markass.R

spoor

Sun Apr 02 08:25:49 2017

```r
# Association Rules for Market Basket Analysis (R)

library(arules)  # association rules
```

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

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

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

```r
library(arulesViz)  # data visualization of association rules
```

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

```r
library(RColorBrewer)  # color palettes for plots

data(Groceries)  # grocery transactions object from arules package

# show the dimensions of the transactions object
print(dim(Groceries))
```

```
## [1] 9835  169
```

```r
print(dim(Groceries)[1])  # 9835 market baskets for shopping trips
```

```
## [1] 9835
```

```r
print(dim(Groceries)[2])  # 169 initial store items
```

```
## [1] 169
```

```r
# examine frequency for each item with support greater than 0.025
pdf(file="fig_market_basket_initial_item_support.pdf",
  width = 8.5, height = 11)
itemFrequencyPlot(Groceries, support = 0.025, cex.names=0.8, xlim = c(0,0.3),
  type = "relative", horiz = TRUE, col = "dark red", las = 1,
  xlab = paste("Proportion of Market Baskets Containing Item",
    "\n(Item Relative Frequency or Support)"))
dev.off()
```

```
## pdf 
##   2
```

```r
# explore possibilities for combining similar items
print(head(itemInfo(Groceries)))
```

```
##               labels  level2           level1
## 1       frankfurter sausage meat and sausage
## 2           sausage sausage meat and sausage
## 3        liver loaf sausage meat and sausage
## 4               ham sausage meat and sausage
## 5              meat sausage meat and sausage
## 6 finished products sausage meat and sausage
```

```r
print(levels(itemInfo(Groceries)[["level1"]]))  # 10 levels... too few
```

```
##  [1] "canned food"          "detergent"            "drinks"
##  [4] "fresh products"       "fruit and vegetables" "meat and sausage"
##  [7] "non-food"             "perfumery"            "processed food"
## [10] "snacks and candies"
```

```r
print(levels(itemInfo(Groceries)[["level2"]]))  # 55 distinct levels
```

```
##  [1] "baby food"                       "bags"
##  [3] "bakery improver"                 "bathroom cleaner"
##  [5] "beef"                            "beer"
##  [7] "bread and backed goods"          "candy"
##  [9] "canned fish"                     "canned fruit/vegetables"
## [11] "cheese"                          "chewing gum"
## [13] "chocolate"                       "cleaner"
## [15] "coffee"                          "condiments"
## [17] "cosmetics"                       "dairy produce"
## [19] "delicatessen"                    "dental care"
## [21] "detergent/softener"              "eggs"
## [23] "fish"                            "frozen foods"
## [25] "fruit"                           "games/books/hobby"
## [27] "garden"                          "hair care"
## [29] "hard drinks"                     "health food"
## [31] "jam/sweet spreads"               "long-life bakery products"
## [33] "meat spreads"                    "non-alc. drinks"
## [35] "non-food house keeping products" "non-food kitchen"
## [37] "packaged fruit/vegetables"       "perfumery"
## [39] "personal hygiene"                "pet food/care"
## [41] "pork"                            "poultry"
## [43] "pudding powder"                  "sausage"
## [45] "seasonal products"               "shelf-stable dairy"
## [47] "snacks"                          "soap"
## [49] "soups/sauces"                    "staple foods"
## [51] "sweetener"                       "tea/cocoa drinks"
## [53] "vegetables"                      "vinegar/oils"
## [55] "wine"
```

```r
# aggregate items using the 55 level2 levels for food categories
# to create a more meaningful set of items
groceries <- aggregate(Groceries, itemInfo(Groceries)[["level2"]])

print(dim(groceries)[1])  # 9835 market baskets for shopping trips
```

```
## [1] 9835
```

```r
print(dim(groceries)[2])  # 55 final store items (categories)
```

```
## [1] 55
```

```r
pdf(file="fig_market_basket_final_item_support.pdf", width = 8.5, height = 11)
itemFrequencyPlot(groceries, support = 0.025, cex.names=1.0, xlim = c(0,0.5),
  type = "relative", horiz = TRUE, col = "blue", las = 1,
  xlab = paste("Proportion of Market Baskets Containing Item",
    "\n(Item Relative Frequency or Support)"))
dev.off()
```

```
## pdf
##   2
```

```
# obtain large set of association rules for items by category and all shoppers
# this is done by setting very low criteria for support and confidence
first.rules <- apriori(groceries,
