 = myrank1[order(myrank1$Cosine_Distance),]


#pragma only after myrank1 defined
myThreshold<- function(threshold){
  local_df = myrank1[myrank1[,3]<threshold,]
  tr = apply(local_df[,1:2],1,function(x) paste(sort(x),collapse='-
'))%>%table
  return(tr[order(tr, decreasing = TRUE)])
}

#These are the authers have lots of similar articles
myThreshold(0.001)%>%head

## .
##  1-14 16-34 16-40  1-11  1-16   1-4
##     8     7     6     5     5     5
```

_myThreshold__(float threshold): exclude the article pairs that have a cosines distance above threshold, and return a sorted vector specify how many similar articles does the pair of authers have. The number "X-Y" correspond to the sequantial number of authors in the test data. i.e.: under 0.001 cosine distance, auther 1(AaronPressman) and auther 14(JanLopatka) have 8 simiar articles. Thus, their articles may be hard to identify. With the table above, we may identify the similar articles and authors.

Practice with association rule mining

```
library(tidyverse)
library(arules)
library(arulesViz)

groceries_raw =
read.transactions("https://raw.githubusercontent.com/jgscott/STA380/master/da
ta/groceries.txt", sep = ",")
str(groceries_raw)

## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:43367] 29 88 118 132 33 157 167 166 38 91 ...
##   .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
##   .. .. ..@ Dim     : int [1:2] 169 9835
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  169 obs. of  1 variable:
##   .. ..$ labels: chr [1:169] "abrasive cleaner" "artif. sweetener" "baby
cosmetics" "baby food" ...
##   ..@ itemsetInfo:'data.frame':  0 obs. of  0 variables

summary(groceries_raw)

## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   29   14   14    9   11    4    6    1    1    1    1    3    1
##
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
groceries_raw <- as (groceries_raw, "transactions")
```

```
freqItems = eclat(groceries_raw, parameter = list(supp = .07, maxlen = 15))
```

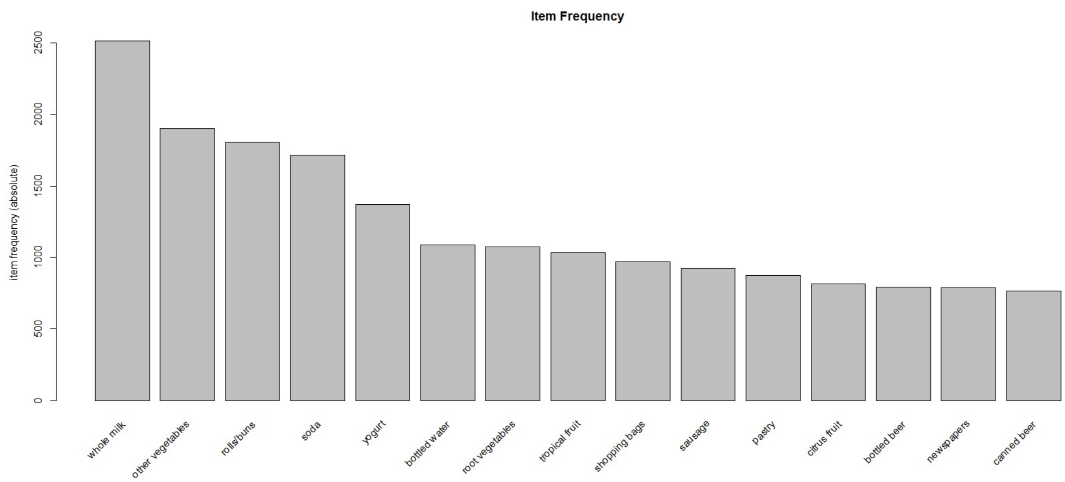
```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target   ext
##     FALSE    0.07      1     15 frequent itemsets FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 688
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating sparse bit matrix ... [18 row(s), 9835 column(s)] done [0.00s].
## writing  ... [19 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
inspect(freqItems)
```

```
##      items                              support    count
## [1]  {other vegetables,whole milk} 0.07483477  736
## [2]  {whole milk}                  0.25551601 2513
## [3]  {other vegetables}            0.19349263 1903
## [4]  {rolls/buns}                  0.18393493 1809
## [5]  {yogurt}                      0.13950178 1372
## [6]  {soda}                        0.17437722 1715
## [7]  {root vegetables}             0.10899847 1072
## [8]  {tropical fruit}              0.10493137 1032
## [9]  {bottled water}               0.11052364 1087
## [10] {sausage}                     0.09395018  924
## [11] {shopping bags}               0.09852567  969
## [12] {citrus fruit}                0.08276563  814
## [13] {pastry}                      0.08896797  875
## [14] {pip fruit}                   0.07564820  744
## [15] {whipped/sour cream}          0.07168277  705
## [16] {fruit/vegetable juice}       0.07229283  711
```

