MBA

R Code

```
#install.packages("arules")
#install.packages("arulesviz")
```

```
#Step 1 - load the relevant libraries
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:arules':
##
##     intersect, recode, setdiff, setequal, union
```

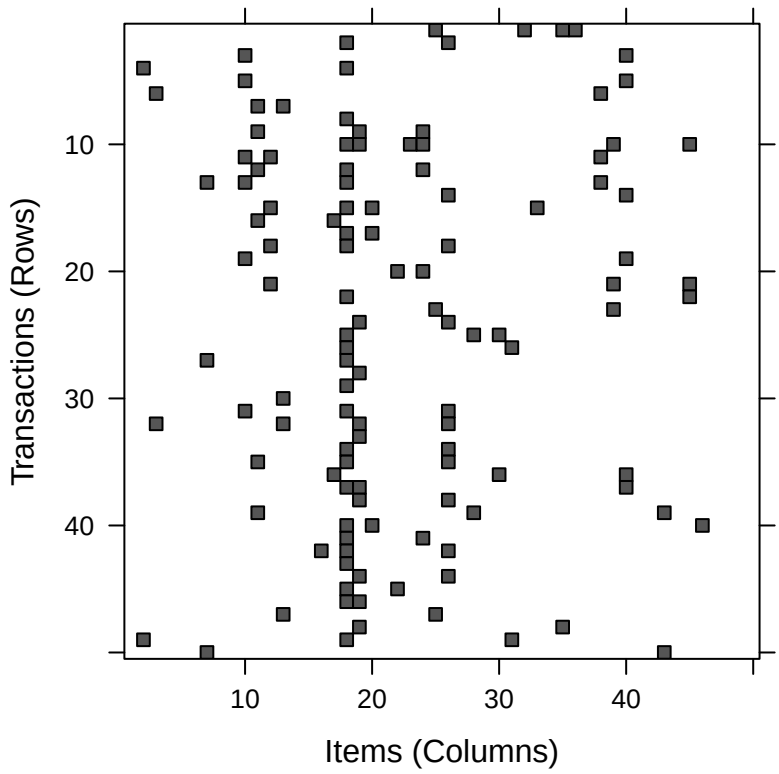
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
chipotle<-read.transactions("chipotleorders.csv", format = "single", header = "True", sep=",", cols = c
```

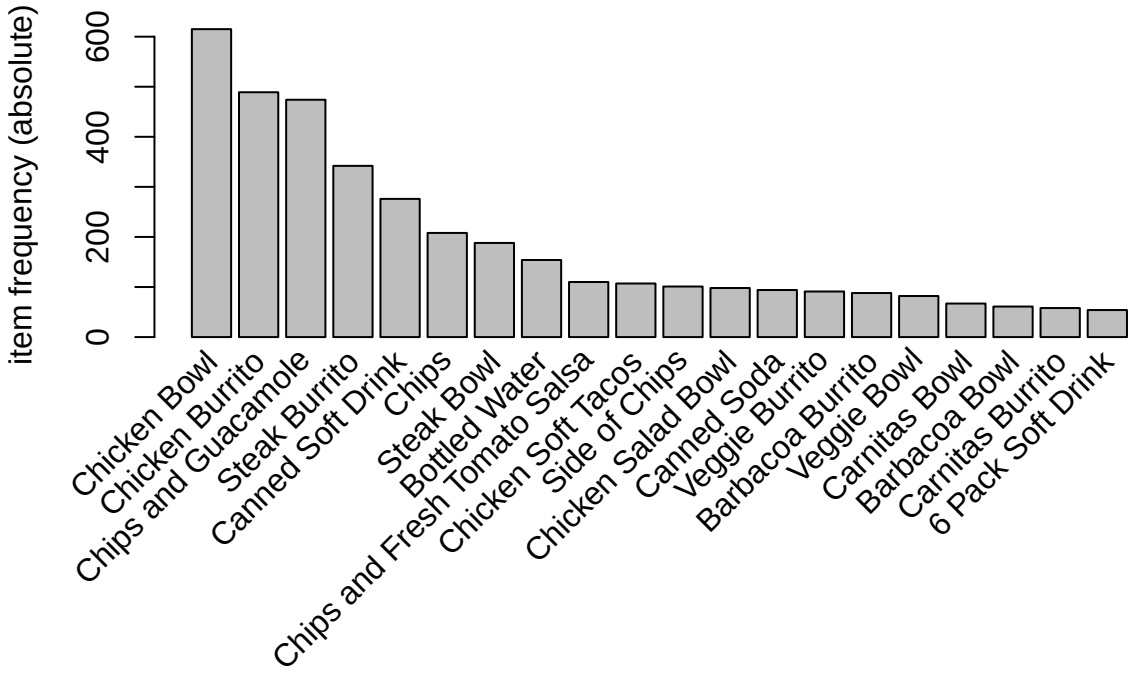
```
#cafe %>% glimpse()
```

