Ex2_Associate_Rule_Learning

Markus Köfler

2023-04-12

The second exercise is about association rule learning. If not stated otherwise, use “retail2.csv” as data set. As always, you can use your own code. Otherwise, you can use the R code/functions provided on Moodle. **Association rule learning** is a popular, unsupervised learning technique for discovering interesting relations between variables based on transactions involving them in large databases. As it is often used for identifying shopping patterns, it is also known as market basket analysis.

```
# local folder / working directory
# retail <- read.csv("retail2.csv", sep=";")

# GitHub Link
retail <- read.csv(
  "https://raw.githubusercontent.com/MarkusStefan/Data_Analytics/main/Exercise2/retail2.csv?token=GHSAT0AAAAAACBJAQ7CZWQDFSY
IQT3UXJVGZBV267Q",
  sep=';')

retail[1:10 ,]
```

```
##    id Bread Yogurt Egg Dog_Food Flowers
## 1   1     1      1   1        0       0
## 2   2     0      0   1        1       0
## 3   3     0      0   0        0       1
## 4   4     1      0   1        0       0
## 5   5     0      1   0        0       0
## 6   6     1      0   0        0       1
## 7   7     0      0   0        1       0
## 8   8     0      0   1        0       0
## 9   9     1      1   0        0       0
## 10 10     1      1   0        0       0
```

(2.1)

In this first question you should practice on calculating support:

(a)

Calculate support for all sets consisting of one item, e.g. bread. Find two items with the highest/lowest support.

with $frq(A)$ as the number of occurrences from the total number of transactions $frq(T) = T$:

$$supp(A) = \frac{frq(A)}{T}$$

For one single set

```
# nrow counts the number of rows
frqA <- retail %>% subset(Bread==1) %>% nrow()
T. <- nrow(retail)
suppA <- frqA/T.
suppA
```

```
## [1] 0.67
```

```
retail[, 2 == 1]
```

```
## data frame with 0 columns and 100 rows
```

packing it into a loop to get the highest support value

```
highest_supp <- c(0, "")
T. <- nrow(retail)
for (i in colnames(retail) ) {
  #frqA <- retail %>% subset(i=1) %>% nrow()
  if (i == "id"){
    next
  }
  #frqA <- retail[, i == 1] %>% nrow()
  frqA <- (retail[[i]]==1) %>% sum()
  suppA <- frqA/T.
  cat(suppA*100, "% (", i, ")\n", sep="" )
  if (suppA > highest_supp[1]) {
    highest_supp[1] <- suppA
    highest_supp[2] <- i
  }
  else {
    next
  }
}
```

```
## 67% (Bread)
## 60% (Yogurt)
## 42% (Egg)
## 22% (Dog_Food)
## 15% (Flowers)
```

```
cat("\nhighest support:\t", (as.numeric(highest_supp[1])*100),
    "% (", highest_supp[2],")", sep="")
```

```
##
## highest support: 67% (Bread)
```

(b)

Calculate support for all sets consisting of two items, e.g. (bread,yogurt). Find two item sets with the highest/lowest support.

```
# creating a list of vector-tuples representing all combinations
bi_sets <- list(c("Bread", "Yogurt"), c("Bread", "Egg"),
                c("Bread", "Dog_Food"), c("Bread", "Flowers"),
                c("Yogurt", "Egg"), c("Yogurt", "Dog_Food"),
                c("Yogurt", "Flowers"), c("Egg", "Dog_Food"),
                c("Egg", "Flowers"), c("Dog_Food", "Flowers"))

T. <- nrow(retail)
for (pair in bi_sets) {
  A <- pair[1]
  B <- pair[2]
  suppAB <- sum(retail[[A]] == 1 & retail[[B]] == 1) / T.
  cat("The support for ", A, " and ", B, " is ",
      suppAB*100, "%\n", sep="")
}
```

```
## The support for Bread and Yogurt is 55%
## The support for Bread and Egg is 27%
## The support for Bread and Dog_Food is 12%
## The support for Bread and Flowers is 6%
## The support for Yogurt and Egg is 22%
## The support for Yogurt and Dog_Food is 9%
## The support for Yogurt and Flowers is 4%
## The support for Egg and Dog_Food is 7%
## The support for Egg and Flowers is 3%
## The support for Dog_Food and Flowers is 2%
```

(2.2)

Calculate the support, the confidence, and the lift for all rules with yogurt as LHS and one item as RHS.

