STAT 4410/8416 Homework 1

Danuwar Ramesh

Due on Sep 18, 2020

1. Based on your reading assignments answer the following questions:

   a) What is data science?

   Data science is the art and science of transforming data into data products. Basically, it is used in the analysis of data.

   b) Explain with an example what you mean by data product.

   Simply, data product is the output of any data science activities.It could be a report based on data.Some of the examples of data products are: credit score results, value of your used car, auto correct on your phone and spell-check on your computer.

   c) Carefully read Cleveland's paper shown in lecture 2 and discuss what he suggested about the field of statistics and data science.

   Cleveland's paper shown in the lecture 2 suggested that data science is not statistics. They are different with each other. It says statistics is a mathematically-based where data is collected based on planned analysis method. In contrast, data science is a multidisciplinary field which uses scientific methods, processes and systems to extract knowledge from data into useful data products.

   d) Explain in a short paragraph how data science is different from computer science.

   Data science is mainly focused on data managment , data technology, data access, data security and data reports where computer science is solely focused on creating , repairing and maintaining of computers.

   e) What is data literacy? Is it important to be data literate in this modern world? Explain why or why not.

   Data Literacy is the ability to read, understand , create and commuincate data as information. It is important to be data literate in this modern world beacuse all most every companies in the world, they use internal and external data literacy. Also, as a business point of view having data literacy will allow us to design better products, understnad customers, and improve efficiency.

   f) In his article, Donoho talked about the Common Task Framework. Explain what it is and why he mentioned it.

   In his article, he defined common task framework in three different ways. First:"A publicly available training dataset involving, for each observation, a list of(possibly many) feature measurements, and a class label for that observation." Second: " A set of enrolled competitors who common task is to infer a class prediction rule from the training data". Lastly:" A scoring referee to which competitors can submit their prediction rule. The referee runs the prediction rule against a testing dataset which is sequestered behind a Chinese wall. The referee objectively and automatically reports the score (prediction accuracy) achieved by the submitted rule." I think the main reason why he mentioned it was to get the users attention and make users aware about what exactly is common task framewrok is. The idea about common task framework is an important idea from machine learning and data science which is lacking in the today's world.

g) According to Donoho, what are the activities of greater data science?

According to Donoho, the activities of greater data science are:

1. Data exploration and preparation
2. Data representation and transforming
3. Computing with Data
4. Data modeling
5. Data visualization and presentation
6. Science about Data science

2. What are the very first few steps one should take once data is loaded onto **R**? Demonstrate them by loading tips data from http://www.ggobi.org/book/data/tips.csv.

```r
url <- "http://www.ggobi.org/book/data/tips.csv "
dat <- read.table(url, header= T, sep=",")
head(dat)
```

```
##   obs totbill  tip sex smoker day  time size
## 1   1   16.99 1.01   F     No Sun Night    2
## 2   2   10.34 1.66   M     No Sun Night    3
## 3   3   21.01 3.50   M     No Sun Night    3
## 4   4   23.68 3.31   M     No Sun Night    2
## 5   5   24.59 3.61   F     No Sun Night    4
## 6   6   25.29 4.71   M     No Sun Night    4
```

3. In our **R** class, we learned about recursive functions that produce a sequence of numbers up to a given number, say n, as demonstrated with the following code:

```r
foo <- function(x) {
  print(x)
  if(x > 1) {
    foo(x - 1)
  }
}

moo <- function(x) {
  if(x > 1) {
    moo(x - 1)
  }
  print(x)
}

foo(3)
```

```
## [1] 3
## [1] 2
## [1] 1
```

```r
moo(3)
```

```
## [1] 1
## [1] 2
## [1] 3
```

Explain why `moo()` prints 1 through 3 while `foo()` prints from 3 to 1.

foo(x): It is printing the value of the argument x first and then it is doing recursive call to foo(x-1). So, for the call foo(3), it prints the argument 3 and then calls foo(2) foo(2) prints the argument 2 and then calls foo(1) foo(2) prints the argument 1.

moo(x): It is doing recursive call to foo(x-1) first and then it is printing the value of the argument x first and so for the call moo(3). It calls moo(2) and after all the recursive calls it prints value 3

4. The function `sqrt()` provides the square root of a non-negative number. Note what happens when you try `sqrt(-1)`. We want to create our own function that either finds the square root of a non-negative number or provides a custom message if we pass it a negative number.

