Intro to Data Science - HW 10

```
# Enter your name here: Hrishikesh Telang
```

Copyright 2021, Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

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

```
# 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

Part 1: Explore Data Set

A. Read the contents of the following URL to a dataframe called **bank** https://intro-datascience.s3.us-east-2.amazonaws.com/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 either read.csv or read.table, with two additional parameters. *sep=";"* defines how the data is seperated (the default is a comma), and *header=TRUE* defines that there is a header line in the dataset.

```r
library(readr) #Initiate readr library
bank <- read.csv('https://intro-datascience.s3.us-east-2.amazonaws.com/bank-full.csv',
                 sep=";", header=TRUE)
#read csv file from the internet
head(bank) #Show the head of the bank
```

```
##   age       job marital   education default housing loan   contact month
## 1  56 housemaid married    basic.4y      no      no   no telephone   may
## 2  57  services married high.school unknown      no   no telephone   may
## 3  37  services married high.school      no     yes   no telephone   may
## 4  40    admin. married    basic.6y      no      no   no telephone   may
## 5  56  services married high.school      no      no  yes telephone   may
## 6  45  services married    basic.9y unknown      no   no telephone   may
##   day_of_week duration campaign pdays previous    poutcome emp.var.rate
## 1         mon      261        1   999        0 nonexistent          1.1
## 2         mon      149        1   999        0 nonexistent          1.1
## 3         mon      226        1   999        0 nonexistent          1.1
## 4         mon      151        1   999        0 nonexistent          1.1
```

```
## 5         mon      307        1   999        0 nonexistent          1.1
## 6         mon      198        1   999        0 nonexistent          1.1
##   cons.price.idx cons.conf.idx euribor3m nr.employed  y
## 1         93.994         -36.4     4.857        5191 no
## 2         93.994         -36.4     4.857        5191 no
## 3         93.994         -36.4     4.857        5191 no
## 4         93.994         -36.4     4.857        5191 no
## 5         93.994         -36.4     4.857        5191 no
## 6         93.994         -36.4     4.857        5191 no
```

```
tail(bank) #Show the tail of the bank
```

```
##       age         job marital           education default housing loan  contact
## 41183  29  unemployed  single            basic.4y      no     yes   no cellular
## 41184  73     retired married professional.course      no     yes   no cellular
## 41185  46 blue-collar married professional.course      no      no   no cellular
## 41186  56     retired married   university.degree      no     yes   no cellular
## 41187  44  technician married professional.course      no      no   no cellular
## 41188  74     retired married professional.course      no     yes   no cellular
##       month day_of_week duration campaign pdays previous    poutcome
## 41183   nov         fri      112        1     9        1     success
## 41184   nov         fri      334        1   999        0 nonexistent
## 41185   nov         fri      383        1   999        0 nonexistent
## 41186   nov         fri      189        2   999        0 nonexistent
## 41187   nov         fri      442        1   999        0 nonexistent
## 41188   nov         fri      239        3   999        1     failure
##       emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed   y
## 41183         -1.1         94.767         -50.8     1.028      4963.6  no
## 41184         -1.1         94.767         -50.8     1.028      4963.6 yes
## 41185         -1.1         94.767         -50.8     1.028      4963.6  no
## 41186         -1.1         94.767         -50.8     1.028      4963.6  no
## 41187         -1.1         94.767         -50.8     1.028      4963.6 yes
## 41188         -1.1         94.767         -50.8     1.028      4963.6  no
```

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

```
#To cross check that, we have used str(), nrow() and ncol()
str(bank) #returns the structure (datatypes) of the bank dataset
```

```
## '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 1 ...
```

```
##  $ pdays         : int  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : int  0 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" ...
```

```
nrow(bank) #returns number of rows
```

```
## [1] 41188
```

```
ncol(bank) #returns number of columns
```

```
## [1] 21
```

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 command. Write a comment describing how the conversion for each numeric variable works and what are the variables in the resulting dataframe.

