Assignment2

Question 1

Importing the data into a dataframe named airports: This data contains information about every commercial flight in 2008 that either departed from or landed at Austin-Bergstrom International Airport Creating deptime_hour and arrivaltime_hour from the departure time and arrival time mentioned in the dataset

```
airports= read.csv('C:/Users/Ramyasai/Desktop/Predictive modelling/STA380-master/STA380-master/data/ABIA.csv')
attach(airports)
airports$DepTime_hour=as.integer(DepTime/100)
airports$ArrTime_hour=as.integer(ArrTime/100)
names(airports)
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay" "DepTime_hour"
## [31] "ArrTime_hour"
```

Aggregating(Sum of) DepDelayy at DayOfWeek, DepTime_hour level

```
dep_agg<- aggregate(DepDelay~DayOfWeek+DepTime_hour,airports,FUN='sum')
```

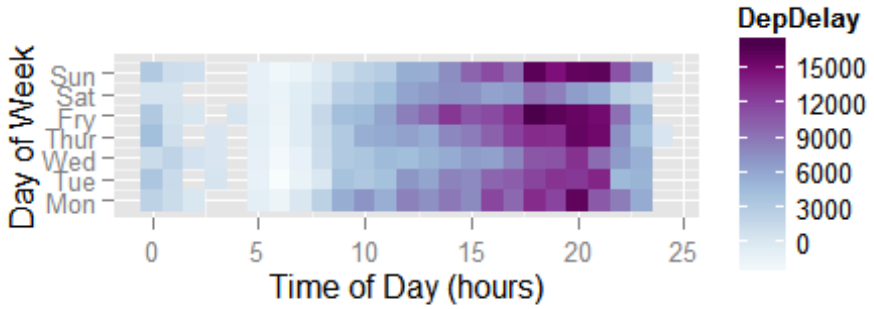
Importing ggplot and RColorBrewer libraries to enable plots

```
library(ggplot2)
library(RColorBrewer)
```

Plotting deptime_hour vs. day of week level

```
ggplot(dep_agg, aes(DepTime_hour,y=DayOfWeek))+
  geom_tile(aes(fill=DepDelay))+
  scale_fill_gradientn(colours=brewer.pal(9,"BuPu"),
                       breaks=seq(0,max(dep_agg$DepDelay),by=3000))+

scale_y_continuous(breaks=7:1,labels=c("Sun","Sat","Fry","Thur","Wed","Tue","Mon"))+
  labs(x="Time of Day (hours)", y="Day of Week")+ coord_fixed()
```



We can see from the plot that during the initial hours of the day for all days there is very less delay, Upto 10 hours there is not much delay

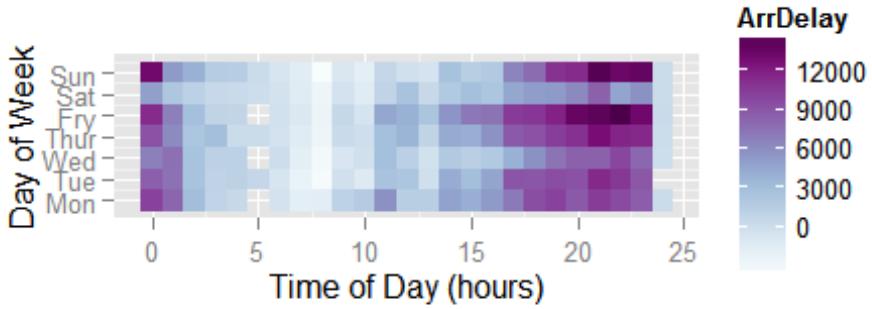
Similarly aggregating (Sum of) Arrival Delay at Day of week and Arrival Time level

```
arr_agg    <- aggregate(ArrDelay~DayOfWeek+ArrTime_hour,airports,FUN='sum')
```

Plotting arrival time aggregated at hourly level vs. day of week

```
ggplot(arr_agg, aes(ArrTime_hour,y=DayOfWeek))+
  geom_tile(aes(fill=ArrDelay))+
  scale_fill_gradientn(colours=brewer.pal(9,"BuPu"),
                       breaks=seq(0,max(arr_agg$ArrDelay),by=3000))+

scale_y_continuous(breaks=7:1,labels=c("Sun","Sat","Fry","Thur","Wed","Tue","
Mon"))+
  labs(x="Time of Day (hours)", y="Day of Week")+ coord_fixed()
```



We can see from the plot that arrival delay is quite less accross the early hours as well. ANd it is very high during sundays after 19 hrs.General trend it is high during the evening and nights

