Statistics 147: LAB #1
10 pts; Summer 2019

NAME: (Please print)	Wesley Chang
ID: (last 4 #s only)	0996

This lab is designed to introduce the student to *SAS Version 9 for Windows* and **R** and perform some elementary calculations using both.

1 Downloading Information from *Blackboard*

Information for the labs and other course materials have been and will be stored on the course WEB site on *Blackboard*. To access these materials, you can use the following procedure (in this lab).

To complete future labs, you will need to download the following files from *iLearn*. They are located under **Data Files**.

cartest1.dat:

```
Car BrandA BrandB
 1    125    133
 2     64     65
 3     94    103
 4     38     37
 5     90    102
 6    106    115
```

gas1.csv:

```
premium regular
35.4    29.7
34.5    29.6
31.6    32.1
32.4    35.4
34.8    34.0
31.7    34.8
35.4    34.6
35.3    34.8
36.6    32.6
36.0    32.2
```

Log on to *iLearn (ilearn.ucr.edu)*. Click on **Statistics 147** → **DataFiles**. You should see a link to *cartest1.dat*. **Right** click on that link and a sub-menu will appear. Select *Save Target As* to open the *Save As* window. Pay attention to where the file was saved.

You should also see a link to *gas1.csv*. Repeat the same steps as above to save the *gas1.csv* file.

Take a moment to open both files to inspect their contents. On a Windows computer, you can use the *Notepad* text editor. On a Mac, use may use *TextEdit*

2 Introduction to SAS

This lab is designed to familiarize the student with entering, executing and saving a file using *SAS Version 9 for Windows.*

1. To launch SAS, double-click on the **SAS** icon on the Windows desktop.

   RECALL: You will type your program in the **Program Editor** window!

   REMINDER: Each time you make a change to your SAS program, remember to save the file and then execute it again to update the output.

   We will use the following example:

2. A statistics teaching assistant was interested in determining whether the average quiz scores differed between his two discussion sections. The data (average for each of seven quizzes) is as follows:

Quiz	1	2	3	4	5	6	7
Section 1	7.6	6.2	8.0	5.6	9.7	7.6	6.6
Section 2	6.6	8.2	6.5	4.8	9.0	7.2	5.7

   (i) First we want to enter the data and print it. Simply type the following in the **Program Editor** window. (See below.)

   NOTE: You must enter a value for *quiz* then a value for *section1* and then a value for *section2*. (See below. Later we'll see an easier way to enter this type of data!)

   NOTE: You may enter comments in the program by inserting them between /* and */. (See below.)

   (Once you have entered finished entering the data, be sure you check to make sure the entries are correct. See below.)

```
/* Set up format of the output */
options nocenter ps = 55 nocenter ls = 78 nodate nonumber formdlim='*';


/* Use DM to clear all windows except the editor window */
DM log "odsresults; clear; out; clear; log; clear;";

ods graphics off;

/* Create a temporary SAS data set and enter the values manually*/
data quizzes;

/* Input variable list
   quiz     represents quiz number
   section1 represents quiz averages for Section 1
   section2 represents quiz averages for Section 2
   @@       allows entry of more than one piece of data per line */

   input quiz section1 section2 @@;

/* Use datalines command to indicate the data is about to follow.
    Enter the values alternating between quiz, section1 and section2. */
```

```
datalines;
1  7.6  6.6  2  6.2  8.2  3  8.0  6.5  4  5.6  4.8
5  9.7  9.0  6  7.6  7.2  7  6.6  5.7
;
run; /* use run statement to separate data step from proc step */

/* Print the data as a check */

proc print data = quizzes;
run;
quit;
```

Note: The two @@ on the end of the input line allow you to enter more than one set of data values on a line. Without them, SAS would read exactly one value for *quiz*, one value for *section1* and one value for *section2* from each line. Try removing the two @@ to see for yourself! Just be sure to put them back and run again.

