Association rules

Monica Jain

March 26, 2018

About this Dataset Context: Random Shopping cart

Content: Date - to add register Id - transaction Product - for id transaction

Acknowledgements: The dataset is Random Shopping cart https://www.kaggle.com/fanatiks/shopping-cart

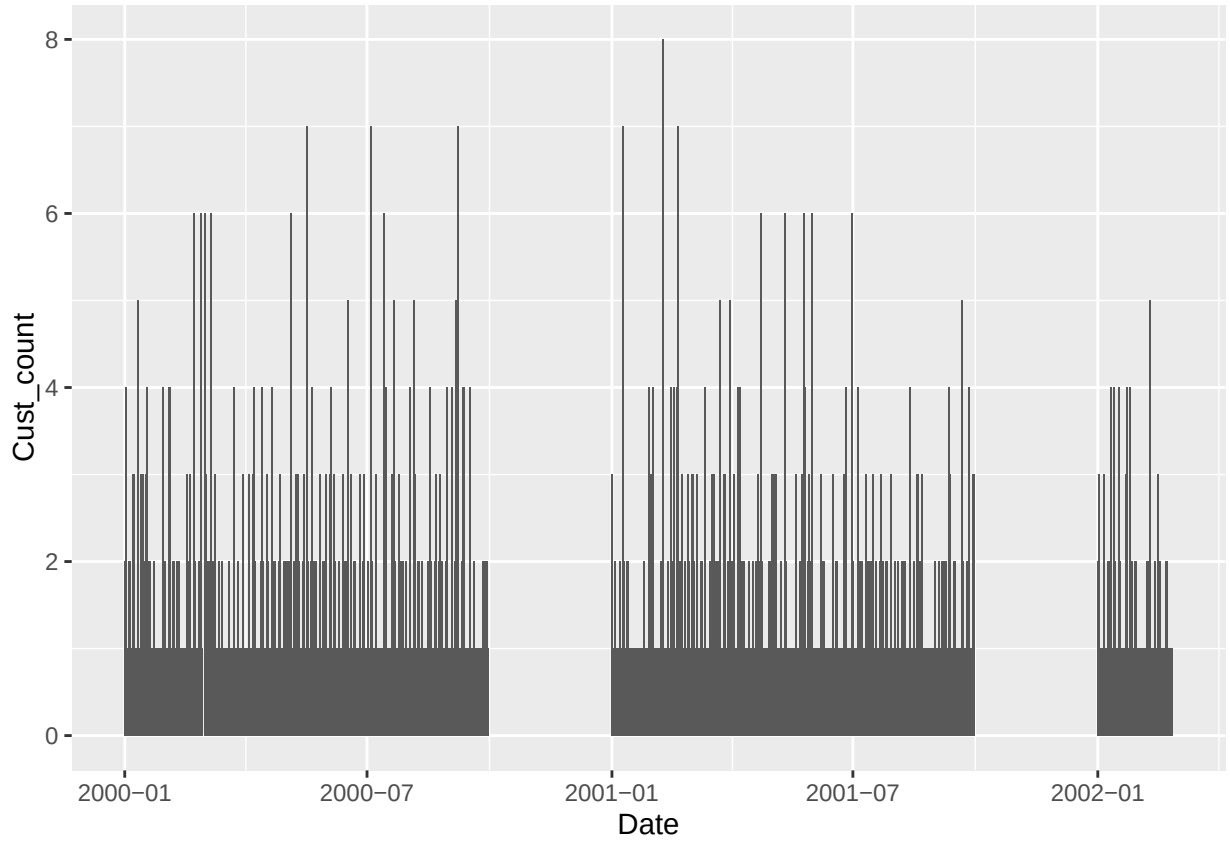
```
#Imorting the file with transactions data
shop_items <- read.csv("dataset_group.csv")
colnames(shop_items) <- c("Date","CustomerNo","Product")

#Analyzing the class of every column and changing the column format accordingly
str(shop_items)
```

```
## 'data.frame':    22342 obs. of  3 variables:
##  $ Date      : Factor w/ 603 levels "2000-01-01","2000-01-02",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ CustomerNo: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Product   : Factor w/ 38 levels "all- purpose",..: 25 27 20 1 12 31 5 36 4 2 ...
```

```
#Converting the date into date format
shop_items$Date <- as.Date(shop_items$Date)

#Since customers number is their identity, we factorise it
shop_items$CustomerNo <- as.factor(shop_items$CustomerNo)

#Analyzing the values to check for NA values or missing data
summary(shop_items)
```