```
## [17] {newspapers}                     0.07981698  785
## [18] {bottled beer}                   0.08052872  792
## [19] {canned beer}                    0.07768175  764
```

```
itemFrequencyPlot(groceries_raw, topN=15, type="absolute", main="Item 
Frequency")
```



We set the parameters to plot top 15 groceries with the largest number of counts in dataset. Based on summary above and this plot of transaction data, ‘wholemilk’ has the biggest frequency and ‘other vegetables’ category follows. This might impact the result of our analysis.

We tried several models with different support and confidence level and found 0.15% of support and 60% of confidence level returns the most interesting and distinguishable result. We also set ‘manlen=3’ since, assuming the retail company puts products in a row at each aisle, retail company often asks that which product should they put on the left and right side of the section. For this practical reason, we choose maxlen=3.

```
rules <- apriori (groceries_raw, parameter = list(supp = 0.0015, conf = 0.60, 
maxlen = 3))

## Apriori
## 
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5  0.0015      1
##  maxlen target   ext
##       3  rules FALSE
## 
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
## 
## Absolute minimum support count: 14
## 
```

```
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [153 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3

## Warning in apriori(groceries_raw, parameter = list(supp = 0.0015, conf =
## 0.6, : Mining stopped (maxlen reached). Only patterns up to a length of 3
## returned!

##  done [0.00s].
## writing ... [255 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules_conf <- sort(rules, by="lift", decreasing = TRUE)
inspect(rules_conf)
```

```
##       lhs                           rhs                     support
confidence      lift count
## [1]   {ham,
##        processed cheese}         => {white bread}      0.001931876
0.6333333 15.045491    19
## [2]   {liquor,
##        red/blush wine}           => {bottled beer}     0.001931876
0.9047619 11.235269    19
## [3]   {rice,
##        yogurt}                   => {root vegetables}  0.001626843
0.6956522  6.382219    16
## [4]   {root vegetables,
##        turkey}                   => {tropical fruit}   0.001525165
0.6000000  5.718023    15
## [5]   {herbs,
##        tropical fruit}           => {root vegetables}  0.001728521
0.6071429  5.570196    17
## [6]   {herbs,
##        rolls/buns}               => {root vegetables}  0.001830198
0.6000000  5.504664    18
## [7]   {curd,
##        soft cheese}              => {yogurt}           0.001525165
0.6818182  4.887523    15
## [8]   {fruit/vegetable juice,
##        soft cheese}              => {yogurt}           0.001830198
0.6666667  4.778912    18
## [9]   {butter milk,
##        pork}                     => {other vegetables} 0.001830198
0.8571429  4.429848    18
## [10]  {margarine,
##        meat}                     => {other vegetables} 0.001728521
0.8500000  4.392932    17
## [11]  {rice,
##        yogurt}                   => {other vegetables} 0.001931876
```

```
0.8260870  4.269346    19
## [12]  {herbs,
##        shopping bags}            => {other vegetables} 0.001931876
0.8260870  4.269346    19
## [13]  {onions,
##        sliced cheese}            => {other vegetables} 0.001525165
0.7894737  4.080123    15
## [14]  {root vegetables,
##        turkey}                   => {other vegetables} 0.001931876
0.7600000  3.927798    19
## [15]  {soft cheese,
##        whipped/sour cream}       => {other vegetables} 0.002236909
0.7333333  3.789981    22
## [16]  {frozen vegetables,
##        soft cheese}              => {other vegetables} 0.001626843
0.7272727  3.758659    16
## [17]  {root vegetables,
##        soft cheese}              => {other vegetables} 0.002440264
0.7272727  3.758659    24
## [18]  {grapes,
##        pork}                     => {other vegetables} 0.001626843
0.7272727  3.758659    16
## [19]  {citrus fruit,
##        herbs}                    => {other vegetables} 0.002135231
0.7241379  3.742457    21
## [20]  {baking powder,
##        root vegetables}          => {other vegetables} 0.002541942
0.7142857  3.691540    25
## [21]  {frozen vegetables,
##        ham}                      => {other vegetables} 0.001525165
0.7142857  3.691540    15
## [22]  {rice,
##        root vegetables}          => {other vegetables} 0.002236909
0.7096774  3.667723    22
## [23]  {herbs,
##        pip fruit}                => {other vegetables} 0.001728521
0.7083333  3.660777    17
## [24]  {onions,
##        white bread}              => {other vegetables} 0.001728521
0.7083333  3.660777    17
## [25]  {canned vegetables,
##        root vegetables}          => {other vegetables} 0.001830198
0.6923077  3.577954    18
## [26]  {margarine,
##        soft cheese}              => {other vegetables} 0.001525165
0.6818182  3.523742    15
## [27]  {cat food,
##        whipped/sour cream}       => {other vegetables} 0.001931876
0.6785714  3.506963    19
## [28]  {frozen meals,
```