Support: It indicates how frequently an item set appears in the data set.

```
# -2 to disregard the index and yoghurt columns
T. <- nrow(retail)
for (i in colnames(retail)[-c(1,3)]) {
  suppAB <- sum(retail$Yogurt == 1 & retail[[i]] == 1) / T.
  cat(paste("The support for Yogurt and", i, "is",
            suppAB*100, "%\n"), sep="")
}
```

```
## The support for Yogurt and Bread is 55 %
## The support for Yogurt and Egg is 22 %
## The support for Yogurt and Dog_Food is 9 %
## The support for Yogurt and Flowers is 4 %
```

Confidence: It says how likely item set B is purchased when item set A is purchased

$$conf(A \rightarrow B) = \frac{supp(A,B)}{supp(A)}$$

with $supp(A,B) = \frac{frq(A,B)}{frq(T)}$ and $supp(A) = \frac{frq(A)}{frq(T)}$ it holds that:

$$conf(A \rightarrow B) = \frac{supp(A,B)}{supp(A)} = \frac{\frac{frq(A,B)}{frq(T)}}{\frac{frq(A)}{frq(T)}} = \frac{frq(A,B)}{frq(A)}$$

```
# -2 to disregard the index and yoghurt columns
for (i in colnames(retail)[-c(1,3)]) {
  # note that the division by T. of both suppAB and suppB is redundant
  suppAB <- sum(retail$Yogurt == 1 & retail[[i]] == 1) / T.
  suppA <- sum(retail$Yogurt == 1) / T.
  confAB <- suppAB / suppA
  cat(paste("The confidence for Yogurt and", i, "is",
            round(confAB,4)*100, "%\n"), sep="")
}
```

```
## The confidence for Yogurt and Bread is 91.67 %
## The confidence for Yogurt and Egg is 36.67 %
## The confidence for Yogurt and Dog_Food is 15 %
## The confidence for Yogurt and Flowers is 6.67 %
```

Lift: It says how likely item set B is purchased when item set A is purchased while controlling for how popular item set B is.

Lift is the ratio of the observed support to that expected if A and B were independent or equivalently the ratio of the confidence of the rule to the expected confidence of the RHS item set by independence.

$$lift(A \rightarrow B) = \frac{conf(A,B)}{supp(B)} = \frac{supp(A,B)}{supp(A) \times supp(B)}$$

Note:

$$lift(A \rightarrow B) == lift(B \rightarrow A)$$

```
# -2 to disregard the index and yoghurt columns
for (i in colnames(retail)[-c(1,3)]) {

  suppAB <- sum(retail$Yogurt == 1 & retail[[i]] == 1) / T.
  suppA <- sum(retail$Yogurt == 1) / T.
  suppB <- sum(retail[[i]] == 1) / T.
  confAB <- suppAB / suppA
  # first method
  liftAB <- confAB / suppB
  # second method
  liftAB <- suppAB / (suppA * suppB)
  cat(paste("The lift for Yogurt and", i, "is",
            round(liftAB,4), "\n"), sep="")
}
```

```
## The lift for Yogurt and Bread is 1.3682
## The lift for Yogurt and Egg is 0.873
## The lift for Yogurt and Dog_Food is 0.6818
## The lift for Yogurt and Flowers is 0.4444
```

(2.3)

Calculate the support, the confidence, and the lift for all rules with one item as LHS and one item as RHS, e.g. bread → yogurt and yogurt → bread. Identify three interesting rules.