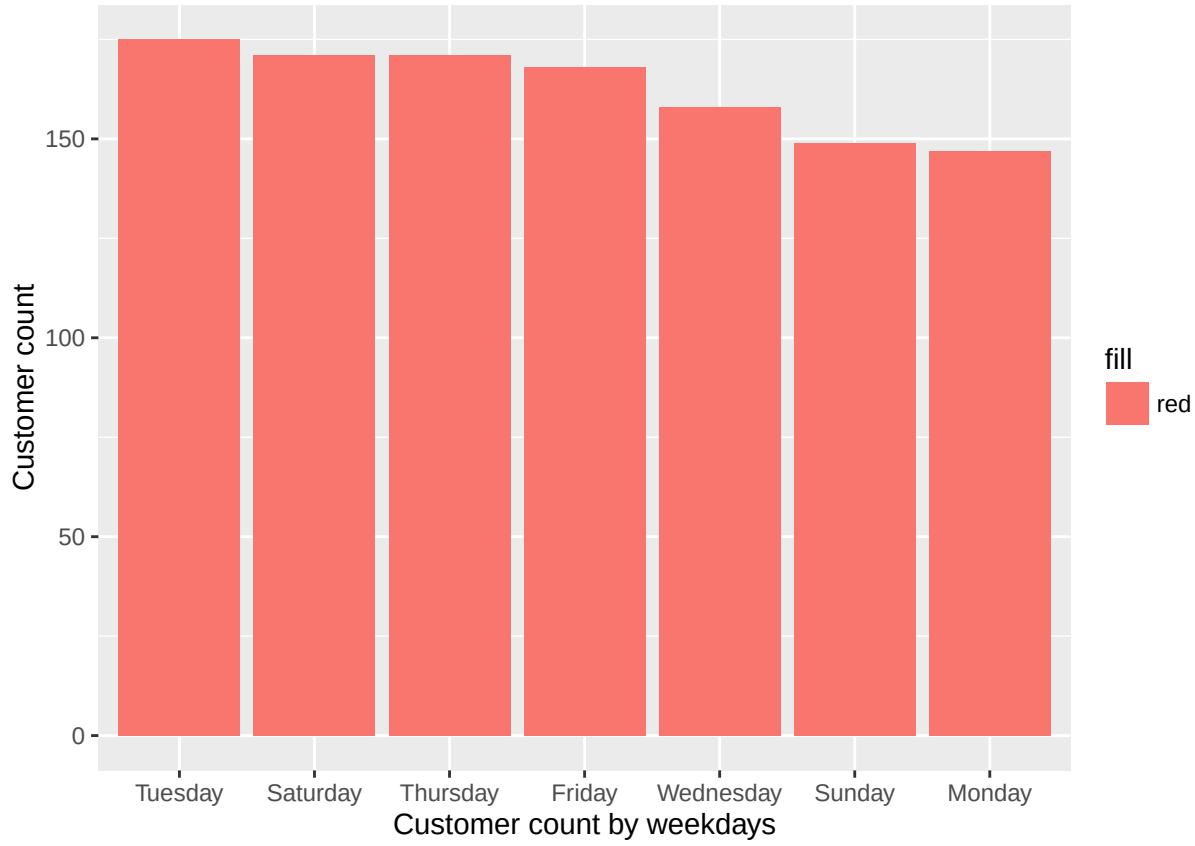
```
##       Date              CustomerNo          Product
##  Min.   :2000-01-01   10     :   34   vegetables: 1702
##  1st Qu.:2000-05-29   156    :   34   poultry   :  640
##  Median :2001-01-30   204    :   34   soda      :  597
##  Mean   :2000-12-21   226    :   34   cereals   :  591
##  3rd Qu.:2001-06-21   253    :   34   ice cream :  579
##  Max.   :2002-02-26   257    :   34   cheeses   :  578
##                       (Other):22138   (Other)   :17655
```

```
#Analyzing number of unique customers visiting on each date
count_cust <- shop_items %>% group_by(Date) %>% summarise(Cust_count = n_distinct(CustomerNo))
ggplot(count_cust) + geom_bar(aes(x=Date, y= Cust_count),stat = "identity")
```



