Confidence Intervals Using R

Orville D. Hombrebueno

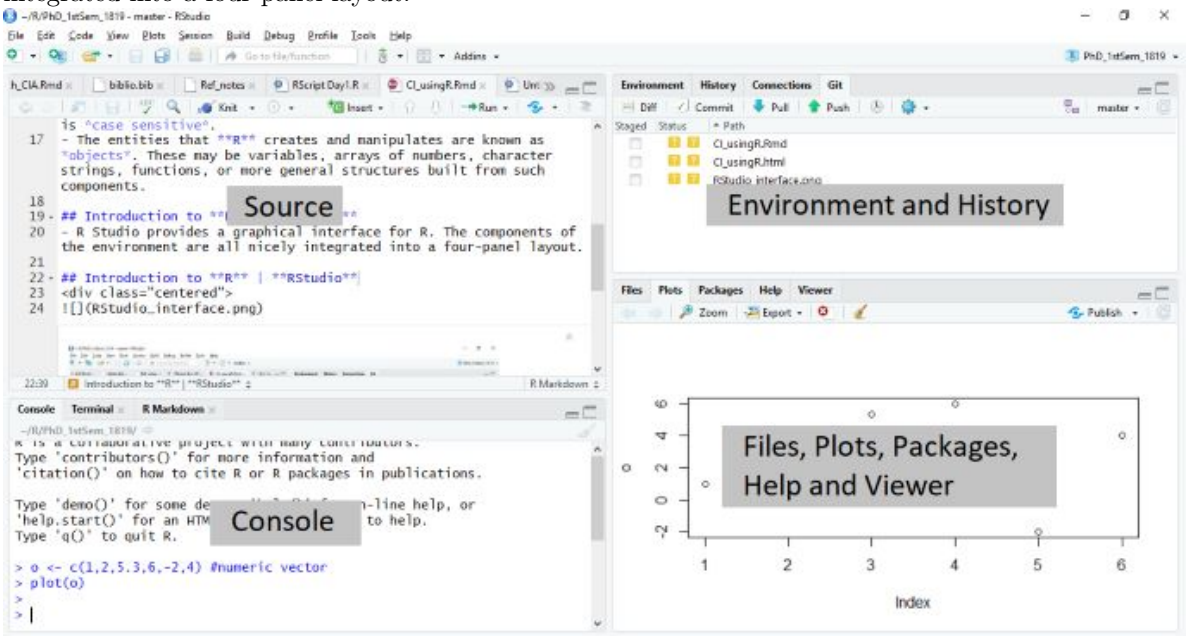
Introduction to R

The R Environment

- **R** is an integrated suite of software facilities for data manipulation, calculation and graphical display.
- It is an open-source software environment for statistical computing and graphics.
- It is an environment within which many classical and modern statistical techniques have been implemented.
- A few of these techniques are built into the base **R** environment, but many are supplied as packages.
- **R** is an *expression language* with a very simple syntax. It is *case sensitive*.
- The entities that **R** creates and manipulates are known as *objects*. These may be variables, arrays of numbers, character strings, functions, or more general structures built from such components.

RStudio

- R Studio provides a graphical interface for **R**. The components of the environment are all nicely integrated into a four-panel layout.



- The RStudio provides most of the desired features of **R** in an Integrated Development Environment (IDE).
- The **console** provides the command-line interface for interactive use of **R**. This is where users issue commands for **R** to evaluate.
- The **source** tab is a built-in text editor.
- The **Environment** tab is an interactive list of **R** objects.
- The **Files** tab displays the files and subdirectories of a given directory.
- Display of graphics is rendered in the **Plots** tab.
- RStudio keeps a stack of past commands and allows one to scroll through them easily. This can be done using the up or down keys. In addition, the **History** tab allows one to scroll through past commands.
- The **Packages** tab allows users to effortlessly load, install, update, and/or delete packages in the library of packages.

- The **Help** tab is an output location for help commands and help search window.
- The **Viewer** tab is an advanced tab for local web content.

