Lab 1: R Language and Simple Linear Regression

1. Download R from http://cran.r-project.org and install it. This basic application includes the command line console and a simple script editor, which are sufficient for the purposes of this course.

2. If you would like to have a more powerful code manager, RStudio is recommended and can be downloaded from www.rstudio.com. RStudio does not work by itself. It requires that R is installed on your computer. Follow step 1 to install R.

3. Useful websites for learning R:
   a. Short introduction tutorial: https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf
   b. R reference card: https://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf
   c. Quick tutorial on using R in various statistical themes: http://www.statmethods.net/
   d. *Practical regression and anova using R* by Julian Faraway: cran.r-project.org/doc/contrib/Faraway-PRA.pdf

4. **Basic arithmetic.** Type the codes below in the **console** (i.e., command window), followed by the RETURN/ENTER key, to carry out the following calculation in R.

   a. Addition / Subtraction. `1430 + 8748`
   b. Multiplication / Division. `24 * 32`
   c. Power. `4 ^ 2`
   d. Natural exponential. `exp(1)`
   e. Natural log. `log(1)`
   f. Square root. `sqrt(225)`
   g. Take the integer part of a quotient. `10 %/% 3`
   h. Take the remainder of a division. `10 %% 3`

5. **Vectors.**
   a. Create a contiguous integer sequence using ":". Type the following in the console, followed by the RETURN/ENTER key, to see what you get:
   ```
   1:3
   6:(-4)
   ```
   b. Create a vector of any chosen numbers using **c( )**. Try the following and see what you get:
   ```
   c(1, 10, 100)
   ```
   c. Create a vector joining single numbers and sequences. Try the following and see what you get:
   ```
   c(1:3, 9, 4:2)
   c(1, 1, 1, 2:10)
   ```

d. Create a sequence of numbers with specified start, end, and step values using **seq( )**.

```
seq(from=1, to=11, by=2)
seq(10, 15)
seq(9, -1, -2)
```

Note: when the arguments follow the correct order, the argument prompts can be omitted such as in (ii) and (iii), but it is recommended that argument prompts are included in the codes especially in more complicated functions.

6. **Using internal help file.** Find out the syntax rule for **seq( )** by type in **help(seq)**. Understand the explanations in the manual page.

   a. What is the name of the function and which library does it reside? (Hint: top left corner)

   b. What is the general purpose of the function?

   c. What are frequently used arguments of this function, what do they mean, and what are their default values?

   d. What are the typical uses of the function?

   e. What are the outputs of the function? (Hint: see under “value” section)

   f. What are related functions that you might find useful?

   g. Read the examples and see if you’ve learned anything new.

7. **Matrix operation.** Try the codes below and understand what each function does. Write notes on the side as you need. An example is given for a few lines.

   Hints:
   (1) You can view the outputs to help figure out the functionality of the codes.
   (2) For explicit functions (explicit functions always have parentheses following the function names, such as **matrix( )**, **rownames( )**, **nrow( )**.) Type in **help( )** with the function name inside to look up the function.

```
A = matrix(data=1:9, nrow=3, ncol=3)
```
Create a matrix of 3 rows and 3 columns containing the numbers 1 through 9, and assign the matrix to an object named "A".

```
rownames(A) = c('r1', 'r2', 'r3')
```
Assigning 'r1', 'r2', and 'r3' as the row names for A.

```
colnames(A) = c('c1', 'c2', 'c3')
```

```
A
```
Show the contents of A on the console.

```
A[1, 2]
A[2, ]
A[1:3, ]
A[c(1, 3), ]
A[-3, ]
```
Remove the third row of A.

```
A[, 'c2']
```
Subset the column named 'c2' in A.

```
nrow(A)
ncol(A)
dim(A)
length(A)

z = c(100, 200, 300)
rbind(A, z)
cbind(A, z)

min(z)
sum(z)
mean(z)
sd(z)
var(z)
```

```
max(A)
sum(A)
rowSums(A)
colSums(A)
rowMeans(A)
colMeans(A)
```

8. **Logical operation.** Logical operation returns **TRUE** or **FALSE**. Try out the following codes and understand what each line does. Write notes as you need.

```
1 < 2
1 > 2
1 <= 2
3 >= 3
1 == 1
1 != 1
z = seq(from=1, to=9, by=2)
4 %in% z
```

