MATH7343 – Applied Statistics Computer Lab 1.

Learning Objectives:

1) Running R in the Rstudio environment;
   Writing and Running a R program;
   Saving R Output and Program files.
2) Learn elementary data analysis tasks in R:
   Graphic and Numerical summaries of data.

Instructions:

(0) Preparations:

(a) You should have already install the R and Rstudio on your machine. See instructions on CANVAS (use your MyNEU password to log-in). Go to the course MATH7343 Applied Statistics, under Module 2 find Lab1.

(b) Download the example R programs and datasets from Canvas (Lab1 in Module 2). The R examples are in the file "lab1.r". There are several text files containing the data sets fsalary.txt, salary.txt, unicef.txt. (The last data sets is also available from the CD in textbook.). Download them to a directory folder on your own machine. (Putting them all in the same folder so that the R session can find them.)

(1) Open the 'lab1.r' file using Rstudio. Click on the file and use right button to select "open with" Rstudio.

2) After starting the Rstudio, you will see a large window with four smaller subwindows in it. The upper-left subwindow is the Editor, where the code in 'lab1.r' now shows. The lower-left subwindow is the Console, this is where the running R commands and outputs will show. The Console shows the exactly the same information as in the interface of basic R installation (Rstudio basically expands the basic interface to show more information to the user.) The windows on the right side shows information like the graphic outputs, which you will learn to appreciate later.

3) Running the R program

Go to Editor subwindow. Notice that there is a "run" button on the upper-right corner of this subwindow. Highlight the part of code you want to run, with your mouse, then click "run" button to submit. Then those commands shows up in the Console below, together with the results of execution.

Notice that, without Rstudio, you enter commands in the Console window directly. You may copy and paste the commands from your own text editor. It is recommended that you always keep your R commands in a text file such as 'lab1.r' (Those are called R scripts). This allows you to easily reproduce the analysis results in the future, by simply executing the saved R scripts.

Let us look at the code in the Editor subwindow in detail to understand what they are doing.

```
##### Example 1: the Faculty Salary data
# Import data set. This is formatted two columns/variables
fsalary.data <- read.table(file="fsalary.txt", header=TRUE)
fsalary.data #display data

# For the categorical variable RANK, do a frequency table
table(fsalary.data$RANK)
# Draw a barchart
barplot(table(fsalary.data$RANK), main='Bar Chart for Rank')

#Draw a histogram for the continuous variable SALARY
hist(fsalary.data$SALARY)
#Define our own breakpoints for histogram: 27.5, 42.5, ..., 97.5
hist(fsalary.data$SALARY, breaks=27.5+(0:14)*5)

#Draw a boxplot
boxplot(fsalary.data$SALARY)
#Boxplot by the rank
boxplot(fsalary.data$SALARY~fsalary.data$RANK)
##A different syntax doing the same task
boxplot(SALARY~RANK, data=fsalary.data)

#Calculate statistics: mean, median, standard deviation
mean(fsalary.data$SALARY)
median(fsalary.data$SALARY)
sd(fsalary.data$SALARY)

#Summary statistics: min, max, 1st/3rd quartile, mean, median.
summary(fsalary.data$SALARY)
#Contrast this with summary of the whole data set (two variables)
summary(fsalary.data)  #For class(nominal) variable 'RANK', it gives the counts.

#More summary statistics, use some package. Here use 'psych' for example.
#The package needs to be installed beforehand. Installation is needed only once
#  on a computer. But needs to load the library once every R session.
#install.packages("psych")
library(psych)
describe(fsalary.data$SALARY)
#Summary statistics by groups (rank)
describeBy(fsalary.data$SALARY, fsalary.data$RANK)

#Clear workspace
#You do not have to do this.
#But it is a good habit to clear out all data after finishing.
#(Just to avoid confusion, may inadvertently use same variable name next time.)
rm(list=ls())

##### Example 2: the Response time data
# Import data set. This is just one variable.
#  We use 'scan' which input unformatted data.
#  This will create a vector variable
MyTime <- scan(file="ResponseTime.txt")
# We can also make this vector into a data frame as above (here only one column)
time.data<-data.frame(MyTime)
# We now add new variables to this data frame
time.data$obs.order<-seq(length(time.data$MyTime))
```

```
time.data$obs.phrase<- ifelse(time.data$obs.order<=5, 1,
                          ifelse(time.data$obs.order<=10, 2,
                                 ifelse(time.data$obs.order<=15, 3, 4)))
# Scatterplot of response time versus observation order
plot(MyTime~obs.order, data=time.data)
# stratified scatterplot of response time by 4 phrases of observations.
plot(MyTime~obs.phrase, data=time.data)

#Clear workspace
rm(list=ls())

##### Example 3: the trees data
# There are datasets that comes with R. So no need to import them.
#  Use data() to see which datasets are there.
#  We use 'trees' data as an example.

#Summary statistics
summary(trees)
#Scatterplot of Volume versus Height
plot(Volume~Height, data=trees)

# We can inspect the contents of a dataset with str()
str(trees)
# Display the first few observations
head(trees)
```