Note that if:

- $lift = 1$, the occurrence of A and B are independent of each other
- $lift < 1$, the occurrence of A has a negative effect on the occurrence of B, and vice versa
- $lift > 1$, the two occurrences of A and B depend on each other

```
bi_sets <- list(c("Bread", "Yogurt"), c("Bread", "Egg"),
                c("Bread", "Dog_Food"), c("Bread", "Flowers"),
                c("Yogurt", "Egg"), c("Yogurt", "Dog_Food"),
                c("Yogurt", "Flowers"), c("Egg", "Dog_Food"),
                c("Egg", "Flowers"), c("Dog_Food", "Flowers"))

T. <- nrow(retail)
for (pair in bi_sets) {
  A <- pair[1]
  B <- pair[2]
  suppAB <- sum(retail[[A]] == 1 & retail[[B]] == 1) / T.
  suppA <- sum(retail[[A]] == 1)/T.; suppB <- sum(retail[[B]] == 1)/T.;
  confAB <- suppAB / suppA; confBA <- suppAB / suppB;
  liftAB <- confAB / suppB; liftBA <- confBA / suppA;
  liftAB <- suppAB / (suppA*suppB); liftBA <- suppAB / (suppB*suppA)

  cat("\nsupp(", A, ",", B, ") \t=\t", suppAB*100, "%\n")
  cat("conf(", A, "->", B,")\t=\t",confAB*100,
      "%\nconf(", B,"->", A, ")\t=\t", confBA*100,"%\n")
  cat("lift(", A, "->", B,")\t=\t",liftAB,
      "\nlift(", B,"->", A, ")\t=\t", liftBA,"\n")
}
```

```
##
## supp( Bread , Yogurt )   =    55 %
## conf( Bread -> Yogurt )  =    82.08955 %
## conf( Yogurt -> Bread )  =    91.66667 %
## lift( Bread -> Yogurt )  =    1.368159
## lift( Yogurt -> Bread )  =    1.368159
##
## supp( Bread , Egg )  =    27 %
## conf( Bread -> Egg ) =    40.29851 %
## conf( Egg -> Bread ) =    64.28571 %
## lift( Bread -> Egg ) =    0.9594883
## lift( Egg -> Bread ) =    0.9594883
##
## supp( Bread , Dog_Food )     =    12 %
## conf( Bread -> Dog_Food )    =    17.91045 %
## conf( Dog_Food -> Bread )    =    54.54545 %
## lift( Bread -> Dog_Food )    =    0.8141113
## lift( Dog_Food -> Bread )    =    0.8141113
##
## supp( Bread , Flowers )  =    6 %
## conf( Bread -> Flowers ) =    8.955224 %
## conf( Flowers -> Bread ) =    40 %
## lift( Bread -> Flowers ) =    0.5970149
## lift( Flowers -> Bread ) =    0.5970149
##
## supp( Yogurt , Egg )     =    22 %
## conf( Yogurt -> Egg )    =    36.66667 %
## conf( Egg -> Yogurt )    =    52.38095 %
## lift( Yogurt -> Egg )    =    0.8730159
## lift( Egg -> Yogurt )    =    0.8730159
##
## supp( Yogurt , Dog_Food )    =    9 %
## conf( Yogurt -> Dog_Food )   =    15 %
## conf( Dog_Food -> Yogurt )   =    40.90909 %
## lift( Yogurt -> Dog_Food )   =    0.6818182
## lift( Dog_Food -> Yogurt )   =    0.6818182
##
## supp( Yogurt , Flowers )     =    4 %
## conf( Yogurt -> Flowers )    =    6.666667 %
## conf( Flowers -> Yogurt )    =    26.66667 %
## lift( Yogurt -> Flowers )    =    0.4444444
## lift( Flowers -> Yogurt )    =    0.4444444
##
## supp( Egg , Dog_Food )   =    7 %
## conf( Egg -> Dog_Food )  =    16.66667 %
## conf( Dog_Food -> Egg )  =    31.81818 %
## lift( Egg -> Dog_Food )  =    0.7575758
## lift( Dog_Food -> Egg )  =    0.7575758
##
## supp( Egg , Flowers )    =    3 %
## conf( Egg -> Flowers )   =    7.142857 %
## conf( Flowers -> Egg )   =    20 %
## lift( Egg -> Flowers )   =    0.4761905
## lift( Flowers -> Egg )   =    0.4761905
##
## supp( Dog_Food , Flowers )   =    2 %
## conf( Dog_Food -> Flowers )  =    9.090909 %
## conf( Flowers -> Dog_Food )  =    13.33333 %
## lift( Dog_Food -> Flowers )  =    0.6060606
## lift( Flowers -> Dog_Food )  =    0.6060606
```

