CIND 123 - Data Analytics: Basic Methods

Assignment 1 (10%)

[Alyzeh Jiwani]

[D20 501106857]

Instructions

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. Review this website for more details on using R Markdown http://rmarkdown.rstudio.com.

Use RStudio for this assignment. Complete the assignment by inserting your R code wherever you see the string "#INSERT YOUR ANSWER HERE".

When you click the **Knit** button, a document (PDF, Word, or HTML format) will be generated that includes both the assignment content as well as the output of any embedded R code chunks.

Submit **both** the rmd and generated output files. Failing to submit both files will be subject to mark deduction.

Sample Question and Solution

Use `seq()` to create the vector $(1, 2, 3, \ldots, 10)$.

```
seq(1,10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Question 1

a) Create and print a vector `x` with all integers 1-100, and a vector `y` with every fifth integer in the same range. What is the difference in lengths of the vectors `x` and `y`?. (8 points) Hint: use `seq()`function, every fifth element of "y" will be like [1,6,11,. . . ..].

```
x <- seq(1,100)
y <- seq(1,100,5)
print("The vector x is:")
```

```
## [1] "The vector x is:"
```

```
(x)
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

```
print("The vector y is:")
```

```
## [1] "The vector y is:"
```

```
(y)
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
```

```
print("The difference between the legths of x and y is")
```

```
## [1] "The difference between the legths of x and y is"
```

```
(length(x)-length(y))
```

```
## [1] 80
```

b) Create a new vector, "x_square", with the square of elements at indices 3, 6, 7, 10, 15, 22, 23, 24, and 30 from the variable "x". Hint: Use indexing rather than a for loop. Calculate the mean and median of the last five values from x_square.

```
(x_square <- c(x[3]^2, x[6]^2, x[7]^2, x[10]^2, x[15]^2, x[22]^2, x[23]^2, x[24]^2, x[30]^2))
```

```
## [1]   9  36  49 100 225 484 529 576 900
```

```
(mean(x_square[-4:-1]))
```

```
## [1] 542.8
```

```
(median(x_square[5:9]))
```

```
## [1] 529
```

c) Would it be correct to use the following commands to convert a factor variable to a numeric variable? Explain your answer.

```
factorVar <- factor(c(1, 6, 5.4, 3.2));as.numeric(factorVar)
```

```
#It would not be correct as as.numeric()
#will convert the individual factor levels of
#the factor variable to numeric values if they werent
#already numeric, hoever they will still
#remain as factor levels.
```

d) Assume that you would read a comma-separated file `dataset.csv` consists of missing values represented by question marks (`?`) and exclamation mark (`!`). How can you read this type of files in R (please include your code in the answer section)?

```
#When importing the csv i would use the read.csv()
#function and set na.strings option to switch any strings
#that match ? or ! to the logical value NA
#(i.e. c("?", "!"))
```

Question 2

a) Compute:

$$\sum_{n=1}^{100} \frac{2^n}{(n-1)!}$$

```
dummy_vec <- seq(1:100)
for(i in 1:100) {
  num <- 2^i
  den <- factorial(i-1)
  dummy_vec[i] <- num/den
}
(sum(dummy_vec))
```

```
## [1] 14.77811
```

b) Compute:

$$\sum_{n=1}^{10} \left(\frac{2^n}{n^2} + \frac{n^4}{4^n} \right)$$

```
dv1 <- seq(1,10)
dv2 <- seq(1,10)
for (i in 1:10){
  n1 <- 2^i
  d1 <- i^2
  n2 <- i^4
  d2 <- 4^i
  dv1[i] <- n1/d1
  dv2[i] <- n2/d2

}
(sum(dv1+dv2))
```

```
## [1] 35.80589
```

c) Compute:

$$\sum_{n=0}^{10} \frac{1}{(n+1)!}$$

(Hint: Use `factorial(n)` to compute $n!$)

```
d_vec <- seq(0:10)
for(i in 0:10){
  d_vec[i] <- 1/factorial(i+1)
}
(sum(d_vec))
```

```
## [1] 11.71828
```

d) Compute:

