Problem Set 8

Matt He

2022-11-28

##Part 1 Question 1 A. actionable: Patients in the emergency room usually have problems with high blood pressure. This could alarm other people with high blood pressure. They should be extremely careful with sudden heard attack.
B. Trivial The sugar consumption is extremely for diabetic patients. C. Inexplicable Study shows that the cause of nearsightedness in younger generation is not long-term using of electronic devices. Instead, the nearsightedness results from lack of natural sunlight. If a child can guarantee one to two hours of outdoor exercising, the nearsightedness is not likely to happen.

Question 2 I used to work at a gift store that sells special local food. During vacations, people like to travel around. For people who takes long road trips or takes train, they are more likely to buy a kind of local cookie that is quite dry and hard. For people who takes plane or takes short trip, the local cookie is not popular for them. I am assuming because this cookie can be stored and carried for long time, so people would like to buy the cookie as snacks for the road. While people who takes shore trip or takes plane does not necessarily need long-lasting food.

Question 3 a What are the 10 least frequently purchased items?

```
groceries <- read_csv('~/Downloads/groceries.csv')
```

```
## Warning: One or more parsing issues, see ‘problems()‘ for details
```

```
## Rows: 9834 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): citrus fruit, semi-finished bread, margarine, ready soups
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
groceries_2 <- read.transactions('~/Downloads/groceries.csv', sep = ',')
```

```
summary(groceries_2)
```

```
## 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
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
s <- summary(groceries_2)
print(s)
```

```
## 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
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
groceries_frequency =tibble(Items = names(itemFrequency(groceries_2)),
                            Frequency = itemFrequency(groceries_2))
groceries_frequency %>%
  arrange(desc(Frequency))%>%
  slice(160:169)
```

```
## # A tibble: 10 x 2
```

```
##    Items                 Frequency
##    <chr>                     <dbl>
##  1 salad dressing         0.000813
##  2 whisky                 0.000813
##  3 toilet cleaner         0.000712
##  4 baby cosmetics         0.000610
##  5 frozen chicken         0.000610
##  6 bags                   0.000407
##  7 kitchen utensil        0.000407
##  8 preservation products  0.000203
##  9 baby food              0.000102
## 10 sound storage medium   0.000102
```

```
#These are the 10 least frequency items
```

b If you change the minimum rule length to 3, how many rules to you generate? What if you change it to 4? (Use the same support / confidence thresholds used in the case study)

```
groceryrules_len3 =
  apriori(groceries_2,
          parameter =list(
            support = 0.015,
            confidence = 0.25,
            minlen = 3
          ) )
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.015      3
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 147
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [73 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [16 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
groceryrules_len4 =
  apriori(groceries_2,
          parameter =list(
            support = 0.015,
            confidence = 0.25,
```

```
          minlen = 4
         ) )
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.015      4
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 147
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [73 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(groceryrules_len3)
```

```
## set of 16 rules
##
## rule length distribution (lhs + rhs):sizes
##  3
## 16
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       3       3       3       3       3
##
## summary of quality measures:
##     support           confidence        coverage            lift
##  Min.   :0.01515   Min.   :0.2704   Min.   :0.02928   Min.   :1.510
##  1st Qu.:0.01556   1st Qu.:0.3067   1st Qu.:0.04230   1st Qu.:1.840
##  Median :0.01749   Median :0.4007   Median :0.04814   Median :2.016
##  Mean   :0.01865   Mean   :0.3905   Mean   :0.04984   Mean   :2.065
##  3rd Qu.:0.02227   3rd Qu.:0.4745   3rd Qu.:0.05618   3rd Qu.:2.212
##  Max.   :0.02318   Max.   :0.5174   Max.   :0.07483   Max.   :2.842
##      count
##  Min.   :149.0
##  1st Qu.:153.0
##  Median :172.0
##  Mean   :183.4
##  3rd Qu.:219.0
##  Max.   :228.0
##
## mining info:
```

```
##         data ntransactions support confidence
##  groceries_2          9835   0.015       0.25
##                                                                                          call
##  apriori(data = groceries_2, parameter = list(support = 0.015, confidence = 0.25, minlen = 3))
```

```
summary(groceryrules_len4)
```

```
## set of 0 rules
```

There will be 16 rules generated for 3 length and 0 rules for 4 length.