```
bank_new <- data.frame(job=as.factor(bank$job),
                      marital=as.factor(bank$marital),
                      housing_loan=as.factor(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$previous<0),
                      success=as.factor(bank$y))
```

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

```
table(bank_new$success) #returns the number of rows that have "no" and "yes" labels
```

```
##
##    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.

```
prop.table(table(bank_new$success)) #returns the percentage of the number of rows
```

```
##
##        no       yes
## 0.8873458 0.1126542
```

```r
#with "no" and "yes" labels
```

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

```r
print("%%%%marital%%%%")
```

```
## [1] "%%%%marital%%%%"
```

```r
table(bank_new$marital) #returns the number of rows that have "no" and "yes" labels
```

```
##
## divorced  married   single  unknown
##     4612    24928    11568       80
```

```r
prop.table(table(bank_new$marital))
```

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

```r
#returns the percentage of the number of rows with "no" and "yes" labels

print("%%%%housing_loan%%%%")
```

```
## [1] "%%%%housing_loan%%%%"
```

```r
table(bank_new$housing_loan) #returns the number of rows that have "no" and "yes" labels
```

```
##
##      no unknown     yes
##   18622     990   21576
```

```r
prop.table(table(bank_new$housing_loan))
```

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

```r
#returns the percentage of the number of rows with "no" and "yes" labels
```

Part 2: Coerce the data frame into transactions

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

```r
#install.packages('arules')
#install.packages('arulesViz')
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

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

```r
library(arulesViz)
```

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

```r
bankX <- as(bank_new, "transactions")
bankX
```

```
## transactions in sparse format with
##  41188 transactions (rows) and
##  26 items (columns)
```

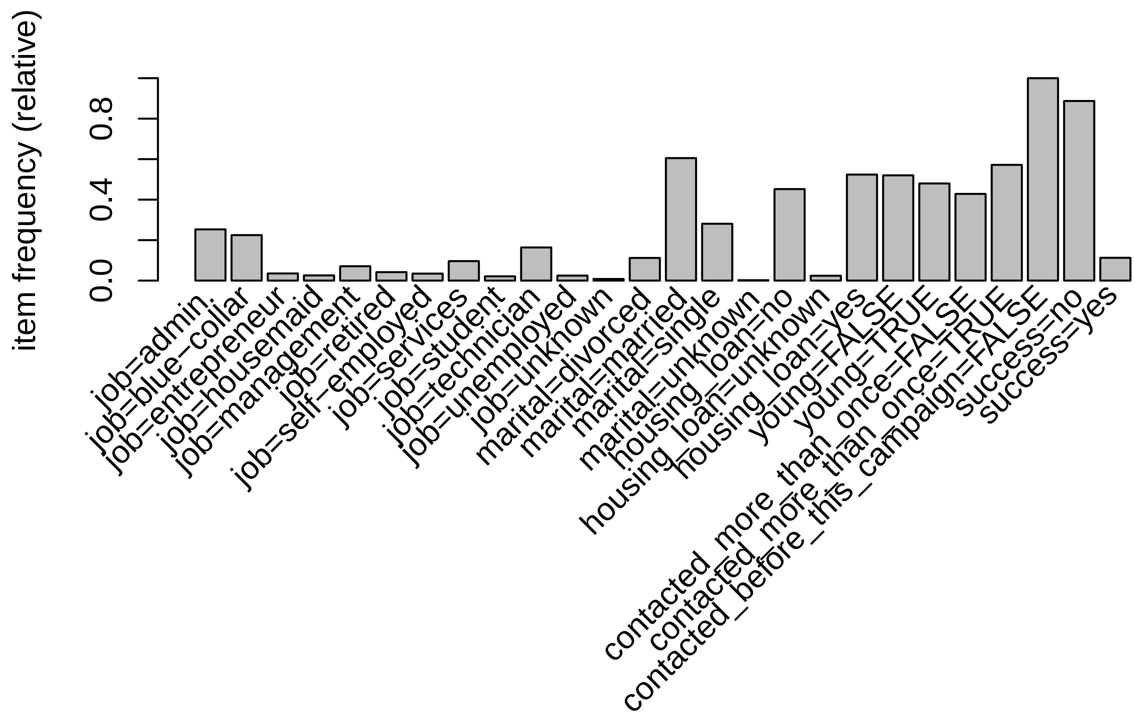
H. Use the itemFrequency( ) and itemFrequencyPlot( ) commands to explore the contents of **bankX**. What do you see?