```
##        whipped/sour cream}        => {other vegetables} 0.001931876
0.6785714  3.506963    19
## [29]  {processed cheese,
##        root vegetables}           => {other vegetables} 0.002135231
0.6774194  3.501009    21
## [30]  {frozen vegetables,
##        onions}                    => {other vegetables} 0.002135231
0.6774194  3.501009    21
## [31]  {root vegetables,
##        sliced cheese}             => {other vegetables} 0.003762074
0.6727273  3.476759    37
## [32]  {frozen dessert,
##        root vegetables}           => {other vegetables} 0.001626843
0.6666667  3.445437    16
## [33]  {ham,
##        pip fruit}                 => {other vegetables} 0.002643620
0.6666667  3.445437    26
## [34]  {hamburger meat,
##        pip fruit}                 => {other vegetables} 0.002948653
0.6590909  3.406284    29
## [35]  {frozen meals,
##        root vegetables}           => {other vegetables} 0.002541942
0.6578947  3.400102    25
## [36]  {soft cheese,
##        tropical fruit}            => {other vegetables} 0.002135231
0.6562500  3.391602    21
## [37]  {ham,
##        margarine}                 => {other vegetables} 0.001728521
0.6538462  3.379179    17
## [38]  {chicken,
##        dessert}                   => {other vegetables} 0.001525165
0.6521739  3.370536    15
## [39]  {brown bread,
##        whipped/sour cream}        => {other vegetables} 0.003050330
0.6521739  3.370536    30
## [40]  {chicken,
##        hamburger meat}            => {other vegetables} 0.002440264
0.6486486  3.352317    24
## [41]  {herbs,
##        whipped/sour cream}        => {other vegetables} 0.002033554
0.6451613  3.334294    20
## [42]  {domestic eggs,
##        soft cheese}               => {other vegetables} 0.002033554
0.6451613  3.334294    20
## [43]  {curd,
##        hamburger meat}            => {other vegetables} 0.002033554
0.6451613  3.334294    20
## [44]  {frozen vegetables,
##        sugar}                     => {other vegetables} 0.002033554
0.6451613  3.334294    20
```

```
## [45]  {onions,
##        tropical fruit}           => {other vegetables} 0.003660397
0.6428571  3.322386    36
## [46]  {cream cheese,
##        long life bakery product} => {other vegetables} 0.001830198
0.6428571  3.322386    18
## [47]  {mayonnaise,
##        root vegetables}          => {other vegetables} 0.001626843
0.6400000  3.307620    16
## [48]  {butter milk,
##        pip fruit}                => {other vegetables} 0.003253686
0.6400000  3.307620    32
## [49]  {citrus fruit,
##        onions}                   => {other vegetables} 0.003558719
0.6363636  3.288826    35
## [50]  {long life bakery product,
##        root vegetables}          => {other vegetables} 0.003355363
0.6346154  3.279791    33
## [51]  {butter,
##        rice}                     => {whole milk}       0.001525165
0.8333333  3.261374    15
## [52]  {dishes,
##        root vegetables}          => {other vegetables} 0.001728521
0.6296296  3.254024    17
## [53]  {semi-finished bread,
##        yogurt}                   => {other vegetables} 0.002236909
0.6285714  3.248555    22
## [54]  {semi-finished bread,
##        whipped/sour cream}       => {other vegetables} 0.001525165
0.6250000  3.230097    15
## [55]  {baking powder,
##        sausage}                  => {other vegetables} 0.001525165
0.6250000  3.230097    15
## [56]  {chicken,
##        oil}                      => {other vegetables} 0.001525165
0.6250000  3.230097    15
## [57]  {hamburger meat,
##        onions}                   => {other vegetables} 0.001525165
0.6250000  3.230097    15
## [58]  {hamburger meat,
##        pork}                     => {other vegetables} 0.002033554
0.6250000  3.230097    20
## [59]  {herbs,
##        tropical fruit}           => {whole milk}       0.002338587
0.8214286  3.214783    23
## [60]  {soups,
##        whole milk}               => {other vegetables} 0.001830198
0.6206897  3.207821    18
## [61]  {ice cream,
##        newspapers}               => {other vegetables} 0.001830198
```

