Exercise 2

J. Zhao (jz5223)

August 14, 2015

Flights at ABIA

```
library(RCurl) # I'm using curl to load data from Github over HTTPS.
abia <- read.csv(text =
getURL("https://raw.githubusercontent.com/jgscott/STA380/master/data/ABIA.csv"))
dim(abia) # abia should have 99260 rows (excl. header) and 29 columns
```

```
## [1] 99260    29
```

I want to be able to estimate the amount of delay I will experience leaving from ABIA, so I can plan layovers and destination activities accordingly. Of course, the layover and destination experience will be impacted by the connecting and destination airports, but I just wanted to understand departure delays at different times of day in Austin.

```
departures <- abia[abia$Origin == 'AUS',]
nrow(departures)/nrow(abia) # 50%, looks about right
```

```
## [1] 0.4999295
```

Also, I don't worry as much about arriving flights, as I usually don't have much going on at home. So I'm taking a subset of just the departing flights.

```
departures$CRSDepHour <- trunc(departures$CRSDepTime/100)
as.factor(departures$CRSDepHour)
```

In order to group-by, I created a column CRSDepHour by truncating the CRSDepTime. Looking at factor levels of CRSDepHour, there were no flights scheduled to depart ABIA between 11:00pm and 11:59pm in 2008. I wanted to check on the late-night scheduled departures so I created the subset check22 which I thought would consist of departing flights with departure times ranging from 2200 to 2259 hours.

```
check22 <- departures[departures$CRSDepHour == '22',]
```

I was surprised that the last scheduled flight departure time is actually 2200 hours exactly with no departures scheduled past that time. This is probably due to a noise curfew ordinance. The 10:00pm departure is Mesa Airlines flight YV 2890 bound for Las Vegas, which ran three times a week only in January and early February.

```
# Code taken from http://stackoverflow.com/questions/19997242
require(pander)
panderOptions('table.split.table', Inf)
cat(pander(check22[,c(2:5,10,18)], style = 'rmarkdown'))
```

	Month	DayofMonth	DayOfWeek	DepTime	FlightNum	Dest
1674	1	6	7	NA	2890	LAS
2841	1	10	4	2149	2890	LAS
3141	1	11	5	2149	2890	LAS
3639	1	13	7	2148	2890	LAS
4802	1	17	4	2158	2890	LAS
5105	1	18	5	2155	2890	LAS
5587	1	20	7	2150	2890	LAS
6762	1	24	4	2155	2890	LAS
7061	1	25	5	2155	2890	LAS
7544	1	27	7	2333	2890	LAS
8709	1	31	4	2150	2890	LAS
9015	2	1	5	2158	2890	LAS
9491	2	3	7	2150	2890	LAS
10672	2	7	4	NA	2890	LAS
10963	2	8	5	2155	2890	LAS
11451	2	10	7	2224	2890	LAS

With no flights past 10pm, I was thus surprised that there is a factor level of 0 in CRSDepHour. Subset check0 reveals two JetBlue flights (flight number B6 1832) scheduled to depart for Boston at 0055 hours. I presume these were exceptions created to move an aircraft overnight or for some stranded passengers after the Thanksgiving long weekend. November 30, 2008 was the Sunday after Thanksgiving.

```
check0 <- departures[departures$CRSDepHour == '0',]
cat(pander(check0[,c(2,3,5,6,10,18)], style = 'rmarkdown'))
```

	Month	DayofMonth	DepTime	CRSDepTime	FlightNum	Dest
91765	11	30	45	55	1832	BOS

92013	12	1	151	55	1832	BOS

Below is a table of Sunday departures to Boston in November and December 2008. This supports my suspicion that B6 1832 was a special exception over Thanksgiving to help passengers get back to work. I also looked at the whole year and there was only one direct flight to Boston on Sundays and Mondays (JetBlue flight B6 1264), always in the early afternoon, except the two midnight flights mentioned above. It is peculiar that only JetBlue got this Thanksgiving exception booked into the schedule, as there were no other flights scheduled past 10pm the whole year, including Thanksgiving weekend.

```
checkBOS <- departures[departures$Dest == 'BOS',]
checkBOSsun <- checkBOS[checkBOS$DayOfWeek == 7,]
checkBOSsun <- checkBOSsun[checkBOSsun$Month >= 11,]
cat(pander(checkBOSsun[,c(2,3,5,6,10,18)], style = 'rmarkdown'))
```