3.Change the minimum rule length back to 2 and produce a list of rules involving either soda or whipped/sour cream (you'll need to study the subset() function)

```
groceryrules_len2 =
  apriori(groceries_2,
          parameter =list(
            support = 0.015,
            confidence = 0.25,
            minlen = 2
          ) )
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.015      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 147
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [73 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [78 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
groceryrules_len2 %>%
  subset(items %in% c('whipped/sour cream','soda'))%>%
  inspect()
```

```
##     lhs                         rhs                support    confidence
## [1] {fruit/vegetable juice} => {soda}             0.01840366 0.2545710
## [2] {whipped/sour cream}    => {yogurt}           0.02074225 0.2893617
## [3] {whipped/sour cream}    => {other vegetables} 0.02887646 0.4028369
```

```
## [4] {whipped/sour cream}    => {whole milk}       0.03223183 0.4496454
## [5] {sausage}               => {soda}             0.02430097 0.2586580
## [6] {bottled water}         => {soda}             0.02897814 0.2621895
##     coverage   lift     count
## [1] 0.07229283 1.459887 181
## [2] 0.07168277 2.074251 204
## [3] 0.07168277 2.081924 284
## [4] 0.07168277 1.759754 317
## [5] 0.09395018 1.483324 239
## [6] 0.11052364 1.503577 285
```

##Part 2 1. Read the transactions into R

```
market_basket <- read.transactions('~/Downloads/Market_Basket_Optimisation.csv', sep = ",")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

2. Use the summary() function to answer the questions:

```
print(summary(market_basket))
```

```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03288973
##
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate
##          1788          1348          1306          1282          1229
##       (Other)
##         22405
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17    4
##   18   19   20
##    1    2    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.914   5.000  20.000
##
## includes extended item information - examples:
##              labels
## 1           almonds
## 2 antioxydant juice
## 3         asparagus
```

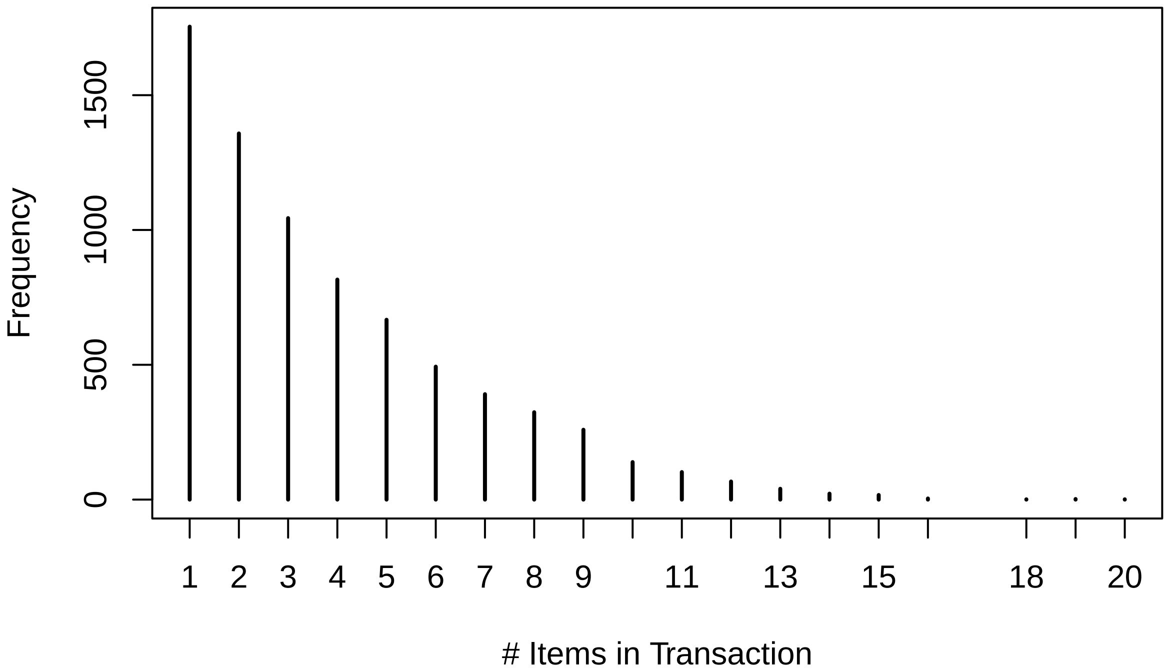
```
market_basket
```

```
## transactions in sparse format with
##  7501 transactions (rows) and
##  119 items (columns)
```

There are 7501 transactions in the data set. The number of distinct items is 119. The number of possible itemsets is 2^119-1 = 6.64614e+35.

3.Using the summary() function output, create a graph showing the distribution of transaction sizes in the data.

```
s = summary(market_basket)
sizes = s@lengths
plot(sizes,
     xlab = "# Items in Transaction",
     ylab = "Frequency")
```



4.Using the itemFrequency() function, create a dataset of items and their frequencies and determine the ten most frequent items, and the ten least frequent items.

