

exercise2

```
airport <- read.csv("../data/ABIA.csv")
names(airport)
```

```
## [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"
```

```
#what are the bad airports to fly to
#aggregate by airport and see mean arrival delays
airport[is.na(airport)] <- 0
airport[is.null(airport)] <- 0
destair = aggregate(airport, by = list(airport$Dest), FUN = mean, na.rm = TRUE)
#grouped by destination and looking at arrival time delay and distance
df = cbind.data.frame(destair$ArrDelay, destair$Distance)
df = cbind.data.frame(df, destair$Group.1)
#df[is.na(df)] <- 0
#df[is.null(df)] <- 0
names(df) <- c("arrival_delay", "distance", "dest_airport")
head(df)
```

```
## arrival_delay distance dest_airport
## 1      3.4321839  619.0000      ABQ
## 2     13.9760213  813.0000      ATL
## 3      7.9645224  704.5998      AUS
## 4      1.9015152  756.0000      BNA
## 5      4.0869565 1698.0000      BOS
## 6     -0.1643836 1342.0000      BWI
```

```
order.delay <- order(df$arrival_delay, decreasing=TRUE)
order.distance <- order(df$distance, decreasing=TRUE)
df.delay = df[order.delay,]
df.distance = df[order.distance,]
#delay by airport
head(df.delay)
```

```
## arrival_delay distance dest_airport
## 13      100.00000      813      DSM
## 34      15.89091     1042      MSP
## 2       13.97602      813      ATL
## 16      13.53003     1504      EWR
## 14      13.00000     1149      DTW
## 28      12.11429     1226      LGB
```

The above shows the destination airports with the highest delay mean.

```
#training dataset to find model
library(tm)
```

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

```
library(plyr)
library(nnet)
#read in english files
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
              id=fname, language='en') }

#get authors from files
author_dirs = Sys.glob('../data/ReutersC50/C50train/*')
file_list = NULL
labels = NULL
for(author in author_dirs) {
  author_name = substring(author, first=29)
  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)))
}

# Need a more clever regex to get better names here
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

#make into corpus
my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list

# Preprocessing
my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove numbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) # remove punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords("SMART"))

DTM = DocumentTermMatrix(my_corpus)
class(DTM)
```

```
## [1] "DocumentTermMatrix"      "simple_triplet_matrix"
```

```
DTM = removeSparseTerms(DTM, 0.975)
X = as.matrix(DTM)
```

```
#get word probability vector for each author
# AP's multinomial probability vector
```

```

# Notice the smoothing factor
# Why?
smooth_count = 1/nrow(X)
w_All = rowsum(X + smooth_count, labels)
w_All = w_All/rowSums(w_All)
w_All = log(w_All)

#do the same for the test set without the last step of making probability vector

#get authors from files
author_dirs_test = Sys.glob('../data/ReutersC50/C50test/*')
file_list = NULL
labels = NULL
for(author in author_dirs_test) {
  author_name = substring(author, first=28)
  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)))
}

# Need a more clever regex to get better names here
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

#make into corpus
my_corpus_test = Corpus(VectorSource(all_docs))
names(my_corpus_test) = file_list

# Preprocessing
my_corpus_test = tm_map(my_corpus_test, content_transformer(tolower)) # make everything lowercase
my_corpus_test = tm_map(my_corpus_test, content_transformer(removeNumbers)) # remove numbers
my_corpus_test = tm_map(my_corpus_test, content_transformer(removePunctuation)) # remove punctuation
my_corpus_test = tm_map(my_corpus_test, content_transformer(stripWhitespace)) ## remove excess white-sp
my_corpus_test = tm_map(my_corpus_test, content_transformer(removeWords), stopwords("SMART"))

train_names_dict = NULL
train_names_dict = dimnames(DTM)[[2]]
#Create testing DTM & matrix using train words only
DTM_test = DocumentTermMatrix(my_corpus_test, list(dictionary=train_names_dict))
#DTM_test = removeSparseTerms(DTM_test, 0.975)

```

```

X_test = as.matrix(DTM_test)

#get the log probabilities of X_test using train naive bayes model
prob = X_test %*% t(w_All)
prediction<- colnames(prob)[apply(prob,1,which.max)]
#head(prob)
actual = labels

#make a final comparison matrix and find accuracy of prediction

compare =cbind.data.frame(actual,prediction)
compare2 = compare
compare2$same = 0.0
mask = (compare2$actual == compare2$prediction)
ind = which(mask %in% TRUE)
compare2$same[c(ind)] = 1.0
Per_accuracy = sum(compare2$same)/length(compare2$same)
final <- ddpdy(compare, .(actual), transform, sum.n = length(actual))
xtab<-ddply(final, .(actual, prediction), summarise, n = length(prediction), prop = n / sum.n[1] * 100)

head(compare2)