	Month	DayofMonth	DepTime	CRSDepTime	FlightNum	Dest
85268	11	2	1202	1220	1264	BOS
86942	11	9	1213	1220	1264	BOS
88605	11	16	1208	1220	1264	BOS
90284	11	23	1216	1220	1264	BOS
91765	11	30	45	55	1832	BOS
91868	11	30	1242	1220	1264	BOS
93525	12	7	1229	1220	1264	BOS
95167	12	14	1217	1220	1264	BOS
96975	12	21	NA	1220	1264	BOS
98415	12	28	1237	1220	1264	BOS

I picked one of the tail numbers of these mystery late flights, and saw that it frequents the AUS-BOS route but does not necessarily always park overnight in AUS or BOS. As shown below, N187JB came in very late on 11/30 which was unusual and perhaps had to be moved for its assignments the next day.

```
tail = 'N187JB'
checktail <- abia[abia$TailNum == tail,]
checktail1 <- checktail[checktail$Month == 11
                                    & checktail$DayofMonth >= 20,]
checktail2 <- checktail[checktail$Month == 12
                                    & checktail$DayofMonth <= 10,]
cat(pander(checktail1[,c(2,3,6,8,17,18)], style = 'rmarkdown'))
```

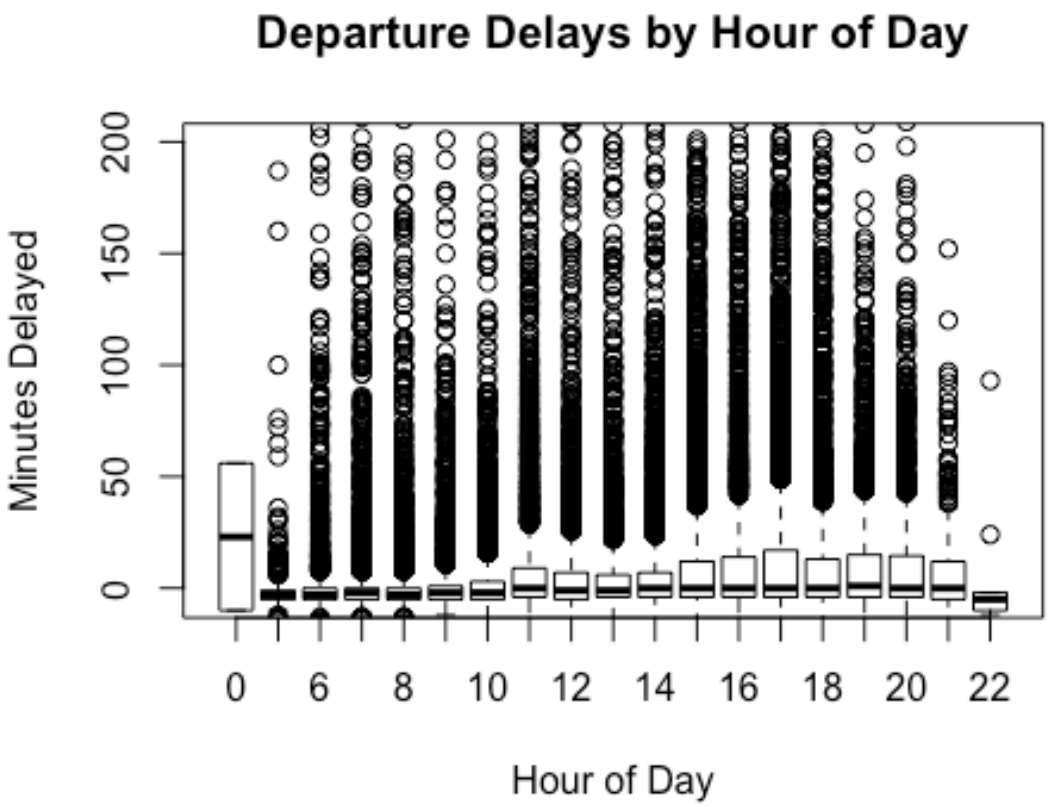
	Month	DayofMonth	CRSDepTime	CRSArrTime	Origin	Dest
90969	11	26	800	1141	BOS	AUS
91034	11	26	1220	1711	AUS	BOS
91594	11	29	925	1252	JFK	AUS
91649	11	29	1330	1814	AUS	JFK
91994	11	30	2005	2344	JFK	AUS

```
cat(pander(checktail2[,c(2,3,6,8,17,18)], style = 'rmarkdown'))
```