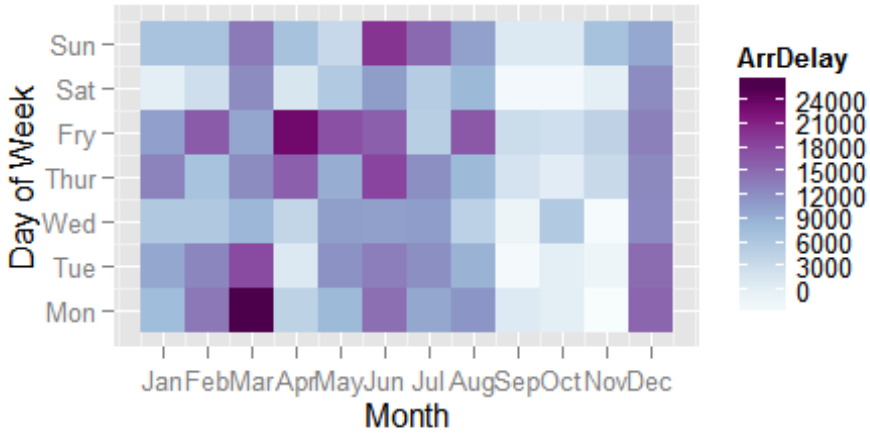
Aggregating ArrivalDelay at Dayofweek and month level

```
arr_agg_month    <- aggregate(ArrDelay~DayOfWeek+Month,airports,FUN='sum')
```

Plotting arrival delay's aggregated at month level vs. month and dayofweek

```
ggplot(arr_agg_month, aes(Month,y=DayOfWeek))+
  geom_tile(aes(fill=ArrDelay))+
  scale_fill_gradientn(colours=brewer.pal(9,"BuPu"),
                       breaks=seq(0,max(arr_agg_month$ArrDelay),by=3000))+

scale_y_continuous(breaks=7:1,labels=c("Sun","Sat","Fry","Thur","Wed","Tue","
Mon"))+
  scale_x_continuous(breaks=1:12,
labels=c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","D
ec"))+
  labs(x="Month", y="Day of Week")+ coord_fixed()
```



From the graph we can see that september, october and november are the months with relatively less arrival delays

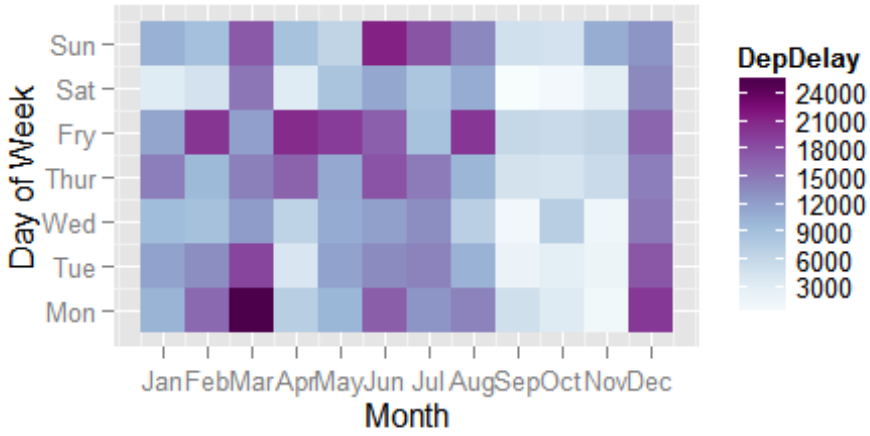
Aggregating DepDelay at DayofWeek and Month level

```
Dep_agg_month     <- aggregate(DepDelay~DayOfWeek+Month,airports,FUN='sum')
```

Plotting Aggregated departed delay vs. month and dayofweek

```
ggplot(Dep_agg_month, aes(Month,y=DayOfWeek))+
  geom_tile(aes(fill=DepDelay))+
  scale_fill_gradientn(colours=brewer.pal(9,"BuPu"),
                       breaks=seq(0,max(Dep_agg_month$DepDelay),by=3000))+

scale_y_continuous(breaks=7:1,labels=c("Sun","Sat","Fry","Thur","Wed","Tue","
Mon"))+ scale_x_continuous(breaks=1:12,
labels=c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","D
ec"))+
  labs(x="Month", y="Day of Week")+ coord_fixed()
```



Departure delay also follows similar trend like arrival delays, it is low during the september, october and november and quite high on sundays,fridays and mondays.