Interpretation:

- The most frequently bought bundle is {Bread, Yoghurt} with 55%, followed by {Yoghurt, Egg} with 22%.
- Surprisingly, the likelihood of “Yoghurt” being bought is 40% if “Dog_Food” is bought. Attention to the interpretation: If the good of the LHS is purchased very frequently, the confidence will most likely be very high, despite the weak relationship. Lift is the better measure to overcome this issue.
- Looking at the patterns, we may draw the conclusion that products and foods which are regarded as staples of the average households consumption are, of course, frequently bought together. This does not imply that they are necessarily complements to each other.
- The results suggest, that only the item set {Bread -> Yoghurt} (and vice versa yields the same result) are positively correlated, for all other item sets, they appear to negatively influence each other in terms of being purchased together. Hence, “Bread” and “Yoghurt” seem to complement each other.

(2.4)

Calculate the support, the confidence, and the lift for all rules with **(bread, egg)** as LHS and one item as RHS, e.g. (bread, egg) → yogurt.

```
# Subset data to include only rows where bread and egg are both 1
sub_retail <- retail[retail$Bread == 1 & retail$Egg == 1,]

T. <- nrow(sub_retail)
# excluding id, bread and egg columns
for(i in colnames(sub_retail)[-c(1,2,4)]) {
  # Calculate support
  suppA <- sum(sub_retail[,c("Bread","Egg",i)] == 1) / T.

  # Calculate confidence
  confAB <- sum(sub_retail[,c("Bread","Egg",i)] == 1) / sum(sub_retail$Bread == 1 & sub_retail$Egg == 1)

  # Calculate lift
  liftAB <- suppA / ((sum(sub_retail$Bread == 1 & sub_retail$Egg == 1) / T.) * (sum(sub_retail[,i] == 1) / T.))

  # Print results
  cat(sprintf("{(%s, %s) -> %s}: support=%.2f, confidence=%.2f, lift=%.2f\n", "Bread", "Egg", i, suppA, confAB, liftAB))
}
```

```
## {(Bread, Egg) -> Yogurt}: support=2.81, confidence=2.81, lift=3.45
## {(Bread, Egg) -> Dog_Food}: support=2.15, confidence=2.15, lift=14.50
## {(Bread, Egg) -> Flowers}: support=2.04, confidence=2.04, lift=55.00
```

(2.5)

Apriori algorithm

- Apriori algorithm is the most popular algorithm used for association rule mining. The objective is to find subsets that are common to at least a minimum number of the item sets. A frequent item set is an item set whose support is greater than or equal to minimum support threshold.
- The Apriori property is a downward closure property, which means that all nonempty subsets of a frequent item set must also be frequent.
- Apriori algorithm uses a bottom-up approach; and the size of frequent subsets is gradually increased, from one-item subsets to two-item subsets, then three-item subsets, and so on. Groups of candidates at each level are tested against the data for minimum support.

(a)