   a) Create a new `R` function `getRootNotVectorized()` that will return the square root of any non-negative number and 'not possible' for a negative number. Further, `getRootNotVectorized()` should **only** successfully return 'not possible' if the negative value is the first element that you pass to the function. Otherwise, your function should return `NaN` for negative values. Demonstrate that your function produces the following outputs:

      ```
      getRootNotVectorized(4) = 2
      getRootNotVectorized(-4) = "not possible"
      getRootNotVectorized(c(-1, -4)) = "not possible"
      getRootNotVectorized(c(0, 1, -1, 4, -4)) = 0 1 NaN 2 NaN.
      ```

   Don't worry about the warning messages that accompany vector inputs with more than one element for now.

```r
getRootNotVectorized=function(x) {
  if (x >=0) {
    return(sqrt(x))
  }else return("not possible")
}
getRootNotVectorized(4)
```

```
## [1] 2
```

```r
getRootNotVectorized(-4)
```

```
## [1] "not possible"
```

```r
getRootNotVectorized(c(-1,-4))
```

```
## Warning in if (x >= 0) {: the condition has length > 1 and only the first
## element will be used
```

```
## [1] "not possible"
```

```r
getRootNotVectorized(c(0,1,-1,4,-4))
```

```
## Warning in if (x >= 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in sqrt(x): NaNs produced

## [1]   0   1 NaN   2 NaN
```

```
b) Now create a second function `getRootVectorized()` that will return the square root of any non-negat
\
`getRootVectorized(4) = 2`
`getRootVectorized(-4) = "not possible"`
`getRootVectorized(c(-1, -4)) = "not possible" "not possible"`
`getRootVectorized(c(0, 1, -1, 4, -4)) = "0" "1" "not possible" "2" "not possible"`.
\
```

```r
getRootVectorized= function(x) {
  returnValue =ifelse(x>=0, sqrt(x), "not possible")
  return(returnValue)
}
getRootVectorized(4)
```

```
## [1] 2
```

```r
getRootVectorized(-4)
```

```
## [1] "not possible"
```

```r
getRootVectorized(c(-1,-4))
```

```
## [1] "not possible" "not possible"
```

```r
getRootVectorized(c(0,1,-1,4,-4))
```

```
## Warning in sqrt(x): NaNs produced

## [1] "0"            "1"            "not possible" "2"            "not possible"
```

```
c) Describe the differences in your code between `getRootNotVectorized()` and `getRootVectorized()` tha
```

In first function, the conditioning citeria is if(x>=0) which takes only fist elememt of vector.

In the second function,ifelse takes the vector and run criteria for each index of vector.Therefore, we run the same operation of each element of matrix.

```
d) Why do you see a difference between the output of the two following lines of code?
```

```
is.numeric(getRootVectorized(c(0, 1, 4)))

is.numeric(getRootVectorized(c(0, 1, -4)))
```

In first cases, all numbers are positive so vector is numeric. In 2nd case, it was numeric until -4 then since at that operation, we are returning the string, the resultant vector converts to character and then the type of vector will be character. So first will be TRUE and 2nd will be FALSE

5. This problem will give you some practice with creating and manipulating vectors.
   a) Using `seq()`, create a vector consisting of an arithmetic sequence of integers from 5 to 50 with a common difference of 5 stored in a variable called `mySeq`. **Report** `mySeq`.

```
mySeq= seq(from=5 ,to= 50, by=5)
    mySeq
```

```
##  [1]  5 10 15 20 25 30 35 40 45 50
```

```
b) Describe how the different arguments in each of the three following commands changes the output of ‘
```

```
#The mySeq replicates 5 times
```

```
rep(mySeq, each=5)
```

```
##  [1]  5  5  5  5  5 10 10 10 10 10 15 15 15 15 15 20 20 20 20 20 25 25 25 25 25
## [26] 30 30 30 30 30 35 35 35 35 35 40 40 40 40 40 45 45 45 45 45 50 50 50 50 50
```

#repeat counting of mySeq variable 5 time

```
rep(mySeq, 5)
```

```
##  [1]  5 10 15 20 25 30 35 40 45 50  5 10 15 20 25 30 35 40 45 50  5 10 15 20 25
## [26] 30 35 40 45 50  5 10 15 20 25 30 35 40 45 50  5 10 15 20 25 30 35 40 45 50
```