(A)
First, notice that there are some lines displayed in color green. Those are comments (after '#'), and will not be executed as R commands. It is a good habit as a programmer to put in comments so that people (including yourself) can understand the code.

The commands are mostly self-explanatory. Let us execute the first two commands. You will see in the Console the following

```
> ##### Example 1: the Faculty Salary data
> # Import data set. This is formatted two columns/variables
> fsalary.data <- read.table(file="fsalary.txt", header=TRUE)
> fsalary.data #display data
   SALARY RANK
1    77.0 Full
2    79.0 Full
3    80.0 Full
4    85.0 Full
5    86.0 Full
...
```

You can see that the data in 'fsalary.txt' has two columns, with first row giving the names 'SALARY' and 'RANK'. The read.table() function reads the data in, and we store it in a table called 'fsalary.data', which has two columns with column(variable) names 'SALARY' and 'RANK'. R automatically recognize the first variable as numeric, the second variable as factor (categorical).

You may also look at the contents of 'fsalary.data' by: in the upper-right subwindow, click on 'environment' and then 'fsalary.data'.

Let us execute the next two commands. In the Console:

```
> # For the categorical variable RANK, do a frequency table
> table(fsalary.data$RANK)

Assi Asso Full
  50   71   73
> # Draw a barchart
> barplot(table(fsalary.data$RANK), main='Bar Chart for Rank')
```

We see the frequency table for the variable RANK here. Notice the variable specification is DataName$VariableName (`fsalary.data$RANK`).

Then we create a bar plot, the graphic output shows in a new graphic subwindow, which comes up in the lower-right subwindow of Rstudio. Please locate it.

When we produce more plots later, they all show in that subwindow. Use the Arrow button there to move from one plot to another.

(B)
Next, let us look at the rest of the first example. The following commands produce graphic and numerical summary statistics, which are explained in the comments.

```
#Draw a histogram for the continuous variable SALARY
hist(fsalary.data$SALARY)
#Define our own breakpoints for histogram: 27.5, 42.5, ..., 97.5
hist(fsalary.data$SALARY, breaks=27.5+(0:14)*5)

#Draw a boxplot
boxplot(fsalary.data$SALARY)
#Boxplot by the rank
boxplot(fsalary.data$SALARY~fsalary.data$RANK)
##A different syntax doing the same task
boxplot(SALARY~RANK, data=fsalary.data)

#Calculate statistics: mean, median, standard deviation
mean(fsalary.data$SALARY)
median(fsalary.data$SALARY)
sd(fsalary.data$SALARY)

#Summary statistics: min, max, 1st/3rd quartile, mean, median.
summary(fsalary.data$SALARY)
#Contrast this with summary of the whole data set (two variables)
summary(fsalary.data)  #For class(nominal) variable 'RANK', it gives the counts.
```

(C) Packages: R base installation has just some basic stuffs. For statistical analysis, we often want more procedures. To do this, we can write R functions to implement the procedures. Those functions can be distributed through 'packages' to other users.

