HW4 F21

your name

10/15/2021

Setup

For this homework, you will investigate two new Association Rules and one already-done Rule, using the same data set with 9835 shoppers.

```
knitr::opts_chunk$set(echo = TRUE)

#  Load the tidyverse, or just dplyr
library(dplyr)
#  Read in the large groceries data set.
G <- read.csv('groceriesALL.csv')
```

```
#  Print the first few lines of the first 5 variables in the data frame
G %>% select(1:5) %>%
  head(3)
```

```
##   frankfurter sausage liver.loaf ham meat
## 1           0       0          0   0    0
## 2           0       0          0   0    0
## 3           0       0          0   0    0
```

```
#  Print the names of all of the variables in the data frame
names(G)
```

```
##   [1] "frankfurter"              "sausage"
##   [3] "liver.loaf"               "ham"
##   [5] "meat"                     "finished.products"
##   [7] "organic.sausage"          "chicken"
##   [9] "turkey"                   "pork"
##  [11] "beef"                     "hamburger.meat"
##  [13] "fish"                     "citrus.fruit"
##  [15] "tropical.fruit"           "pip.fruit"
##  [17] "grapes"                   "berries"
##  [19] "nuts.prunes"              "root.vegetables"
##  [21] "onions"                   "herbs"
##  [23] "other.vegetables"         "packaged.fruit.vegetables"
##  [25] "whole.milk"               "butter"
##  [27] "curd"                     "dessert"
##  [29] "butter.milk"              "yogurt"
##  [31] "whipped.sour.cream"       "beverages"
##  [33] "UHT.milk"                 "condensed.milk"
##  [35] "cream"                    "soft.cheese"
##  [37] "sliced.cheese"            "hard.cheese"
##  [39] "cream.cheese"             "processed.cheese"
##  [41] "spread.cheese"            "curd.cheese"
##  [43] "specialty.cheese"         "mayonnaise"
##  [45] "salad.dressing"           "tidbits"
##  [47] "frozen.vegetables"        "frozen.fruits"
##  [49] "frozen.meals"             "frozen.fish"
##  [51] "frozen.chicken"           "ice.cream"
##  [53] "frozen.dessert"           "frozen.potato.products"
##  [55] "domestic.eggs"            "rolls.buns"
##  [57] "white.bread"              "brown.bread"
##  [59] "pastry"                   "roll.products"
##  [61] "semi.finished.bread"      "zwieback"
##  [63] "potato.products"          "flour"
##  [65] "salt"                     "rice"
##  [67] "pasta"                    "vinegar"
##  [69] "oil"                      "margarine"
##  [71] "specialty.fat"            "sugar"
##  [73] "artif..sweetener"         "honey"
##  [75] "mustard"                  "ketchup"
##  [77] "spices"                   "soups"
##  [79] "ready.soups"              "Instant.food.products"
##  [81] "sauces"                   "cereals"
##  [83] "organic.products"         "baking.powder"
##  [85] "preservation.products"    "pudding.powder"
##  [87] "canned.vegetables"        "canned.fruit"
##  [89] "pickled.vegetables"       "specialty.vegetables"
##  [91] "jam"                      "sweet.spreads"
##  [93] "meat.spreads"             "canned.fish"
##  [95] "dog.food"                 "cat.food"
##  [97] "pet.care"                 "baby.food"
##  [99] "coffee"                   "instant.coffee"
## [101] "tea"                      "cocoa.drinks"
## [103] "bottled.water"            "soda"
## [105] "misc..beverages"          "fruit.vegetable.juice"
## [107] "syrup"                    "bottled.beer"
```

```
## [109] "canned.beer"               "brandy"
## [111] "whisky"                    "liquor"
## [113] "rum"                       "liqueur"
## [115] "liquor..appetizer."        "white.wine"
## [117] "red.blush.wine"            "prosecco"
## [119] "sparkling.wine"            "salty.snack"
## [121] "popcorn"                   "nut.snack"
## [123] "snack.products"            "long.life.bakery.product"
## [125] "waffles"                   "cake.bar"
## [127] "chewing.gum"               "chocolate"
## [129] "cooking.chocolate"         "specialty.chocolate"
## [131] "specialty.bar"             "chocolate.marshmallow"
## [133] "candy"                     "seasonal.products"
## [135] "detergent"                 "softener"
## [137] "decalcifier"               "dish.cleaner"
## [139] "abrasive.cleaner"          "cleaner"
## [141] "toilet.cleaner"            "bathroom.cleaner"
## [143] "hair.spray"                "dental.care"
## [145] "male.cosmetics"            "make.up.remover"
## [147] "skin.care"                 "female.sanitary.products"
## [149] "baby.cosmetics"            "soap"
## [151] "rubbing.alcohol"           "hygiene.articles"
## [153] "napkins"                   "dishes"
## [155] "cookware"                  "kitchen.utensil"
## [157] "cling.film.bags"           "kitchen.towels"
## [159] "house.keeping.products"    "candles"
## [161] "light.bulbs"               "sound.storage.medium"
## [163] "newspapers"                "photo.film"
## [165] "pot.plants"                "flower.soil.fertilizer"
## [167] "flower..seeds."            "shopping.bags"
## [169] "bags"
```

```
#  Setup -- 2 pts
```

