OR/SYST-568 Applied Predictive Analytics (Spring 2019)

Assignment 1 – Basics of R, Descriptive Statistics and Data Preprocessing

Due Date: 02/13/2019 Wednesday 11:59 PM

Submission:

1. Prepare a pdf file with answers to homework questions, with brief explanations for the analysis/implementation you perform. Please provide your R codes at the end of the submitted file as appendix or in a separate R file.
2. Name the file as "LastName, FirstName-HW1.pdf" (for example, "Ji,Ran-HW1.pdf") and submit it on blackboard.

Part I: (30pts) Review of R Basics.

This part is an opportunity to try the R statistical package and to start to learn some of its basic behaviors and options. Part I contains tutorials and questions. Please read through the descriptions and tutorials before doing the questions.

- Text like this will be general comments.
- **`Text like this will be my commands to R, the R prompt is a "greater than" sign (>).`**
- `Text like this will be output from R in my examples.`
- **Text like this will be problems for you to do and turn in. (There are 4 in this problem.)**

0. Assignment and basics

Assignment to an object name may be done using 1) an equals sign =, 2) a "left arrow" <- (less than, hyphen), or 3) a "right arrow" -> (hyphen, greater than).

You can type the name of any object to look at that object.

```
> n <- 15
> n
[1] 15
> a = 12
> a
[1] 12
> 24 -> z
> z
[1] 24
```

Variables must start with a letter, but may also contain numbers and periods. R is case sensitive.

```
> N <- 26.42
> N
[1] 26.42
> n
[1] 15
```

To see a list of your objects, use ls( ). The ( ) is required, even though there are no arguments.

```
> ls()
[1] "a" "n" "N" "z"
```

Use rm to delete objects you no longer need.

```
> rm(n)
> ls()
[1] "a" "N" "z"
```

You may see online help about a function using the help command or a question mark.

```
> ?ls
> help(rm)
```

Several commands are available to help find a command whose name you don't know. Note that anything after a pound sign (#) is a comment and will not have any effect on R.

```
> apropos(help)  # "help" in name
[1] ".helpForCall"   "help"           "help.search"    "help.start"
[5] "link.html.help"

> help.search("help") # "help" in name or summary; note quotes!

> help.start()   # also remember the R Commands web page (link on
                 # class page)
```

Other data types are available. You do not need to declare these; they will be assigned automatically.

```
> name <- "Mike"      # Character data
> name
[1] "Mike"

> q1 <- TRUE          # Logical data
> q1
[1] TRUE

> q2 <- F
> q2
[1] FALSE
```

1. Simple calculation

R may be used for simple calculation, using the standard arithmetic symbols +, -, *, /, as well as parentheses and ^ (exponentiation).

```
> a <- 12+14
> a
[1] 26
> 3*5
[1] 15
> (20-4)/2
[1] 8
> 7^2
[1] 49
```

Standard mathematical functions are available.

```
> exp(2)
[1] 7.389056
> log(10)  # Natural log
[1] 2.302585
> log10(10)      # Base 10
[1] 1
> log2(64) # Base 2
[1] 6
> pi
[1] 3.141593
> cos(pi)
[1] -1
> sqrt(100)
[1] 10
```

Q1: Use R as a calculator to compute the following values. After you do so, cut and paste your input and output from R to Word. Add numbering in Word to identify each part of each problem. (Do this for every problem from now on.)

(a) 27*(38-17)

(b) $ln(147)$

(c) $\sqrt{\frac{436}{12}}$

2. Vectors

Vectors may be created using the c command, separating your elements with commas.

```
> a <- c(1, 7, 32, 16)
> a
[1]  1  7 32 16
```

Sequences of integers may be created using a colon (:).

```
> b <- 1:10
> b
 [1]  1  2  3  4  5  6  7  8  9 10

> c <- 20:15
> c
[1] 20 19 18 17 16 15
```

Other regular vectors may be created using the seq (sequence) and rep (repeat) commands.

```
> d <- seq(1, 5, by=0.5)
> d
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

> e <- seq(0, 10, length=5)
```

```
> e
[1]  0.0  2.5  5.0  7.5 10.0

> f <- rep(0, 5)
> f
[1] 0 0 0 0 0

> g <- rep(1:3, 4)
> g
 [1] 1 2 3 1 2 3 1 2 3 1 2 3

> h <- rep(4:6, 1:3)
> h
[1] 4 5 5 6 6 6
```