```

```

##          actual    prediction same
## 1 AaronPressman AaronPressman    1
## 2 AaronPressman AaronPressman    1
## 3 AaronPressman  SamuelPerry    0
## 4 AaronPressman AaronPressman    1
## 5 AaronPressman  SamuelPerry    0
## 6 AaronPressman AaronPressman    1

```

```

#second model PCA multinomial logistic regression
pc_Xtrain = prcomp(X, scale=TRUE)
pc_Xtest = prcomp(X_test, scale=TRUE)
K = 15
#rotation vectors
V_train = pc_Xtrain$rotation[,1:K]
#V_test = pc_Xtest$rotation[,1:K]

#get scores to model off of
scores_train = X %*% V_train
scores_test = predict(pc_Xtrain,X_test)

dftrain =cbind.data.frame(labels,scores_train)
dftest =cbind.data.frame(labels,scores_test)

#multinomial logistics regression

model <- multinom(labels ~., dftrain)

```

```

## # weights:  850 (784 variable)
## initial   value 9780.057514

```

```
## iter 10 value 5196.611030
## iter 20 value 4795.231273
## iter 30 value 4494.568843
## iter 40 value 4174.130979
## iter 50 value 3946.119237
## iter 60 value 3724.909552
## iter 70 value 3578.416096
## iter 80 value 3460.340411
## iter 90 value 3330.290474
## iter 100 value 3113.782079
## final value 3113.782079
## stopped after 100 iterations
```

```
#plot(pc_Xtrain)
#pc_Xtrain$sdev

prediction =predict(model,dftest)

pca_compare =cbind.data.frame(labels,prediction)
pca_compare$same = 0.0
mask = (pca_compare$labels == pca_compare$prediction)
ind = which(mask %in% TRUE)
pca_compare$same[c(ind)] = 1.0
pca_Per_accuracy = sum(pca_compare$same)/length(pca_compare$same)
head(pca_compare)
```

```
##      labels      prediction same
## 1 AaronPressman AaronPressman   1
## 2 AaronPressman AaronPressman   1
## 3 AaronPressman AaronPressman   1
## 4 AaronPressman AaronPressman   1
## 5 AaronPressman AaronPressman   1
## 6 AaronPressman AaronPressman   1
```

The models used were Naive Bayes and PCA multinomial logistic regression. The PCA method had a 32% accuracy on the test data while the Naive Bayes had a 61% accuracy. Sidenote: couldn't get the PCA to use more than 15 vectors for multinomial. Thus, Naive Bayes was a better method to use.

```
# Association rule mining
# Adapted from code by Matt Taddy
library(arules) # has a big ecosystem of packages built around it
```

```
## 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
```

```
library(reshape)
```

```
##
## Attaching package: 'reshape'
##
## The following object is masked from 'package:Matrix':
##
##     expand
##
## The following objects are masked from 'package:plyr':
##
##     rename, round_any
```

```
# Read in playlists from users
#grocery = read.table("../data/groceries.txt", header = FALSE, sep = ",", col.names = paste0("V",seq_len(ncol(grocery2))))
grocery2 <- file("../data/groceries.txt")
grocery = strsplit(readLines(grocery2),",")
close(grocery2)
```

```
#n = dim(grocery)[1]
#id = c(1:n)
#grocery = cbind(id,grocery)
#grocery$id <- factor(grocery$id)
#grocery = melt(grocery,id = 'id')
```

```
# First create a list of baskets: vectors of items by consumer
# Analagous to bags of words
```

```
# First split data into a list of artists for each user
#grocery2 <- split(x=grocery$value, f=grocery$id)
#head(grocery2)
## Remove duplicates ("de-dupe")
grocery <- lapply(grocery, unique)
```

```
## Cast this variable as a special arules "transactions" class.
grocery <- as(grocery, "transactions")
```

```
# Now run the 'apriori' algorithm
# Look at rules with support > .01 & confidence >.5 & length (# artists) <= 4
grocery2 <- apriori(grocery, parameter=list(support=.01, confidence=.2, maxlen=4))
```

```
##
## Parameter specification:
## confidence minval smax arem aval originalSupport support minlen maxlen
##      0.2    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 [0.00s].
## checking subsets of size 1 2 3 4 done [0.02s].
## writing ... [232 rule(s)] done [0.00s].
## creating S4 object ... done [0.00s].
```

```
# Look at the output
inspect(grocery2)
```