Type the needed quantities below. For support, confidence and lift values, SHOW the numeric values that you divide to get the result (i.e., show your work).

Rule 1 {Yogurt} → {Whole Milk}

- Number of times Yogurt purchased: nYogurt = 1372
- Number of times Milk purchased: nMilk = 2513
- Number of times both purchased: nYogurtMilk = 3885
- Support of Yogurt: supYogurt = nYogurt / N = 1372 / 9835 = 0.14
- Support of Milk: supMilk = nMilk / N = 2513 / 9835 = 0.26
- Support of both: supYogurtMilk = nYogurtMilk / N = 3885 / 9835 = 0.40
- Confidence of the Rule: conf = supYogurtMilk / supYogurt = 0.40 / 0.14 = 2.83
- Lift of the Rule: lift = conf / supMilk = 1.55 / 0.26 = 11.08

```
# For each of the three rules, write R code to calculate all of the sums, as well a
s the support, confidence and lift values. Save the results as a data frame, as we
did in class, and print the data frame.  You should already have code for Rule 1; D
O include it here:
G %>% summarize(nYogurt = sum(yogurt),
                nMilk = sum(whole.milk),
                nYogurtAndMilk = nMilk + nYogurt,
                N = n(),
                supYogurt = nYogurt / N,
                supMilk = nMilk / N,
                supYogurtMilk = nYogurtAndMilk / N,
                confidence = supYogurtMilk / supYogurt,
                lift = confidence / supMilk )
```

```
##   nYogurt nMilk nYogurtAndMilk    N supYogurt  supMilk supYogurtMilk confidence
## 1    1372  2513           3885 9835 0.1395018 0.255516     0.3950178   2.831633
##       lift
## 1 11.08202
```

```
# Code worth 1 pts; writing numbers worth 1 pts.  (did this in class)
```

Rule 2 {Whole Milk} → {Yogurt}

- Number of times Milk purchased: nMilk = 2513
- Number of times Yogurt purchased: nYogurt = 1372
- Number of times both purchased: nMilkYogurt = 3885
- Support of Milk: supMilk = nMilk / N = 2513 / 9835 = 0.26
- Support of Yogurt: supYogurt = nYogurt / N = 1372 / 9835 = 0.14
- Support of both: supMilkYogurt = nMilkYogurt / N = 3885 / 9835 = 0.40
- Confidence of the Rule: conf = supMilkYogurt / supMilk = 0.40 / 0.26 = 1.55
- Lift of the Rule: lift = conf / supYogurt = 1.55 / 0.14 = 11.08

```
# Write similar R code below, for Rule 2.
G %>% summarize(nMilk = sum(whole.milk),
                 nYogurt = sum(yogurt),
                nMilkYogurt = nMilk + nYogurt,
                N = n(),
                supMilk = nMilk / N,
                supYogurt = nYogurt / N,
                supMilkYogurt = nMilkYogurt / N,
                confidence = supMilkYogurt / supMilk ,
                lift = confidence / supYogurt )
```

```
##   nMilk nYogurt nMilkYogurt    N  supMilk supYogurt supMilkYogurt confidence
## 1  2513    1372        3885 9835 0.255516 0.1395018     0.3950178   1.545961
##       lift
## 1 11.08202
```

```
# Code worth 2 pts; writing numbers worth 3 pts.
```

