HW 9

April 26, 2021

1 IST 387 HW 9

[4]:
```
# Enter your name here: Connor Hanan
```

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

[5]:
```
# 1. I did this homework by myself, with help from the book and the professor.
```

Association mining can be applied to many data problems beyond the well-known example of **finding relationships between different products in customer shopping data**. In this homework assignment, we will explore **real data** from the banking sector and look for **patterns associated with the likelihood of responding positively to a direct marketing campaign and signing up for a term deposit with the bank (stored in the variable "y")**. You can find out more about the variables in this dataset here: https://archive.ics.uci.edu/ml/datasets/bank+marketing

1.1 Part 1: Explore Data Set

A. Copy the contents of the following URL to a dataframe called bank: https://ist387.s3.us-east-2.amazonaws.com/data/bank-full.csv

Hint: Even though this is a .csv file, chances are R won't be able to read it in correctly using the read_csv() function. If you take a closer look at the contents of the URL file, you may notice each field is separated by a **semicolon** (;) rather than a comma. In situations like this, consider using something like this:

[6]:
```
bank <- read.table("https://ist387.s3.us-east-2.amazonaws.com/data/bank-full.
↪csv", sep=";", header = TRUE)
```

[7]:
```
str(bank)
```

```
'data.frame':   41188 obs. of  21 variables:
 $ age           : int  56 57 37 40 56 45 59 41 24 25 …
 $ job           : chr  "housemaid" "services" "services" "admin." …
 $ marital       : chr  "married" "married" "married" "married" …
 $ education     : chr  "basic.4y" "high.school" "high.school" "basic.6y" …
 $ default       : chr  "no" "unknown" "no" "no" …
```

```
 $ housing       : chr  "no" "no" "yes" "no" …
 $ loan          : chr  "no" "no" "no" "no" …
 $ contact       : chr  "telephone" "telephone" "telephone" "telephone" …
 $ month         : chr  "may" "may" "may" "may" …
 $ day_of_week   : chr  "mon" "mon" "mon" "mon" …
 $ duration      : int  261 149 226 151 307 198 139 217 380 50 …
 $ campaign      : int  1 1 1 1 1 1 1 1 1 …
 $ pdays         : int  999 999 999 999 999 999 999 999 999 999 …
 $ previous      : int  0 0 0 0 0 0 0 0 0 …
 $ poutcome      : chr  "nonexistent" "nonexistent" "nonexistent" "nonexistent"
…
 $ emp.var.rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 …
 $ cons.price.idx: num  94 94 94 94 94 …
 $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4
-36.4 …
 $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 …
 $ nr.employed   : num  5191 5191 5191 5191 5191 …
 $ y             : chr  "no" "no" "no" "no" …
```

Make sure there are **41,188** rows and **21** columns in your **bank** df.

B. Next, we will focus on some key factor variables from the dataset, and convert a few numeric ones to factor variables. Execute the following commands and write a comment describing how the conversion for each numeric variable works and what the variables in the resulting dataframe are.

[8]:
```
bank_new <- data.frame(job=bank$job,
                       marital=bank$marital,
                       housing_loan=bank$housing,
                       young=as.factor((bank$age<median(bank$age))),
                       contacted_more_than_once=as.factor((bank$campaign>1)),
                       contacted_before_this_campaign=as.
↪factor((bank$previous<0)),
                       success=(bank$y))
```

C. Count the number of successful term deposit sign-ups, using the table( ) command on the **success** variable.

[10]:
```
table(bank$y)
```

```
   no   yes
36548  4640
```

D. Express the results of problem C as percentages by sending the results of the table( ) command into the prop.table( ) command.

[12]:
```
prop.table(table(bank$y))
```

```
       no       yes
0.8873458 0.1126542
```

E. Using the same techniques, show the percentages for the **marital** and **housing_loan** variables as well.

[13]:
```
prop.table(table(bank$marital))
prop.table(table(bank$housing))
```

```
   divorced     married      single     unknown
0.111974361 0.605224823 0.280858502 0.001942313
```

```
        no    unknown        yes
0.45212198 0.02403613 0.52384190
```

1.2 Part 2: Coerce the data frame into transactions

F. Install and library two packages: **arules** and **arulesViz**.

[15]:
```
install.packages("arules")
install.packages('arulesViz')
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done

also installing the dependencies ‘viridisLite’, ‘gridExtra’, ‘viridis’,
‘gtools’, ‘caTools’, ‘TSP’, ‘qap’, ‘cluster’, ‘gclus’, ‘dendextend’, ‘gplots’,
‘registry’, ‘htmlwidgets’, ‘seriation’, ‘vcd’, ‘igraph’, ‘scatterplot3d’,
‘plotly’, ‘visNetwork’


Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done
```

