Intro to Data Science Lab 10

```
# Enter your name here: Hrishikesh Telang
```

Copyright Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor and using
#StackOverflow as the Internet source
```

#Instructions: Association rules mining, also known as market basket analysis, is an unsupervised data mining technique that discovers patterns in the form of if-then rules. The technique is "unsupervised" in the sense that there is no prediction or classification happening. We are simply trying to find interesting patterns.

#In addition to working with "baskets" of objects, association rules mining is good at working with any kind of data that can be expressed as lists of attributes. For example, a trip to Washington DC might consist of the following attributes: train, July, morning departure, afternoon arrival, Union Station, first class, express.

#In these exercises we will work with a built in data set called groceries. Make sure to library the arules and arulesViz packages before running the following:

```
#install.packages('arules') #Install the package 'arules'
#install.packages('arulesViz') #Install the package 'arulesViz'
library(arules) #Load the package 'arules'
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

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

```
library(arulesViz) #Load the package 'arulesViz'
data (Groceries) # Load data into memory
myGroc <- Groceries # Make a copy for safety
summary(myGroc) # What is the structure?
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
```

```
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##        labels  level2           level1
## 1 frankfurter sausage meat and sausage
## 2     sausage sausage meat and sausage
## 3  liver loaf sausage meat and sausage
```

#1. Examine the data structure that summary() reveals. This is called a sparse matrix and it efficiently stores a set of market baskets along with meta-data. Report in a comment about some of the item labels.

```
summary(myGroc)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##        labels  level2           level1
## 1 frankfurter sausage meat and sausage
## 2     sausage sausage meat and sausage
## 3  liver loaf sausage meat and sausage
```

```
#The sparse matrix contains 9835 rows and 169 columns
#It also details on the most frequent items, the popular ones being whole milk,
#other vegetables, rolls/buns, soda, and yogurt.
```

```
#This code returns the labels along with its association rules.
#For example, the labels 'frankfurter', 'sausage' and 'liver loaf' have the same
#association rules ie: the frequency of choosing meat and sausage together
```