Logical operation can be used to locate elements in a vector that satisfy the condition. Try out the following codes and understand what they do.

```
z>=5
which(z>=5)
which(z>=5 & z<9)
which(z>=5 | z<9)
z[which(z>=5)]
z[z>=5]
```

9. **Data management.**

   a. See current working directory by **getwd( )**.
   b. Set current working directory to your own choice by **setwd("…")**. Replace "…" in the parentheses with your directory path. (*Note:* If you use a Windows PC, replace all the "\" with "/" or "\\" in the path. You can also set working directory using the drop-down menu.)
   c. Download "TRUCKING.txt" from Canvas and put the file in your working directory folder.

d. Type **truck = read.table('TRUCKING.txt')** to read in the data. Then type in **truck** to view the imported data. Is there anything wrong?

e. Type **help(read.table)** or **?read.table** to bring up the manual and find out how to fix the problem. What was your solution?

f. After you correctly read in the data, type **class(truck)**. What type of object is it?

10. **Library management.**
    a. Install the "faraway" package by typing in **install.packages("faraway")**. You may need to choose a server mirror next; choose one that locates close to you. Once you have installed a package on your computer, you do not need to install it again unless you re-installed R or need to update the package.
    b. After you have successfully installed the package, every time you start a new R session, you need to load the package in order to use the functions and/or datasets included in it. Load the "faraway" package by typing in **library(faraway)**.
    c. You can load any dataset included in the package by **data( )**. Try **data(stat500)**.

11. **Simple data analysis.** Load the "stat500" dataset in the "faraway" package as instructed above. Try the following commands and understand what they do.

```
stat500
?stat500
head(stat500)
tail(stat500)
nrow(stat500)
summary(stat500)
stat500[, 1]
stat500$midterm
```

The $ sign is used to call a column that resides within a specific dataset (of the "data.frame" type). Only calling the column (e.g., midterm) will not work since the column does not exist by itself in the workspace.

```
hist(stat500$total)
```

You may attach a dataset so that you do not have to call the dataset name every time you call the variables within the dataset.

```
attach(stat500)
hist(total)

boxplot(total)
boxplot(midterm, final, names=c("midterm", "final"))

total.cut <- cut(total, c(0, 59.4, 100))
total.count <- table(total.cut)
barplot(total.count)

pie(total.count)
pie(total.count, label=c("fail", "pass"))
```

12. **Simple linear regression.** Try out the following codes. Understand what they do.

```
mod <- lm(final ~ midterm, data=stat500)
summary(mod)

coef(mod)
predict(mod)
predict(mod, interval="confidence")
predict(mod, interval="confidence", level=.90)
residuals(mod)
rstandard(mod)

ls(mod)
mod$coefficients

plot(midterm, final, xlim=c(10,35), ylim=c(10,35))
abline(coef(mod))
text(30, 10, sprintf("Pearson's correlation: %.3f",
     cor(midterm, final)))
```

13. **For loop and if statement.** Below is a sample of "for" loop and "if" statement, basic components of programming with R. Run these codes and try to understand the syntax structure. Section 11 in *Short-R-Intro.pdf* available on Canvas has more explanation.

```
y = sample(-5:5, 5)


for (i in 1:5){
  if (y[i] > 0) {
    cat(sprintf('%d is a positive number.\n', y[i]))
  } else if (y[i] < 0) {
    cat(sprintf('%d is a negative number.\n', y[i]))
  } else {
    cat(sprintf('%d is zero.\n', y[i]))
  }
}
```

14. **Find critical values for hypothesis testing and confidence interval.**
    a. Type in **help(pt)** to bring up the manual of four functions related to *t*-distribution. Understand what each function does and each argument in the functions corresponds to.
    b. Use **pt()** to find the *one-sided left-tail* p-value for $t = -1.98$ with $df = 8$. Report the result.

    c. Change pertinent arguments in **pt()** to find the *one-sided right-tail* p-value for $t = 1.98$ with $df = 8$. Report the result.

    d. What is the *two-sided* p-value for $t = 1.98$ with $df = 8$. Report the result.

    e. Use **qt()** to find the critical *t*-value for constructing a 90% confidence interval with $df = 30$. Report the result. (Hint: to find a 95% CI, the specification would be **qt(p=.975, df=30)** or **qt(p=.025, df=30, lower.tail=F)**.)