$$\prod_{n=3}^{33}\left(3n+\frac{3}{\sqrt[3]{n}}\right)$$

```
dummy_v <- seq(3,33)
for(i in 3:33){
  a <- 3*i
  b <- 3/(i^(1/3))
  dummy_v[i-2] <- a+b
}
(dummy_v)
```

```
##  [1] 11.08008 13.88988 16.75441 19.65096 22.56827 25.50000 28.44225 31.39248
##  [9] 34.34893 37.31037 40.27587 43.24474 46.21644 49.19055 52.16673 55.14471
## [17] 58.12427 61.10521 64.08738 67.07065 70.05490 73.04004 76.02599 79.01266
## [25] 82.00000 84.98795 87.97646 90.96549 93.95499 96.94494 99.93530
```

```
for(i in 4:33){
  dummy_v[i-2] <- dummy_v[i-3]*dummy_v[i-2]
}
(dummy_v)
```

```
##  [1] 1.108008e+01 1.539011e+02 2.578521e+03 5.067043e+04 1.143544e+06
##  [6] 2.916038e+07 8.293867e+08 2.603650e+10 8.943261e+11 3.336764e+13
## [11] 1.343911e+15 5.811707e+16 2.685964e+18 1.321240e+20 6.892480e+21
## [16] 3.800838e+23 2.209209e+25 1.349942e+27 8.651425e+28 5.802567e+30
## [21] 4.064983e+32 2.969065e+34 2.257261e+36 1.783522e+38 1.462488e+40
## [26] 1.242939e+42 1.093493e+44 9.947016e+45 9.345718e+47 9.060201e+49
## [31] 9.054339e+51
```

```
(dummy_v[31])
```

```
## [1] 9.054339e+51
```

e) Explain the output of this R-command: `c(0:5)[NA]`

```
#the output of this r command is NA NA NA NA NA NA
#this is because the square brackets are used to find
#subsets corresponding to thindex inside the brackets
#if the index of the subset we want to select is NA
#(not available) then for each element in the vector
#c(0:5) its representative in the subset [NA]
#will also be unavailable.
```

f) What is the difference between is.vector() and is.numeric() functions?

```
#is.vector() checkes to see whether an object in
#r is a vector. is.numeric() takes in a vector as input
#and checks to see if that vector is of numeric class.
```

g) There are lost of packages in R. `RShiny` is one of it (https://shiny.rstudio.com/). Please investigate this package and list at least three advantages and three disadvantages of using `RShiny` package?

```
#helps us build web applications without having to know
#any html, css or javascript knowledge
#its free
#its compatible with R, and highly cusomisable/extensible
```

Question 3

`iris` dataset gives the measurements in centimeters of the variables sepal length, sepal width, petal length and petal width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

Install the `iris` dataset on your computer using the command `install.packages("datasets")`. Then, load the `datasets` package into your session using the following command.

```
library(datasets)
```

a) Display the first ten rows of the `iris` data set.

```
head(iris, 10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

b) Compute the average of the first four variables (Sepal.Length, Sepal.Width, Petal.Length and Petal.Width) using `sapply()` function.

Hint: You might need to consider removing the `NA` values, otherwise the average will not be computed.

```
sapply(iris[c('Sepal.Length', 'Sepal.Width', 'Petal.Length', 'Petal.Width')], mean)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```
m_count <- sum(is.na(iris))
if(("NA"%in% iris)|
   ("na"%in% iris) | (""%in% iris)|(NA %in% iris) ){
  print("True")
  m_count<- m_count +1
}
str_c("There are ", m_count, " missing values")
```

```
## [1] "There are 0 missing values"
```

c) Show how to use R to replace the missing values in this dataset with plausible ones.

```
#first when we read in the csv file we replace any
#variants of NA with the logical value NA
#(As shown above).
#when using variables or instances in the data set in
#functions or as part of calculations we can use
#na.omit() to omit the missing values from our calculations.
#depending on how frequently NA values appear in certain
#variables, and how useful those variables are to our
#analysis we can remove those variables entirely
# likewise, if certain variables our vital to our
#analysis, we can even consider removing entires
#instances where these variables have a NA value, or if
#these instances have a high occurence of NA values.
```

d) Compute the standard deviation for only the first and the third variables (Sepal.Length and Petal.Length)