#2. Use the itemFrequency(myGroc) command to generate a list of item frequencies. Save that list in a new data object. Run str( ) on the data object and write a comment describing what it is. Run sort( ) on the data object and save the results. Run head( ) and tail( ) on the sorted object to show the most and least frequently occurring items. What's the most frequently purchased item?

```
data <- itemFrequency(myGroc)
data #Display the data
```

```
##                frankfurter                  sausage               liver loaf
##               0.0589730554             0.0939501779             0.0050838841
##                        ham                     meat        finished products
##               0.0260294865             0.0258261312             0.0065073716
##            organic sausage                  chicken                   turkey
##               0.0022369090             0.0429079817             0.0081342145
##                       pork                     beef           hamburger meat
##               0.0576512456             0.0524656838             0.0332486019
##                       fish             citrus fruit           tropical fruit
##               0.0029486528             0.0827656329             0.1049313676
##                  pip fruit                   grapes                  berries
##               0.0756481952             0.0223690900             0.0332486019
##                 nuts/prunes           root vegetables                  onions
##               0.0033553635             0.1089984748             0.0310116929
##                      herbs         other vegetables packaged fruit/vegetables
##               0.0162684291             0.1934926284             0.0130147433
##                 whole milk                   butter                     curd
##               0.2555160142             0.0554143366             0.0532791052
##                    dessert              butter milk                   yogurt
##               0.0371123538             0.0279613625             0.1395017794
##          whipped/sour cream                beverages                 UHT-milk
##               0.0716827656             0.0260294865             0.0334519573
##             condensed milk                    cream              soft cheese
##               0.0102694459             0.0013218099             0.0170818505
##              sliced cheese              hard cheese             cream cheese
##               0.0245043213             0.0245043213             0.0396542959
##           processed cheese            spread cheese              curd cheese
##               0.0165734621             0.0111845450             0.0050838841
##           specialty cheese               mayonnaise           salad dressing
##               0.0085409253             0.0091509914             0.0008134215
##                    tidbits        frozen vegetables            frozen fruits
##               0.0023385867             0.0480935435             0.0012201322
##                frozen meals              frozen fish           frozen chicken
##               0.0283680732             0.0116929334             0.0006100661
##                  ice cream           frozen dessert   frozen potato products
##               0.0250127097             0.0107778343             0.0084392476
##              domestic eggs               rolls/buns              white bread
##               0.0634468734             0.1839349263             0.0420945602
##                brown bread                   pastry            roll products
##               0.0648703610             0.0889679715             0.0102694459
##         semi-finished bread                 zwieback          potato products
```

```
##               0.0176919166             0.0069140824             0.0028469751 
##                      flour                     salt                     rice 
##               0.0173868836             0.0107778343             0.0076258261 
##                      pasta                  vinegar                      oil 
##               0.0150482969             0.0065073716             0.0280630402 
##                  margarine            specialty fat                    sugar 
##               0.0585663447             0.0036603965             0.0338586680 
##           artif. sweetener                    honey                  mustard 
##               0.0032536858             0.0015251652             0.0119979664 
##                    ketchup                   spices                    soups 
##               0.0042704626             0.0051855618             0.0068124047 
##                ready soups   Instant food products                   sauces 
##               0.0018301983             0.0080325369             0.0054905948 
##                    cereals        organic products           baking powder 
##               0.0056939502             0.0016268429             0.0176919166 
##     preservation products          pudding powder        canned vegetables 
##               0.0002033554             0.0023385867             0.0107778343 
##               canned fruit      pickled vegetables    specialty vegetables 
##               0.0032536858             0.0178952720             0.0017285206 
##                        jam           sweet spreads            meat spreads 
##               0.0053889171             0.0090493137             0.0042704626 
##                canned fish                dog food                 cat food 
##               0.0150482969             0.0085409253             0.0232841891 
##                   pet care                baby food                   coffee 
##               0.0094560244             0.0001016777             0.0580579563 
##             instant coffee                      tea             cocoa drinks 
##               0.0074224708             0.0038637519             0.0022369090 
##              bottled water                     soda          misc. beverages 
##               0.1105236401             0.1743772242             0.0283680732 
##     fruit/vegetable juice                    syrup             bottled beer 
##               0.0722928317             0.0032536858             0.0805287239 
##                canned beer                   brandy                   whisky 
##               0.0776817489             0.0041687850             0.0008134215 
##                     liquor                      rum                  liqueur 
##               0.0110828673             0.0044738180             0.0009150991 
##        liquor (appetizer)               white wine           red/blush wine 
##               0.0079308592             0.0190137265             0.0192170819 
##                   prosecco          sparkling wine             salty snack 
##               0.0020335536             0.0055922725             0.0378240976 
##                    popcorn               nut snack          snack products 
##               0.0072191154             0.0031520081             0.0030503305 
##  long life bakery product                  waffles                 cake bar 
##               0.0374173869             0.0384341637             0.0132180986 
##                chewing gum                chocolate       cooking chocolate 
##               0.0210472801             0.0496187087             0.0025419420 
##       specialty chocolate            specialty bar   chocolate marshmallow 
##               0.0304016268             0.0273512964             0.0090493137 
##                      candy        seasonal products               detergent 
##               0.0298932384             0.0142348754             0.0192170819 
##                   softener              decalcifier             dish cleaner 
##               0.0054905948             0.0015251652             0.0104728012 
##          abrasive cleaner                  cleaner          toilet cleaner 
##               0.0035587189             0.0050838841             0.0007117438 
##          bathroom cleaner               hair spray              dental care 
```

```
##              0.0027452974              0.0011184545              0.0057956279
##            male cosmetics            make up remover                 skin care
##              0.0045754957              0.0008134215              0.0035587189
##  female sanitary products            baby cosmetics                      soap
##              0.0061006609              0.0006100661              0.0026436197
##            rubbing alcohol          hygiene articles                   napkins
##              0.0010167768              0.0329435689              0.0523640061
##                    dishes                  cookware           kitchen utensil
##              0.0175902389              0.0027452974              0.0004067107
##            cling film/bags            kitchen towels    house keeping products
##              0.0113879004              0.0059989832              0.0083375699
##                   candles               light bulbs       sound storage medium
##              0.0089476360              0.0041687850              0.0001016777
##                 newspapers                photo/film                pot plants
##              0.0798169802              0.0092526690              0.0172852059
##    flower soil/fertilizer            flower (seeds)              shopping bags
##              0.0019318760              0.0103711235              0.0985256736
##                      bags
##              0.0004067107
```

```
str(data) #It returns the overall frequency of items in a randomised order.
```

```
##  Named num [1:169] 0.05897 0.09395 0.00508 0.02603 0.02583 ...
##  - attr(*, "names")= chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
```

```
sorted_data <- sort(data) #Sorting the data
#Printing head and tail
print("Head")
```

```
## [1] "Head"
```

```
head(sorted_data)
```

```
##            baby food  sound storage medium preservation products
##         0.0001016777          0.0001016777          0.0002033554
##      kitchen utensil                  bags        frozen chicken
##         0.0004067107          0.0004067107          0.0006100661
```

```
print("Tail")
```

```
## [1] "Tail"
```