Random vectors can be created with a set of functions that start with r, such as rnorm (normal) or runif (uniform).

```
> x <- rnorm(5)  # Standard normal random variables
> x
[1] -1.4086632  0.3085322  0.3081487  0.2317044 -0.6424644

> y <- rnorm(7, 10, 3)      # Normal r.v.s with mu = 10, sigma = 3
> y
[1] 10.407509 13.000935  8.438786  8.892890 12.022136  9.817101  9.330355

> z <- runif(10) # Uniform(0, 1) random variables
> z
 [1] 0.925665659 0.786650785 0.417698083 0.619715904 0.768478685
0.676038428
 [7] 0.050055548 0.727041628 0.008758944 0.956625536
```

If a vector is passed to an arithmetic calculation, it will be computed element-by-element.

```
> c(1, 2, 3) + c(4, 5, 6)
[1] 5 7 9
```

If the vectors involved are of different lengths, the shorter one will be repeated until it is the same length as the longer.

```
> c(1, 2, 3, 4) + c(10, 20)
[1] 11 22 13 24

> c(1, 2, 3) + c(10, 20)
[1] 11 22 13
Warning message:
longer object length
        is not a multiple of shorter object length in: c(1, 2, 3) + c(10,
20)
```

Basic mathematical functions will apply element-by-element.

```
> sqrt(c(100, 225, 400))
[1] 10 15 20
```

To select subsets of a vector, use square brackets ([ ]).

```
> d
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> d[3]
[1] 2
> d[5:7]
[1] 3.0 3.5 4.0
```

A logical vector in the brackets will return the TRUE elements.

```
> d > 2.8
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> d[d > 2.8]
[1] 3.0 3.5 4.0 4.5 5.0
```

The number of elements in a vector can be found with the length command.

```
> length(d)
[1] 9
> length(d[d > 2.8])
[1] 5
```

Q2: Create the following vectors in R.

a = (5, 10, 15, 20, ..., 160)

b = (87, 86, 85, ..., 56)

Use vector arithmetic to multiply these vectors and call the result d. Select subsets of d to identify the following.

(a) What are the 19th, 20th, and 21st elements of d?

(b) What are all of the elements of d which are less than 2000?

(c) How many elements of d are greater than 6000?

3. Simple statistics

There are a variety of mathematical and statistical summaries which can be computed from a vector.

```
> 1:4
[1] 1 2 3 4
> sum(1:4)
[1] 10
> prod(1:4)            # product
[1] 24
> max(1:10)
[1] 10
> min(1:10)
[1] 1
```

```
> range(1:10)
[1]  1 10

> X <- rnorm(10)
> X
 [1]  0.2993040 -1.1337012 -0.9095197 -0.7406619 -1.1783715  0.7052832
 [7]  0.4288495 -0.8321391  1.1202479 -0.9507774

> mean(X)
[1] -0.3191486

> sort(X)
 [1] -1.1783715 -1.1337012 -0.9507774 -0.9095197 -0.8321391 -0.7406619
 [7]  0.2993040  0.4288495  0.7052832  1.1202479

> median(X)
[1] -0.7864005

> var(X)
[1] 0.739266

> sd(X)
[1] 0.8598058
```

Q3: Using d from problem Q2, use R to compute the following statistics of d:

- **(a) sum**
- **(b) mean and median**
- **(c) standard deviation**

4. Graphics

R has functions to automatically plot many standard statistical graphics. Histograms and boxplots may be generated with hist and boxplot, respectively.

Once you have a graphic you're happy with, you can copy the entire thing. Make sure that the graphics window is the active (selected) window, and select "Copy to clipboard as bitmap" from the file menu. You can then paste your figure into Word and resize to taste.

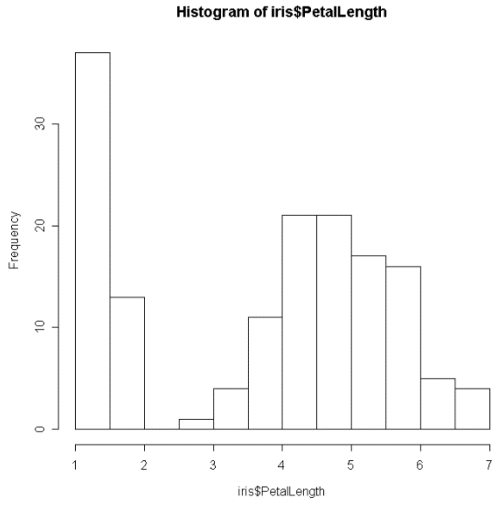
There is an alternative way to save the figure into a single PDF file using the R codes below.