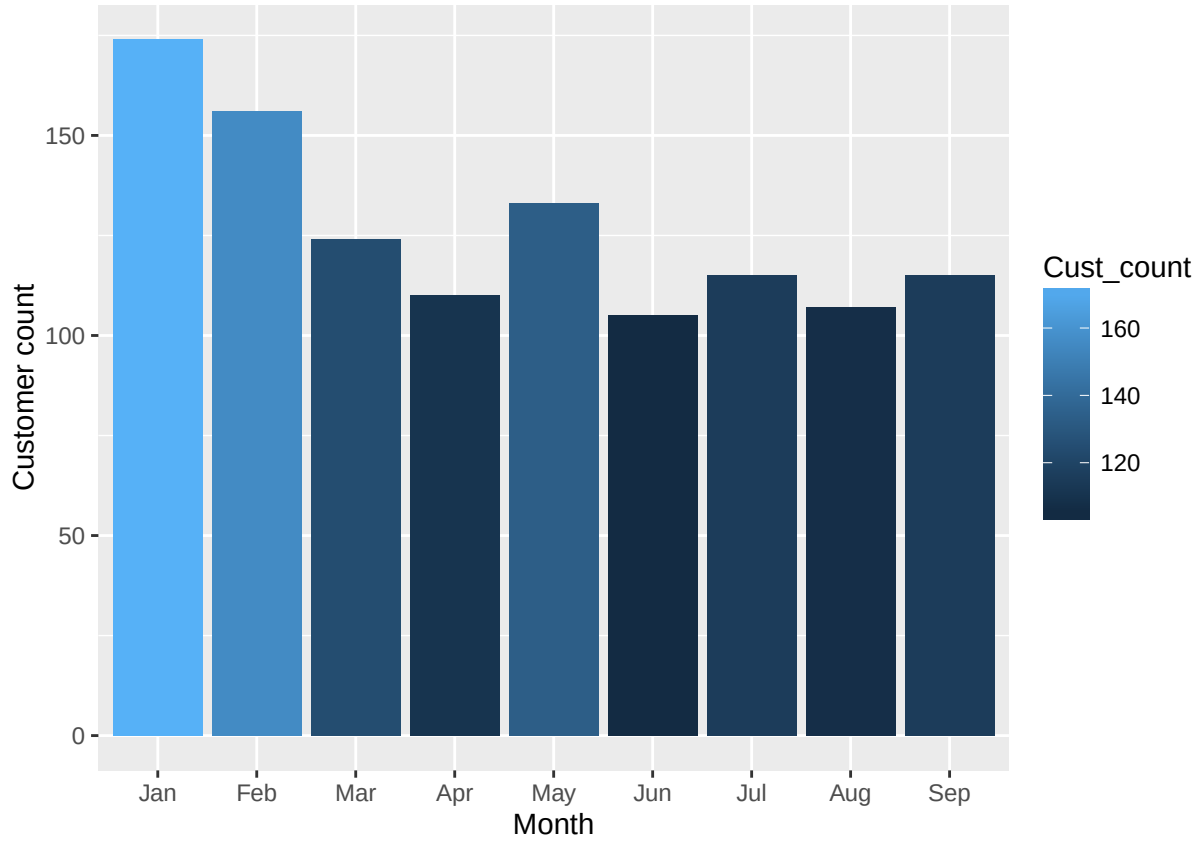
We cannot see any particular pattern from the above chart. We can see the data has missing date transaction. To inspect more, let us look into weekdays and months for any trend in the data.

```
#Adding day of the week to the data set using the Date column
shop_items$Day <- weekdays(shop_items$Date)
shop_items$Day <- as.factor(shop_items$Day)

#Analyzing customer count by weekdays and weekends
count_cust_day <- shop_items %>% group_by(Day) %>%
  summarise(Cust_count = n_distinct(CustomerNo)) %>%
                      arrange(desc(Cust_count))
ggplot(count_cust_day)+
  geom_bar(aes(x= reorder(Day,-Cust_count), y=Cust_count,fill="red"), stat="identity") +
  xlab("Customer count by weekdays") +
  ylab("Customer count")
```