#automated length repetition

```
rep(mySeq, mySeq)
```

```
##   [1]  5  5  5  5  5 10 10 10 10 10 10 10 10 10 10 15 15 15 15 15 15 15 15 15 15
##  [26] 15 15 15 15 15 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
##  [51] 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
##  [76] 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
## [101] 30 30 30 30 30 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35
## [126] 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 40 40 40 40 40 40 40 40 40 40
## [151] 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
## [176] 40 40 40 40 40 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45
## [201] 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45
## [226] 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
## [251] 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
c) Concatenate the sequence ‘1:14‘ to the end of the vector described by ‘rep(mySeq,mySeq)‘ and store
```

the resulting vector in the same `mySeq` variable. **Report** the length of `mySeq`.

```
mySeq=c(rep(mySeq,mySeq),1:14)
length(mySeq)
```

```
## [1] 289
```

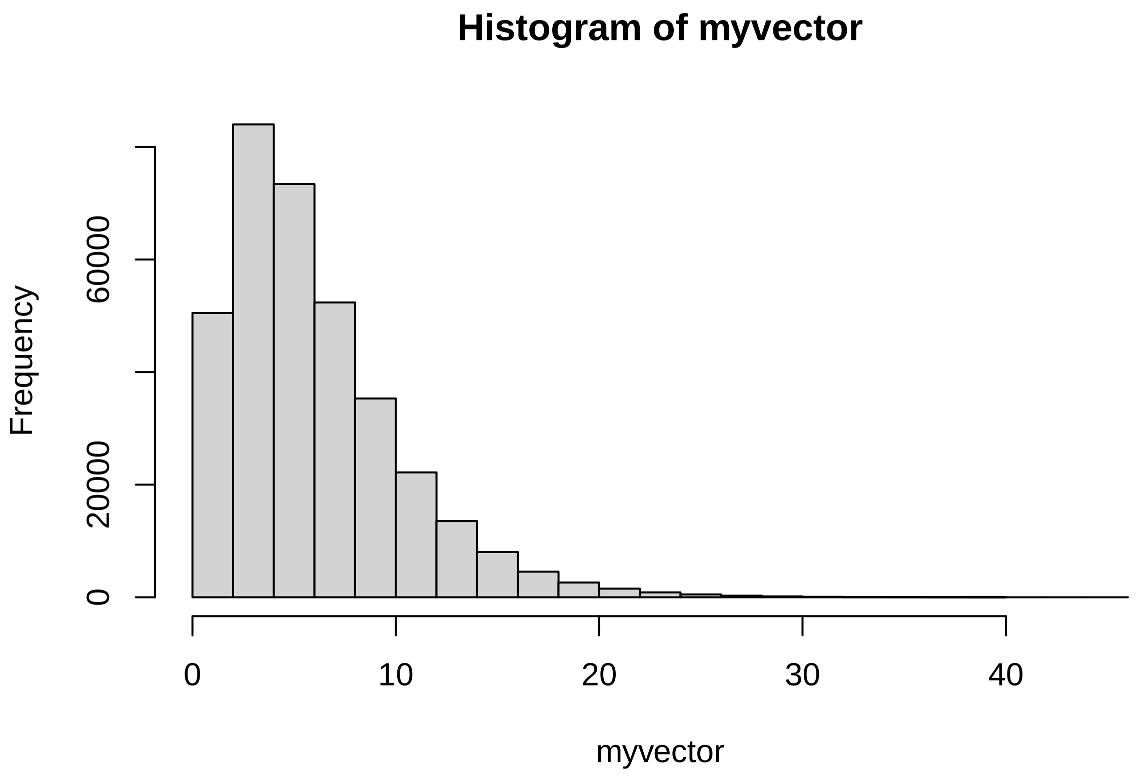
```
d) Create a square matrix populated row-wise from your ‘mySeq‘ vector and store it in a variable called
```

```
sqMtrx=matrix(mySeq,sqrt(length(mySeq)), byrow=TRUE)
margin.table(sqMtrx,2)
```

```
##  [1] 585 585 585 541 552 558 559 565 566 567 573 574 575 581 582 588 594
```