Use “Groceries” data set. Calculate all interesting (non-empty) rules with minimal support = 0.01 and minimal confidence = 0.4. Identify three rules with the highest support, three rules with the highest confidence and three rules with the highest lift.

```
library(arules)
library(datasets)

data("Groceries")

Groceries
```

```
## transactions in sparse format with
##  9835 transactions (rows) and
##  169 items (columns)
```

```
Groceries_rules <- apriori(Groceries,
                           parameter = list(support = 0.01,
                                            confidence = 0.4))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5    0.01      1
##  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: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [62 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# top 3 ruels with highest support
inspect(sort(Groceries_rules, by = "support",
             decreasing = TRUE)[1:3])
```

```
##     lhs                  rhs                support    confidence coverage
## [1] {yogurt}          => {whole milk}       0.05602440 0.4016035  0.1395018
## [2] {root vegetables} => {whole milk}       0.04890696 0.4486940  0.1089985
## [3] {root vegetables} => {other vegetables} 0.04738180 0.4347015  0.1089985
##     lift     count
## [1] 1.571735 551
## [2] 1.756031 481
## [3] 2.246605 466
```

```
# top 3 rules with highest confidence
inspect(sort(Groceries_rules, by = "confidence",
             decreasing = TRUE)[1:3])
```

```
##     lhs                                rhs                support
## [1] {citrus fruit, root vegetables}   => {other vegetables} 0.01037112
## [2] {tropical fruit, root vegetables} => {other vegetables} 0.01230300
## [3] {curd, yogurt}                    => {whole milk}       0.01006609
##     confidence coverage   lift     count
## [1] 0.5862069  0.01769192 3.029608 102
## [2] 0.5845411  0.02104728 3.020999 121
## [3] 0.5823529  0.01728521 2.279125  99
```

```
# top 3 rules with highest lift
inspect(sort(Groceries_rules, by = "lift",
             decreasing = TRUE)[1:3])
```

```
##     lhs                                rhs                support
## [1] {citrus fruit, root vegetables}   => {other vegetables} 0.01037112
## [2] {tropical fruit, root vegetables} => {other vegetables} 0.01230300
## [3] {root vegetables, rolls/buns}     => {other vegetables} 0.01220132
##     confidence coverage   lift     count
## [1] 0.5862069  0.01769192 3.029608 102
## [2] 0.5845411  0.02104728 3.020999 121
## [3] 0.5020921  0.02430097 2.594890 120
```

Interpretation:

This means that 5.6% of all transactions contain both “yoghurt” and “whole milk”, and that among those transactions, if “yoghurt” is purchased, the likelihood for “whole milk” to be purchased is 40.2% (so, of all purchases containing “yoghurt”, 40.2% of them also contain “whole milk”). The lift value of 1.57 indicates that the presence of “yoghurt” and “whole milk” in a transaction is 1.57 times more likely than would be expected if the two items were independent.

Data Mining:

- The top 3 bundles with the highest support value, consist of ~more healthy food choices, which is nice to see. To be more precise, they consist of dairy and vegetable products.
- support rankings are comprised of sets of 2 items; the confidence and lift rankings consist of item sets of 2 items, whereas the LHS is a tuple of 2 items
- the confidence ranking suggests, that if products from one category are bought, then there is a slightly higher than 58% chance that other items from the same product category are bought. E.g.: 58% confidence for other vegetables to be purchased in conjunction to citrus fruits and root vegetables (category: fruits & vegetables); similarily, we can be confident that 58% of purchases indlucing curd and yoghurt also include whole milk (category: dairy).
- The bundles with the highest confidence have (roughly) the highest lift values, indicating a (strongly) positive relationship.
- The highest lift values are comprised of “other vegetables” on the RHS.

(b)