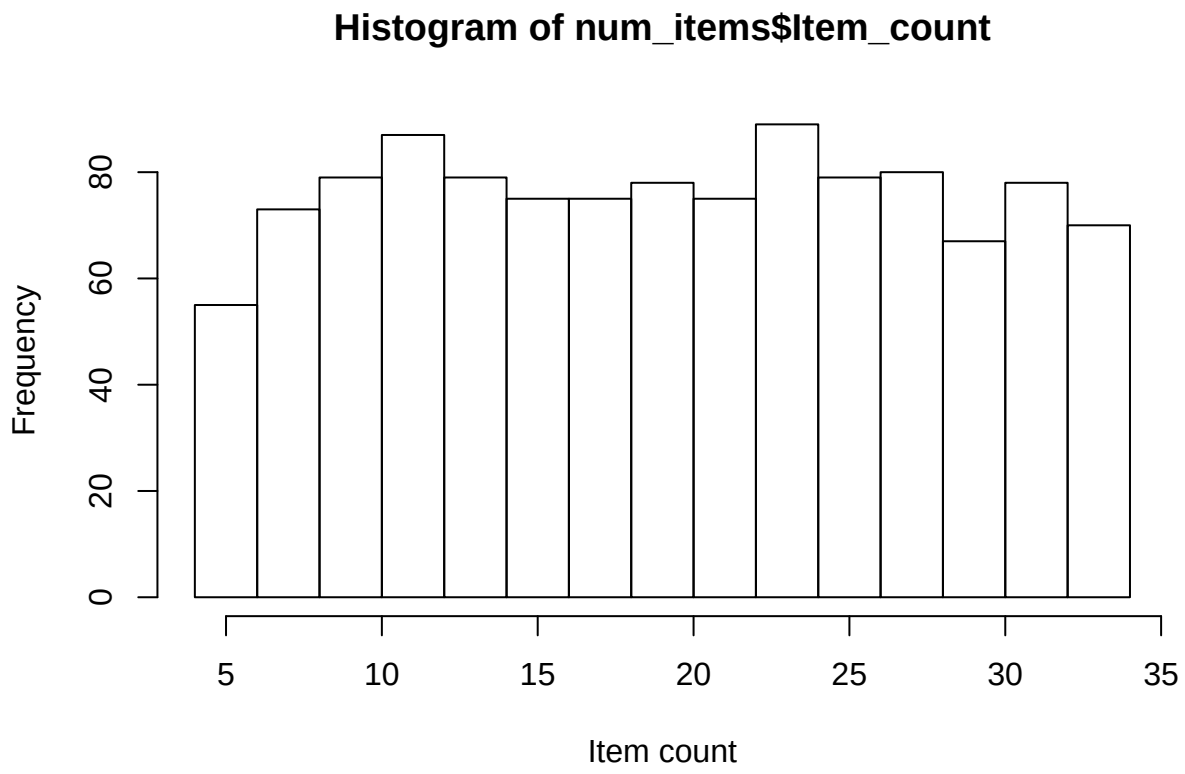
We can see Tuesdays and Saturdays have the highest customer trafic.

```
#Analyzing customer count by month
shop_items$Month <- month(shop_items$Date,label=TRUE, abbr = TRUE)
cust_count_month <- shop_items %>% group_by(Month) %>%
  summarise(Cust_count = n_distinct(CustomerNo))
ggplot(cust_count_month) +
  geom_bar(aes(x=Month, y= Cust_count, fill = Cust_count), stat="identity") +
  ylab("Customer count")
```



We can see that January has the highest customer count, as indicative by the colour shading as well.

```
#Analyzing the number of items purchased per customer
num_items <- shop_items %>% group_by(CustomerNo) %>% summarise(Item_count = n())
hist(num_items$Item_count, xlab = "Item count")
```