  parameter = list(support = 0.001, confidence = 0.05))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    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: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[55 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [54 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.03s].
## writing ... [69921 rule(s)] done [0.04s].
## creating S4 object  ... done [0.06s].
```

```
print(summary(first.rules))  # yields 69,921 rules... too many
```

```
## set of 69921 rules
##
## rule length distribution (lhs + rhs):sizes
##     1     2     3     4     5     6     7     8
##    21  1205 10467 23895 22560  9888  1813    72
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   4.000   4.502   5.000   8.000
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.001017   Min.   :0.0500   Min.   : 0.4475
##  1st Qu.:0.001118   1st Qu.:0.2110   1st Qu.: 1.8315
##  Median :0.001525   Median :0.4231   Median : 2.2573
##  Mean   :0.002488   Mean   :0.4364   Mean   : 2.5382
##  3rd Qu.:0.002339   3rd Qu.:0.6269   3rd Qu.: 2.9662
##  Max.   :0.443010   Max.   :1.0000   Max.   :16.1760
##
## mining info:
```

```
##       data ntransactions support confidence
##  groceries          9835   0.001       0.05
```

```
# select association rules using thresholds for support and confidence
second.rules <- apriori(groceries,
  parameter = list(support = 0.025, confidence = 0.05))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.025      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: 245
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[55 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [32 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [344 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
print(summary(second.rules))  # yields 344 rules
```

```
## set of 344 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3   4
##  21 162 129  32
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     2.0     2.0     2.5     3.0     4.0
##
## summary of quality measures:
##     support          confidence           lift
##  Min.   :0.02542   Min.   :0.05043   Min.   :0.6669
##  1st Qu.:0.03030   1st Qu.:0.18202   1st Qu.:1.2498
##  Median :0.03854   Median :0.39522   Median :1.4770
##  Mean   :0.05276   Mean   :0.37658   Mean   :1.4831
##  3rd Qu.:0.05236   3rd Qu.:0.51271   3rd Qu.:1.7094
##  Max.   :0.44301   Max.   :0.79841   Max.   :2.4073
##
## mining info:
##       data ntransactions support confidence
##  groceries          9835   0.025       0.05
```

```
# data visualization of association rules in scatter plot
pdf(file="fig_market_basket_rules.pdf", width = 8.5, height = 8.5)
plot(second.rules,
  control=list(jitter=2, col = rev(brewer.pal(9, "Greens")[4:9])),
  shading = "lift")
dev.off()
```

```
## pdf
##   2
```

```
# grouped matrix of rules
pdf(file="fig_market_basket_rules_matrix.pdf", width = 8.5, height = 8.5)
plot(second.rules, method="grouped",
  control=list(col = rev(brewer.pal(9, "Greens")[4:9])))
dev.off()
```

```
## pdf
##   2
```

```
# select rules with vegetables in consequent (right-hand-side) item subsets
vegie.rules <- subset(second.rules, subset = rhs %pin% "vegetables")
inspect(vegie.rules)  # 41 rules
```

```
##      lhs                            rhs          support confidence      lift
## [1]  {}                          => {vegetables} 0.27300458  0.2730046 1.0000000
## [2]  {poultry}                   => {vegetables} 0.02897814  0.5745968 2.1047148
## [3]  {pork}                      => {vegetables} 0.03009659  0.5220459 1.9122238
## [4]  {staple foods}              => {vegetables} 0.02613116  0.5160643 1.8903136
## [5]  {eggs}                      => {vegetables} 0.03141840  0.4951923 1.8138608
## [6]  {games/books/hobby}         => {vegetables} 0.02785968  0.3145809 1.1522918
## [7]  {long-life bakery products} => {vegetables} 0.02907982  0.3492063 1.2791227
## [8]  {perfumery}                 => {vegetables} 0.03213015  0.4056483 1.4858662
## [9]  {beef}                      => {vegetables} 0.04585663  0.5595533 2.0496116
## [10] {bags}                      => {vegetables} 0.03141840  0.3175745 1.1632571
## [11] {vinegar/oils}              => {vegetables} 0.04199288  0.4666667 1.7093731
## [12] {chocolate}                 => {vegetables} 0.03192679  0.2934579 1.0749195
## [13] {beer}                      => {vegetables} 0.03406202  0.2189542 0.8020168
## [14] {frozen foods}              => {vegetables} 0.04738180  0.4052174 1.4842879
## [15] {cheese}                    => {vegetables} 0.05531266  0.4365971 1.5992300