	Month	DayofMonth	CRSDepTime	CRSArrTime	Origin	Dest
92013	12	1	55	542	AUS	BOS
92309	12	2	800	1141	BOS	AUS
92365	12	2	1220	1711	AUS	BOS
92553	12	3	800	1141	BOS	AUS
92615	12	3	1220	1711	AUS	BOS
92866	12	4	1300	1617	JFK	AUS
92932	12	4	1655	2138	AUS	JFK

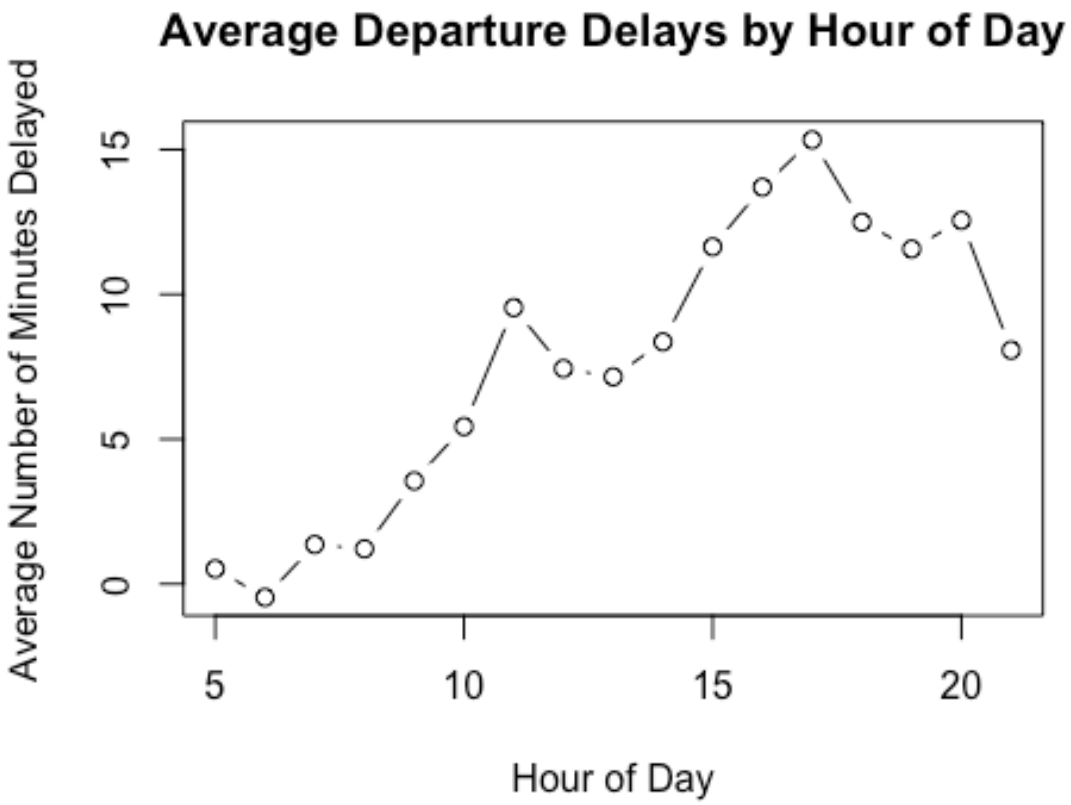
Back on topic, let's look at the departure delays by time of day.

```
boxplot(DepDelay~CRSDepHour, data = departures,
        main = 'Departure Delays by Hour of Day',
        xlab = 'Hour of Day', ylab = 'Minutes Delayed',
        ylim = c(-5,200))
```