The frequency histogram shows that maximum customers buy items between 20 to 25.

```
#Applying association rules for inspecting the transaction rules
#Converting data into a binary transaction format
transaction <- dcast(shop_items, CustomerNo~ Product)
```

```
## Using Month as value column: use value.var to override.
```

```
## Aggregation function missing: defaulting to length
```

```
head(transaction[,1:6])
```

```
##   CustomerNo all- purpose aluminum foil bagels beef butter
## 1          1            3             1      0    1      1
## 2          2            0             1      0    0      0
## 3          3            0             0      1    0      0
## 4          4            1             0      0    0      0
## 5          5            1             0      0    0      0
## 6          6            1             1      3    0      2
```

```
#Converting the data into a matrix, removing the customer ID
transaction <- data.matrix(transaction[,-1])
head(transaction[,1:6])
```

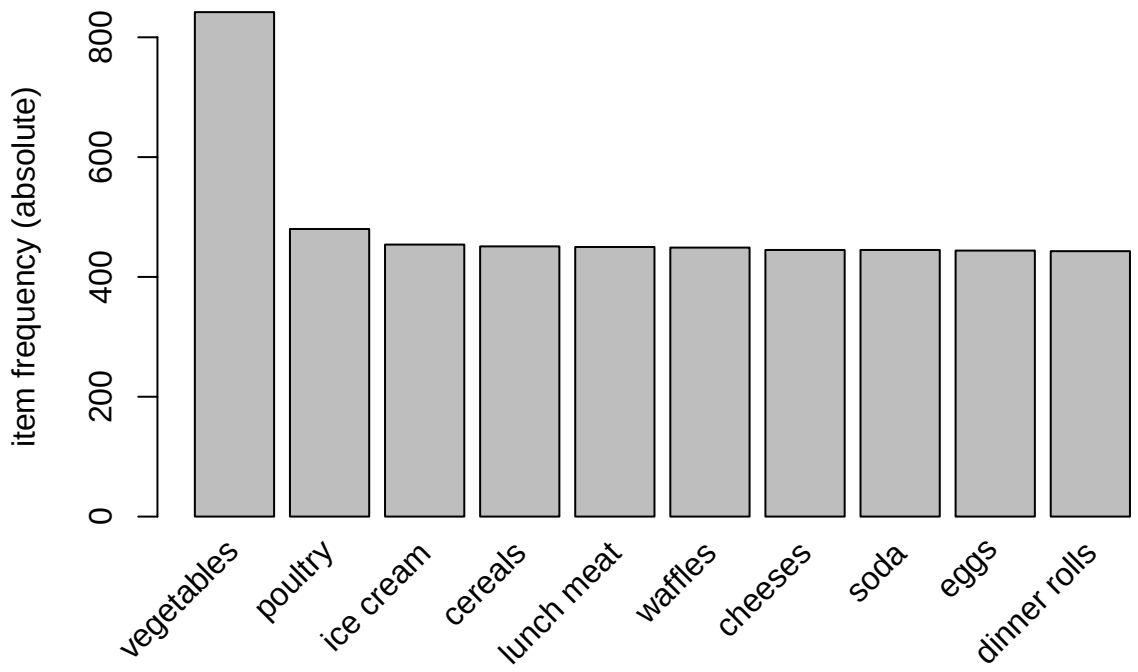
```
##      all- purpose aluminum foil bagels beef butter cereals
## [1,]            3             1      0    1      1       0
## [2,]            0             1      0    0      0       1
## [3,]            0             0      1    0      0       1
## [4,]            1             0      0    0      0       1
```

```
## [5,]            1             0      0    0      0       0
## [6,]            1             1      3    0      2       2
```

```
#Converting the data into an itemMatrix, required for apriori rules
transaction <- as(transaction, "itemMatrix")
```

```
## Warning in asMethod(object): matrix contains values other than 0 and 1!
## Setting all entries != 0 to 1.
```

```
#Analyzing the top 10 products pruchased by all customers, based on frequency
itemFrequencyPlot(transaction,topN=10,type="absolute")
```



Applying apriori analysis The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation, and groups of candidates are tested against the data).