(i) To **save** the *.sas* script, select **File** → **Save As**. The **Save As** window open. Navigate to the desired folder where you will be saving your work. In the box next to *File name:*, type in **saslab1** and click on **Save**. This will save your file and return you to the **Program Editor** window.

(ii) To **execute** your program, select **Run** → **Submit** or click on the *Stickman Running Figure* on the main toolbar.

Upon executing the file in SAS, you will see information being written to the **Log** window. This is a listing of all the errors/executions. If everything goes well, you will also get output written to the **Output** window. This **Output** window will take up the entire screen.

The output you should see is

Obs	quiz	section1	section2
1	1	7.6	6.6
2	2	6.2	8.2
3	3	8.0	6.5
4	4	5.6	4.8
5	5	9.7	9.0
6	6	7.6	7.2
7	7	6.6	5.7

To return to the **Program Editor** window, click on the **-** in the corner of the **Output** window to minimize the **Output** window.

(iii) Suppose you would like to calculate an overall average (for both sections combined) for each quiz. This may be accomplished by modifying the existing code as indicated below; Look through the following program and just add the appropriate lines of code to your existing program. The result should be a new column/variable called *overall_average*

```
/* Set up format of the output */
options nocenter ps = 55 nocenter ls = 78 nodate nonumber formdlim='*';

/* Use DM to clear all windows except the editor window */
DM log "odsresults; clear; out; clear; log; clear;";
ods graphics off;

/* Create a temporary SAS data set and enter the values manually*/
```

```
data quizzes;

/* Input variable list
   quiz     represents quiz number
   section1 represents quiz averages for Section 1
   section2 represents quiz averages for Section 2
   @@       allows entry of more than one piece of data per line */

   input quiz section1 section2 @@;

/* ***************************NEW CODE BEGINS HERE ***********************/
/* Use mean function to calc the average for both sections for each quiz */
           overall_average = mean(of section1 - section2);
/* ***************************NEW CODE ENDS HERE *************************/

/* Use datalines command to indicate the data is about to follow.
    Enter the values alternating between quiz, section1 and section2. */
datalines;
1  7.6  6.6  2  6.2  8.2  3  8.0  6.5  4  5.6  4.8
5  9.7  9.0  6  7.6  7.2  7  6.6  5.7
;
run; /* use run statement to separate data step from proc step */

/* Print the data as a check */

proc print data = quizzes;
run;

quit;
```

NOTE: Now save your program by selecting **File** → **Save** or by clicking on the diskette on the main toolbar. Don't forget to re-run your program through SAS to get the updated output file! (RECALL: Select **Run** → **Submit** or click on the *Stickman Running Figure* on the main toolbar.

NOTE: Every time you make a change to your SAS program file, you must save it and re-submit/re-run/re-execute it!

Complete the following table.

quiz	section1	section2	overall_average
1	7.6	6.6	7.10
2	6.2	8.2	7.20
3	8.0	6.5	7.25
4	5.6	4.8	5.20
5	9.7	9.0	9.35
6	7.6	7.2	7.40
7	6.6	5.7	6.15