```
0.6206897  3.207821    18
## [62]  {ice cream,
##        root vegetables}          => {other vegetables} 0.001830198
0.6206897  3.207821    18
## [63]  {onions,
##        whipped/sour cream}       => {other vegetables} 0.003152008
0.6200000  3.204256    31
## [64]  {hard cheese,
##        root vegetables}          => {other vegetables} 0.003457041
0.6181818  3.194860    34
## [65]  {onions,
##        pip fruit}                => {other vegetables} 0.002135231
0.6176471  3.192096    21
## [66]  {pork,
##        waffles}                  => {other vegetables} 0.002135231
0.6176471  3.192096    21
## [67]  {pip fruit,
##        pork}                     => {other vegetables} 0.003762074
0.6166667  3.187029    37
## [68]  {tropical fruit,
##        turkey}                   => {other vegetables} 0.001626843
0.6153846  3.180403    16
## [69]  {ice cream,
##        whipped/sour cream}       => {other vegetables} 0.001626843
0.6153846  3.180403    16
## [70]  {oil,
##        yogurt}                   => {other vegetables} 0.003253686
0.6153846  3.180403    32
## [71]  {grapes,
##        root vegetables}          => {other vegetables} 0.002745297
0.6136364  3.171368    27
## [72]  {hard cheese,
##        whipped/sour cream}       => {other vegetables} 0.002745297
0.6136364  3.171368    27
## [73]  {cereals,
##        yogurt}                   => {whole milk}       0.001728521
0.8095238  3.168192    17
## [74]  {bottled beer,
##        hamburger meat}           => {whole milk}       0.001728521
0.8095238  3.168192    17
## [75]  {whipped/sour cream,
##        white bread}              => {other vegetables} 0.003355363
0.6111111  3.158317    33
## [76]  {curd,
##        hamburger meat}           => {whole milk}       0.002541942
0.8064516  3.156169    25
## [77]  {grapes,
##        yogurt}                   => {other vegetables} 0.002846975
0.6086957  3.145834    28
## [78]  {cat food,
```

```
##        root vegetables}          => {other vegetables} 0.002846975
0.6086957  3.145834    28
## [79]  {butter,
##        candy}                    => {other vegetables} 0.001728521
0.6071429  3.137809    17
## [80]  {chicken,
##        onions}                   => {other vegetables} 0.001728521
0.6071429  3.137809    17
## [81]  {coffee,
##        oil}                      => {other vegetables} 0.002033554
0.6060606  3.132215    20
## [82]  {herbs,
##        rolls/buns}               => {whole milk}       0.002440264
0.8000000  3.130919    24
## [83]  {butter milk,
##        whipped/sour cream}       => {other vegetables} 0.002338587
0.6052632  3.128094    23
## [84]  {pip fruit,
##        whipped/sour cream}       => {other vegetables} 0.005592272
0.6043956  3.123610    55
## [85]  {onions,
##        root vegetables}          => {other vegetables} 0.005693950
0.6021505  3.112008    56
## [86]  {frozen fish,
##        tropical fruit}           => {other vegetables} 0.001525165
0.6000000  3.100893    15
## [87]  {herbs,
##        rolls/buns}               => {other vegetables} 0.001830198
0.6000000  3.100893    18
## [88]  {baking powder,
##        rolls/buns}               => {other vegetables} 0.002135231
0.6000000  3.100893    21
## [89]  {grapes,
##        tropical fruit}           => {other vegetables} 0.003660397
0.6000000  3.100893    36
## [90]  {chocolate,
##        frozen vegetables}        => {other vegetables} 0.001525165
0.6000000  3.100893    15
## [91]  {rice,
##        tropical fruit}           => {whole milk}       0.001525165
0.7894737  3.089723    15
## [92]  {detergent,
##        whipped/sour cream}       => {whole milk}       0.001525165
0.7894737  3.089723    15
## [93]  {rice,
##        yogurt}                   => {whole milk}       0.001830198
0.7826087  3.062856    18
## [94]  {rice,
##        root vegetables}          => {whole milk}       0.002440264
0.7741935  3.029922    24
```

```
## [95]  {butter milk,
##        whipped/sour cream}        => {whole milk}       0.002948653
0.7631579  2.986732    29
## [96]  {bottled water,
##        mustard}                   => {whole milk}       0.001525165
0.7500000  2.935237    15
## [97]  {curd,
##        herbs}                     => {whole milk}       0.001830198
0.7500000  2.935237    18
## [98]  {curd,
##        ham}                       => {whole milk}       0.001830198
0.7500000  2.935237    18
## [99]  {butter,
##        onions}                    => {whole milk}       0.003050330
0.7500000  2.935237    30
## [100] {butter,
##        soft cheese}               => {whole milk}       0.002033554
0.7407407  2.898999    20
## [101] {cream cheese,
##        sugar}                     => {whole milk}       0.002033554
0.7407407  2.898999    20
## [102] {cat food,
##        curd}                      => {whole milk}       0.001728521
0.7391304  2.892697    17
## [103] {curd,
##        domestic eggs}             => {whole milk}       0.004778851
0.7343750  2.874086    47
## [104] {oil,
##        sugar}                     => {whole milk}       0.001626843
0.7272727  2.846290    16
## [105] {berries,
##        frankfurter}               => {whole milk}       0.001626843
0.7272727  2.846290    16
## [106] {curd,
##        onions}                    => {whole milk}       0.001830198
0.7200000  2.817827    18
## [107] {chicken,
##        sugar}                     => {whole milk}       0.001830198
0.7200000  2.817827    18
## [108] {butter,
##        curd}                      => {whole milk}       0.004880529
0.7164179  2.803808    48
## [109] {specialty cheese,
##        yogurt}                    => {whole milk}       0.002033554
0.7142857  2.795464    20
## [110] {roll products,
##        rolls/buns}                => {whole milk}       0.001525165
0.7142857  2.795464    15
## [111] {domestic eggs,
##        herbs}                     => {whole milk}       0.001525165
```