##	lhs	rhs	support	confidence	lift
## 1	{}	=> {whole milk}	0.25551601	0.2555160	1.0000000
## 2	{hard cheese}	=> {whole milk}	0.01006609	0.4107884	1.6076815
## 3	{butter milk}	=> {other vegetables}	0.01037112	0.3709091	1.9169159
## 4	{butter milk}	=> {whole milk}	0.01159126	0.4145455	1.6223854
## 5	{ham}	=> {whole milk}	0.01148958	0.4414062	1.7275091
## 6	{sliced cheese}	=> {whole milk}	0.01077783	0.4398340	1.7213560
## 7	{oil}	=> {whole milk}	0.01128622	0.4021739	1.5739675
## 8	{onions}	=> {other vegetables}	0.01423488	0.4590164	2.3722681
## 9	{onions}	=> {whole milk}	0.01209964	0.3901639	1.5269647
## 10	{berries}	=> {yogurt}	0.01057448	0.3180428	2.2798477
## 11	{berries}	=> {other vegetables}	0.01026945	0.3088685	1.5962805
## 12	{berries}	=> {whole milk}	0.01179461	0.3547401	1.3883281
## 13	{hamburger meat}	=> {other vegetables}	0.01382816	0.4159021	2.1494470
## 14	{hamburger meat}	=> {whole milk}	0.01474326	0.4434251	1.7354101
## 15	{hygiene articles}	=> {whole milk}	0.01281139	0.3888889	1.5219746
## 16	{salty snack}	=> {other vegetables}	0.01077783	0.2849462	1.4726465
## 17	{salty snack}	=> {whole milk}	0.01118454	0.2956989	1.1572618
## 18	{sugar}	=> {other vegetables}	0.01077783	0.3183183	1.6451186
## 19	{sugar}	=> {whole milk}	0.01504830	0.4444444	1.7393996
## 20	{waffles}	=> {other vegetables}	0.01006609	0.2619048	1.3535645
## 21	{waffles}	=> {whole milk}	0.01270971	0.3306878	1.2941961
## 22	{long life bakery product}	=> {other vegetables}	0.01067616	0.2853261	1.4746096
## 23	{long life bakery product}	=> {whole milk}	0.01352313	0.3614130	1.4144438
## 24	{dessert}	=> {other vegetables}	0.01159126	0.3123288	1.6141636
## 25	{dessert}	=> {whole milk}	0.01372649	0.3698630	1.4475140
## 26	{cream cheese }	=> {yogurt}	0.01240468	0.3128205	2.2424123
## 27	{cream cheese }	=> {other vegetables}	0.01372649	0.3461538	1.7889769
## 28	{cream cheese }	=> {whole milk}	0.01647178	0.4153846	1.6256696
## 29	{chicken}	=> {root vegetables}	0.01087951	0.2535545	2.3262206
## 30	{chicken}	=> {other vegetables}	0.01789527	0.4170616	2.1554393
## 31	{chicken}	=> {whole milk}	0.01759024	0.4099526	1.6044106
## 32	{white bread}	=> {soda}	0.01026945	0.2439614	1.3990437
## 33	{white bread}	=> {other vegetables}	0.01372649	0.3260870	1.6852681
## 34	{white bread}	=> {whole milk}	0.01708185	0.4057971	1.5881474
## 35	{chocolate}	=> {soda}	0.01352313	0.2725410	1.5629391
## 36	{chocolate}	=> {rolls/buns}	0.01179461	0.2377049	1.2923316
## 37	{chocolate}	=> {other vegetables}	0.01270971	0.2561475	1.3238103
## 38	{chocolate}	=> {whole milk}	0.01667514	0.3360656	1.3152427
## 39	{coffee}	=> {other vegetables}	0.01342145	0.2311734	1.1947400
## 40	{coffee}	=> {whole milk}	0.01870869	0.3222417	1.2611408