Data Type and Objects

- Scalars – atomic quantity and can hold only one value at a time.
  Examples: number, logical value, character(string)
- Vector – a sequence of data elements of the same basic type.
- Matrix – a collection of data elements in a rectangular layout.
- Data Frame – more general than a matrix, in that different columns can have different basic types.
- List – a generic vector containing other objects.

Basic Commands

- Before everything else, set your working directory. Setting the working directory is choosing a folder to save your work. You can set the working directory using the File tab or the function `setwd()`.
- Typing `?funcname` will cause **R** to open a new help file window with additional information about the function `funcname`
- We can assign values in **R** using `<-` operator.
  Example

```
> x <- 143
> x
```

```
[1] 143
```

```
> y <- 198
> y
```

```
[1] 198
```

```
> x + y
```

```
[1] 341
```

```
> z <- x + y
> z
```

```
[1] 341
```

- **R** uses *functions* to perform operations. To run a function called `funcname`, we type `funcname(input1, input2)`
  Example
  We use the function `c()` to create a vector of numbers

```
> x <- c(1, 4, 3, 4, 4)
> x
```

```
[1] 1 4 3 4 4
```

- Note that the prompt, `>`, is not part of the command; rather, it is printed by **R** to indicate that it is ready for another command to be entered.
- Hitting the *up arrow key* multiple times will display the previous commands, which can then be edited.
- a `+` sign replacing `>` indicates that your code is not complete and that **R** is asking you to complete your code.

Creating Functions in R

- We can write functions in R!

Example
Suppose we want to write a function for average. The formula for average is

$$average = \frac{\sum x}{n}$$

where x is a vector and n is a scalar containing the number of elements in x.

```
> average <- function(x){
+   sum(x)/length(x)
+ }
```

```
> y <- c(4, 3, 5, 1, 7)
```

```
> average(y)
```

```
[1] 4
```

or

```
> average <- function(x){
+   a <- sum(x)
+   b <- length(x)
+   c <- a/b
+   return(c)
+ }
```

```
> average(y)
```

```
[1] 4
```

or

```
> mean(y)
```

```
[1] 4
```

Confidence Intervals for the Mean (Large Samples)

Finding a Confidence Interval for a Population Mean

1. Find the sample statistics n and $\bar{x}$.

$$\bar{x} = \frac{\sum x}{n}$$

   In **R** we have the function `mean()`.

2. Specify σ, if known. Otherwise, if $n \geq 30$, find the sample standard deviation s and use it as an estimate for σ.

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

   In **R** we have the function `sd()`.

3. Find the critical value z_c that corresponds to the given level of confidence.

Example
For $c = 95\%$ or 0.95

```
> qnorm((1+0.95)/2)
```

```
[1] 1.959964
```

4. Find the margin of error E.

$$E = z_c \frac{\sigma}{\sqrt{n}}$$

5. Find the left and right endpoints and form the confidence interval.

LEP

$$\bar{x} - E$$

REP

$$\bar{x} + E$$

Interval

$$\bar{x} - E < \mu < \bar{x} + E$$

or

$$(\bar{x} - E,\ \bar{x} + E)$$

Example
Construct a 95% confidence interval for the population mean. Interpret your answer.

1. The stem-and-leaf plot shows the result of a random sample of airfare prices (in dollars) for a one-way ticket from Boston, MA to Chicago, IL.

```
The decimal point is 1 digit(s) to the right of the |

18 | 33
19 | 7
20 | 99
21 | 222333333366
22 | 2222366888889
23 | 88
```

2. A random sample of the closing stock prices for the Oracle Corporation for a recent year.

```
 [1] 18.41 18.32 22.86 14.47 16.91 18.65 20.86 19.06 16.83 20.71 20.74
[12] 18.42 17.72 20.66 22.05 20.85 15.54 21.04 21.42 21.43 15.56 21.74
[23] 22.34 21.97 18.01 22.13 22.83 21.81 19.11 21.96 24.34 19.79 22.16
[34] 17.97
```