```
0.7142857  2.795464    15
## [112] {hard cheese,
##        margarine}                => {whole milk}       0.001525165
0.7142857  2.795464    15
## [113] {butter milk,
##        long life bakery product} => {whole milk}       0.001525165
0.7142857  2.795464    15
## [114] {butter milk,
##        dessert}                  => {whole milk}       0.002033554
0.7142857  2.795464    20
## [115] {domestic eggs,
##        sugar}                    => {whole milk}       0.003558719
0.7142857  2.795464    35
## [116] {baking powder,
##        yogurt}                   => {whole milk}       0.003253686
0.7111111  2.783039    32
## [117] {sliced cheese,
##        whipped/sour cream}       => {whole milk}       0.002745297
0.7105263  2.780751    27
## [118] {butter,
##        cat food}                 => {whole milk}       0.001728521
0.7083333  2.772168    17
## [119] {butter,
##        pork}                     => {whole milk}       0.003863752
0.7037037  2.754049    38
## [120] {butter,
##        coffee}                   => {whole milk}       0.003355363
0.7021277  2.747881    33
## [121] {butter,
##        hamburger meat}           => {whole milk}       0.003050330
0.6976744  2.730453    30
## [122] {butter,
##        hygiene articles}         => {whole milk}       0.003050330
0.6976744  2.730453    30
## [123] {root vegetables,
##        soft cheese}              => {whole milk}       0.002338587
0.6969697  2.727695    23
## [124] {curd,
##        frozen meals}             => {whole milk}       0.001626843
0.6956522  2.722538    16
## [125] {frankfurter,
##        sliced cheese}            => {whole milk}       0.001626843
0.6956522  2.722538    16
## [126] {frozen potato products,
##        other vegetables}         => {whole milk}       0.001830198
0.6923077  2.709449    18
## [127] {frozen fish,
##        root vegetables}          => {whole milk}       0.001830198
0.6923077  2.709449    18
## [128] {brown bread,
```

```
##        ham}                         => {whole milk}       0.001830198
0.6923077  2.709449    18
## [129] {berries,
##        margarine}                   => {whole milk}       0.001830198
0.6923077  2.709449    18
## [130] {frozen fish,
##        yogurt}                      => {whole milk}       0.002236909
0.6875000  2.690634    22
## [131] {frozen meals,
##        root vegetables}             => {whole milk}       0.002643620
0.6842105  2.677760    26
## [132] {frozen potato products,
##        yogurt}                      => {whole milk}       0.001525165
0.6818182  2.668397    15
## [133] {butter,
##        detergent}                   => {whole milk}       0.001525165
0.6818182  2.668397    15
## [134] {frozen vegetables,
##        soft cheese}                 => {whole milk}       0.001525165
0.6818182  2.668397    15
## [135] {butter milk,
##        domestic eggs}               => {whole milk}       0.001525165
0.6818182  2.668397    15
## [136] {cream cheese,
##        domestic eggs}               => {whole milk}       0.003457041
0.6800000  2.661281    34
## [137] {chocolate,
##        frozen vegetables}           => {whole milk}       0.001728521
0.6800000  2.661281    17
## [138] {baking powder,
##        bottled water}               => {whole milk}       0.001931876
0.6785714  2.655690    19
## [139] {domestic eggs,
##        soft cheese}                 => {whole milk}       0.002135231
0.6774194  2.651182    21
## [140] {butter,
##        sugar}                       => {whole milk}       0.002135231
0.6774194  2.651182    21
## [141] {curd,
##        sugar}                       => {whole milk}       0.002338587
0.6764706  2.647468    23
## [142] {napkins,
##        white bread}                 => {whole milk}       0.002338587
0.6764706  2.647468    23
## [143] {margarine,
##        white bread}                 => {whole milk}       0.002541942
0.6756757  2.644357    25
## [144] {butter,
##        cream cheese}                => {whole milk}       0.002745297
0.6750000  2.641713    27
```