## 41	{frozen vegetables}	=> {root vegetables}	0.01159126	0.2410148	2.2111759
## 42	{frozen vegetables}	=> {yogurt}	0.01240468	0.2579281	1.8489235
## 43	{frozen vegetables}	=> {rolls/buns}	0.01016777	0.2114165	1.1494092
## 44	{frozen vegetables}	=> {other vegetables}	0.01779359	0.3699789	1.9121083
## 45	{frozen vegetables}	=> {whole milk}	0.02043721	0.4249471	1.6630940
## 46	{beef}	=> {root vegetables}	0.01738688	0.3313953	3.0403668
## 47	{beef}	=> {yogurt}	0.01169293	0.2228682	1.5976012
## 48	{beef}	=> {rolls/buns}	0.01362481	0.2596899	1.4118576
## 49	{beef}	=> {other vegetables}	0.01972547	0.3759690	1.9430662
## 50	{beef}	=> {whole milk}	0.02125064	0.4050388	1.5851795
## 51	{curd}	=> {root vegetables}	0.01087951	0.2041985	1.8734067
## 52	{curd}	=> {yogurt}	0.01728521	0.3244275	2.3256154
## 53	{curd}	=> {other vegetables}	0.01718353	0.3225191	1.6668288
## 54	{curd}	=> {whole milk}	0.02613116	0.4904580	1.9194805
## 55	{napkins}	=> {soda}	0.01199797	0.2291262	1.3139687
## 56	{napkins}	=> {yogurt}	0.01230300	0.2349515	1.6842183
## 57	{napkins}	=> {rolls/buns}	0.01169293	0.2233010	1.2140216
## 58	{napkins}	=> {other vegetables}	0.01443823	0.2757282	1.4250060
## 59	{napkins}	=> {whole milk}	0.01972547	0.3766990	1.4742678
## 60	{pork}	=> {root vegetables}	0.01362481	0.2363316	2.1682099
## 61	{pork}	=> {soda}	0.01189629	0.2063492	1.1833495
## 62	{pork}	=> {other vegetables}	0.02165735	0.3756614	1.9414764
## 63	{pork}	=> {whole milk}	0.02216573	0.3844797	1.5047187
## 64	{frankfurter}	=> {rolls/buns}	0.01921708	0.3258621	1.7716161
## 65	{frankfurter}	=> {other vegetables}	0.01647178	0.2793103	1.4435193
## 66	{frankfurter}	=> {whole milk}	0.02053889	0.3482759	1.3630295
## 67	{bottled beer}	=> {soda}	0.01698017	0.2108586	1.2092094
## 68	{bottled beer}	=> {other vegetables}	0.01616675	0.2007576	1.0375464
## 69	{bottled beer}	=> {whole milk}	0.02043721	0.2537879	0.9932367
## 70	{brown bread}	=> {yogurt}	0.01453991	0.2241379	1.6067030
## 71	{brown bread}	=> {other vegetables}	0.01870869	0.2884013	1.4905025
## 72	{brown bread}	=> {whole milk}	0.02521607	0.3887147	1.5212930
## 73	{margarine}	=> {yogurt}	0.01423488	0.2430556	1.7423115
## 74	{margarine}	=> {rolls/buns}	0.01474326	0.2517361	1.3686151
## 75	{margarine}	=> {other vegetables}	0.01972547	0.3368056	1.7406635
## 76	{margarine}	=> {whole milk}	0.02419929	0.4131944	1.6170980
## 77	{butter}	=> {root vegetables}	0.01291307	0.2330275	2.1378971
## 78	{butter}	=> {yogurt}	0.01464159	0.2642202	1.8940273
## 79	{butter}	=> {rolls/buns}	0.01342145	0.2422018	1.3167800
## 80	{butter}	=> {other vegetables}	0.02003050	0.3614679	1.8681223
## 81	{butter}	=> {whole milk}	0.02755465	0.4972477	1.9460530
## 82	{newspapers}	=> {rolls/buns}	0.01972547	0.2471338	1.3435934
## 83	{newspapers}	=> {other vegetables}	0.01931876	0.2420382	1.2508912
## 84	{newspapers}	=> {whole milk}	0.02735130	0.3426752	1.3411103
## 85	{domestic eggs}	=> {root vegetables}	0.01433655	0.2259615	2.0730706
## 86	{domestic eggs}	=> {yogurt}	0.01433655	0.2259615	1.6197753
## 87	{domestic eggs}	=> {rolls/buns}	0.01565836	0.2467949	1.3417510
## 88	{domestic eggs}	=> {other vegetables}	0.02226741	0.3509615	1.8138238
## 89	{domestic eggs}	=> {whole milk}	0.02999492	0.4727564	1.8502027
## 90	{fruit/vegetable juice}	=> {soda}	0.01840366	0.2545710	1.4598869
## 91	{fruit/vegetable juice}	=> {yogurt}	0.01870869	0.2587904	1.8551049
## 92	{fruit/vegetable juice}	=> {rolls/buns}	0.01453991	0.2011252	1.0934583
## 93	{fruit/vegetable juice}	=> {other vegetables}	0.02104728	0.2911392	1.5046529
## 94	{fruit/vegetable juice}	=> {whole milk}	0.02663955	0.3684951	1.4421604