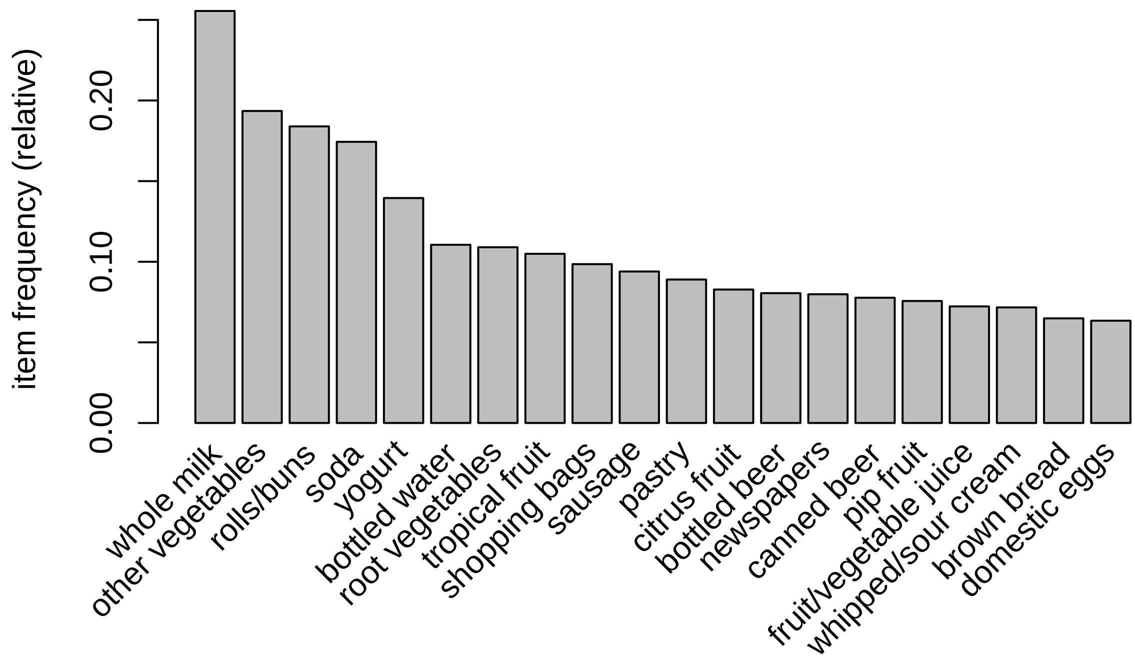
```
tail(sorted_data)
```

```
##    bottled water           yogurt             soda       rolls/buns
##        0.1105236        0.1395018        0.1743772        0.1839349
## other vegetables       whole milk
##        0.1934926        0.2555160
```

```
# Whole milk is the most frequently purchased item in this dataset
```

#3. Create a frequency plot with itemFrequencyPlot(myGroc, topN=20) and confirm that the plot shows the most frequently purchased item with the left-most bar. Write a comment describing the meaning of the Y-axis.

```
itemFrequencyPlot(myGroc, topN=20)
```




```
#The Y axis signifies the frequency of the number of times the item has been selected.
#(In other words, Y axis indicates the support.)
#This has been taken from the myGroc dataset.
```

#4. Create a cross table with ct <- crossTable(myGroc, sort=TRUE). Examine the first few rows and columns of ct by using the square brackets subsetting technique. For example, the first two rows and first three columns would be ct[1:2, 1:3]. Write a comment describing one of values. Write a comment describing what is on the diagonal of the matrix.

```
ct  <- crossTable(myGroc,   sort=TRUE)
ct[1:2, 1:3]
```

```
##                    whole milk other vegetables rolls/buns
## whole milk               2513              736        557
## other vegetables          736             1903        419
```

```r
#The crosstable value is pretty straightforward. It checks the frequency for the
#amount of items for every item in the dataset, but with subsetting, the items drill down to a
#2 x 3 dataset. Thus, the frequency of whole milk with other vegetables is 736, and with
#rolls/buns it is 557 and so forth.
```

#5. Run the following analysis:

```r
rules1 <- apriori(myGroc,
parameter=list(supp=0.0008, conf=0.55),
control=list(verbose=F),
appearance=list(default="lhs",rhs=("bottled beer")))
```