```
## [145] {other vegetables,
##        rice}                    => {whole milk}       0.002643620
0.6666667  2.609099    26
## [146] {detergent,
##        rolls/buns}              => {whole milk}       0.002033554
0.6666667  2.609099    20
## [147] {soft cheese,
##        whipped/sour cream}      => {whole milk}       0.002033554
0.6666667  2.609099    20
## [148] {chocolate,
##        hamburger meat}          => {whole milk}       0.001626843
0.6666667  2.609099    16
## [149] {frankfurter,
##        hamburger meat}          => {whole milk}       0.002236909
0.6666667  2.609099    22
## [150] {hygiene articles,
##        root vegetables}         => {whole milk}       0.003558719
0.6603774  2.584485    35
## [151] {butter,
##        whipped/sour cream}      => {whole milk}       0.006710727
0.6600000  2.583008    66
## [152] {baking powder,
##        root vegetables}         => {whole milk}       0.002338587
0.6571429  2.571827    23
## [153] {soft cheese,
##        tropical fruit}          => {whole milk}       0.002135231
0.6562500  2.568332    21
## [154] {hamburger meat,
##        pork}                    => {whole milk}       0.002135231
0.6562500  2.568332    21
## [155] {citrus fruit,
##        herbs}                   => {whole milk}       0.001931876
0.6551724  2.564115    19
## [156] {fruit/vegetable juice,
##        processed cheese}        => {whole milk}       0.001931876
0.6551724  2.564115    19
## [157] {curd,
##        oil}                     => {whole milk}       0.001728521
0.6538462  2.558924    17
## [158] {turkey,
##        yogurt}                  => {whole milk}       0.001525165
0.6521739  2.552380    15
## [159] {citrus fruit,
##        specialty chocolate}     => {whole milk}       0.001525165
0.6521739  2.552380    15
## [160] {hard cheese,
##        yogurt}                  => {whole milk}       0.004168785
0.6507937  2.546978    41
## [161] {cream cheese,
##        pip fruit}               => {whole milk}       0.003965430
```

```
0.6500000  2.543872    39
## [162] {berries,
##        butter}                     => {whole milk}       0.002440264
0.6486486  2.538583    24
## [163] {pip fruit,
##        whipped/sour cream}         => {whole milk}       0.005998983
0.6483516  2.537421    59
## [164] {frozen meals,
##        tropical fruit}             => {whole milk}       0.003558719
0.6481481  2.536624    35
## [165] {onions,
##        pip fruit}                  => {whole milk}       0.002236909
0.6470588  2.532361    22
## [166] {chicken,
##        curd}                       => {whole milk}       0.002236909
0.6470588  2.532361    22
## [167] {sugar,
##        whipped/sour cream}         => {whole milk}       0.003152008
0.6458333  2.527565    31
## [168] {processed cheese,
##        root vegetables}            => {whole milk}       0.002033554
0.6451613  2.524935    20
## [169] {beef,
##        oil}                        => {whole milk}       0.002033554
0.6451613  2.524935    20
## [170] {cereals}                    => {whole milk}       0.003660397
0.6428571  2.515917    36
## [171] {baking powder,
##        pip fruit}                  => {whole milk}       0.001626843
0.6400000  2.504735    16
## [172] {chicken,
##        white bread}                => {whole milk}       0.001626843
0.6400000  2.504735    16
## [173] {hamburger meat,
##        root vegetables}            => {whole milk}       0.003965430
0.6393443  2.502169    39
## [174] {chicken,
##        domestic eggs}              => {whole milk}       0.003965430
0.6393443  2.502169    39
## [175] {butter,
##        yogurt}                     => {whole milk}       0.009354347
0.6388889  2.500387    92
## [176] {hygiene articles,
##        pip fruit}                  => {whole milk}       0.003050330
0.6382979  2.498074    30
## [177] {butter,
##        root vegetables}            => {whole milk}       0.008235892
0.6377953  2.496107    81
## [178] {oil,
##        root vegetables}            => {whole milk}       0.004473818
```