You can play with the required support and confidence, and sort the rules by decreasing order of lift. Confidence calculates how strong an association is. It is the conditional probability of purchasing RHS if one has purchased LHS. It is calculated by by P(LHS U RHS)/P(LHS). This reprsents the probability of having RHS in the market basket, given the presence of LHS in the basket already.

The support of an item or item set is the fraction of transactions in our data set that contain that item or item set. It is the probability of finding an itemset in all the transactions. It is calculated using [count(itemset)/total transactions].

The lift value of an association rule is the ratio of the confidence of the rule and the expected confidence of the rule. It is the probability of finding certain itemsets toegther in a transaction, compared to the probability of finding them individually.

```
rules <- apriori(data=transaction, parameter = list(support= 0.3, confidence= 0.7))
```

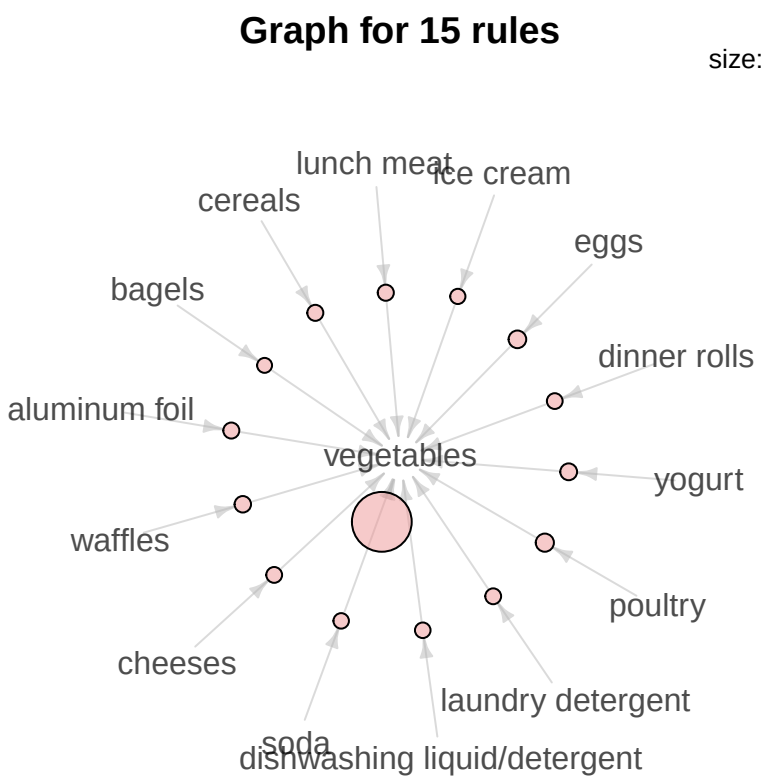
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.7    0.1    1 none FALSE            TRUE       5     0.3      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 341
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[38 item(s), 1139 transaction(s)] done [0.00s].
## sorting and recoding items ... [38 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