## [16] {sausage}                   => {vegetables} 0.07625826  0.4032258 1.4769929
## [17] {fruit}                     => {vegetables} 0.10706660  0.4297959 1.5743176
## [18] {non-alc. drinks}           => {vegetables} 0.09456024  0.2974097 1.0893944
## [19] {bread and backed goods}    => {vegetables} 0.11621759  0.3363743 1.2321198
## [20] {dairy produce}             => {vegetables} 0.17041179  0.3846683 1.4090180
## [21] {beef,
##       dairy produce}             => {vegetables} 0.02989324  0.6074380 2.2250104
## [22] {dairy produce,
##       vinegar/oils}              => {vegetables} 0.03141840  0.5355286 1.9616103
## [23] {dairy produce,
##       frozen foods}              => {vegetables} 0.03436706  0.5121212 1.8758704
## [24] {cheese,
##       fruit}                     => {vegetables} 0.02674123  0.5197628 1.9038613
## [25] {bread and backed goods,
##       cheese}                    => {vegetables} 0.02887646  0.4536741 1.6617821
## [26] {cheese,
##       dairy produce}             => {vegetables} 0.04219624  0.4987981 1.8270686
## [27] {fruit,
##       sausage}                   => {vegetables} 0.03426538  0.5290424 1.9378517
## [28] {non-alc. drinks,
##       sausage}                   => {vegetables} 0.03029995  0.4156206 1.5223944
## [29] {bread and backed goods,
##       sausage}                   => {vegetables} 0.04382308  0.4229637 1.5492916
## [30] {dairy produce,
##       sausage}                   => {vegetables} 0.05266904  0.4905303 1.7967842
## [31] {fruit,
##       non-alc. drinks}           => {vegetables} 0.04361973  0.4657980 1.7061914
## [32] {bread and backed goods,
##       fruit}                     => {vegetables} 0.05124555  0.4763705 1.7449177
## [33] {dairy produce,
##       fruit}                     => {vegetables} 0.07869853  0.5032510 1.8433793
## [34] {bread and backed goods,
##       non-alc. drinks}           => {vegetables} 0.04636502  0.3731588 1.3668590
## [35] {dairy produce,
##       non-alc. drinks}           => {vegetables} 0.06446365  0.4243641 1.5544213
## [36] {bread and backed goods,
##       dairy produce}             => {vegetables} 0.08195221  0.4366197 1.5993128
## [37] {dairy produce,
##       fruit,
##       sausage}                   => {vegetables} 0.02714794  0.5741935 2.1032378
## [38] {bread and backed goods,
##       dairy produce,
##       sausage}                   => {vegetables} 0.03284189  0.5135135 1.8809704
## [39] {dairy produce,
##       fruit,
##       non-alc. drinks}           => {vegetables} 0.03304525  0.5183413 1.8986543
## [40] {bread and backed goods,
##       dairy produce,
##       fruit}                     => {vegetables} 0.04077275  0.5276316 1.9326840
## [41] {bread and backed goods,
##       dairy produce,
##       non-alc. drinks}           => {vegetables} 0.03345196  0.4627286 1.6949480
```

```r
# sort by lift and identify the top 10 rules
top.vegie.rules <- head(sort(vegie.rules, decreasing = TRUE, by = "lift"), 10)
inspect(top.vegie.rules)
```

```
##      lhs                         rhs             support confidence     lift
## [1]  {beef,
##       dairy produce}          => {vegetables} 0.02989324  0.6074380 2.225010
## [2]  {poultry}                => {vegetables} 0.02897814  0.5745968 2.104715
## [3]  {dairy produce,
##       fruit,
##       sausage}                => {vegetables} 0.02714794  0.5741935 2.103238
## [4]  {beef}                   => {vegetables} 0.04585663  0.5595533 2.049612
## [5]  {dairy produce,
##       vinegar/oils}           => {vegetables} 0.03141840  0.5355286 1.961610
## [6]  {fruit,
##       sausage}                => {vegetables} 0.03426538  0.5290424 1.937852
## [7]  {bread and backed goods,
##       dairy produce,
##       fruit}                  => {vegetables} 0.04077275  0.5276316 1.932684
## [8]  {pork}                   => {vegetables} 0.03009659  0.5220459 1.912224
## [9]  {cheese,
##       fruit}                  => {vegetables} 0.02674123  0.5197628 1.903861
## [10] {dairy produce,
##       fruit,
##       non-alc. drinks}        => {vegetables} 0.03304525  0.5183413 1.898654
```

```r
pdf(file="fig_market_basket_farmer_rules.pdf", width = 11, height = 8.5)
plot(top.vegie.rules, method="graph",
  control=list(type="items"),
  shading = "lift")
dev.off()
```

```
## pdf
##   2
```

```r
# Suggestions for the student:
# Suppose your client is someone other than the local farmer,
# a meat producer/butcher, dairy, or brewer perhaps.
# Determine association rules relevant to your client's products
# guided by the market basket model. What recommendations
# would you make about future marketplace actions?
```