[16]:
```
library(arules)
library(arulesViz)
library(tidyverse)
```

```
Loading required package: Matrix


Attaching package: ‘arules’
```

```
The following objects are masked from 'package:base':

    abbreviate, write


  Attaching packages                                  tidyverse
1.3.0

  ggplot2 3.3.2      purrr   0.3.4
  tibble  3.0.4      dplyr   1.0.2
  tidyr   1.1.2      stringr 1.4.0
  readr   1.4.0      forcats 0.5.0

  Conflicts
tidyverse_conflicts()
  tidyr::expand() masks Matrix::expand()
  dplyr::filter() masks stats::filter()
  dplyr::lag()    masks stats::lag()
  tidyr::pack()   masks Matrix::pack()
  dplyr::recode() masks arules::recode()
  tidyr::unpack() masks Matrix::unpack()
```

G. Coerce the **bank_new** dataframe into a **sparse transactions matrix** called **bankX**.

[17]:
```
bankX <- as(bank_new,'transactions')
```

```
Warning message:
"Column(s) 1, 2, 3, 7 not logical or factor. Applying default discretization
(see '? discretizeDF')."
```

H. Use the itemFrequency( ) and itemFrequencyPlot( ) commands to explore the contents of **bankX**. What do you see?

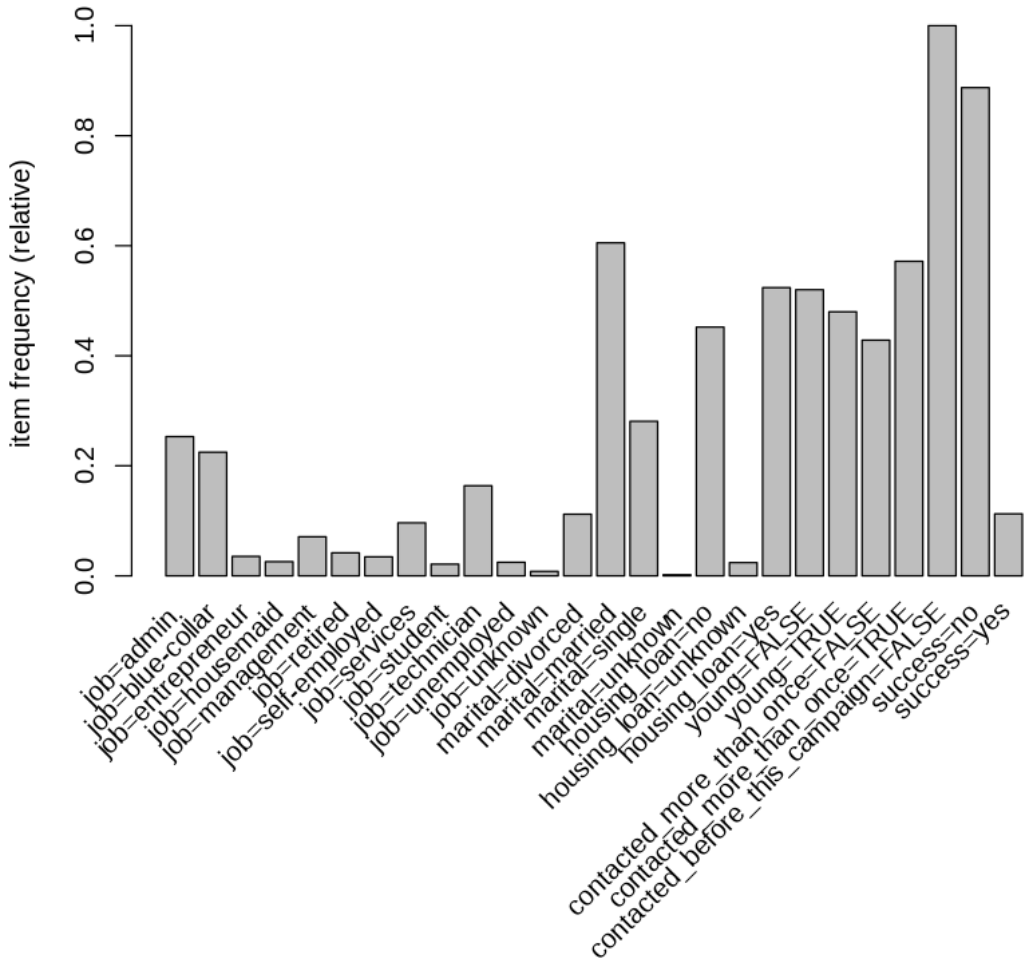
[18]:
```
itemFrequency(bankX)

itemFrequencyPlot(bankX)

#a large majority of the attempts ended in failure, regarless of the varying␣
 ↪situations that the people found themselves in
```

job=admin. 0.253034864523648 **job=blue-collar** 0.224677090414684 **job=entrepreneur** 0.035350101971448 **job=housemaid** 0.0257356511605322 **job=management** 0.0709915509371662 **job=retired** 0.0417597358453919 **job=self-employed** 0.0345003399048266 **job=services** 0.0963630183548606 **job=student** 0.0212440516655336 **job=technician** 0.16371273186365 **job=unemployed** 0.0246188210158299 **job=unknown** 0.00801204234242983 **marital=divorced** 0.111974361464504 **marital=married** 0.605224822763912 **marital=single** 0.280858502476449 **marital=unknown** 0.00194231329513451 **housing_loan=no**