## 95 {whipped/sour cream}	=> {root vegetables}	0.01708185	0.2382979	2.1862496
## 96 {whipped/sour cream}	=> {yogurt}	0.02074225	0.2893617	2.0742510
## 97 {whipped/sour cream}	=> {rolls/buns}	0.01464159	0.2042553	1.1104760
## 98 {whipped/sour cream}	=> {other vegetables}	0.02887646	0.4028369	2.0819237
## 99 {whipped/sour cream}	=> {whole milk}	0.03223183	0.4496454	1.7597542
## 100 {pip fruit}	=> {tropical fruit}	0.02043721	0.2701613	2.5746476
## 101 {pip fruit}	=> {root vegetables}	0.01555669	0.2056452	1.8866793
## 102 {pip fruit}	=> {yogurt}	0.01799695	0.2379032	1.7053777
## 103 {pip fruit}	=> {other vegetables}	0.02613116	0.3454301	1.7852365
## 104 {pip fruit}	=> {whole milk}	0.03009659	0.3978495	1.5570432
## 105 {pastry}	=> {soda}	0.02104728	0.2365714	1.3566647
## 106 {pastry}	=> {rolls/buns}	0.02094560	0.2354286	1.2799558
## 107 {pastry}	=> {other vegetables}	0.02257245	0.2537143	1.3112349
## 108 {pastry}	=> {whole milk}	0.03324860	0.3737143	1.4625865
## 109 {citrus fruit}	=> {tropical fruit}	0.01992883	0.2407862	2.2947022
## 110 {citrus fruit}	=> {root vegetables}	0.01769192	0.2137592	1.9611211
## 111 {citrus fruit}	=> {yogurt}	0.02165735	0.2616708	1.8757521
## 112 {citrus fruit}	=> {rolls/buns}	0.01677682	0.2027027	1.1020349
## 113 {citrus fruit}	=> {other vegetables}	0.02887646	0.3488943	1.8031403
## 114 {citrus fruit}	=> {whole milk}	0.03050330	0.3685504	1.4423768
## 115 {shopping bags}	=> {soda}	0.02460600	0.2497420	1.4321939
## 116 {shopping bags}	=> {other vegetables}	0.02318251	0.2352941	1.2160366
## 117 {shopping bags}	=> {whole milk}	0.02450432	0.2487100	0.9733637
## 118 {sausage}	=> {soda}	0.02430097	0.2586580	1.4833245
## 119 {sausage}	=> {yogurt}	0.01962379	0.2088745	1.4972889
## 120 {sausage}	=> {rolls/buns}	0.03060498	0.3257576	1.7710480
## 121 {sausage}	=> {other vegetables}	0.02694459	0.2867965	1.4822091
## 122 {sausage}	=> {whole milk}	0.02989324	0.3181818	1.2452520
## 123 {bottled water}	=> {soda}	0.02897814	0.2621895	1.5035766
## 124 {bottled water}	=> {yogurt}	0.02297916	0.2079117	1.4903873
## 125 {bottled water}	=> {rolls/buns}	0.02419929	0.2189512	1.1903734
## 126 {bottled water}	=> {other vegetables}	0.02480935	0.2244710	1.1601012
## 127 {bottled water}	=> {whole milk}	0.03436706	0.3109476	1.2169396
## 128 {tropical fruit}	=> {root vegetables}	0.02104728	0.2005814	1.8402220
## 129 {tropical fruit}	=> {yogurt}	0.02928317	0.2790698	2.0004746
## 130 {yogurt}	=> {tropical fruit}	0.02928317	0.2099125	2.0004746
## 131 {tropical fruit}	=> {rolls/buns}	0.02460600	0.2344961	1.2748863
## 132 {tropical fruit}	=> {other vegetables}	0.03589222	0.3420543	1.7677896
## 133 {tropical fruit}	=> {whole milk}	0.04229792	0.4031008	1.5775950
## 134 {root vegetables}	=> {yogurt}	0.02582613	0.2369403	1.6984751
## 135 {root vegetables}	=> {rolls/buns}	0.02430097	0.2229478	1.2121013
## 136 {root vegetables}	=> {other vegetables}	0.04738180	0.4347015	2.2466049
## 137 {other vegetables}	=> {root vegetables}	0.04738180	0.2448765	2.2466049
## 138 {root vegetables}	=> {whole milk}	0.04890696	0.4486940	1.7560310
## 139 {soda}	=> {rolls/buns}	0.03833249	0.2198251	1.1951242
## 140 {rolls/buns}	=> {soda}	0.03833249	0.2084024	1.1951242
## 141 {soda}	=> {whole milk}	0.04006101	0.2297376	0.8991124
## 142 {yogurt}	=> {rolls/buns}	0.03436706	0.2463557	1.3393633
## 143 {yogurt}	=> {other vegetables}	0.04341637	0.3112245	1.6084566
## 144 {other vegetables}	=> {yogurt}	0.04341637	0.2243826	1.6084566
## 145 {yogurt}	=> {whole milk}	0.05602440	0.4016035	1.5717351
## 146 {whole milk}	=> {yogurt}	0.05602440	0.2192598	1.5717351
## 147 {rolls/buns}	=> {other vegetables}	0.04260295	0.2316197	1.1970465
## 148 {other vegetables}	=> {rolls/buns}	0.04260295	0.2201787	1.1970465