Rule 3 {Cereals} → {Yogurt}

- Number of times Cereals purchased: nCereal = 56
- Number of times Yogurt purchased: nYogurt = 1372
- Number of times both purchased: nCerealYogurt = 1428
- Support of Cereals: supCereal = nCereal / N = 56 / 9835 = 0.01
- Support of Yogurt: supYogurt = nYogurt / N = 1372 / 9835 = 0.14
- Support of both: supCerealYogurt = 1428 / 9835 = 0.15
- Confidence of the Rule: conf = supCerealYogurt / supCereal = 0.15 / 0.01 = 25.5
- Lift of the Rule: lift = conf / supYogurt = 25.5 / 0.14 = 182.79

(Because of rounding the calculations sometimes do not correspond with the result but they are correct taking into account all the decimals)

```
# Write similar R code below, for Rule 3.
G %>% summarize(nCereal = sum(cereals),
                nYogurt = sum(yogurt),
               nCerealYogurt = nCereal + nYogurt,
               N = n(),
               supCereal = nCereal / N,
               supYogurt = nYogurt / N,
               supCerealYogurt = nCerealYogurt / N,
               confidence = supCerealYogurt / supCereal ,
               lift = confidence / supYogurt )
```

```
##   nCereal nYogurt nCerealYogurt    N  supCereal supYogurt supCerealYogurt
## 1      56    1372          1428 9835 0.00569395 0.1395018       0.1451957
##   confidence     lift
## 1       25.5 182.7934
```

```
# Code worth 2 pts; writing numbers worth 3 pts.
```

Rule 4 {Liquor} → {Soda}

*Choose a food that you believe might predict sale of Soda. Show code and results for the rule, as you did for the three rules above (fill in the name of the food in all of the ___). Did you come up with a good rule to predict Soda purchases? Explain briefly. *

I chose liquor because it is very typical to drink liquor with a soda chaser. I believe that this is a good rule to predict Soda puchases because the Lift is bigger than 1, indicating a positive association between the two. Taking into account that the chances of buying liquor are low, as seen in its support, I think my instincts were correct.

- Number of times Liquor purchased: nLiquor = 109
- Number of times Soda purchased: nSoda = 1715
- Number of times both purchased: nLiquorSoda = 1824
- Support of Liquor: supLiquor = nLiquor / N = 109 / 9835 = 0.01
- Support of Soda: supSoda = nSoda / N = 1715 / 9835 = 0.17
- Support of both: supLiquorSoda = nLiquorSoda / N = 1824 / 9835 = 0.19
- Confidence of the Rule: conf = supLiquorSoda / supLiquor = 0.19 / 0.01 = 16.73
- Lift of the Rule: lift = conf / supSoda = 16.73 / 0.17 = 95.96

```
# Write similar R code below, for Rule 4.
G %>% summarize(nLiquor = sum(liquor),
                nSoda = sum(soda),
               nLiquorSoda = nLiquor + nSoda,
               N = n(),
               supLiquor = nLiquor / N,
               supSoda = nSoda / N,
               supLiquorSoda = nLiquorSoda / N,
               confidence = supLiquorSoda / supLiquor ,
               lift = confidence / supSoda )
```

```
##   nLiquor nSoda nLiquorSoda    N  supLiquor   supSoda supLiquorSoda confidence
## 1     109  1715        1824 9835 0.01108287 0.1743772     0.1854601   16.73394
##       lift
## 1 95.96405
```

```
# Code worth 2 pts; writing numbers worth 3 pts.
```

Question 1

Of Rule 2 and Rule 3, which is more useful in predicting yogurt purchases? Explain how you know. 1 pt

Both rules have a lift greater than 1 which signifies that the items we are trying to relate to Yogurt both have a positive association with Yogurt. However, Rule 3 is much more useful at predicting yogurt purchases because the lift is much higher than in Rule 2, meaning that it provides more information to understanding the purchase of the item.

Question 2

Rule 1 and Rule 2 both look at yogurt and milk. Did you get the same results for confidence and lift? For each of confidence and lift, explain why it is the same or different for the two rules. 2 pts

For confidence it will always be different, because association rules cannot be reversed and it would be very weird if both products had the exact same purchases (although it could happen).

The lift however has the following formula: Lift = confidence / supSecondItem

which if we develop it further it is: Lift = confidence / supSecondItem = (supFirstSecondItem / supFirstItem) / supSecondItem

Due to division rules, thanks to the way lift is formulated it will always be the same for firstItem and the secondItem.