0.452121977274934 **housing_loan=unknown** 0.0240361270272895 **housing_loan=yes**
0.523841895697776 **young=FALSE** 0.520054384772264 **young=TRUE** 0.479945615227736
contacted_more_than_once=FALSE 0.428328639409537
contacted_more_than_once=TRUE 0.571671360590463
contacted_before_this_campaign=FALSE 1 **success=no** 0.887345828882199
success=yes 0.112654171117801



I. This is a fairly large dataset, so we will explore only the first 10 observations in the **bankX** transaction matrix:

[19]:
```
inspect(bankX[1:10])
```

```
    items                                   transactionID
[1] {job=housemaid,
```

```
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    1
[2]  {job=services,
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    2
[3]  {job=services,
      marital=married,
      housing_loan=yes,
      young=TRUE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    3
[4]  {job=admin.,
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    4
[5]  {job=services,
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    5
[6]  {job=services,
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    6
[7]  {job=admin.,
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    7
```

```
[8]  {job=blue-collar,
      marital=married,
      housing_loan=no,
      young=FALSE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    8
[9]  {job=technician,
      marital=single,
      housing_loan=yes,
      young=TRUE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    9
[10] {job=services,
      marital=single,
      housing_loan=yes,
      young=TRUE,
      contacted_more_than_once=FALSE,
      contacted_before_this_campaign=FALSE,
      success=no}                                    10
```

Explain the difference between **bank_new** and **bankX** in a block comment:

[20]:
```
#bank_new is just a normal data frame, so each row is a single observation
#bankX is a transaction matrix, which has a data frame to store each row of␣
↪values as a single transaction, as well as another data frame to store the␣
↪labels
```

1.3 Part 3: Use arules to discover patterns

Support is the proportion of times that a particular set of items occurs relative to the whole dataset. **Confidence** is proportion of times that the consequent occurs when the antecedent is present.

J. Use **apriori** to generate a set of rules with support over 0.005 and confidence over 0.3, and trying to predict who successfully signed up for a term deposit. **Hint:** You need to define the **right-hand side rule (rhs)**.

[23]:
```
rules <- apriori(bankX,
                 parameter=list(supp=0.005,conf=0.3),
                 control=list(verbose=F),
                 appearance=list(default='lhs',rhs=("success=yes")))
```

K. Use inspect() to review of the **ruleset**.

[24]:
```
inspect(rules)
```

```
    lhs                                    rhs                support
```

```
     confidence   coverage     lift count
[1] {job=student}                           => {success=yes} 0.006676702
0.3142857 0.02124405 2.789828   275
[2] {job=student,
     marital=single}                        => {success=yes} 0.006409634
0.3203883 0.02000583 2.843999   264
[3] {job=student,
     young=TRUE}                            => {success=yes} 0.006579586
0.3180751 0.02068564 2.823465   271
[4] {job=student,
     contacted_before_this_campaign=FALSE} => {success=yes} 0.006676702
0.3142857 0.02124405 2.789828   275
[5] {job=student,
     marital=single,
     young=TRUE}                            => {success=yes} 0.006312518
0.3233831 0.01952025 2.870582   260
[6] {job=student,
     marital=single,
     contacted_before_this_campaign=FALSE} => {success=yes} 0.006409634
0.3203883 0.02000583 2.843999   264
[7] {job=student,
     young=TRUE,
     contacted_before_this_campaign=FALSE} => {success=yes} 0.006579586
0.3180751 0.02068564 2.823465   271
[8] {job=student,
     marital=single,
     young=TRUE,
     contacted_before_this_campaign=FALSE} => {success=yes} 0.006312518
0.3233831 0.01952025 2.870582   260
```

L. Use the output of inspect( ) or inspectDT( ) and describe **any 2 rules** the algorithm found.

[25]:
```
#[1]: when the job is student, there is a 31% chance that there is a success;␣
↪however, the job is student only about 0.7% of the time

#[2]: when the job is student and the marital status is single, there is a 32%␣
↪chance that there is a success; however, this case only happens about 0.6%␣
↪of the time
```

[ ]:
```

```