6. Write a program that will do the following. Include your codes and necessary outputs to demonstrate your work.
    a) Generate 350,000 random numbers from a gamma distribution with `shape = 2` and `scale = 3` and store these numbers in a vector called `myVector`. **Report** a histogram of the numbers you just generated.

```
myvector=rgamma(350000, shape = 2, scale=3)
hist(myvector)
```



```
b) Convert ‘myVector‘ into a matrix with 5,000 rows and assign it to an object called ‘myMatrix‘. **Repo
```

```
myMatrix=matrix(myvector,ncol = 1, nrow = 5000)
dim(myMatrix)
```

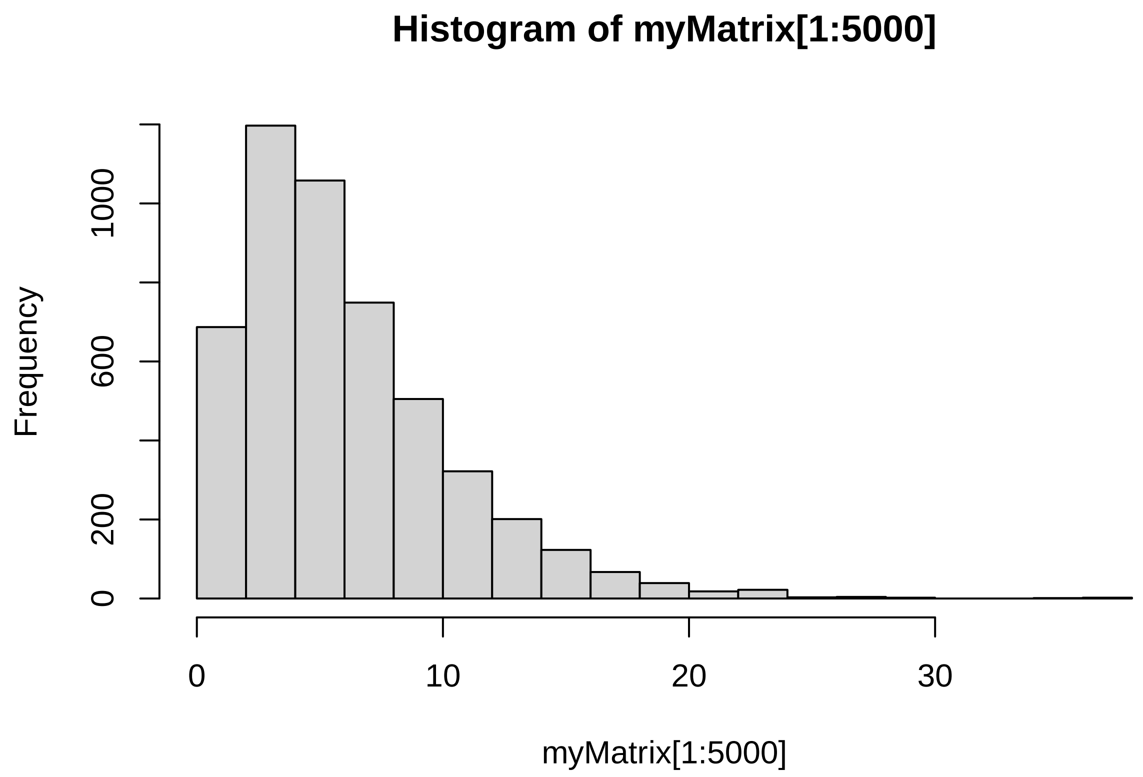
```
## [1] 5000    1
```

```
c) Compute the row means of ‘myMatrix‘ and **report** a histogram of those row means.
```

```
mean(myMatrix[1:5000],)
```

```
## [1] 6.092197
```

```
hist(myMatrix[1:5000],)
```



```
d) Explain why the two histograms you created in (6a) and (6c) have different shapes.
```

The histograms that i created in (6a) and (6c) have different shapes becasue they have different frequency whcih also made their axis scale different but both have positively skewness shape.