```
sapply(iris[c('Sepal.Length', 'Petal.Length')], sd)
```

```
## Sepal.Length Petal.Length
##    0.8280661    1.7652982
```

e) Construct a boxplot for `Sepal.Width` variable, then display the values of all the outliers. Explain how these outliers have been calculated.

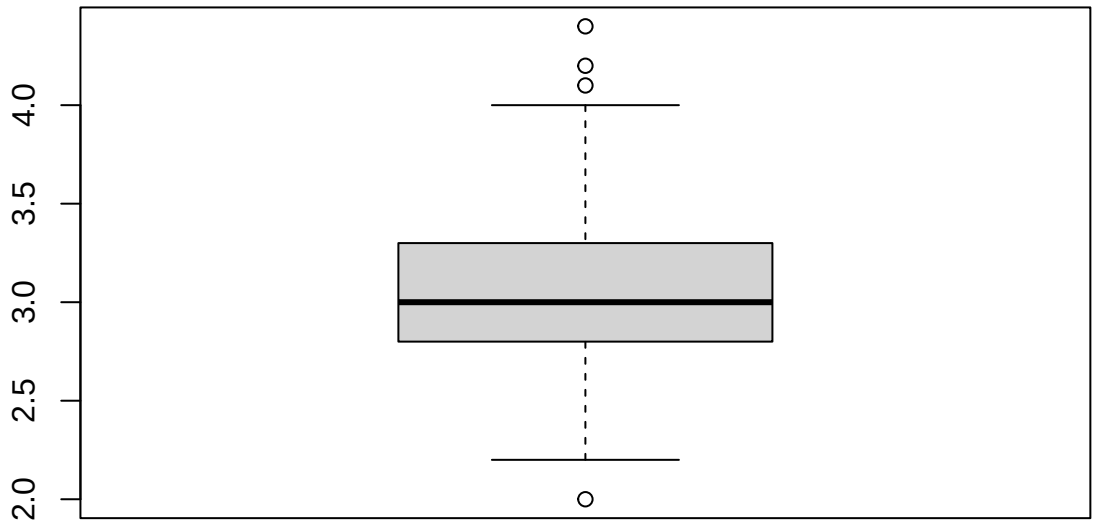
```
summary(iris["Sepal.Width"])
```

```
##   Sepal.Width
##  Min.   :2.000
##  1st Qu.:2.800
##  Median :3.000
##  Mean   :3.057
##  3rd Qu.:3.300
##  Max.   :4.400
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
sw_box <- boxplot(iris$Sepal.Width)
```




```
sw_box$out # this gives us the outliers of sepal.width; 4.4 4.1 4.2 2.0
```

```
## [1] 4.4 4.1 4.2 2.0
```

```
#how have these been calculated? explain
```

f) Compute the upper quartile of the `Sepal.Width` variable with two different methods.

```
sw_box$stats[4,] #give us the upper quartile
```

```
## [1] 3.3
```

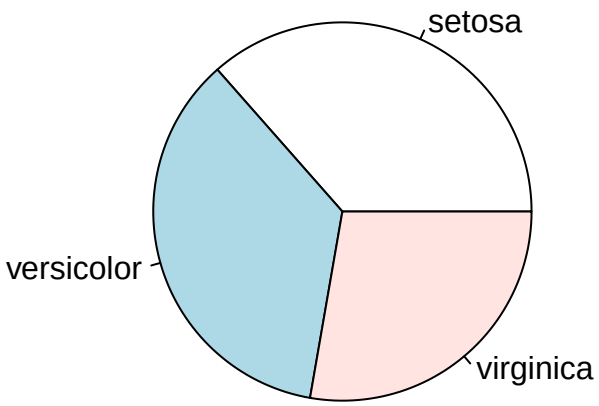
g) Construct a pie chart to describe the species with 'Sepal.Length' less than 7 centimeters.

```
species_counts <- c(0,0,0)
for (row in 1:nrow(iris)){
  if ((iris[row, "Species"] == "setosa") & (iris[row, "Sepal.Length"]< 7)){
    species_counts[1] <- species_counts[1] +1
  }
  if((iris[row, "Species"] == "versicolor") & (iris[row, "Sepal.Length"]< 7)){
    species_counts[2] <- species_counts[2] +1
  }
  if((iris[row, "Species"] == "virginica") & (iris[row, "Sepal.Length"]< 7)){
```

```
    species_counts[3] <- species_counts[3] +1
  }}
species_counts
```

```
## [1] 50 49 38
```

```
pie(species_counts, labels = c("setosa","versicolor", "virginica" ))
```



END of Assignment #1.