```
image(chipotle[1:50,1:50])
```




```
# Step 3 - Create an item frequency plot for the top 20 items

itemFrequencyPlot(chipotle,topN=20,type="absolute")
```




```
# Step 4 - Get the rules setting min support to 0.001 and minimum confidence to 0.8

rules <- apriori(chipotle, parameter = list(supp = 0.001, conf = 0.80))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.001      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: 1
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[50 item(s), 1834 transaction(s)] done [0.00s].
## sorting and recoding items ... [45 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [216 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Step 5 - Limit the digits to 2 for reference and sort the rules by confidence before
# inspecting the top 5 rules found
rules<-sort(rules, by="confidence", decreasing=TRUE)
options(digits=2)
inspect(rules[1:5])
```

```
##     lhs                      rhs                   support confidence  lift count
## [1] {Carnitas Soft Tacos,
##      Steak Salad}         => {Steak Bowl}           0.0011          1   9.8     2
## [2] {Steak Bowl,
##      Steak Salad}         => {Carnitas Soft Tacos}  0.0011          1  48.3     2
## [3] {Carnitas Soft Tacos,
##      Steak Bowl}          => {Steak Salad}          0.0011          1 458.5     2
## [4] {Carnitas Soft Tacos,
##      Steak Salad}         => {Chips and Guacamole}  0.0011          1   3.9     2
## [5] {Steak Bowl,
##      Steak Salad}         => {Chips and Guacamole}  0.0011          1   3.9     2
```

```
rules<-apriori(data=chipotle, parameter=list(supp=0.001,conf = 0.8,minlen=6),
               appearance = list(default="lhs",rhs="Bottled Water"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="lift")
rules
```

```
## set of 1 rules
```

```
inspect(rules[1:1])
```

```
##     lhs                     rhs             support confidence lift count
## [1] {Canned Soft Drink,
##      Carnitas Bowl,
##      Chicken Bowl,
##      Chicken Soft Tacos,
##      Veggie Bowl}        => {Bottled Water}  0.0011          1   12     2
```

```
rules<-apriori(data=chipotle, parameter=list(supp=0.001,conf = 0.01,minlen=2),
               appearance = list(default="rhs",lhs="Chicken Bowl"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
rules
```

```
## set of 25 rules
```

```
inspect(rules[1:10])
```

```
##      lhs                rhs                                    support confidence
## [1]  {Chicken Bowl} => {Chips and Guacamole}                0.081   0.242
## [2]  {Chicken Bowl} => {Chips}                              0.067   0.198
## [3]  {Chicken Bowl} => {Canned Soft Drink}                  0.061   0.180
## [4]  {Chicken Bowl} => {Bottled Water}                      0.038   0.112
## [5]  {Chicken Bowl} => {Chicken Burrito}                    0.036   0.107
## [6]  {Chicken Bowl} => {Side of Chips}                      0.018   0.054
## [7]  {Chicken Bowl} => {Chicken Salad Bowl}                 0.016   0.049
## [8]  {Chicken Bowl} => {Chips and Fresh Tomato Salsa}       0.016   0.049
## [9]  {Chicken Bowl} => {Chips and Tomatillo Red Chili Salsa} 0.016   0.047
## [10] {Chicken Bowl} => {Steak Bowl}                         0.016   0.047
##      lift count
## [1]  0.94 149
## [2]  1.75 122
## [3]  1.20 111
## [4]  1.34  69
## [5]  0.40  66
## [6]  0.97  33
## [7]  0.91  30
## [8]  0.81  30
## [9]  1.88  29
## [10] 0.46  29
```

```
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
```

```
## Warning in `[<-`(`*tmp*`, as.vector(i), value = NA): x[.] <- val: x is
## "ngTMatrix", val not in {TRUE, FALSE} is coerced; NA |--> TRUE.
```

```
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned
```

```
#plot(rules[1:10], method = "graph", engine="interactive", control = list(verbose = TRUE))
```

```
#plot(rules[1:10], method = "graph",  engine = "htmlwidget")
```