Question 2

We need to build two models to predict the author of the article based on the article's textual content. Let us start off with importing the files and creating the corpus Importing the necessary libraries

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.2.2
```

```
## Loading required package: NLP
```

```
## Warning: package 'NLP' was built under R version 3.2.2
```

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

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(e1071)
library(rpart)
library(ggplot2)
library(caret)

## Loading required package: lattice
```

Lets create the reader function which reads each of the text file through the readPlain function which is present in the tm library

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

TRAINING CORPUS Importing the training corpus and creating the training dataset Training dataset has text files written by 50 authors

```
author_dirs = Sys.glob('C:/Users/Ramyasai/Desktop/Predictive
modelling/STA380-master/STA380-master/data/ReutersC50/C50train/*')
file_list = NULL
train_labels = NULL
for(author in author_dirs) {
  author_name = substring(author, first=23)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  train_labels = append(train_labels, rep(author_name, length(files_to_add)))
}
```

Applying the reader function for each list and extracting the names and cleaning the files

```
# Named conversion & cleanup
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))
```

Initializing training corpus by passing all the docs in the training dataset and setting the names of the file list as the name of the corpus

```
train_corpus = Corpus(VectorSource(all_docs))
names(train_corpus) = file_list
```

Tokenization of training corpus based on the following steps 1. Converting all the txt into lowercase 2. Removing the numbers 3. Removing all the punctuation 4. Removing all the whitespaces 5. Removing all the words which we defined as the remove words and stop words

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

Creating training DocumentTermMatrix & dense matrix by removing the sparse terms and converting the documentTermMatrix to a matrix format

```
DTM_train = DocumentTermMatrix(train_corpus)
DTM_train = removeSparseTerms(DTM_train, 0.975)
DTM_train = as.matrix(DTM_train)
```

TESTING CORPUS

Now we are repeating the same procedure for creating the test corpus from the test data

```
author_dirs = Sys.glob('C:/Users/Ramyasai/Desktop/Predictive
modelling/STA380-master/STA380-master/data/ReutersC50/C50test/*')
file_list = NULL
test_labels = NULL
for(author in author_dirs) {
  author_name = substring(author, first=22)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  test_labels = append(test_labels, rep(author_name, length(files_to_add)))
}
```

Named conversion and cleanup followed by initializing the testing corpus and then tokenization of the test corpus with the same rules which we followed in the training corpus

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

test_corpus = Corpus(VectorSource(all_docs))
names(test_corpus) = file_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"))
```

Dictionary creation We need a dictionary of terms from the training corpus inorder to extract terms from the test corpus

```
reuters_dict = NULL
reuters_dict = dimnames(DTM_train)[[2]]
```

Creating testing DTM & matrix using dictionary words only

```
DTM_test = DocumentTermMatrix(test_corpus, list(dictionary=reuters_dict))
DTM_test = removeSparseTerms(DTM_test, 0.975)
DTM_test = as.matrix(DTM_test)
```

Convert DTMs into Data Frames for use in classifier models

```
DTM_train_df = as.data.frame((DTM_train))
#DTM_train$auth_name = train_labels
DTM_test_df = as.data.frame((DTM_test))
#DTM_test$auth_name = test_labels
```

Running Naive Bayes Model with the DTM_train dataframe

```
model_NB = naiveBayes(x=DTM_train_df, y=as.factor(train_labels), laplace=1)
```

Predicting the author names using the test data using the naive bayes function generated

```
pred_NB = predict(model_NB, DTM_test_df)
table_NB = as.data.frame(table(pred_NB,test_labels))
```

The accuracy in this case : 18.5%

RANDOM FORESTS:

Since the accuracy is quite low for the Naive Bayes model, we are implementing Random Forests model for the same. To deal with the words which apply in the training data but not in the test data , we are adding empty columns in the test data for those words. We are doing this because Random forests require same number of words in Training and test data sets

```
DTM_test = as.matrix(DTM_test)
DTM_train = as.matrix(DTM_train)

xx <- data.frame(DTM_test[,intersect(colnames(DTM_test),
colnames(DTM_train))])
yy <- read.table(textConnection(""), col.names = colnames(DTM_train),
colClasses = "integer")

library(plyr)
DTM_test_clean = rbind.fill(xx, yy)

DTM_test_df = as.data.frame(DTM_test_clean)
```

Running random forest on the training data withe number of trees as 200 and then number of columns to be selected each time is 3 Predicting the author names passing the test data into the model and creating table_RF which has predicted author names for the test data and the actual author names for the test data