The box part of the Departure Delays box plot reveals that delays are mostly minimal. Many significant delays do exist, hence the outliers. Disregarding the insufficient sample in the late hours, peak delays occur at 5-6pm.

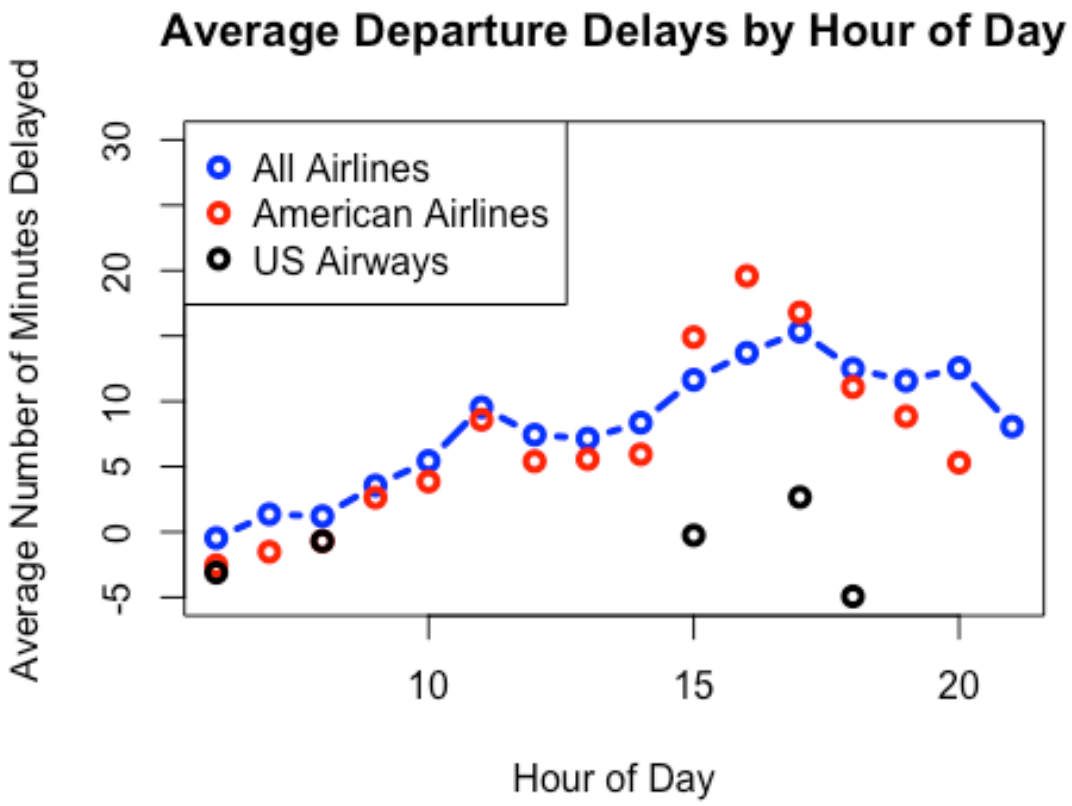
```
delay = aggregate(. ~ departures$CRSDepHour, departures['DepDelay'],
mean)
delay = delay[-c(1, 19),]
plot(delay, type = "b",
     main = 'Average Departure Delays by Hour of Day',
     xlab = 'Hour of Day', ylab = 'Average Number of Minutes Delayed')
```