3. Now sort the data according to the values of *overall_average* using **proc sort**. Recall, that the **sort** routine does NOT print the data, so you must add a **proc print** after the **sort** routine. (Look through the following program and just add the appropriate lines of code to your existing program.)

```
/* Set up format of the output */
options nocenter ps = 55 nocenter ls = 78 nodate nonumber formdlim='*';
 /* ls = linesize,  ps = pagesize
           nocenter      justifies the output so it is not centered on the page
           nodate        suppresses printing of today's date on each page of output
           nonumber      suppresses printing of page number on each page of output
           formdlim      overrides the internal page breaks and replaces them
                            with the designated symbol*/

/* Use DM to clear all windows except the editor window */
DM log "odsresults; clear; out; clear; log; clear;";

ods graphics off;

/* Create SAS data set  and variable input list*/
/*  @@ allows entry of more than one piece of data per line */
data quizzes;
/* Input variable list
                 quiz     represents quiz number
                 section1 represents quiz averages for Section 1
                 section2 represents quiz averages for Section 2 */
            input quiz section1 section2 @@;

/* Use mean function to calculate an average for both sections for each quiz */
           overall_average = mean(of section1 - section2);
/* Use datalines command to indicate the data is about to follow.
    Enter the values alternating between quiz, section1 and section2. */
datalines;
1  7.6  6.6  2  6.2  8.2  3  8.0  6.5  4  5.6  4.8
5  9.7  9.0  6  7.6  7.2  7  6.6  5.7
;      /* use semi-colon to separate data step from proc step */
 /* Print the data as a check */
proc print;
/* ************************NEW CODE BEGINS HERE *********************** */
proc sort data = quizzes;
      by overall_average;
/* Print the sorted data */
proc print data = quizzes;
/* ************************NEW CODE ENDS HERE *********************** */
run;
quit;
```

(Don't forget, you must run/execute the program through SAS to get an updated output file! Save the file and then submit it for execution.)

(a) Which **quiz** had the highest overall average? 5

What was the largest value of *overall_average*? 9.35

(b) Which **quiz** had the lowest overall average? 4

What was the smallest value of *overall_average*? 5.20

4. Suppose you only want to print the overall averages and also do not want the observation numbers printed. Modify the **proc print** statement and code as follows:

```
/* ****************************************************
    noobs    suppresses printing of observation numbers
    Use var statement to select which variables to print
   ***************************************************** */
proc print noobs;
    var  overall_average;
```

   Be sure to save and execute your program!

   When your output appears on the screen, ask a lab partner or the TA to check your work and then write their initials here. ____________________

5. You do not need to save your output file for this lab. To **Exit** SAS, select **File** → **Exit** or click on the **X** in the upper corner. You will be asked *Are you sure you want to end the session?*. Click on **OK**.

3 Introduction to R

1. **To invoke R**: Double-click on the *R 3.x.y* icon, where *x* and *y* represent the current installed version.
2. When **R** opens you should see the main menu and a window called the **R Console**. The **R Console** window is where you will complete all your work.
3. **R** is command-line driven. All commands must follow the > prompt (similar to the **MTB** prompt in Minitab.
4. You always tell **R** to execute a command line by pressing the *Enter* key.
5. Commands are separated either by a semi-colon (;), or by a newline. Elementary commands can be grouped together into one compound expression by braces (and).
6. The R Console window allows command editing. The left and right arrow keys, home, end, backspace, insert, and delete work exactly as one would expect. The up and down arrow keys can be used to scroll through recent commands. Thus, if you make a mistake all you need to do is press the up key to recall your last command and edit it.

NOTE: The backslash (/) character has a special meaning to R. To specify a Windows path in the RConsole, either

1. use double backslashes (\\), or
2. use a forward slash (/). (It is suggested that you use this option.)

NOTE: R programs end in **.r**.

NOTE: Comments in **R** begin with a # sign.

```
> # This is a comment
```

NOTE: To quit **R**, type

```
> quit()   # or q()
```

NOTE: To list the contents of your workspace,

```
> ls()                          # List the contents of the workspace.
> rm(list=ls())                 # This completely clears the workspace.
> ls()
character(0)                    # This means "nothing to see here"
```

Setting your working Directory:

First create a new folder called **RSpace** in the location of your choice. To create a **working directory**, use the **setwd** command. (You might need to do this each time you start a new **R** session, depending on your system.)

```
> setwd("c:/linda/summer2019/RSpace") # creates working directory,
> # be sure to change the path to your location
```

If you don't remember your working directory,

```
> getwd()    # Displays the location of the current working directory
```

If you would like to see the contents of your working directory,

```
> dir()    # Displays the contents of your current working directory
```

Open R now and set up your working directory! The location should be your flash drive or another location you access on a regular basis.