For example, here we want more summary statistics than those given by 'summary()". We can write the function our own, or use some package written by others. Here, we use the psych() package.

```
> #More summary statistics, use some package. Here use 'psych' for example.
> #The package needs to be installed beforehand. Installation is needed only once
> #  on a computer. But needs to load the library once every R session.
> #install.packages("psych")
> library(psych)
> describe(fsalary.data$SALARY)
  vars   n  mean    sd median trimmed   mad  min  max range skew kurtosis   se
1    1 194 56.41 10.72   55.4   56.08 11.05 35.1 89.9  54.8 0.37    -0.19 0.77
> #Summary statistics by groups (rank)
> describeBy(fsalary.data$SALARY, fsalary.data$RANK)
group: Assi
  vars  n  mean   sd median trimmed  mad min  max range  skew kurtosis   se
1    1 50 47.17 5.24  48.35   47.44 5.26  36 55.8  19.8 -0.47    -0.72 0.74
------------------------------------------------------------------------------
--
group: Asso
  vars  n  mean   sd median trimmed  mad  min  max range  skew kurtosis   se
1    1 71 52.72 7.23   53.9   52.96 6.52 35.1 69.3  34.2 -0.29    -0.19 0.86
------------------------------------------------------------------------------
--
group: Full
  vars  n  mean   sd median trimmed  mad  min  max range skew kurtosis   se
1    1 73 66.33 8.12   66.3   66.23 7.56 46.5 89.9  43.4  0.2     0.58 0.95
```

Here we get a few more summary statistics.

NOTICE: If you have not installed the 'psych' package before, then you will get an error message and not able to get these outputs. The error messages are displayed in red color in the Console. For debugging, it helps to locate the first red error message and start from there.

The installation is done by the command

```
install.packages("psych")
```

Here I commented it out, since it only needs to be done on a computer once. Do not reinstall it every time.

(D) Another good habit is to clear up your workspace after each analysis. This avoids confusion later. Without regularly doing this, your R session will contain many variables from older analysis after a while. The clearance is done with command

```
rm(list=ls())
```

ls() list all object names in the workspace. rm() removes the objects listed.

(E) Another data import method: scan()

Let us look at the next example in the code.

```
##### Example 2: the Response time data
# Import data set. This is just one variable.
#  We use 'scan' which input unformatted data.
#  This will create a vector variable
MyTime <- scan(file="ResponseTime.txt")
```

The scan() method takes unformatted data. That is, just read in the entries one by one and put them into one vector. (Even if your text file is in a table of two columns like fsalary.txt in Example 1, scan() will read them all into one vector.)
Here ResponseTime.txt contains the times I took from an java-applet.

Notice, if you want the data to be in the standard data set format, set it to data.frame.

```
# We can also make this vector into a data frame as above (here only one column)
time.data<-data.frame(MyTime)
```

(F) Assign values to variables. You can also define new variables as the following example.

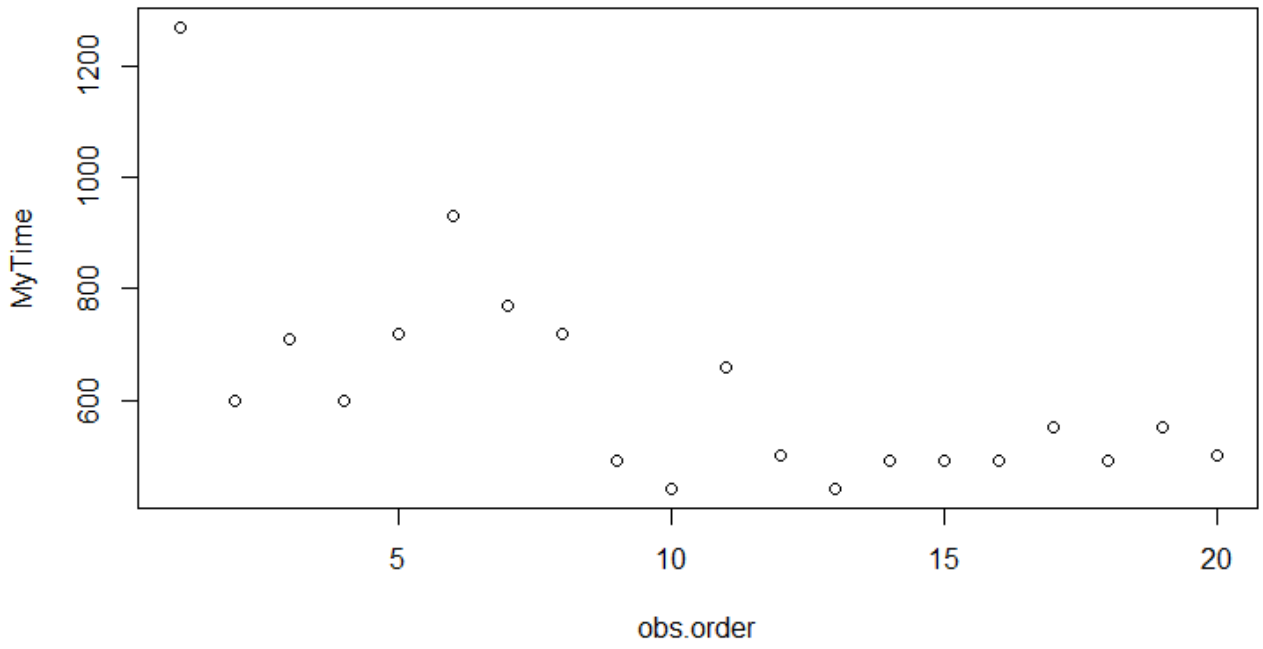
```
# We now add new variables to this data frame
time.data$obs.order <- seq(length(time.data$MyTime))
time.data$obs.phrase <- ifelse(time.data$obs.order<=5, 1,
                          ifelse(time.data$obs.order<=10, 2,
                                 ifelse(time.data$obs.order<=15, 3, 4)))
```