Use “Groceries” data set. Calculate all interesting (non-empty) rules with minimal support = 0.005 and minimal confidence = 0.3. Identify three rules with the highest support, three rules with the highest confidence and three rules with the highest lift.

```
Groceries_rules <- apriori(Groceries,
                           parameter = list(support = 0.005,
                                            confidence = 0.3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5   0.005      1
##  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: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [482 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# top 3 ruels with highest support
inspect(sort(Groceries_rules, by = "support",
             decreasing = TRUE)[1:3])
```

```
##     lhs                   rhs          support    confidence coverage  lift
## [1] {other vegetables} => {whole milk} 0.07483477 0.3867578  0.1934926 1.513634
## [2] {rolls/buns}       => {whole milk} 0.05663447 0.3079049  0.1839349 1.205032
## [3] {yogurt}           => {whole milk} 0.05602440 0.4016035  0.1395018 1.571735
##     count
## [1] 736
## [2] 557
## [3] 551
```

```
# top 3 rules with highest confidence
inspect(sort(Groceries_rules, by = "confidence",
             decreasing = TRUE)[1:3])
```

```
##     lhs                     rhs              support confidence    coverage     lift count
## [1] {tropical fruit,
##      root vegetables,
##      yogurt}             => {whole milk} 0.005693950      0.700 0.008134215 2.739554    56
## [2] {pip fruit,
##      root vegetables,
##      other vegetables}   => {whole milk} 0.005490595      0.675 0.008134215 2.641713    54
## [3] {butter,
##      whipped/sour cream} => {whole milk} 0.006710727      0.660 0.010167768 2.583008    66
```

```
# top 3 rules with highest lift
inspect(sort(Groceries_rules, by = "lift",
             decreasing = TRUE)[1:3])
```

```
##     lhs                    rhs                   support confidence   coverage     lift count
## [1] {citrus fruit,
##      other vegetables,
##      whole milk}        => {root vegetables} 0.005795628  0.4453125 0.01301474 4.085493    57
## [2] {herbs}             => {root vegetables} 0.007015760  0.4312500 0.01626843 3.956477    69
## [3] {citrus fruit,
##      pip fruit}         => {tropical fruit}  0.005592272  0.4044118 0.01382816 3.854060    55
```

Data Mining:

- Now, the highest lift values become even larger
- For the ranking of the top 3 bundles according to lift value, the highest ranking one consists of a tuple of 3 items on the LHS. Thereby, not all items are of the same product category

(c)

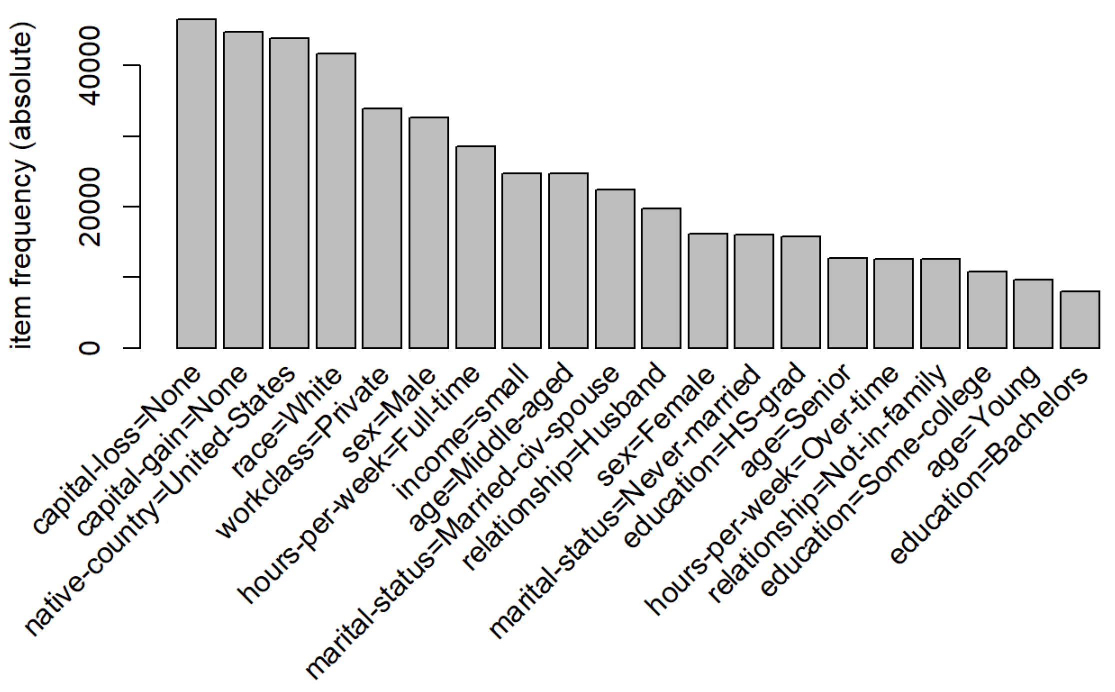
Use “Adult” data set[1] . Calculate all interesting (non-empty) rules with minimal support = 0.5 and minimal confidence = 0.9. Identify three rules with the highest support, three rules with the highest confidence and three rules with the highest lift.