## 149 {rolls/buns}	=> {whole milk}	0.05663447	0.3079049	1.2050318
## 150 {whole milk}	=> {rolls/buns}	0.05663447	0.2216474	1.2050318
## 151 {other vegetables}	=> {whole milk}	0.07483477	0.3867578	1.5136341
## 152 {whole milk}	=> {other vegetables}	0.07483477	0.2928770	1.5136341
## 153 {curd, yogurt}	=> {whole milk}	0.01006609	0.5823529	2.2791250
## 154 {curd, whole milk}	=> {yogurt}	0.01006609	0.3852140	2.7613555
## 155 {other vegetables, pork}	=> {whole milk}	0.01016777	0.4694836	1.8373939
## 156 {pork, whole milk}	=> {other vegetables}	0.01016777	0.4587156	2.3707136
## 157 {butter, other vegetables}	=> {whole milk}	0.01148958	0.5736041	2.2448850
## 158 {butter, whole milk}	=> {other vegetables}	0.01148958	0.4169742	2.1549874
## 159 {domestic eggs, other vegetables}	=> {whole milk}	0.01230300	0.5525114	2.1623358
## 160 {domestic eggs, whole milk}	=> {other vegetables}	0.01230300	0.4101695	2.1198197
## 161 {fruit/vegetable juice, other vegetables}	=> {whole milk}	0.01047280	0.4975845	1.9473713
## 162 {fruit/vegetable juice, whole milk}	=> {other vegetables}	0.01047280	0.3931298	2.0317558
## 163 {whipped/sour cream, yogurt}	=> {other vegetables}	0.01016777	0.4901961	2.5334096
## 164 {other vegetables, whipped/sour cream}	=> {yogurt}	0.01016777	0.3521127	2.5240730
## 165 {other vegetables, yogurt}	=> {whipped/sour cream}	0.01016777	0.2341920	3.2670620
## 166 {whipped/sour cream, yogurt}	=> {whole milk}	0.01087951	0.5245098	2.0527473
## 167 {whipped/sour cream, whole milk}	=> {yogurt}	0.01087951	0.3375394	2.4196066
## 168 {other vegetables, whipped/sour cream}	=> {whole milk}	0.01464159	0.5070423	1.9843854
## 169 {whipped/sour cream, whole milk}	=> {other vegetables}	0.01464159	0.4542587	2.3476795
## 170 {other vegetables, pip fruit}	=> {whole milk}	0.01352313	0.5175097	2.0253514
## 171 {pip fruit, whole milk}	=> {other vegetables}	0.01352313	0.4493243	2.3221780
## 172 {other vegetables, pastry}	=> {whole milk}	0.01057448	0.4684685	1.8334212
## 173 {pastry, whole milk}	=> {other vegetables}	0.01057448	0.3180428	1.6436947
## 174 {citrus fruit, root vegetables}	=> {other vegetables}	0.01037112	0.5862069	3.0296084
## 175 {citrus fruit, other vegetables}	=> {root vegetables}	0.01037112	0.3591549	3.2950455
## 176 {other vegetables, root vegetables}	=> {citrus fruit}	0.01037112	0.2188841	2.6446257
## 177 {citrus fruit, yogurt}	=> {whole milk}	0.01026945	0.4741784	1.8557678