3.1 Basic Data Types

There are several data types in **R**.

Definition 3.1. A **vector** is a sequence of data elements of the same basic type. Members in a vector are officially called components.

Note: Vectors of length one include

♠ **numeric:** real numbers

♠ **integer:** integers; To create an integer variable in **R**, invoke the **as.integer()** function.

♠ **complex:** complex numbers of the form $a + bi$

♠ **logical:** comparison operations

♠ **character:** represents string values in R. We convert objects into character values with the **as.character()** function.

NOTE: Data may be assigned to a variable using either the = sign or using $<-$.

Let's use an *R script* (the text document that is SEPARATE from the *R Console*) to enter our commands. Open **R**. From the main menu select **File** → **New script**. The **R Editor** window will open. (It will say *untitled* until you save the script.)

★ Move the cursor to the **R Editor** window and type in the following:

```
# Statistics 147 Lab #1 Summer 2019
# Your name goes here
#
```

From the main menu in **R**, select **File** → **Save As**. The *Save Script As* window will open. Select a destination where you would like to save your script. In the box next to *Filename*, type **lab1_R_147_su19** and click **Save**.

Example 3.1. Complete the following by typing in the given **R** code in the **R Editor** window.

```
x = 21.56  # Assigns a value to the variable x
x          # Prints the value of x
```

Save your **R** script. (Select **File** → **Save**). To execute your script, from the main menu, select **Edit** → **Run All**. Complete the following from the **R** console window.

```
> # Statistics 147 Lab #1 Summer 2019
> # Your name goes here
> #
> x = 21.56  # Assigns a value to the variable x
> x          # Prints the value of x

[1] ______________________
```

NOTE: Once can also just execute a block of code.

Type the following block of **R** code in the **R Editor** window.

```
class(x)   # Prints the class of x
is.integer(x)  # is x an integer?
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
> class(x)   # Prints the class of x

[1] ______________________
 > is.integer(x)  # is x an integer?

[1] ________________
```

One can coerce a numeric value into an integer with the *as.integer* function. —em as.integer does not round to the nearest integer. It truncates the decimal portion.

Type the following block of **R** code in the **R Editor** window.

```
x1 <- as.integer(x)
x1   # Print the value of x1
class(x1)  # Print the class of x1
```

Save your **R** script. (Select **File** → **Save**.) Highlight the lines of code you just typed. To execute those lines of code, from the main menu, select **Edit** → **Run line or selection**. Complete the following from the **R** console window.

```
> x1 <- as.integer(x)
> x1  # Print the value of x1

[1] ________________
> class(x1)  # Print the class of x1

[1] ________________
```

NOTE: Small data sets can be entered using the **c()** function, where **c** is short for **concatenate**. This produces a data vector.

Example 3.2. Suppose you have the following data:

```
               4   6   2   7   9   5
```

1. Use the **c()** function in **R** to read in the data.

   Type the following block of **R** code in the **R Editor** window.

   ```
   my_data = c(4,6,2,7,9,5)  # read in the data
   my_data                   # Print the data
   class(my_data)            # Print the class name of the data
   ```

   Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

   ▲ highlight the new text you just typed.

   ▲ From the main menu, select **Edit** → **Run line or selection**.

   Complete the following from the **R Console** window.

   ```
   > my_data = c(4,6,2,7,9,5)  # read in the data
   > my_data                   # Print the data

   [1] __________________
   > class(my_data)            # Print the class name of the data

   [1]___________________
   ```

2. Use the sort function to create a new variable representing your sorted data

   Type the following block of **R** code in the **R Editor** window.

   ```
   my_data_sorted = sort(my_data)  # Create new variable containing the sorted data
   my_data_sorted                  # Print the sorted data
   ```

   Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

   ▲ highlight the new text you just typed.

   ▲ From the main menu, select **Edit** → **Run line or selection**.