7. Perform the following reproducible procedure:
    a) Set a seed for the `R` random number generator using `set.seed()` and seed value 2019.

```
set.seed(2019)
    rnorm(4)
```

```
## [1]  0.7385227 -0.5147605 -1.6401813  0.9160368
```

b) Create a vector called ‘x‘ of 1,000 values from a normal distribution with mean 100 and standard devi

```
number1<-seq(1,1000)
    x <-rnorm(number1, mean=100, sd=20)
    summary (x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   35.28   85.45   97.39   98.47  111.95  170.83
```

c) Create a second vector called ‘y‘ of 1,000 values from a normal distribution with mean 0 and standar

```
number2<-seq(1,1000)
    y <-rnorm(number2, mean=0, sd=4)
    summary (y)
```

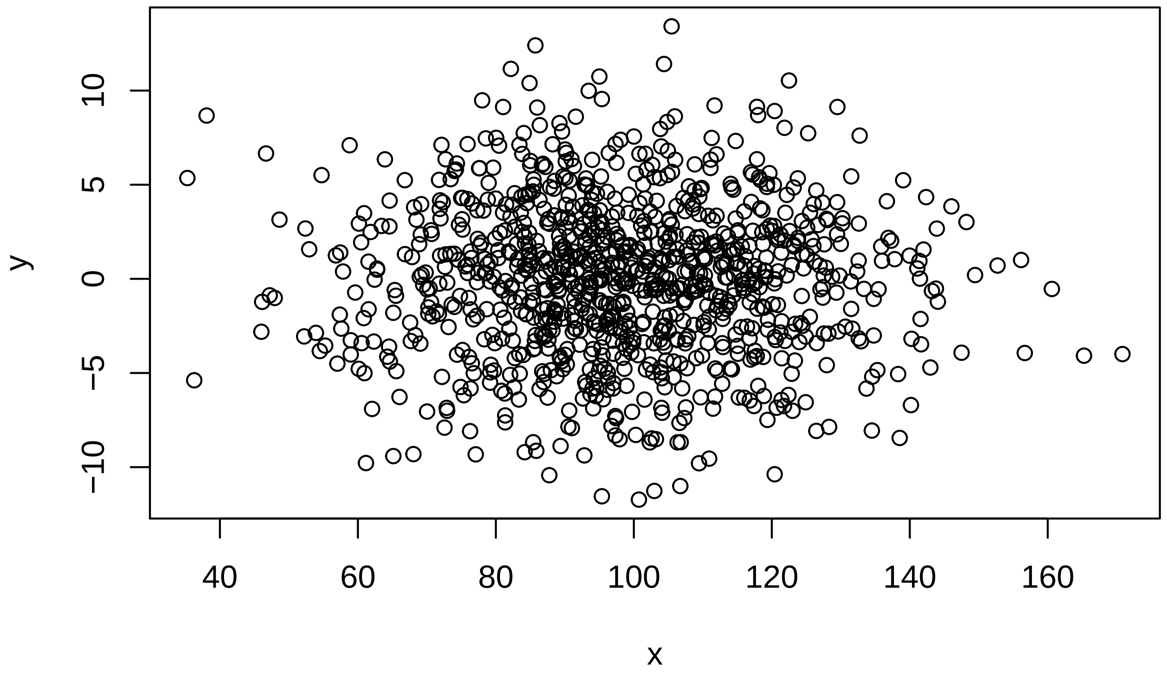
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -11.72496  -2.77565   0.07867  -0.02886   2.56996  13.40945
```

g) Create a data frame called ‘df‘ from your ‘x‘ and ‘y‘ vectors.

```
df=data.frame(x,y)
```

h) Generate a scatterplot of ‘df‘.

```
plot(df)
```



```
i) **Report** the ‘tail()‘ of ‘df‘ as a nice table using ‘kable()‘.
```

```r
library(knitr)
kable(tail(df))
```

	x	y
995	84.73008	1.825883
996	119.20879	1.137595
997	132.60611	-3.164516
998	109.42086	3.444231
999	85.38751	4.641964
1000	98.87863	-3.716250

8. Based on our lecture notes, answer the following questions. Show your answer presenting the relavent `R` code.

   a) We have a vector of values `x = c(2,4,5, "3.5")`. What would be the mode of the vector `x`?

```r
x = c(2,4,5, "3.5")
mode(x)
```

```
## [1] "character"
```

b) How do you load a package into ‘R‘? Show that loading ‘ggplot2‘ package.

install.packages(“ggplot2”)

library(ggplot2)

c) Missing values are shown as ‘NA‘ in ‘R‘. How can you check if there is any missing values in a vecto

```
y = c(3,5,8, NA, 6)
is.na(y)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE
```