```
0.6376812  2.495660    44
## [179] {cream cheese,
##        frankfurter}              => {whole milk}       0.002135231
0.6363636  2.490504    21
## [180] {curd,
##        frozen vegetables}        => {whole milk}       0.002846975
0.6363636  2.490504    28
## [181] {ham,
##        whipped/sour cream}       => {whole milk}       0.002643620
0.6341463  2.481826    26
## [182] {curd,
##        tropical fruit}           => {whole milk}       0.006507372
0.6336634  2.479936    64
## [183] {pot plants,
##        tropical fruit}           => {whole milk}       0.001931876
0.6333333  2.478644    19
## [184] {beef,
##        waffles}                  => {whole milk}       0.001931876
0.6333333  2.478644    19
## [185] {butter,
##        napkins}                  => {whole milk}       0.003152008
0.6326531  2.475982    31
## [186] {beef,
##        butter}                   => {whole milk}       0.003660397
0.6315789  2.471778    36
## [187] {flour,
##        root vegetables}          => {whole milk}       0.002948653
0.6304348  2.467300    29
## [188] {bottled beer,
##        domestic eggs}            => {whole milk}       0.002948653
0.6304348  2.467300    29
## [189] {house keeping products,
##        other vegetables}         => {whole milk}       0.001728521
0.6296296  2.464149    17
## [190] {detergent,
##        frozen vegetables}        => {whole milk}       0.001728521
0.6296296  2.464149    17
## [191] {fruit/vegetable juice,
##        soft cheese}              => {whole milk}       0.001728521
0.6296296  2.464149    17
## [192] {specialty chocolate,
##        whipped/sour cream}       => {whole milk}       0.001728521
0.6296296  2.464149    17
## [193] {dessert,
##        ham}                      => {whole milk}       0.001728521
0.6296296  2.464149    17
## [194] {butter,
##        dessert}                  => {whole milk}       0.001728521
0.6296296  2.464149    17
## [195] {baking powder,
```

```
##        rolls/buns}              => {whole milk}      0.002236909
0.6285714  2.460008    22
## [196] {detergent,
##        root vegetables}         => {whole milk}      0.002745297
0.6279070  2.457408    27
## [197] {beef,
##        domestic eggs}           => {whole milk}      0.003762074
0.6271186  2.454322    37
## [198] {herbs,
##        pip fruit}               => {whole milk}      0.001525165
0.6250000  2.446031    15
## [199] {fruit/vegetable juice,
##        semi-finished bread}     => {whole milk}      0.001525165
0.6250000  2.446031    15
## [200] {baking powder,
##        frozen vegetables}       => {whole milk}      0.001525165
0.6250000  2.446031    15
## [201] {flour,
##        whipped/sour cream}      => {whole milk}      0.002541942
0.6250000  2.446031    25
## [202] {butter,
##        grapes}                  => {whole milk}      0.001525165
0.6250000  2.446031    15
## [203] {brown bread,
##        specialty chocolate}     => {whole milk}      0.001525165
0.6250000  2.446031    15
## [204] {domestic eggs,
##        hamburger meat}          => {whole milk}      0.002541942
0.6250000  2.446031    25
## [205] {chocolate,
##        sugar}                   => {whole milk}      0.001525165
0.6250000  2.446031    15
## [206] {domestic eggs,
##        pip fruit}               => {whole milk}      0.005388917
0.6235294  2.440275    53
## [207] {curd,
##        pip fruit}               => {whole milk}      0.004880529
0.6233766  2.439677    48
## [208] {butter,
##        tropical fruit}          => {whole milk}      0.006202339
0.6224490  2.436047    61
## [209] {domestic eggs,
##        margarine}               => {whole milk}      0.005185562
0.6219512  2.434099    51
## [210] {beef,
##        coffee}                  => {whole milk}      0.002338587
0.6216216  2.432809    23
## [211] {butter,
##        domestic eggs}           => {whole milk}      0.005998983
0.6210526  2.430582    59
```

```
## [212] {pastry,
##        processed cheese}         => {whole milk}       0.001830198
0.6206897  2.429161    18
## [213] {frankfurter,
##        frozen meals}             => {whole milk}       0.001830198
0.6206897  2.429161    18
## [214] {bottled water,
##        hamburger meat}           => {whole milk}       0.001830198
0.6206897  2.429161    18
## [215] {dessert,
##        long life bakery product} => {whole milk}       0.001830198
0.6206897  2.429161    18
## [216] {cream cheese,
##        pork}                     => {whole milk}       0.001830198
0.6206897  2.429161    18
## [217] {meat,
##        root vegetables}          => {whole milk}       0.003152008
0.6200000  2.426462    31
## [218] {butter milk,
##        root vegetables}          => {whole milk}       0.003152008
0.6200000  2.426462    31
## [219] {waffles,
##        whipped/sour cream}       => {whole milk}       0.003152008
0.6200000  2.426462    31
## [220] {hamburger meat,
##        whipped/sour cream}       => {whole milk}       0.002643620
0.6190476  2.422735    26
## [221] {cream cheese,
##        whipped/sour cream}       => {whole milk}       0.003965430
0.6190476  2.422735    39
## [222] {domestic eggs,
##        pork}                     => {whole milk}       0.003457041
0.6181818  2.419347    34
## [223] {curd,
##        white bread}              => {whole milk}       0.002135231
0.6176471  2.417254    21
## [224] {beef,
##        frankfurter}              => {whole milk}       0.002948653
0.6170213  2.414805    29
## [225] {rice}                     => {whole milk}       0.004677173
0.6133333  2.400371    46
## [226] {herbs,
##        whipped/sour cream}       => {whole milk}       0.001931876
0.6129032  2.398688    19
## [227] {baking powder,
##        domestic eggs}            => {whole milk}       0.001931876
0.6129032  2.398688    19
## [228] {butter,
##        pip fruit}                => {whole milk}       0.004473818
0.6111111  2.391674    44
```