#6. Examine the resulting rule set with inspect( ) and make sense of the results. There should be four rules in total.

```r
inspect(rules1)
```

```
##     lhs                                rhs            support      confidence
## [1] {liquor,red/blush wine}         => {bottled beer} 0.0019318760 0.9047619
## [2] {soda,liquor}                   => {bottled beer} 0.0012201322 0.5714286
## [3] {red/blush wine,napkins}        => {bottled beer} 0.0008134215 0.5714286
## [4] {soda,liquor,red/blush wine} => {bottled beer} 0.0008134215 1.0000000
##     coverage     lift     count
## [1] 0.0021352313 11.23527 19
## [2] 0.0021352313  7.09596 12
## [3] 0.0014234875  7.09596  8
## [4] 0.0008134215 12.41793  8
```

```r
#It can be observed that the lift of {soda,liquor,red/blush wine}   =>  {bottled beer}
#is the highest, with confidence being 100%, followed by {liquor,red/blush wine}    =>  {bottled beer},
#with lift=11.23527 and confidence as 90%. The lifts and confidences of the
#remaining association rules are similar, so we check the corresponding supports.
#We can observe that the support {liquor,red/blush wine}    =>  {bottled beer} is higher
#than {red/blush wine,napkins}  =>  {bottled beer}.
```

#7. Adjust the support parameter to a new value so that you get more rules. Anywhere between 10 and 30 rules would be fine. Examine the new rule set with inspect( ). Does your interpretation of the situation still make sense?

```r
rules2 <- apriori(myGroc, parameter=list(supp=0.0006, conf=0.45),
control=list(verbose=F),
appearance=list(default="lhs",rhs=("bottled beer")))
inspect(rules2)
```

```
##      lhs                        rhs                  support confidence     coverage      lift count
## [1]  {liquor (appetizer),
##       dishes}               => {bottled beer} 0.0006100661   0.8571429 0.0007117438 10.643939     6
## [2]  {liquor,
##       red/blush wine}       => {bottled beer} 0.0019318760   0.9047619 0.0021352313 11.235269    19
## [3]  {soda,
##       liquor}               => {bottled beer} 0.0012201322   0.5714286 0.0021352313  7.095960    12
```

```
## [4]  {yogurt,
##       flower (seeds)}      => {bottled beer} 0.0007117438  0.5000000 0.0014234875  6.208965     7
## [5]  {frozen dessert,
##       bottled water}       => {bottled beer} 0.0007117438  0.4666667 0.0015251652  5.795034     7
## [6]  {red/blush wine,
##       napkins}             => {bottled beer} 0.0008134215  0.5714286 0.0014234875  7.095960     8
## [7]  {canned fish,
##       hygiene articles}    => {bottled beer} 0.0006100661  0.5454545 0.0011184545  6.773416     6
## [8]  {soda,
##       liquor,
##       red/blush wine}      => {bottled beer} 0.0008134215  1.0000000 0.0008134215 12.417929     8
## [9]  {whole milk,
##       canned fish,
##       hygiene articles}    => {bottled beer} 0.0006100661  0.5454545 0.0011184545  6.773416     6
## [10] {citrus fruit,
##       herbs,
##       bottled water}       => {bottled beer} 0.0006100661  0.4615385 0.0013218099  5.731352     6
## [11] {herbs,
##       other vegetables,
##       bottled water}       => {bottled beer} 0.0007117438  0.5000000 0.0014234875  6.208965     7
## [12] {butter,
##       rolls/buns,
##       napkins}             => {bottled beer} 0.0006100661  0.5454545 0.0011184545  6.773416     6
## [13] {root vegetables,
##       herbs,
##       other vegetables,
##       bottled water}       => {bottled beer} 0.0006100661  0.6000000 0.0010167768  7.450758     6
```

```
#Yes, the   interpretation  of the  situation   still   makes sense. This time, we have 13
#rules that correspond to a threshold support of 0.006 and confidence of 45%.
#The rule {soda,liquor,red/blush wine}  =>  {bottled beer} has the highest lift
#of 12.417929 with 100% confidence, followed by lift of 11.235269 for
#{liquor,red/blush wine}    =>  {bottled beer} and confidence of 90%. This goes on for the remaining
#10 rules.
```