If we were to only look at averages (see graph above), the peak late afternon / evening is easier to spot.

```
AAdepartures <- departures[departures$UniqueCarrier == 'AA',]
USdepartures <- departures[departures$UniqueCarrier == 'US',]
AAdelay = aggregate(AAdepartures$DepDelay,
                    by = list(AAdepartures$CRSDepHour),
                    FUN = mean, na.rm= TRUE)
USdelay = aggregate(USdepartures$DepDelay,
                    by = list(USdepartures$CRSDepHour),
                    FUN = mean, na.rm= TRUE)
delay = delay[-c(1),]
delay$'Group.1' = delay$`departures$CRSDepHour`
delay$`departures$CRSDepHour` = NULL
delayAA <- merge(x=delay, y=AAdelay,by='Group.1',all.x=TRUE)
delayAAUS <- merge(x=delayAA, y=USdelay,by='Group.1',all.x=TRUE)
plot(delayAAUS$Group.1, delayAAUS$DepDelay, type = "b",
     ylim = c(-5,30), xlab = "Hour of Day",
     ylab = "Average Number of Minutes Delayed",
     col = "blue", lwd = 3,
     main = "Average Departure Delays by Hour of Day" )
lines(delayAAUS$Group.1, delayAAUS$'x.x',
```

```
      type = "p", col = "red", lwd = 3)
lines(delayAAUS$Group.1, delayAAUS$'x.y',
      type = "p", col = "black", lwd = 3)
legend ("topleft",
        c("All Airlines", "American Airlines", "US Airways"),
        pch = 1, pt.lwd =3, col = c('blue','red', 'black'))
```



I have a lot of miles with US Airways, which recently completed its merger with American Airlines. The graph above helps me to see that there is an average delay of up to 20 minutes, peaking between 4-6pm. With the AA network, I have longer average delays out of ABIA at peak hours but I am gaining more travel options compared to the limited flights/hours of US Airways.

Author Attribution

For this question, I will try Naive Bayes and Random Forest. Due to the complexity of the dataset, intuition tells me that Random Forest will win. First, I have to construct the training and validation corpra like we did in class, and build a dictionary using the training document-term matrix to apply on the validation corpus.

```
author_dirs = Sys.glob('../data/ReutersC50/C50train/*')
file_list = NULL
labels = NULL
for(author in author_dirs) {
    author_name = substring(author, first=30)
    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)))
}
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en')}
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))
training_corpus = Corpus(VectorSource(all_docs))
names(training_corpus) = labels
```

So far I have loaded the file names and authors from C50train, and truncated the ".txt" from the file names. The training_corpus has the clean file names and authors, initialzed using the Corpus function from tm.

```
training_corpus = tm_map(training_corpus, content_transformer(tolower))
training_corpus = tm_map(training_corpus,
content_transformer(removeNumbers))
training_corpus = tm_map(training_corpus,
content_transformer(removePunctuation))
training_corpus = tm_map(training_corpus,
content_transformer(stripWhitespace))
training_corpus = tm_map(training_corpus,
content_transformer(removeWords), stopwords("SMART"))
training_dtm = DocumentTermMatrix(training_corpus)
training_dtm99 = removeSparseTerms(training_dtm, 0.99)
training_df = as.data.frame(inspect(training_dtm99)) # for NB
training_mt = as.matrix(training_dtm99) # for RF

training_dtm99
```

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

```
dim(training_df) # should come out to 3076 columns
```

```
## [1] 2500 3076
```

```
training_dict = dimnames(training_dtm99)[[2]] # the non-trivial step
```

