Lab 2: Means and beyond--more descriptive statistics, define functions, install and use R packages

Introduction and Objective:

In this lab, we are going to learn how to use R to calculate the geometric mean, harmonic mean, coefficient of variation, etc. Also, we are going to learn how to install R packages and use the functions in these R packages to perform some statistical analyses. Moreover, we will learn how to create and define our own functions. There is a group assignment at the end of the lab notes.

In this lab notes/manual, except in the introduction parts, all the R command lines are in **bold**; all the notes are following a # and in blue; all the R results directly follow the R codes/command lines and are in red.

A. Mean, Median, and more descriptive statistics

As we talked about in Lab 1, the first thing after we start a new R session is setting the working directory, which is like a working bench you are using in a wet lab. If you are using a PC, type the following R command line to set the working directory, assuming you have created a folder named "R" in the C drive. If you are using a Mac, remember in Lab 1 we talked about how to find the path of the folder that you'd like to use as the working directory, and how to set up the working directory.

setwd("C:/R")

In the lecture, we talked about some descriptive statistics. Certainly, we can calculate them by hand or use a calculator. However, R would absolutely make your job easier. R has some built-in functions (In fact, these functions are the ones from packages that come with the base R. Later, we will talk about what an R package is. To make it easier, we just call these the built-in functions that come with the base R.), such as mean(), sum(), and summary(), as we talked about in Lab 1. Besides the ones that have been mentioned in Lab 1, there are many others that we'd like to introduce in this lab.

In the lecture, we mentioned the median. R has a built-in function for finding the median of an assemblage of data.

Let's first define an object "a", which includes 5 numbers:

```
a<-c(5, 3, 1, 7, 9)
```

Then we can use the R built-in function median() to find the median of these 5 numbers:

```
median(a)
```

```
[1] 5
```

Here is another example, in which the object "b" has two "1":

```
b<-c(1, 3, 1, 7, 9)
```

```
median(b)
```

```
[1] 3
```

Just try to remind you as we talked about in the first lab: depends on where it locates, the lower case "c" can either be defined as an object or work as a function:

```
c<-c(1, 3, 1, 7, 9)
```

```
median(c)
```

```
[1] 3
```

If you defined an object, for example, the little c, and then you define it again, the object would be overwritten, and thus the data set the object represents would have been replaced by the new one:

```
c<-c(5, 2, 3, 1, 7, 9)
```

```
c
```

```
[1] 5 2 3 1 7 9
```

```
median(c)
```

```
[1] 4
```

You can see, the little c itself is no longer [1] 1 3 1 7 9, and the median has been changed from 3 to 4.

The object is always determined by the newest definition.

In addition, R can recognize different formats of the numbers being input, for instance:

d<-c(10^5,10^7,10^9)

sum(d)

[1] 1010100000

d<-c(1e+5,1e+7,1e+9)

sum(d)

[1] 1010100000

Both inputs were treated as the same, and R gives you the same result.

If you want to input 0.01, you can either type:

0.01 or **10^-2** or **1e-2**

You cannot type 1e(-2), otherwise, you will get an error message, because R will consider 1e which is followed by a "()" as a function. However, there is no such function as 1e().

Here is another built-in function we mentioned in the first lab:

sd(d)

[1] 574455977

B. Install and use R packages:

Although R has many built-in functions, there are no built-in functions for performing certain statistical analyses. Of course, as we will learn today, if this is the case, then we can define our own R functions. However, you need to realize that many R users have performed similar calculations many times before you even think about it. To save the time of people to type in similar codes, again and again, some R users and developers put the reproducible codes in some fundamental units. These fundamental units of R are called "packages". If we think

each R function is a “tool” that allows you to perform some tasks, then each R package is like a toolbox that contains a variety of tools. The base R comes with some packages. That is like, once you purchased (of course, R actually is free) a toolbox called “R”, you have some commonly used tools in this toolbox. However, let us say, if you want to renovate your bathroom by yourself and need to cut tiles, you may need a diamond blade, which is not in the toolbox as it is not a commonly used tool. Then you have two options. Either you go to Lowe’s or Homedepot or any other hardware stores to buy one, or you can make one by yourself (for some cases, like making a diamond blade by oneself would be very hard). As for R programming, we are not going to shop for just one “tool”. Usually, people upload the packages they developed and share them with all R users. And these packages have many handy functions, including one that may suit your task. So, if we want to perform some similar tasks, you can simply download the R packages developed by other people and use the functions in the packages directly instead of typing in the same or similar codes again and again.

Today we are going to download and install our first two R packages, which allow us to calculate the geometric mean, harmonic mean, and coefficient of variation.

There are two ways to install R packages. The first way is to type the R command, and the second way is using the pull-down manual.

We are going to use one way to install the first package, and then learn to use the other way to install the second package.

The first package allows us to calculate more descriptive statistics: geometric mean and harmonic mean.

1) Use the R command

We want to install a package called “psych” by typing the R command.

To install it, type the following command line in your R session:

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

Note:

1) The “packages” within the writing of the function “install.packages()” should be plural; the package is followed by an “s”.

2) The writing of the function “install.packages()” starts with a little “i”. It should not start with the capital letter “I”.

If you typed the command line in the base R session, you will see a window “CRAN mirror” pops up. If you type it in the Console or run it use the code editor in the RStudio, you will not see the “CRAN mirror” window. The “CRAN mirrors” are the mirror sites with the repositories of the packages.