```
model_RF = randomForest(x=DTM_train_df, y=as.factor(train_labels), mtry=3,
ntree=200)
pred_RF = predict(model_RF, data=DTM_test_clean)

table_RF = as.data.frame(table(pred_RF,test_labels))
```

The accuracy in this case using Random Forests is 69.4%

Random forests is better than Naive bayes in this case because of better accuracy and also because Random forests deals better with large data than Naive Bayes

Assumption: As stated above there are two ways to deal with the words which are present in the test set that are not present in the training set 1. Just taking the intersection of the words present in both training dataset and test dataset 2. Adding empty columns in the test dataset for those words which would ensure that we will only consider the common words

As Random Forests needs same number of variables in the test and training data set we need to do this additional computation, also we need to do this when we have extra words in the test dataset which are not present in the train dataset

Question 3:

Importing the groceries file

```
library(arules)
```

```
## Warning: package 'arules' was built under R version 3.2.2
```

```
## Loading required package: Matrix
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:tm':
##
##     inspect
##
## The following objects are masked from 'package:base':
##
##     %in%, write
```

```
groceries1<- read.transactions("C:/Users/Ramyasai/Desktop/Predictive
modelling/STA380-master/STA380-master/data/groceries.txt", format ="basket",
sep = ",",rm.duplicates = TRUE)
```

The data in groceries shows the various baskets and the items in each basket. Each row corresponds to a basket and the items brought in that basket

```
groceries <- apriori(groceries1, parameter=list(support=.01, confidence=.5,
maxlen=4))
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.5    0.1    1 none FALSE            TRUE    0.01      1      4
##  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 [0.01s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Passing the groceries file and checking the associatons between the products purchased together This will list the possible association rules with the support, confidence and lift for each of the association rule Looking at the output

```
inspect(groceries)
```

```
##    lhs                     rhs                    support confidence
lift
## 1  {curd,
##     yogurt}             => {whole milk}       0.01006609  0.5823529
2.279125
## 2  {butter,
##     other vegetables}   => {whole milk}       0.01148958  0.5736041
2.244885
## 3  {domestic eggs,
##     other vegetables}   => {whole milk}       0.01230300  0.5525114
2.162336
## 4  {whipped/sour cream,
##     yogurt}             => {whole milk}       0.01087951  0.5245098
2.052747
## 5  {other vegetables,
##     whipped/sour cream} => {whole milk}       0.01464159  0.5070423
1.984385
## 6  {other vegetables,
##     pip fruit}          => {whole milk}       0.01352313  0.5175097
2.025351
## 7  {citrus fruit,
##     root vegetables}    => {other vegetables} 0.01037112  0.5862069
3.029608
## 8  {root vegetables,
##     tropical fruit}     => {other vegetables} 0.01230300  0.5845411
3.020999
## 9  {root vegetables,
##     tropical fruit}     => {whole milk}       0.01199797  0.5700483
2.230969
## 10 {tropical fruit,
##     yogurt}             => {whole milk}       0.01514997  0.5173611
2.024770
```

```
## 11 {root vegetables,
##     yogurt}             => {other vegetables} 0.01291307  0.5000000
2.584078
## 12 {root vegetables,
##     yogurt}             => {whole milk}       0.01453991  0.5629921
2.203354
## 13 {rolls/buns,
##     root vegetables}    => {other vegetables} 0.01220132  0.5020921
2.594890
## 14 {rolls/buns,
##     root vegetables}    => {whole milk}       0.01270971  0.5230126
2.046888
## 15 {other vegetables,
##     yogurt}             => {whole milk}       0.02226741  0.5128806
2.007235
```

This generated 15 association rules with various support, confidence and lift values

To understand the various associations, we need to look the support, confidence and lift values - A lift value greater than 1 indicates that X and Y appear more often together than expected; this means that the occurrence of X has a positive effect on the occurrence of Y or that X is positively correlated with Y

Selecting the association rules with various lift values - Citrus fruits, root vegetables and other vegetables are highly affined with lift 3.02 - Root vegetables, tropical fruit and other vegetables are highly affined with lift 3.02

```
inspect(subset(groceries, subset=lift > 3))
```

```
##   lhs                   rhs                     support confidence     lift
## 1 {citrus fruit,
##    root vegetables} => {other vegetables} 0.01037112  0.5862069 3.029608
## 2 {root vegetables,
##    tropical fruit}  => {other vegetables} 0.01230300  0.5845411 3.020999
```