## 178 {citrus fruit, ## whole milk}	=> {yogurt}	0.01026945	0.3366667	2.4133503
## 179 {citrus fruit, ## other vegetables}	=> {whole milk}	0.01301474	0.4507042	1.7638982
## 180 {citrus fruit, ## whole milk}	=> {other vegetables}	0.01301474	0.4266667	2.2050797
## 181 {other vegetables, ## sausage}	=> {whole milk}	0.01016777	0.3773585	1.4768487
## 182 {sausage, ## whole milk}	=> {other vegetables}	0.01016777	0.3401361	1.7578760
## 183 {bottled water, ## other vegetables}	=> {whole milk}	0.01077783	0.4344262	1.7001918
## 184 {bottled water, ## whole milk}	=> {other vegetables}	0.01077783	0.3136095	1.6207825
## 185 {root vegetables, ## tropical fruit}	=> {other vegetables}	0.01230300	0.5845411	3.0209991
## 186 {other vegetables, ## tropical fruit}	=> {root vegetables}	0.01230300	0.3427762	3.1447798
## 187 {other vegetables, ## root vegetables}	=> {tropical fruit}	0.01230300	0.2596567	2.4745380
## 188 {root vegetables, ## tropical fruit}	=> {whole milk}	0.01199797	0.5700483	2.2309690
## 189 {tropical fruit, ## whole milk}	=> {root vegetables}	0.01199797	0.2836538	2.6023653
## 190 {root vegetables, ## whole milk}	=> {tropical fruit}	0.01199797	0.2453222	2.3379305
## 191 {tropical fruit, ## yogurt}	=> {other vegetables}	0.01230300	0.4201389	2.1713431
## 192 {other vegetables, ## tropical fruit}	=> {yogurt}	0.01230300	0.3427762	2.4571457
## 193 {other vegetables, ## yogurt}	=> {tropical fruit}	0.01230300	0.2833724	2.7005496
## 194 {tropical fruit, ## yogurt}	=> {whole milk}	0.01514997	0.5173611	2.0247698
## 195 {tropical fruit, ## whole milk}	=> {yogurt}	0.01514997	0.3581731	2.5675162
## 196 {whole milk, ## yogurt}	=> {tropical fruit}	0.01514997	0.2704174	2.5770885
## 197 {rolls/buns, ## tropical fruit}	=> {whole milk}	0.01098119	0.4462810	1.7465872
## 198 {tropical fruit, ## whole milk}	=> {rolls/buns}	0.01098119	0.2596154	1.4114524
## 199 {other vegetables, ## tropical fruit}	=> {whole milk}	0.01708185	0.4759207	1.8625865
## 200 {tropical fruit, ## whole milk}	=> {other vegetables}	0.01708185	0.4038462	2.0871397
## 201 {other vegetables, ## whole milk}	=> {tropical fruit}	0.01708185	0.2282609	2.1753349
## 202 {root vegetables, ## yogurt}	=> {other vegetables}	0.01291307	0.5000000	2.5840778
## 203 {other vegetables, ## root vegetables}	=> {yogurt}	0.01291307	0.2725322	1.9536108
## 204 {other vegetables, ## yogurt}	=> {root vegetables}	0.01291307	0.2974239	2.7286977