Notice here, seq(n) function outputs the vector (1,2,…,n). There are 20 observations in this data set, so it is (1, 2, 3, … , 20).
We also separate these observations into four phrases, every 5 observations in one phrase: the first 5, from 6 to 10, from 11 to 15, and from 16 to 20. Notice how I did this with ifelse(): ⩽5 becomes phrase 1, >5 but ⩽10 becomes phrase 2, >10 but ⩽ 15 becomes phrase 3 and >15 becomes phrase 4.

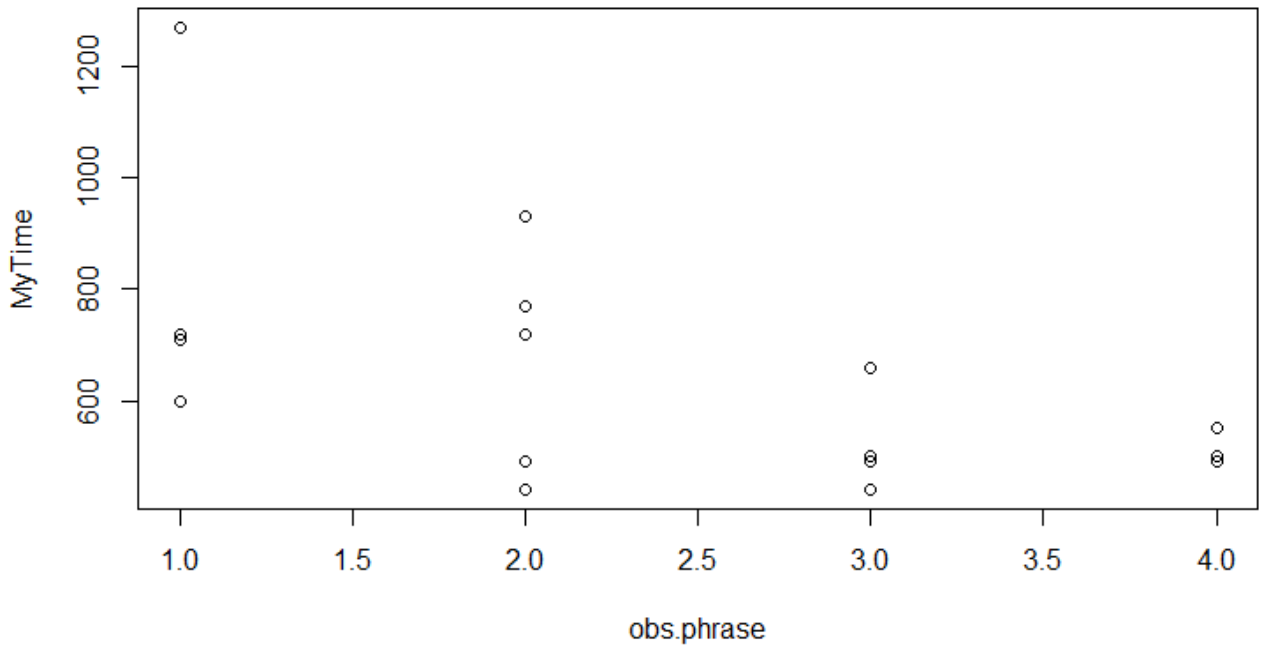
The assigning of values are done using the ‘ <- ’ symbol. In R, you can also use ‘ = ’ to do the exact same thing.