```
library(tidyverse)
market_basket_freq = tibble(Items= names(itemFrequency(market_basket)),
                            Frequency = itemFrequency(market_basket))
market_basket_freq%>%
  arrange(desc(Frequency))%>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    Items                Frequency
##    <chr>                    <dbl>
##  1 mineral water          0.238
##  2 eggs                   0.180
##  3 spaghetti              0.174
##  4 french fries           0.171
##  5 chocolate              0.164
##  6 green tea              0.132
##  7 milk                   0.130
```

```
##   8 ground beef          0.0983
##   9 frozen vegetables    0.0953
##  10 pancakes             0.0951
```

```
market_basket_freq%>%
  arrange(desc(Frequency))%>%
  slice(110:119)
```

```
## # A tibble: 10 x 2
##    Items           Frequency
##    <chr>               <dbl>
##  1 ketchup          0.00440
##  2 oatmeal          0.00440
##  3 chocolate bread  0.00427
##  4 chutney          0.00413
##  5 mashed potato    0.00413
##  6 tea              0.00387
##  7 bramble          0.00187
##  8 cream            0.000933
##  9 napkins          0.000667
## 10 water spray      0.000400
```

5.Use descriptives statistics on the item frequencies to determine a reasonable support threshold (use confidence=0.25 and minlen = 2) and generate the association rules using the apriori algorithm.

```
#The midian value of item frequency can be minimum support threshold should, which is 0.016
market_basket_freq%>%
  select(Frequency)%>%
  summary()
```

```
##    Frequency
##  Min.   :0.0003999
##  1st Qu.:0.0077323
##  Median :0.0157312
##  Mean   :0.0328897
##  3rd Qu.:0.0381283
##  Max.   :0.2383682
```

```
market_basket_rules =
  apriori(market_basket,
          parameter =list(
            support = 0.016,
            confidence = 0.25,
            minlen = 2
          ) )
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.016      2
##  maxlen target  ext
```

```
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 120
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [59 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [42 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

6.Evaluate the rules and answer:

```
summary(market_basket_rules)
```

```
## set of 42 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 39  3
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.071   2.000   3.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.01626   Min.   :0.2506   Min.   :0.03919   Min.   :1.174
##  1st Qu.:0.01760   1st Qu.:0.2881   1st Qu.:0.05979   1st Qu.:1.440
##  Median :0.02286   Median :0.3295   Median :0.06839   Median :1.572
##  Mean   :0.02681   Mean   :0.3322   Mean   :0.08280   Mean   :1.648
##  3rd Qu.:0.02856   3rd Qu.:0.3670   3rd Qu.:0.09505   3rd Qu.:1.758
##  Max.   :0.05973   Max.   :0.4565   Max.   :0.23837   Max.   :2.908
##      count
##  Min.   :122.0
##  1st Qu.:132.0
##  Median :171.5
##  Mean   :201.1
##  3rd Qu.:214.2
##  Max.   :448.0
##
## mining info:
##          data ntransactions support confidence
##  market_basket          7501   0.016       0.25
##                                                                                     call
##  apriori(data = market_basket, parameter = list(support = 0.016, confidence = 0.25, minlen = 2))
```

1.There are 42 rules 2.There are 39 rules for two itemset, 3 rules for 3 itemset. 3.12 top cofident rule

```
market_basket_rules%>%
  sort(by='confidence')%>%
  head(n= 12)%>%
  inspect()
```

```
##      lhs                             rhs             support    confidence
## [1]  {soup}                       => {mineral water} 0.02306359 0.4564644
## [2]  {ground beef, spaghetti}     => {mineral water} 0.01706439 0.4353741
## [3]  {olive oil}                  => {mineral water} 0.02759632 0.4190283
## [4]  {ground beef, mineral water} => {spaghetti}     0.01706439 0.4169381
## [5]  {ground beef}                => {mineral water} 0.04092788 0.4165536
## [6]  {salmon}                     => {mineral water} 0.01706439 0.4012539
## [7]  {ground beef}                => {spaghetti}     0.03919477 0.3989145
## [8]  {cooking oil}                => {mineral water} 0.02013065 0.3942559
## [9]  {chicken}                    => {mineral water} 0.02279696 0.3800000
## [10] {frozen vegetables}          => {mineral water} 0.03572857 0.3748252
## [11] {milk}                       => {mineral water} 0.04799360 0.3703704
## [12] {tomatoes}                   => {mineral water} 0.02439675 0.3567251
##      coverage   lift     count
## [1]  0.05052660 1.914955 173
## [2]  0.03919477 1.826477 128
## [3]  0.06585789 1.757904 207
## [4]  0.04092788 2.394681 128
## [5]  0.09825357 1.747522 307
## [6]  0.04252766 1.683336 128
## [7]  0.09825357 2.291162 294
## [8]  0.05105986 1.653978 151
## [9]  0.05999200 1.594172 171
## [10] 0.09532062 1.572463 268
## [11] 0.12958272 1.553774 360
## [12] 0.06839088 1.496530 183
```