[1] For more information see: [Adults]{http://archive.ics.uci.edu/ml/datasets/Adult (http://archive.ics.uci.edu/ml/datasets/Adult)}.

Note: Labels have are stated on the website. Data set was originally intended for classification purposes (prediction whether a person makes over 50K a year) deploying machine learning algorithms such as Naive-Bayes, Logistic Regression, K-Means Clustering or Artificial Neural Networks.

```
url <- "https://raw.githubusercontent.com/MarkusStefan/Data_Analytics/main/Exercise2/adult.data"
adults <- read.table(url, header = FALSE, sep = ",") %>% data.frame()
header <- c("age", "workclass", "fnlwgt", "education", "education-num",
            "merital-status", "occupation", "relationship", "race"
            ,"sex", "capital-gain", "capital-loss","hours-per-week",
            "native-country", "income")
colnames(adults) <- header
#adults %>% View()
```

```
library(arules)
library(datasets)
data("Adult")
library(arulesViz)
itemFrequencyPlot(Adult, type = "absolute", topN = 20)
```



(d)

Use “Adult” data set. Calculate all interesting (non-empty) rules with minimal support = 0.4 and minimal confidence = 0.8. Identify three rules with the highest support, three rules with the highest confidence and three rules with the highest lift.

```
rules <- apriori(Adult, parameter =
                   list(support = 0.4, confidence = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.4      1
##  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: 19536
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.03s].
## sorting and recoding items ... [11 item(s)] done [0.00s].
## creating transaction tree ... done [0.02s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [169 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# rules by support
rules_supp <- sort(rules, by = "support", decreasing = TRUE)

# top three rules by support
top_rules_supp <- head(rules_supp, n = 3)

# rules by confidence
rules_conf <- sort(rules, by = "confidence", decreasing = TRUE)

# top three rules by confidence
top_rules_conf <- head(rules_conf, n = 3)

# rules by lift
rules_lift <- sort(rules, by = "lift", decreasing = TRUE)

# Get the top three rules by lift
top_rules_lift <- head(rules_lift, n = 3)
```

```
inspect(top_rules_supp)
```

```
##     lhs    rhs                            support   confidence coverage lift
## [1] {}  => {capital-loss=None}            0.9532779 0.9532779  1        1
## [2] {}  => {capital-gain=None}            0.9173867 0.9173867  1        1
## [3] {}  => {native-country=United-States} 0.8974243 0.8974243  1        1
##     count
## [1] 46560
## [2] 44807
## [3] 43832
```

```
#cat("\n")
inspect(top_rules_conf)
```

```
##     lhs                                     rhs                                  support confidence  coverage     lift c
ount
## [1] {relationship=Husband}               => {sex=Male}                         0.4036485  0.9999493 0.4036690 1.495851 1
9715
## [2] {marital-status=Married-civ-spouse,
##      relationship=Husband}               => {sex=Male}                         0.4034028  0.9999492 0.4034233 1.495851 1
9703
## [3] {relationship=Husband}               => {marital-status=Married-civ-spouse} 0.4034233  0.9993914 0.4036690 2.181164 1
9704
```

```
#cat("\n")
inspect(top_rules_lift)
```

```
##     lhs                                     rhs                                  support confidence  coverage     lift c
ount
## [1] {marital-status=Married-civ-spouse,
##      sex=Male}                           => {relationship=Husband}             0.4034028  0.9901503 0.4074157 2.452877 1
9703
## [2] {marital-status=Married-civ-spouse}  => {relationship=Husband}             0.4034233  0.8804683 0.4581917 2.181164 1
9704
## [3] {relationship=Husband}               => {marital-status=Married-civ-spouse} 0.4034233  0.9993914 0.4036690 2.181164 1
9704
```

Data Mining:

- not very useful conclusions, as some variables are linked, hence, imply each other. E.g.: if relationship = Husband, intuition tells us that the sex must be male; or if the merital-status = Married-civ-spouse, then we can induce that the subject’s relationship = Husband