```
## [229] {hamburger meat,
##        yogurt}                    => {whole milk}       0.003965430
0.6093750  2.384880    39
## [230] {pot plants,
##        rolls/buns}                => {whole milk}       0.001728521
0.6071429  2.376144    17
## [231] {berries,
##        napkins}                   => {whole milk}       0.001728521
0.6071429  2.376144    17
## [232] {sugar,
##        white bread}               => {whole milk}       0.001728521
0.6071429  2.376144    17
## [233] {domestic eggs,
##        tropical fruit}            => {whole milk}       0.006914082
0.6071429  2.376144    68
## [234] {detergent,
##        tropical fruit}            => {whole milk}       0.002033554
0.6060606  2.371909    20
## [235] {hygiene articles,
##        tropical fruit}            => {whole milk}       0.004067107
0.6060606  2.371909    40
## [236] {long life bakery product,
##        salty snack}               => {whole milk}       0.002033554
0.6060606  2.371909    20
## [237] {napkins,
##        sugar}                     => {whole milk}       0.002033554
0.6060606  2.371909    20
## [238] {pasta,
##        root vegetables}           => {whole milk}       0.002338587
0.6052632  2.368788    23
## [239] {beef,
##        pip fruit}                 => {whole milk}       0.002948653
0.6041667  2.364496    29
## [240] {bottled water,
##        butter}                    => {whole milk}       0.005388917
0.6022727  2.357084    53
## [241] {root vegetables,
##        turkey}                    => {whole milk}       0.001525165
0.6000000  2.348189    15
## [242] {frozen fish,
##        tropical fruit}            => {whole milk}       0.001525165
0.6000000  2.348189    15
## [243] {bottled water,
##        pot plants}                => {whole milk}       0.001525165
0.6000000  2.348189    15
## [244] {herbs,
##        yogurt}                    => {whole milk}       0.002135231
0.6000000  2.348189    21
## [245] {detergent,
##        pip fruit}                 => {whole milk}       0.001525165
```

```
0.6000000  2.348189    15
## [246] {baking powder,
##        margarine}                => {whole milk}      0.001830198
0.6000000  2.348189    18
## [247] {butter,
##        meat}                     => {whole milk}      0.001830198
0.6000000  2.348189    18
## [248] {frozen meals,
##        white bread}              => {whole milk}      0.001525165
0.6000000  2.348189    15
## [249] {hard cheese,
##        newspapers}               => {whole milk}      0.001525165
0.6000000  2.348189    15
## [250] {coffee,
##        sliced cheese}            => {whole milk}      0.001525165
0.6000000  2.348189    15
## [251] {oil,
##        pastry}                   => {whole milk}      0.001830198
0.6000000  2.348189    18
## [252] {oil,
##        rolls/buns}               => {whole milk}      0.003050330
0.6000000  2.348189    30
## [253] {hamburger meat,
##        margarine}                => {whole milk}      0.001830198
0.6000000  2.348189    18
## [254] {coffee,
##        napkins}                  => {whole milk}      0.002440264
0.6000000  2.348189    24
## [255] {beef,
##        tropical fruit}           => {whole milk}      0.004575496
0.6000000  2.348189    45
```

Among top 10 assciations with highest lift, we found few interesting facts. First, considering high association between 'ham, processed cheese' and 'white bread', this retail company should place few selections of white bread neary by 'ham and processed cheese' section for customers who want to make ham-cheese sandwiches. Since bread doesn't require any thermoregulation, the company can easily place small shelf for bread. Since many people buy things impulsively, this placement might only directly target customers who were look for ham-cheese sandwich previously but also remind them ham-cheese sandwiches. From the 4th association, many people buy turkey, vegetables and tropical fruit together and this combination seems to be a common lunch box; turkey sandwich with bananas. Since those associations are combinations with clear intentions, the compamy can not only place them together, but also do some marketing which helps people to remind 'ham-cheese sandwich' or 'turkey sandwich lunch box'.

The 9th and 10th association, however, show that customers who bought meat and dairy products, they are 4 times more likely to buy vegetables. Based on this fact, the company may should put three sections together for sale increase effect from complementary goods.