#8. Power User: use mtcars to create a new dataframe with factors (e.g., cyl attribute). Then create an mpg column with "good" or "bad" (good MPG is above 25). Convert the dataframe to a transactions dataset and then predict rules for having bad MPG.

```
#str(mtcars)
#Creating mtattr attribute of only factors
mtattr <- mtcars[ ,c("cyl", "vs", "am", "gear", "carb")]
#I used a function ifelse from StackOverflow to get "good" and "bad" labels
mtattr$goodorbadmpg <- ifelse(mtcars$mpg > 25, "good", "bad")
#mtattr

str(mtattr) #Displaying the whole structure of mtattr
```

```
## 'data.frame':    32 obs. of  6 variables:
##  $ cyl         : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ vs          : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am          : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear        : num  4 4 4 3 3 3 3 4 4 4 ...
```

```
##  $ carb       : num  4 4 1 1 2 1 4 2 2 4 ...
##  $ goodorbadmpg: chr  "bad" "bad" "bad" "bad" ...
```

```r
mtmatr <- as(mtattr, "transactions") #Converting the dataframe to a transactions dataset
```

```
## Warning: Column(s) 1, 2, 3, 4, 5, 6 not logical or factor. Applying default
## discretization (see '? discretizeDF').

## Warning in discretize(x = c(6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, : The calculated breaks are
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Warning in discretize(x = c(0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, : The calculated breaks are
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Warning in discretize(x = c(1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, : The calculated breaks are
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Warning in discretize(x = c(4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, : The calculated breaks are
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.
```

```r
#Using apriori to predict rules for having bad  MPG.
rules3 <- apriori(mtmatr, parameter = list(supp=0.25, conf= 0.80, minlen=4), appearance=list(default="ll
summary(rules3) #Displying summary
```

```
## set of 9 rules
##
## rule length distribution (lhs + rhs):sizes
## 4 5
## 7 2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   4.000   4.000   4.222   4.000   5.000
##
## summary of quality measures:
##     support         confidence    coverage           lift           count
##  Min.   :0.3750   Min.   :1    Min.   :0.3750   Min.   :1.231   Min.   :12
##  1st Qu.:0.3750   1st Qu.:1    1st Qu.:0.3750   1st Qu.:1.231   1st Qu.:12
##  Median :0.4375   Median :1    Median :0.4375   Median :1.231   Median :14
##  Mean   :0.4375   Mean   :1    Mean   :0.4375   Mean   :1.231   Mean   :14
##  3rd Qu.:0.4375   3rd Qu.:1    3rd Qu.:0.4375   3rd Qu.:1.231   3rd Qu.:14
##  Max.   :0.6562   Max.   :1    Max.   :0.6562   Max.   :1.231   Max.   :21
##
## mining info:
##    data ntransactions support confidence
##  mtmatr            32    0.25        0.8
```

```r
inspect(rules3) #Displaying the whole transactions dataset
```

```
##     lhs               rhs                support confidence coverage     lift count
## [1] {cyl=[4.67,8],
##      vs=[0,1],
```

```
##      carb=[4,8]}   => {goodorbadmpg=bad} 0.37500          1  0.37500 1.230769    12
## [2] {cyl=[4.67,8],
##      am=[0,1],
##      carb=[4,8]}   => {goodorbadmpg=bad} 0.37500          1  0.37500 1.230769    12
## [3] {vs=[0,1],
##      am=[0,1],
##      carb=[4,8]}   => {goodorbadmpg=bad} 0.37500          1  0.37500 1.230769    12
## [4] {cyl=[4.67,8],
##      vs=[0,1],
##      gear=[3,4)}   => {goodorbadmpg=bad} 0.43750          1  0.43750 1.230769    14
## [5] {cyl=[4.67,8],
##      am=[0,1],
##      gear=[3,4)}   => {goodorbadmpg=bad} 0.43750          1  0.43750 1.230769    14
## [6] {vs=[0,1],
##      am=[0,1],
##      gear=[3,4)}   => {goodorbadmpg=bad} 0.46875          1  0.46875 1.230769    15
## [7] {cyl=[4.67,8],
##      vs=[0,1],
##      am=[0,1]}     => {goodorbadmpg=bad} 0.65625          1  0.65625 1.230769    21
## [8] {cyl=[4.67,8],
##      vs=[0,1],
##      am=[0,1],
##      carb=[4,8]}   => {goodorbadmpg=bad} 0.37500          1  0.37500 1.230769    12
## [9] {cyl=[4.67,8],
##      vs=[0,1],
##      am=[0,1],
##      gear=[3,4)}   => {goodorbadmpg=bad} 0.43750          1  0.43750 1.230769    14
```

```
#The whole set of commands gives me a transactions table of 9 rules with equal lifts and
#conmfidences, but with varying supports. {cyl=[4.67,8],vs=[0,1],am=[0,1]}  =>  {goodorbadmpg=bad}
#has a support of 0.65625
```