   Complete the following from the **R Console** window.

   ```
   > my_data_sorted = sort(my_data)  # Create new variable containing the sorted data
   > my_data_sorted                  # Print the sorted data

   [1]___________________
   ```

NOTE: To change the i^{th} value in the list, address it as dataname[i].

Example 3.3. Suppose the 2nd data value should be a 3, not a 6.

```
> my_data[2] = 3
> my_data
[1] 4 3 2 7 9 5
```

Example 3.4. Suppose you have the following character data:

```
           brown  red  yellow  blue  orange  green
```

NOTE: To enter character data in **R**, enclose the values in double quotes.

Type the following block of **R** code in the **R Editor** window.

```
my_colors = c("brown", "red", "yellow", "blue", "orange", "green")   # enter the data
my_colors          # Print the data
class(my_colors)   # Print the class name of the data
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
> my_colors = c("brown", "red", "yellow", "blue", "orange", "green")   # enter the data
> my_colors          # Print the data

[1] ______________________________________________________________
> class(my_colors)   # Print the class name of the data

[1] ______________________________________________________________
```

NOTE: Vectors can be combined using the **c()** function.

Example 3.5. Consider Example 3.2 and Example 3.4. Use the **c()** function in **R** to combine the two sets of data.

Type the following block of **R** code in the **R Editor** window.

```
combined1 = c(my_data,my_colors)  # Combine the two data sets
combined1                         # Print the new data set
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
> combined1 = c(my_data,my_colors)  # Combine the two data sets
> combined1                         # Print the new data set

 [1] __________________________________________________________________
```

3.2 Arithmetic Operations

Arithmetic operations of vectors are performed memberwise (member-by-member).

Example 3.6. Consider the following data:

```
x = 1,2,3,4,5
y = 2,4,6,8,10
```

1. Enter the data in **R**.

   Type the following block of **R** code in the **R Editor** window.

   ```
   x = c( 1,2,3,4,5)   # enter values of x
   x                   # print x
   y = c(2,4,6,8,10)   # enter values of y
   y                   # print values of y
   ```

   Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

   ▲ highlight the new text you just typed.

   ▲ From the main menu, select **Edit** → **Run line or selection**.

   Complete the following from the **R Console** window.

   ```
   > x = c( 1,2,3,4,5)   # enter values of x
   > x                   # print x

   [1] ________________________
   > y = c(2,4,6,8,10)   # enter values of y
   > y                   # print values of y

    [1] _________________________
   ```

2. Create a new variable x1 = 3.5x.

   Type the following block of **R** code in the **R Editor** window.

   ```
   x1 = 3.5*x    # create  a new variable x1 = 3.5x
   x1            # print x1
   ```

   Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

   ▲ highlight the new text you just typed.

   ▲ From the main menu, select **Edit** → **Run line or selection**.

   Complete the following from the **R Console** window.

   ```
   > x1 = 3.5*x    # create  a new variable x1 = 3.5x
   > x1            # print x1

   [1] ___________________________
   ```

3. Create a new variable sum1 = x + y.

   Type the following block of **R** code in the **R Editor** window.

   ```
   # Create new variable sum1 = x + y
   sum1 = x + y
   sum1    # print sum1
   ```

   Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

   ▲ highlight the new text you just typed.

   ▲ From the main menu, select **Edit** → **Run line or selection**.

   Complete the following from the **R Console** window.

   ```
   > # Create new variable sum1 = x + y
   > sum1 = x + y
   > sum1    # print sum1

    [1] _______________________________
   ```

4. Create a new variable diff1 = x - y.

   Type the following block of **R** code in the **R Editor** window.

   ```
   # Create new variable diff1 = x - y
   diff1 = x - y
   diff1    # print diff1
   ```

   Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

   ▲ highlight the new text you just typed.

   ▲ From the main menu, select **Edit** → **Run line or selection**.

   Complete the following from the **R Console** window.