If you typed the command line in the base R session and the “CRAN mirror” window popped up, then you need to scroll down and highlight to select “USA (PA 1)” (or any other mirror sites), and then click “OK”. After that, you will see the package of “psych” be installed on your computer. If you are using the RStudio to type the command line, after you hit “enter” or “return” on your keyboard, you will see the package be installed.

We are going to use the other method to install the second package, named “raster”, which allows us to calculate the coefficient of variation (CV).

2) Use the pull-down manual:

In the pull-down manual of base R, there is a tab named “Packages”. To install package “raster”, follow this selection: “Packages”→Install package(s)→select “raster”

Then the package “raster” is installed.

If you are using RStudio, there is a tab named “Packages” in the lower-right quadrant/panel. Click it. Then click “install”. In the pop-up window, type “raster” in the space below the sentence that reads “Packages (separate multiple with space or comma)”, then click “Install”.

Activate R packages:

In an R session, before you can use any specific R packages, you have to activate the packages first. We use the R function "library()" to activate a package. If you close and restart a new R session, you need to activate the package again if you want to use it in the new R session. But if you stay in the same R session, once you activate the R package, you do not have to activate the same package before you use it for the second time.

To activate package "psych", we type the following command:

library(psych)

Note:

1) The function "library()" starts with a lower case "l".

2) Within the parentheses, there are no quotation marks around the name of the package.

Now, we can calculate the geometric mean of the object "a" by using the function "geometric.mean()" provided by the package "psych":

geometric.mean(a)

[1] 3.936283

We can define a new object "f", and do the same thing:

f<-c(1, 3, 5, 7, 10)

geometric.mean(f)

[1] 4.020109

Also, we can use the function provided by this package to calculate the harmonic mean:

harmonic.mean(f)

[1] 2.815013

The second package we installed allows us to calculate the coefficient of variation (CV).

For instance, we'd like to calculate the CV of these 5 numbers: 32.88, 40.63, 34.43, 39.97, 36.9

We can type:

s<-c(32.88, 40.63, 34.43, 39.97, 36.9)

library(raster)

cv(s)

[1] 9.13274

Please note that the CV calculated by function "cv()" from package "raster" is the percentage of the actual CV. For instance, the 9.13274 is the percentage of the actual coefficient of variation (CV), which is 9.13274 %. When you report the CV, you should either write 0.0913274 or 9.13274 %.

C. Define your own functions

What if you'd like to repeatedly use an equation to calculate some numbers, however, there is no R built-in function for this task or you can't find an R package or know one that has a function suitable for this task. If this is the case, you may consider defining your own function.

Assume, we do not know the package "raster" has a function that can calculate CV. But we learned in the lecture, how CV can be calculated:

CV=sd/mean

Then, of course, we can do the step-by-step calculations in R as:

ssd<-sd(s)

sm<-mean(s)

scv<-ssd/sm

scv

[1] 0.09132741

However, if you do it in this way, and when you calculate the CV for a second data set, you need type similar codes again. To save your time, you can define a function called "CV" and resue it for future calcuations, by typing:

CV<-function(sd,mean){sd/mean}

scv<-CV(ssd,sm)

scv

[1] 0.09132741

Note here, the function "function()" is the function you need to define your own function (it is mouth-full of functions in this sentence ☺)

In the first command line (**CV<-function(sd,mean){sd/mean}**), in between the () following "function" are the two variables you have to input (provide to R) when you use your own function after you define it. In between the {} is the equation that contains and shows the relationships of the two variables. The first command line defined a function CV(), which is not an R built-in function, but a function made by yourself. Once you define it, you can use it to calculate CVs. For instance, here, you tell R, objects "ssd" and "sm" (you defined these objects already) are the two variables which are corresponding to the "sd" and "mean" when you were defining CV(). And then, R calculates the CV using objects "ssd" and "sm" and deposit the results in the object "scv" as you requested. Then to see the result calculated by your own function CV(), you type "scv" to see what is in it.

The above command lines used two vairalbes. If you want to input only one variable later, you can define your function as:

CV<-function(a){sd(a)/mean(a)}

scv<-CV(s)

scv

[1] 0.09132741

Now, by defining your own function, you can reduce the work and save time for not typing similar command lines. You can reuse your own function, again and again as using the built-in R functions, as long as you do not exit the current R session. If you closed the current R session and start a new session, you need to redefine your own function if you wish to use it.

Group Assignment 1

1. Use the following data set:

Sample1	Sample2	Sample3	Sample4	Sample5	Sample6	Sample7	Sample8	Sample9	Sample10	Sample11	Sample12
32.88	40.63	34.43	39.97	36.9	34.71	57.74	57.42	61.3	56.47	57.87	59.39

1) Create an object named "samples" that includes all the 12 numbers of the samples. Calculate the arithmetic mean of "samples". Show your R command lines and the result. (2pts)

2) Calculate the median of "samples". Show your R command lines and the result. (1pt)

3) Calculate the standard deviation of "samples". Show your R command lines and the result. (1pt)

4) Calculate the geometric mean of "samples". Show all your R command lines that are necessary and the result. (1pt)

5) Calculate the harmonic mean of "samples". Show all your R command lines that are necessary and the result. (1pt)

6) Calculate the coefficient of variation of "samples Show all your R command lines that are necessary and the result. (1pt)

7) If you have an equation: $A=SD/\sqrt{n}$, define your own function named as "B", and use it to calculate the A for the "samples". Show your R command lines and the result. (3pts)