The code chunck above cleanses the terms into words for a document-term matrix. I then removed the sparse terms. Next, I repeat these steps to make another document-term matrix based on the testing corpus. It's common practice for the testing to be based on training words.

```
author_dirs = Sys.glob('../data/ReutersC50/C50test/*')
file_list = NULL
testing_labels = NULL
for(author in author_dirs) {
    author_name = substring(author, first=30)
    files_to_add = Sys.glob(paste0(author, '/*.txt'))
    file_list = append(file_list, files_to_add)
    testing_labels = append(testing_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))
testing_corpus = Corpus(VectorSource(all_docs))
names(testing_corpus) = labels
testing_corpus = tm_map(testing_corpus, content_transformer(tolower))
testing_corpus = tm_map(testing_corpus,
content_transformer(removeNumbers))
testing_corpus = tm_map(testing_corpus,
content_transformer(removePunctuation))
testing_corpus = tm_map(testing_corpus,
content_transformer(stripWhitespace))
testing_corpus = tm_map(testing_corpus,
content_transformer(removeWords), stopwords("SMART"))
testing_dtm = DocumentTermMatrix(testing_corpus,
                                 list(dictionary=training_dict)) # the
non-trivial step
testing_dtm99 = removeSparseTerms(testing_dtm, 0.99)
testing_df = as.data.frame(inspect(testing_dtm99)) # for NB
testing_mt = as.matrix(testing_dtm99) # for RF

testing_dtm99
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 2727)>>
## Non-/sparse entries: 305489/6512011
## Sparsity           : 96%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
dim(testing_df) # should come out to 2727 columns
```

```
## [1] 2500 2727
```

Now it is time to run my training_df and testing_df into models for out-of-sample validation!

```
nbayes = naiveBayes(x = training_df,
                    y = as.factor(labels), laplace=1)
nbayes_predict = predict(nbayes,
                         newdata = testing_df)
confusionMatrix(table(rforest_predict,
                      testing_labels))$overall[1]
```

```
##     Accuracy
##    0.3009000
```

The accuracy of Naive Bayes seems less than desirable.

```
col_count = data.frame(testing_mt[,intersect(colnames(training_mt),
colnames(testing_mt))])
col_names = read.table(textConnection(""), col.names =
colnames(training_mt), colClasses = "integer")
testing_bound = rbind.fill(col_count, col_names)
```

Random Forest will not run unless testing matrix width is equal to that of the training matrix. The new dataframe testing_bound matches up to the number of columns (3076) in the training_df.

```
rforest = randomForest(x = training_mt,
                       y = as.factor(labels),
                       mtry = 4, ntree = 200)
rforest_predict = predict(rforest,
                          data = testing_bound)
confusionMatrix(table(rforest_predict,
                      testing_labels))$overall[1]
```

```
##     Accuracy
##    0.7114000
```

My model "rforest" is around 71% accurate when classifying the author using 200 trees.

Practice with Association Rule Mining

For this question, I will adjust confidence and support of apriori() from package arules to find a good number of association rules. I will then sort/filter by lift to narrow down on a few associations that appear most frequently. The final associations will be displayed on a 3D or 4D graph but using only two axes, per the graphing best practice taught by Dr. Scott.

```
library(arules)
library(arulesViz)
download.file("https://raw.githubusercontent.com/jgscott/STA380/master/
data/groceries.txt",
              destfile = "groceries.txt", method = "curl")
groceries <- read.transactions("groceries.txt",
                               format = 'basket', sep = ',')
rules <- apriori(groceries,
                 parameter = list(supp=.001,
                                  conf=.4))
inspect(rules)
```

I started with a minimum support of .01 but there weren't enough rules, and almost everything pointed toward whole milk. Now I see why supermarkets make milk cheap and put it all the way in the back. They know you're gonna buy milk, so it lures you in. At support of .001 and confidence of .5, there were sufficient rules (5000+) but not enough variety in RHS as I would have liked. In order to find other RHS items that have high lift, I decreased the confidence to .4 which provided 8955 rules at a support of .002. The highest lift associations are vegetables that lead to buying wine, ham, snacks and sugar -- the mom's basket.

```
plot(rules, method="grouped", control=list(k=20))
```