## 205 {root vegetables, ## yogurt}	=> {whole milk}	0.01453991	0.5629921	2.2033536
## 206 {root vegetables, ## whole milk}	=> {yogurt}	0.01453991	0.2972973	2.1311362
## 207 {whole milk, ## yogurt}	=> {root vegetables}	0.01453991	0.2595281	2.3810253
## 208 {rolls/buns, ## root vegetables}	=> {other vegetables}	0.01220132	0.5020921	2.5948898
## 209 {other vegetables, ## root vegetables}	=> {rolls/buns}	0.01220132	0.2575107	1.4000100
## 210 {other vegetables, ## rolls/buns}	=> {root vegetables}	0.01220132	0.2863962	2.6275247
## 211 {rolls/buns, ## root vegetables}	=> {whole milk}	0.01270971	0.5230126	2.0468876
## 212 {root vegetables, ## whole milk}	=> {rolls/buns}	0.01270971	0.2598753	1.4128652
## 213 {rolls/buns, ## whole milk}	=> {root vegetables}	0.01270971	0.2244165	2.0588959
## 214 {other vegetables, ## root vegetables}	=> {whole milk}	0.02318251	0.4892704	1.9148326
## 215 {root vegetables, ## whole milk}	=> {other vegetables}	0.02318251	0.4740125	2.4497702
## 216 {other vegetables, ## whole milk}	=> {root vegetables}	0.02318251	0.3097826	2.8420820
## 217 {soda, ## yogurt}	=> {whole milk}	0.01047280	0.3828996	1.4985348
## 218 {soda, ## whole milk}	=> {yogurt}	0.01047280	0.2614213	1.8739641
## 219 {other vegetables, ## soda}	=> {whole milk}	0.01392984	0.4254658	1.6651240
## 220 {soda, ## whole milk}	=> {other vegetables}	0.01392984	0.3477157	1.7970490
## 221 {rolls/buns, ## yogurt}	=> {other vegetables}	0.01148958	0.3343195	1.7278153
## 222 {other vegetables, ## yogurt}	=> {rolls/buns}	0.01148958	0.2646370	1.4387534
## 223 {other vegetables, ## rolls/buns}	=> {yogurt}	0.01148958	0.2696897	1.9332351
## 224 {rolls/buns, ## yogurt}	=> {whole milk}	0.01555669	0.4526627	1.7715630
## 225 {whole milk, ## yogurt}	=> {rolls/buns}	0.01555669	0.2776770	1.5096478
## 226 {rolls/buns, ## whole milk}	=> {yogurt}	0.01555669	0.2746858	1.9690488
## 227 {other vegetables, ## yogurt}	=> {whole milk}	0.02226741	0.5128806	2.0072345
## 228 {whole milk, ## yogurt}	=> {other vegetables}	0.02226741	0.3974592	2.0541308
## 229 {other vegetables, ## whole milk}	=> {yogurt}	0.02226741	0.2975543	2.1329789
## 230 {other vegetables, ## rolls/buns}	=> {whole milk}	0.01789527	0.4200477	1.6439194
## 231 {rolls/buns, ## whole milk}	=> {other vegetables}	0.01789527	0.3159785	1.6330258

```
## 232 {other vegetables,
##     whole milk}          => {rolls/buns}          0.01789527  0.2391304 1.3000817
```

```
## Choose a subset
```

```
inspect(subset(grocery2, subset=support > .01 & confidence > 0.5 & lift>2))
```

##	lhs	rhs	support	confidence	lift
## 1	{curd,	=> {whole milk}	0.01006609	0.5823529	2.279125
##	yogurt}				
## 2	{butter,	=> {whole milk}	0.01148958	0.5736041	2.244885
##	other vegetables}				
## 3	{domestic eggs,	=> {whole milk}	0.01230300	0.5525114	2.162336
##	other vegetables}				
## 4	{whipped/sour cream,	=> {whole milk}	0.01087951	0.5245098	2.052747
##	yogurt}				
## 5	{other vegetables,	=> {whole milk}	0.01352313	0.5175097	2.025351
##	pip fruit}				
## 6	{citrus fruit,	=> {other vegetables}	0.01037112	0.5862069	3.029608
##	root vegetables}				
## 7	{root vegetables,	=> {other vegetables}	0.01230300	0.5845411	3.020999
##	tropical fruit}				
## 8	{root vegetables,	=> {whole milk}	0.01199797	0.5700483	2.230969
##	tropical fruit}				
## 9	{tropical fruit,	=> {whole milk}	0.01514997	0.5173611	2.024770
##	yogurt}				
## 10	{root vegetables,	=> {whole milk}	0.01453991	0.5629921	2.203354
##	yogurt}				
## 11	{rolls/buns,	=> {other vegetables}	0.01220132	0.5020921	2.594890
##	root vegetables}				
## 12	{rolls/buns,	=> {whole milk}	0.01270971	0.5230126	2.046888
##	root vegetables}				
## 13	{other vegetables,	=> {whole milk}	0.02226741	0.5128806	2.007235
##	yogurt}				

you can tell that whole milk was usually predicted.