(G) Collecting outputs.
Here we do two more plots. The plots show up in the graphic subwindow. I have copied them below.

```
# Scatterplot of response time versus observation order
plot(MyTime~obs.order, data=time.data)
```




```
# stratified scatterplot of response time by 4 phrases of observations.
plot(MyTime~obs.phrase, data=time.data)
```



The copying was done using the 'Export' button on the subwindow. Then choose 'Copy to clipboard", then 'copy plot'. I then paste the plot into a MSword file. (Microsoft's Office365 is available to all Northeastern students for free.)
You can also save the plots as image files using other options under 'Export'.

Important: Your homework solutions should contain clearly organized relevant R outputs (graphical and numerical – from the Console).
You can organize them by copy and paste, into one Word file, as illustrated above. Then wrote your answers to questions.

(H) Use existing data sets. There are data sets already in the base R installation. They are easy to use as no import is needed. This is illustrated in Example 3.

```
##### Example 3: the trees data
# There are datasets that comes with R. So no need to import them.
#  Use data() to see which datasets are there.
#  We use 'trees' data as an example.

#Summary statistics
summary(trees)
#Scatterplot of Volume versus Height
plot(Volume~Height, data=trees)
```

If you do not know what is in an R object (data set here). You can use str() and head() to get a quick glance.

```
> # We can inspect the contents of a dataset with str()
> str(trees)
'data.frame':   31 obs. of  3 variables:
 $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
 $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
 $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
> # Display the first few observations
> head(trees)
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
```

4) **Summary**.
We have through the above examples taught some basic operations of R on data sets.

(A) You should be familiar with the Rstudio environment. Know how to edit, run a R code. How to save your output (numerical and graphical).
(B) Know how to import data from .txt file, both formatted and unformatted.
(C) Getting simple numerical summary statistics and plots.

Further help:

To know how the syntax of any R command, use "?" to bring up the help file. For example, type in the Console: `?head`

You will get to the help file for command 'head' in the lower-right subwindow (where the graph subwindow is).

Also, google usually is a helpful tool to find how something is done in R. We taught how to import the .txt data files. Other type of files are also common, such as Excel file. You can use ?scan, ?read.table to see if these commands take .csv files. Or simply google 'r import .csv files'.

---

5) *Mini-project: quick fingers -- collecting and analyzing your response time.*

Now you will collect your own response time and modify the above program to produce a scatterplot and a stratified scatterplot with 3 phrases.

In the module 2 there is a link A webpage that measure your response time and instructions on using it. Click on the link, a new browser window will pop up with the webpage. For today's project, you should collect thirty response times clicking the "Medium" sized box. Follow the instructions on the page.

Click 'clear' button. Then select the button in each step: "one sample" in step 1, "Medium" in step 2. Hit the "Start" button in step 3, then hit on the popped-up square quickly. Your response time is now recorded in the table below.

Copy and paste your response times (in three batch of ten each) into a .txt file. Then write an R code to produce a scatterplot and a stratified scatterplot with 3 phrases (the first 10, the next 10, and the last 10 response times).

You can do this by modifying the above example program.

There is an issue of the extra three columns of zeros when you copy and paste. One way is to delete the zeros manually yourself, then input the data as unformatted data as in the example 2. Or you can paste the data in table format with the extra three columns of zeros, input the data as formatted data as in example 1, then ignore the last three columns. I posted how I inputted twenty responses times as formatted data (see the files: ExampleTimeCollection.rtf, ExampleTimeCollection.txt, ExampleTimeCollection.r)