Filtering for various support, confidence and lift values to identify interesting associations of various products To identify most frequently occuring associations we need to check support followed by confidence because less support means that assocation is very less important(across all the baskets) though it might have high confidence So to identify the high frequent associations we first put a filter on support and then check the confidence and lift for those associations

Support of an item or item set is the fraction of transactions in our data set that contain that item or item set. In general, it is nice to identify rules that have a high support, as these will be applicable to a large number of transactions

Support is very low for the rules generated, most of it is around 0.01-0.02 range. One association with support 0.0222 is mentioned below

```
inspect(subset(groceries,subset=support> 0.02))
```

```
##   lhs                    rhs             support confidence     lift
## 1 {other vegetables,
##    yogurt}           => {whole milk} 0.02226741  0.5128806 2.007235
```

- This shows that other vegetables, yogurt ang whole milk have higher product affinity

confidence of a rule is the likelihood that it is true for a new transaction that contains the items on the LHS of the rule. (I.e. it is the probability that the transaction also contains the item(s) on the RHS.)

3 rules are identified with confidence > 0.58 which are: 1. Curd, yogurt and whole milk 2. citrus fruit, root vegetables and other vegetables 3. root vegetables, tropical fruit and other vegetables

```
inspect(subset(groceries, subset=confidence > 0.58))
```

```
##   lhs                   rhs                    support confidence     lift
## 1 {curd,
##    yogurt}           => {whole milk}       0.01006609  0.5823529 2.279125
## 2 {citrus fruit,
##    root vegetables} => {other vegetables} 0.01037112  0.5862069 3.029608
## 3 {root vegetables,
##    tropical fruit}  => {other vegetables} 0.01230300  0.5845411 3.020999
```

Now we will indetify rules with combinations of support, confidence and lift values

```
inspect(subset(groceries, subset=support > 0.02 & confidence >=0.4))
```

```
##   lhs                    rhs             support confidence     lift
## 1 {other vegetables,
##    yogurt}           => {whole milk} 0.02226741  0.5128806 2.007235
```

Other vegetables, yogurt and whole milk are highly affined which is quite expected. Customers who come to purchase few items(smaller baskets) make this kind of purchases.

```
inspect(subset(groceries, subset=support > 0.01 & confidence >=0.3))
```

```
##    lhs                      rhs                    support confidence
lift
## 1  {curd,
##     yogurt}              => {whole milk}       0.01006609  0.5823529
2.279125
## 2  {butter,
##     other vegetables}    => {whole milk}       0.01148958  0.5736041
2.244885
## 3  {domestic eggs,
##     other vegetables}    => {whole milk}       0.01230300  0.5525114
2.162336
## 4  {whipped/sour cream,
##     yogurt}              => {whole milk}       0.01087951  0.5245098
2.052747
## 5  {other vegetables,
```

```
##     whipped/sour cream} => {whole milk}       0.01464159  0.5070423
1.984385
## 6  {other vegetables,
##     pip fruit}          => {whole milk}       0.01352313  0.5175097
2.025351
## 7  {citrus fruit,
##     root vegetables}    => {other vegetables} 0.01037112  0.5862069
3.029608
## 8  {root vegetables,
##     tropical fruit}     => {other vegetables} 0.01230300  0.5845411
3.020999
## 9  {root vegetables,
##     tropical fruit}     => {whole milk}       0.01199797  0.5700483
2.230969
## 10 {tropical fruit,
##     yogurt}             => {whole milk}       0.01514997  0.5173611
2.024770
## 11 {root vegetables,
##     yogurt}             => {other vegetables} 0.01291307  0.5000000
2.584078
## 12 {root vegetables,
##     yogurt}             => {whole milk}       0.01453991  0.5629921
2.203354
## 13 {rolls/buns,
##     root vegetables}    => {other vegetables} 0.01220132  0.5020921
2.594890
## 14 {rolls/buns,
##     root vegetables}    => {whole milk}       0.01270971  0.5230126
2.046888
## 15 {other vegetables,
##     yogurt}             => {whole milk}       0.02226741  0.5128806
2.007235
```

Rules like curd, yogurt and whole milk are quite common

Some more interesting associations are 1. rolls/buns, root vegetables and whole milk 2. Root vegetables, troipcal fruits and whole milk 3. Vegetables ,whipped cream and sour cream

To summarise: Associations with high support/confidence/lift values are:

1. Curd, yogurt and whole milk
2. citrus fruit, root vegetables and other vegetables
3. root vegetables, tropical fruit and other vegetables
4. Vegetables ,whipped cream and sour cream
5. vegetables, yogurt and whole milk