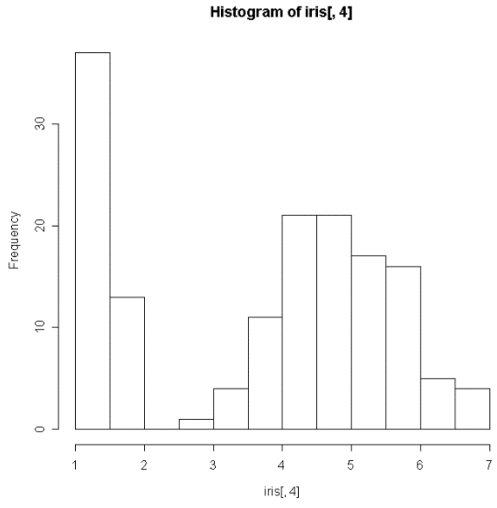
```
PDFPath = "C:\\Users\\<insert computer user name
here>\\Desktop\\NameOfFile.pdf"
pdf(file = PDFPath)
<insert your subject plot/chart/etc. code here>
dev.off()
```

You can use whatever path you want, but each sub-folder needs to be separated by a double backwards slash (\\). What is given above will save your PDF to the Desktop. The last piece of code closes the file.

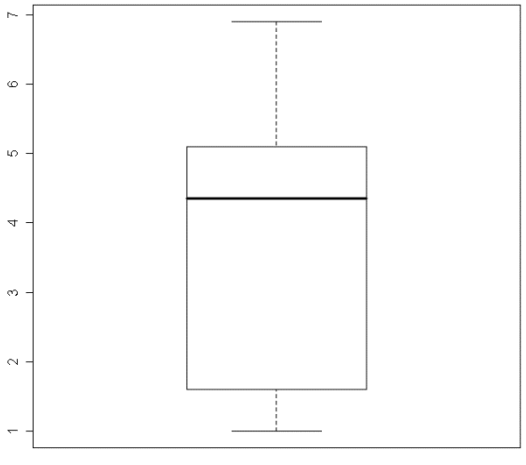
The code will print whatever is plotted and save it to a single PDF file, making post-processing your plots much more convenient.

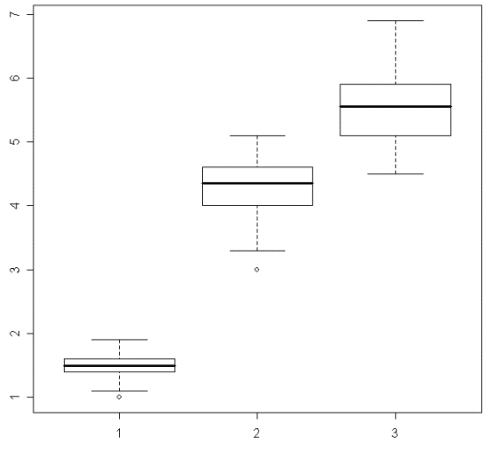
```
> data(iris)
> head(iris)
> hist(iris$Petal.Length)
> hist(iris[,4])# alternative specification
```






```
> boxplot(iris$Petal.Length)
> boxplot(PetalLength~Species, data=iris) # Formula description,
                                          # side-by-side boxplots
```





Q4: Read the dataset named "cars" and check the first several rows of the dataset.

> data(cars)

> head(cars)

(a) Plot a histogram of distance using the hist function

(b) Generate a boxplot of speed.

(c) Use the plot(,) function (e.g. plot(variableX, variableY) to create a scatterplot of dist against speed.

[Note: You can also create the graphs in parts (a) to (c) using ggplot2 package].

Part II: (35pts) Data Preprocessing (Exercise 3.1 of APM Book).

The UC Irvine Machine Learning Repository contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

The data can be accessed via:

```
> library(mlbench)
> data(Glass)
> str(Glass)
'data.frame':        214 obs. of  10 variables:
 $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
 $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
 $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
 $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
 $ Si  : num  71.8 72.7 73 72.6 73.1 ...
 $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
 $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
 $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
 $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
```

(a) Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors. Provide the pairwise scatter plots and investigate the correlation matrix.

(b) Do there appear to be any outliers in the data? Are any predictors skewed?

(c) Are there any relevant transformations of one or more predictors that might improve the classification model? (Hint: You could transform the predictors using the BoxCox Transformation. This can be done using mathematical formulation, or using the “preprocess” function in the AppliedPredictiveModeling package).

Part III: (35pts) Data Preprocessing (Exercise 3.2 of APM Book).

The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

The data can be loaded via:

```
> library(mlbench)
> data(Soybean)
> ## See ?Soybean for details
```

(a) Investigate the frequency distributions for the categorical predictors. Are there any extremely unbalanced categorical predictors? In the extreme case, if there is only one value for the predictor, it is called a degenerate case. Such degenerate predictors should be removed from subsequent analysis.

(b) Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

(c) Develop a strategy for handling missing data, either by eliminating predictors or imputation.