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

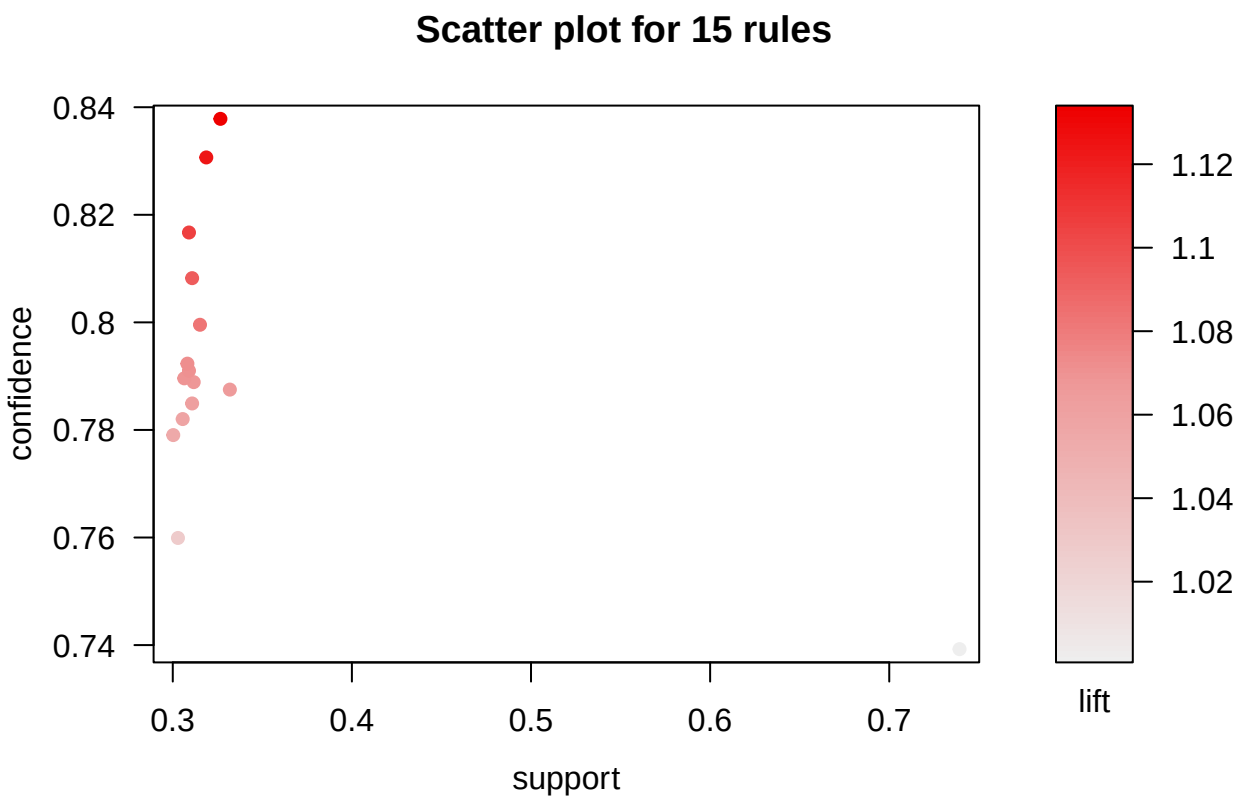
```
##      lhs                               rhs          support   confidence
## [1]  {eggs}                         => {vegetables} 0.3266023 0.8378378
## [2]  {yogurt}                       => {vegetables} 0.3187006 0.8306636
## [3]  {laundry detergent}            => {vegetables} 0.3090430 0.8167053
## [4]  {aluminum foil}                => {vegetables} 0.3107989 0.8082192
## [5]  {waffles}                      => {vegetables} 0.3151888 0.7995546
## [6]  {dinner rolls}                 => {vegetables} 0.3081651 0.7923251
## [7]  {cheeses}                      => {vegetables} 0.3090430 0.7910112
## [8]  {dishwashing liquid/detergent} => {vegetables} 0.3064091 0.7895928
## [9]  {lunch meat}                   => {vegetables} 0.3116769 0.7888889
## [10] {poultry}                      => {vegetables} 0.3318701 0.7875000
## [11] {cereals}                      => {vegetables} 0.3107989 0.7849224
## [12] {soda}                         => {vegetables} 0.3055312 0.7820225
## [13] {bagels}                       => {vegetables} 0.3002634 0.7790433
## [14] {ice cream}                    => {vegetables} 0.3028973 0.7599119
## [15] {}                             => {vegetables} 0.7392450 0.7392450
##      lift
## [1]  1.133370
## [2]  1.123665
## [3]  1.104783
## [4]  1.093304
## [5]  1.081583
## [6]  1.071803
## [7]  1.070026
## [8]  1.068107
## [9]  1.067155
## [10] 1.065276
```

```
## [11] 1.061789
## [12] 1.057867
## [13] 1.053836
## [14] 1.027957
## [15] 1.000000
```

```
plot(rules,method="graph",shading=NA)
```




```
plot(rules,method="scatter",shading="lift")
```