Grouped matrix for 8955 rules
size: support
color: lift
LHS
{tropical fruit, +48 items} - 153 rules
{other vegetables, +87 items} - 735 rules
{other vegetables, +109 items} - 877 rules
{citrus fruit, +27 items} - 96 rules
{whole milk, +24 items} - 136 rules
{bottled water, +54 items} - 203 rules
{other vegetables, +37 items} - 322 rules
{whole milk, +20 items} - 68 rules
{root vegetables, +74 items} - 1203 rules
{whole milk, +92 items} - 989 rules
{tropical fruit, +60 items} - 1032 rules
{root vegetables, +25 items} - 142 rules
{other vegetables, +33 items} - 251 rules
{other vegetables, +55 items} - 308 rules
{other vegetables, +63 items} - 416 rules
{whole milk, +52 items} - 414 rules
{yogurt, +45 items} - 410 rules
{other vegetables, +46 items} - 453 rules
{other vegetables, +53 items} - 348 rules
{whole milk, +78 items} - 399 rules
RHS
{red/blush wine}
{ham}
{salty snack}
{hamburger meat}
{sugar}
{cream cheese }
{white bread}
{chicken}
{curd}
{beef}
{frozen vegetables}
{chocolate}
{butter}
{margarine}
{frankfurter}
{domestic eggs}
{brown bread}
{whipped/sour cream}
{bottled beer}
{fruit/vegetable juice}
{pip fruit}
{citrus fruit}
{pastry}
{sausage}
{tropical fruit}
{root vegetables}
{shopping bags}
{bottled water}
{yogurt}
{other vegetables}
{soda}
{rolls/buns}
{whole milk}

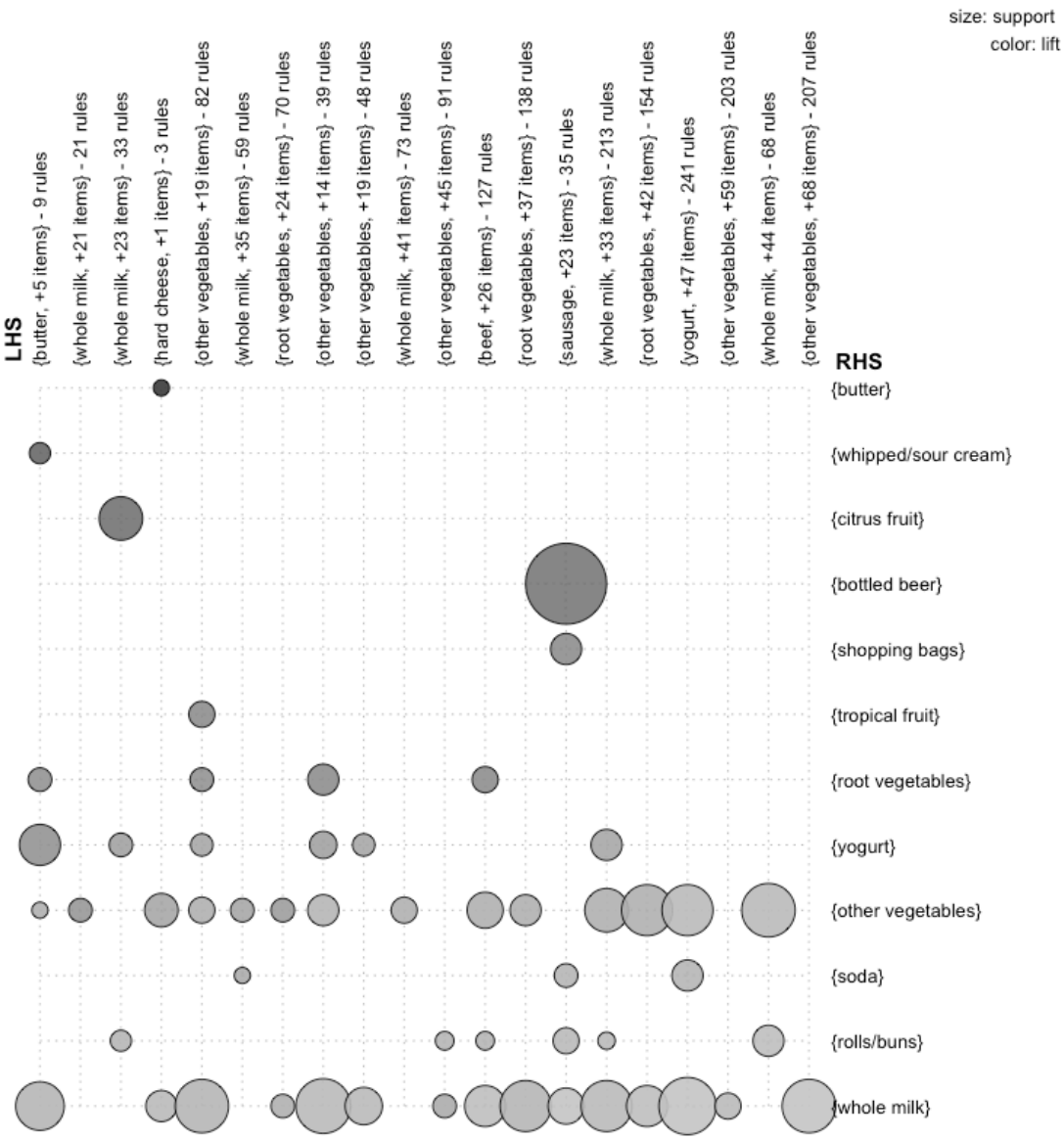
There are some interesting baskets. For example: hamberger meat and instant foods; wine and yogurt; white bread and processed cheese. The problem here is that you can't see for example, what the yogurt is doing with fruits, because of too many rules and lower average lift. All you see is that yogurt and eggs are related to everything.

```
rules <- apriori(groceries,
                 parameter = list(supp=.002,
                                  conf=.4))

##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen
maxlen
##         0.4    0.1    1 none FALSE            TRUE   0.002      1
10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [147 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 done [0.01s].
## writing ... [1914 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].

plot(rules, method="grouped", control=list(k=20))
```