   ```
   > # Create new variable diff1 = x - y
   > diff1 = x - y
   > diff1    # print diff1

   [1] _______________________________
   ```

5. Create a new variable prod1 = x*y.

   Type the following block of **R** code in the **R Editor** window.

   ```
   # Create new variable prod1 = x*y
   prod1 = x*y
   prod1  # print prod1
   ```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
> # Create new variable prod1 = x*y
> prod1 = x*y
> prod1  # print prod1

[1] _______________________________
```

6. Create a new variable div1 = x/y.

   Type the following block of **R** code in the **R Editor** window.

```
# Create new variable div1 = x/y
div1 = x/y
div1   # print div1
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
> # Create new variable div1 = x/y
> div1 = x/y
> div1   # print div1

 [1] ________________________________
```

7. Using the **sqrt** function in **R**, create a new variable $s = 2x + 3\sqrt{y}$.

   Type the following block of **R** code in the **R Editor** window.

```
# Create a new variable s = 2x+ 3sqrt(y)

s = __________2*x + 3*sqrt(y)__________  # fill in your code
s                                        # print the values of s
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
> # Create a new variable s = 2x+ 3sqrt(y)


> s = __________________________________  # fill in your code (same as above)
> s                                       # print the values of s

 [1] _______________________________________________________________________________
```

NOTE: You may find that your vector covers two or more lines in the console when it prints out, so instead of simply seeing the `[1]` on the first line, you may see another number in brackets on the second line, such as `[7]` or `[12]`. These numbers simply tell which observation in the printed vector starts on that line; so if you see a `[7]` on the second line, that means that the first observation printed on that line is the 7^{th} observation in the vector as a whole!

3.3 Reading Data from an External File

One may read in data from external files of various formats, including text files, csv files, Excel files, Minitab and SAS files.

The **read.table** function is very useful when reading in ASCII files that contain rectangular data. When the file contains the variable names in the first line of data the option **header** should be set to **TRUE**. The default delimiter is blank space. Other delimiters must be specified by using the **sep** option and setting it equal to the delimiter in quotes (i.e., sep= ";" for the semicolon delimited data file).

The general format is, for example:

```
read.table(file="filename",header=TRUE,sep="")
```

NOTE: If your file does not contain headers, then set header = FALSE. You will need to open and inspect the file in a text editor (Textpad or TextEdit) to confirm this.

NOTE: If your file has more than 1 line of headers and/or lines that should be skipped, one can add the **skip** option. You will need to open and inspect the file in a text editor (Textpad or TextEdit) to confirm this.

```
read.table(file="filename",header=TRUE/FALSE, skip = integer_value)
```

NOTE: If the file is not located in your **R** working directory, you must give the complete path to your file.

NOTE: Don't forget, you can always read more about what each parameter for a given function by typing `?function_name` in the console.

Example 3.7. Consider the data file , **cartest1.dat**. Note that the data file includes headings in Line 1. Feel free to open the file in a test etidor to see this.

To read in this data file, type the following block of **R** code in the **R Editor** window.

```
cartest = read.table(file = "c:/linda/summer2018/su18147/datafiles/cartest1.dat", header = TRUE)
cartest
# Use attach() function to make columns accessible individually
attach(cartest)
# Use the names() function to obtain the column names
names(cartest)
```

NOTE: You will have to change the path to the location of your data file. If you saved it in your **RSpace** directory, you do not need to include the path.

Read in your data file now!

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
# Read the file into a variable called "cartest"
cartest = read.table(file = "c:/linda/summer2018/su18147/datafiles/cartest1.dat", header = TRUE)

# Print the dataframe to check its contents
cartest
```

Now, highlight the variable `cartest` and submit it to the console by pressing *Ctrl + R* (Windows) or *Command + Enter* (Mac). You should see the following in the **R Console**:

```
> cartest
  Car BrandA BrandB
1   1    125    133
2   2     64     65
3   3     94    103
4   4     38     37
5   5     90    102
6   6    106    115
```

This is an **R** `data.frame`. Each column represents a different variable (Car, BrandA BrandB) as a vector. We can access each of these vectors separately in one of two ways: Either type the name of the data frame followed by a $ sign

```
cartest$Car
cartest$BrandA
cartest$BrandB
```

... or we can use the attach() function to make each column of our data frame into its own vector variable.