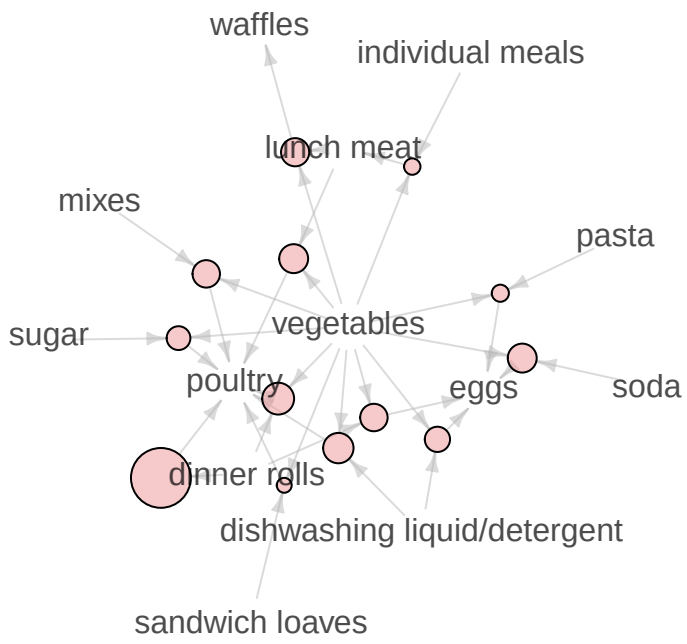

```
#We saw rules where vegetables is the most purchases itemset. Let's try to inspect rules where vegetabl
#is not a part of the transactions

rules_2 <- apriori(data=transaction, parameter = list(support= 0.1, confidence= 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 113
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[38 item(s), 1139 transaction(s)] done [0.00s].
## sorting and recoding items ... [38 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [716 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules_2 <- subset(rules_2, !(rhs %in% "vegetables"))
rules_2 <- sort(rules_2, by="lift", decreasing = TRUE)
inspect(rules_2)
```

```
##      lhs                                rhs            support confidence     lift
## [1]  {soda,
##       vegetables}                    => {eggs}       0.1580334  0.5172414 1.326887
## [2]  {dinner rolls,
##       vegetables}                    => {eggs}       0.1562774  0.5071225 1.300929
## [3]  {pasta,
##       vegetables}                    => {eggs}       0.1439860  0.5030675 1.290527
## [4]  {dishwashing liquid/detergent,
##       vegetables}                    => {eggs}       0.1536435  0.5014327 1.286333
## [5]  {lunch meat,
##       vegetables}                    => {waffles}    0.1571554  0.5042254 1.279093
## [6]  {individual meals,
##       vegetables}                    => {lunch meat} 0.1431080  0.5015385 1.269450
## [7]  {mixes,
##       vegetables}                    => {poultry}    0.1562774  0.5281899 1.253351
## [8]  {dinner rolls,
##       vegetables}                    => {poultry}    0.1615452  0.5242165 1.243922
## [9]  {dishwashing liquid/detergent,
##       vegetables}                    => {poultry}    0.1597893  0.5214900 1.237452
## [10] {sugar,
##       vegetables}                    => {poultry}    0.1518876  0.5103245 1.210957
## [11] {lunch meat,
##       vegetables}                    => {poultry}    0.1580334  0.5070423 1.203169
## [12] {dinner rolls}                  => {poultry}    0.1949078  0.5011287 1.189137
## [13] {sandwich loaves,
##       vegetables}                    => {poultry}    0.1413521  0.5000000 1.186458
```

```r
plot(rules_2,method="graph",shading=NA)
```




```
#Let's try finding rules where vegetables is neither in LHS nor in RHS
rules_3 <- apriori(data=transaction, parameter = list(support= 0.1, confidence= 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 113
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[38 item(s), 1139 transaction(s)] done [0.00s].
## sorting and recoding items ... [38 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [716 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules_3 <- subset(rules_3, !(rhs %in% "vegetables" | lhs %in% "vegetables"))
rules_3 <- sort(rules_3, by="lift", decreasing = TRUE)
```

```r
inspect(rules_3)
```

```
##     lhs                rhs       support   confidence lift
## [1] {dinner rolls} => {poultry} 0.1949078 0.5011287  1.189137
```

```r
plot(rules_3,method="graph",control=list(layout=igraph::in_circle()))
```