Grouped matrix for 1914 rules



Then, I tried confidence of .4 and support of .002 with 1914 rules. I see that butter, cheese, and whip cream go together. We still have bottled water and soda. Interestingly, people tend to pick up a shopping bag with sausage. Maybe people are embarrased about buying sausage? I also resolved the too-many-yogurt-associations problem as the RHS yogurt here is related to less number of product categories. Finally, there is beer, milk, rolls/buns, and sausage for what I call basic-breakfast type of baskets.

```
rules <- apriori(groceries,
                 parameter = list(supp=.0015,
                                  conf=.4))
```

Eventually, I decided to compromise at support of .0015 for more variety, and filtered to keep only the rules with lift higher than 3, for a total of 1130 rules.

```
rules = subset(rules, subset=lift > 3)
plot(rules, method="grouped", control=list(k=20))
```


Grouped matrix for 1130 rules
size: support
color: lift
LHS
{bottled beer, +1 items} - 1 rules
{processed cheese, +1 items} - 1 rules
{Instant food products, +1 items} - 1 rules
{ham, +2 items} - 2 rules
{flour, +1 items} - 2 rules
{tropical fruit, +22 items} - 27 rules
{root vegetables, +39 items} - 81 rules
{whole milk, +22 items} - 15 rules
{yogurt, +30 items} - 74 rules
{other vegetables, +21 items} - 66 rules
{other vegetables, +4 items} - 9 rules
{other vegetables, +26 items} - 87 rules
{other vegetables, +15 items} - 17 rules
{root vegetables, +60 items} - 186 rules
{other vegetables, +41 items} - 140 rules
{root vegetables, +25 items} - 91 rules
{pip fruit, +11 items} - 28 rules
{whole milk, +50 items} - 185 rules
{whole milk, +21 items} - 60 rules
{root vegetables, +38 items} - 57 rules
RHS
{red/blush wine}
{ham}
{hamburger meat}
{sugar}
{white bread}
{domestic eggs}
{bottled beer}
{butter}
{fruit/vegetable juice}
{whipped/sour cream}
{pip fruit}
{citrus fruit}
{shopping bags}
{sausage}
{tropical fruit}
{bottled water}
{root vegetables}
{yogurt}
{soda}
{other vegetables}
{rolls/buns}
{whole milk}