Try this: Enter the three column names into the **R Console**:

```
> Car
Error: object 'Car' not found
> BrandA
Error: object 'BrandA' not found
> BrandB
Error: object 'BrandB' not found
```

R can't find them because we never defined these variables! Now, let's use the attach() function and try again.

```
> # Use attach() function to make columns accessible individually
> attach(cartest)

> # Use the names() function to obtain the column names
> names(cartest)
[1] "Car"    "BrandA" "BrandB"

> Car
[1] 1 2 3 4 5 6
> BrandA
[1] 125  64  94  38  90 106
> BrandB
[1] 133  65 103  37 102 115
```

Example 3.8. Refer to Example 3.7. Using arithmetic operations in **R**, calculate the sum of BrandA and BrandB.

Type the following block of **R** code in the **R Editor** window.

```
# Calculate sum of Brand A and Brand B

sum2 = _______BrandA + BrandB_________ # fill in your code
sum2                                   # print the values of sum2
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

Complete the following from the **R Console** window.

```
># Calculate sum of Brand A and Brand B
> sum2 = ____________________________________ # fill in your code (same as above)
> sum2                                        # print the values of sum2
```

```
[1] __________________________________________
```

Complete the following table.

Car	BrandA	BrandB	SUM2
1	125	133	258
2	64	65	129
3	94	103	197
4	38	37	75
5	90	102	192
6	106	115	221

Example 3.9. Consider the csv file, **gas1.csv**. Note that the file includes headings in Line 1. To read in this data file, type the following in the **R Editor** window. (Be sure to change the path to the datafile to your path!)

```
# Use read.csv to read in csv file
gas_data = read.csv("c:/linda/summer2019/su19147/datafiles/gas1.csv", header = TRUE)
gas_data

# Use the names function to see the variable names
names(gas_data)

# Use attach function to separate columns of data
attach(gas_data)

# Print the data
premium
regular
```

Make sure your cursor is in the **R Editor** window. Save your **R** script (Select **File** → **Save**.) and then

▲ highlight the new text you just typed.

▲ From the main menu, select **Edit** → **Run line or selection**.

You should see the following in the **R Console** window.

```
> gas_data = read.csv("c:/linda/summer2019/su19147/datafiles/gas1.csv",header = TRUE)
> gas_data
   premium regular
1     35.4    29.7
```

```
2     34.5    29.6
3     31.6    32.1
4     32.4    35.4
5     34.8    34.0
6     31.7    34.8
7     35.4    34.6
8     35.3    34.8
9     36.6    32.6
10    36.0    32.2
> # Use the names function to see the variable names
> names(gas_data)
[1] "premium" "regular"
> # Use attach function to separate columns of data
> attach(gas_data)
> premium
 [1] 35.4 34.5 31.6 32.4 34.8 31.7 35.4 35.3 36.6 36.0
> regular
 [1] 29.7 29.6 32.1 35.4 34.0 34.8 34.6 34.8 32.6 32.2
```

When the output appears in the **R Console**, ask a lab partner or the TA to check your work and write their initials here. ________________

NOTE:

- ♣ Be sure to save your **R** script onto your computer, flash drive, or Google Drive. You can open it any time and run any of the code.
- ♣ If you want to save the contents of the **R Console** window, copy and paste it into Word/Notepad/LaTeX.

Feel free to quit **R**.

You have now successfully completed **Lab #1**. Please turn in your lab worksheet.

Thanks and have a nice day!

Luke & Ruihan