```r
itemFrequency(bankX) #Shows item frequency for each categorical value
```

```
##                        job=admin.                   job=blue-collar
##                       0.253034865                       0.224677090
##                  job=entrepreneur                     job=housemaid
##                       0.035350102                       0.025735651
##                    job=management                       job=retired
##                       0.070991551                       0.041759736
##                 job=self-employed                      job=services
##                       0.034500340                       0.096363018
##                       job=student                    job=technician
##                       0.021244052                       0.163712732
##                    job=unemployed                       job=unknown
##                       0.024618821                       0.008012042
##                  marital=divorced                   marital=married
##                       0.111974361                       0.605224823
##                    marital=single                   marital=unknown
##                       0.280858502                       0.001942313
##                   housing_loan=no              housing_loan=unknown
##                       0.452121977                       0.024036127
##                  housing_loan=yes                       young=FALSE
##                       0.523841896                       0.520054385
##                        young=TRUE       contacted_more_than_once=FALSE
```

```
##                         0.479945615                           0.428328639
##       contacted_more_than_once=TRUE contacted_before_this_campaign=FALSE
##                         0.571671361                           1.000000000
##                          success=no                           success=yes
##                         0.887345829                           0.112654171
```

```
itemFrequencyPlot(bankX) #Plots the frequency distribution
```




```
# I see a frequency graph of all the occurences of attributes from the sparse
#transactions matrix
#The Y axis signifies the frequency of the number of times the item has been selected.
#(In other words, Y axis indicates the support.)
#Based on this observation, attributes such as contacted_before_this_campaign has
#had the highest occurence, followed by success, then marital and so forth.
```

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

```
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
```

```
# With this code, we are exploring only the first 10 observations in the 'bankX' transaction matrix
```

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

```
#bank_new is the new dataset that is returned after all the variables which looked
#categorical in nature are converted into factor variables, whereas, bankX is the
#a sparse transaction matrix that results from the categorical valuables of bank_new.
#It contains the frequency values as a probability distribution of all the
#different columns along with their categorical values.
```

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)**.

```
#I have listed the supp=0.005 and conf= 0.3 to satisfy the question of the prompt.
#I also used success=yes to predict who successfully signed up for a term deposit.
rules3 <- apriori(bank_new, parameter = list(supp=0.005, conf= 0.3),
                  appearance=list(default="lhs", rhs="success=yes"),
                  control=list(verbose=F))
```

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

```
inspect(rules3)
```

```
##     lhs                                     rhs             support confidence   coverage     li
## [1] {job=student}                        => {success=yes} 0.006676702  0.3142857 0.02124405 2.7898
## [2] {job=student,
##      marital=single}                     => {success=yes} 0.006409634  0.3203883 0.02000583 2.8439
## [3] {job=student,
##      young=TRUE}                         => {success=yes} 0.006579586  0.3180751 0.02068564 2.8234
## [4] {job=student,
##      contacted_before_this_campaign=FALSE} => {success=yes} 0.006676702  0.3142857 0.02124405 2.7898
```

```
## [5] {job=student,
##      marital=single,
##      young=TRUE}                           => {success=yes} 0.006312518  0.3233831 0.01952025 2.87058
## [6] {job=student,
##      marital=single,
##      contacted_before_this_campaign=FALSE} => {success=yes} 0.006409634  0.3203883 0.02000583 2.84399
## [7] {job=student,
##      young=TRUE,
##      contacted_before_this_campaign=FALSE} => {success=yes} 0.006579586  0.3180751 0.02068564 2.82340
## [8] {job=student,
##      marital=single,
##      young=TRUE,
##      contacted_before_this_campaign=FALSE} => {success=yes} 0.006312518  0.3233831 0.01952025 2.87058
```

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

```
#The rule {job=student, marital=single, young=TRUE, contacted_before_this_campaign=FALSE}
#=> {success=yes} shows a lift of 2.870582. This is the highest value computed for the given
#rule in the ruleset. This is followed by two second highest rules {job=student, marital=single}
#=> {success=yes} and {job=student, marital=single, contacted_before_this_campaign=FALSE}
#=> {success=yes} with equal lifts, confidences and supports and will be treated
#equally based on the conditions set at training the apriori model.
```