4.Printout the top 12 association rules by lift.

```
market_basket_rules%>%
  sort(by='lift')%>%
  head(n= 12)%>%
  inspect()
```

```
##      lhs                             rhs             support    confidence
## [1]  {mineral water, spaghetti}   => {ground beef}   0.01706439 0.2857143
## [2]  {ground beef, mineral water} => {spaghetti}     0.01706439 0.4169381
## [3]  {ground beef}                => {spaghetti}     0.03919477 0.3989145
## [4]  {olive oil}                  => {spaghetti}     0.02293028 0.3481781
## [5]  {olive oil}                  => {milk}          0.01706439 0.2591093
## [6]  {soup}                       => {mineral water} 0.02306359 0.4564644
## [7]  {herb & pepper}              => {spaghetti}     0.01626450 0.3288410
## [8]  {burgers}                    => {eggs}          0.02879616 0.3302752
## [9]  {ground beef, spaghetti}     => {mineral water} 0.01706439 0.4353741
## [10] {grated cheese}              => {spaghetti}     0.01653113 0.3155216
## [11] {olive oil}                  => {mineral water} 0.02759632 0.4190283
## [12] {tomatoes}                   => {spaghetti}     0.02093054 0.3060429
```

```
##      coverage   lift     count
## [1]  0.05972537 2.907928 128
## [2]  0.04092788 2.394681 128
## [3]  0.09825357 2.291162 294
## [4]  0.06585789 1.999758 172
## [5]  0.06585789 1.999567 128
## [6]  0.05052660 1.914955 173
## [7]  0.04946007 1.888695 122
## [8]  0.08718837 1.837830 216
## [9]  0.03919477 1.826477 128
## [10] 0.05239301 1.812196 124
## [11] 0.06585789 1.757904 207
## [12] 0.06839088 1.757755 157
```

7.Using the subset() function, printout the top 10 association rules by lift, that do not include the 6 most frequent items

```
freq_top6 <- market_basket_freq%>%
  arrange(desc(Frequency))%>%
  slice(1:6)%>%
  c()

market_basket_rules%>%
  subset(!items %in% freq_top6$Items)%>%
  sort(by = 'lift')%>%
  head(n = 10)%>%
  inspect()
```

```
##     lhs             rhs    support    confidence coverage   lift     count
## [1] {olive oil} => {milk} 0.01706439 0.2591093  0.06585789 1.999567 128
```

8.Discuss a couple of the rules you find most interesting and explain how you think they might be used in a retail context.

```
market_basket_rules%>%
  subset(items %in% c('burgers','eggs','milk'))%>%
  inspect()
```

```
##     lhs             rhs             support    confidence coverage   lift
## [1] {turkey}    => {eggs}          0.01946407 0.3113006  0.06252500 1.732245
## [2] {olive oil} => {milk}          0.01706439 0.2591093  0.06585789 1.999567
## [3] {burgers}   => {french fries}  0.02199707 0.2522936  0.08718837 1.476173
## [4] {burgers}   => {eggs}          0.02879616 0.3302752  0.08718837 1.837830
## [5] {burgers}   => {mineral water} 0.02439675 0.2798165  0.08718837 1.173883
## [6] {milk}      => {spaghetti}     0.03546194 0.2736626  0.12958272 1.571779
## [7] {milk}      => {mineral water} 0.04799360 0.3703704  0.12958272 1.553774
## [8] {eggs}      => {mineral water} 0.05092654 0.2833828  0.17970937 1.188845
##     count
## [1] 146
## [2] 128
## [3] 165
## [4] 216
```

```
## [5] 183
## [6] 266
## [7] 360
## [8] 382
```

According to this chart, we can find out there is a strong relationship between burgers and eggs. However, I do not see a relationship between eggs and milks. This reminds me a interesting point. I used to go to Costco for grocery shopping. Recently, the Costco I go to changed the eggs next to milk. The eggs were closer to burgers before. I think the manager of Costco made a wrong decision to change the place for eggs because there is not relationship strong relationship between eggs and milk.