Introduction to R

1. What is R? Why use R?
2. Installing R and RStudio
3. Working with R

Examples:

1. R Example #1
2. Regression Example
3. ANOVA Example
4. *R Help Hints*

1. What is R? Why use R?

- R is a command line programming language for statistical computing.
- R is free!
- It makes great graphics.
- It provides a wide range of packages and functions for statistical analysis.
- Command line programming helps achieve “reproducible research”.
- It is open source software.

- R comes with a standard set of packages, but many more (thousands!) are available for download.
- The best thing and worst thing about R is the large number of packages that are available".
- This is a "good" thing because it greatly extends R's functionality.
- This is a "bad" thing because the packages are written by many different people and the syntax is not consistent.

2. Installing R and RStudio

- To install R go to: www.r-project.org
- Click on “download R”.
- Select a CRAN mirror.
- Download R (binaries for base distribution) for your operating system.
- Be aware that R is updated frequently, with a few major releases each year.

- RStudio is an “integrated development environment” or IDE.
- It is a program that makes it more convenient to use R.
- Go to the website www.Rstudio.com and download RStudio. We will use the free Desktop Open Source License.

3. Working with R

- An object can be created with the “assign" operators “<-” or “=“.
- R is case sensitive!
- # is the comment character in R. You can use it to add notes to your code. Anything following # on a single line will be ignored by R.

Objects vs Functions

- R objects store information.
  - Every object in R has a type (class).
- R functions "do things".
  - The syntax of functions is the function name followed by open parentheses, a comma separated list of arguments (possibly named) and then a closing parentheses:
    `functionname(name1= arg1, name2 = arg2, …)`

Hint: In RStudio, use the tab key to see function arguments!

Object Types

- In most cases imported data will be a data.frame.
- When we created an object using `c()`, the resulting object was a vector.
- Other object types include arrays, matrices and lists (to name a few).

Referencing a column from a data.frame

1. $ approach

   Ex: `mean(chickwts$weight)`

2. with() function

   Ex: `with(mean(weight), data=chickwts)`

3. Some (but not all) functions have a data = option

   Ex: `aggregate()` and `boxplot()`

4. Use attach() to identify an "active" dataset

This means the dataset does not need to be specified within a function but can be dangerous. Use with caution!

Finding Help

- Finding help with R:
  - To find help on a specific topic (for example *mean*) just type `?mean` or go to Help and search for “mean”.
  - Rseek.org and Quick-R are helpful sites.
- Always look at the data! The `View()` and `str()` functions are very helpful.

R Scripts

- An R script is a convenient way to save your code. With the code, you can recreate all your results later.
- To open a script (in Rstudio), choose File > New File > R script.
- I like to work in a script (making changes as I go) and **then save the script at the end of the R session**.
- An entire script can be run using the `source()` function.
- Typically, I just open the script and run as needed.

Saving Work

- Save the script (preferred)
- Save R objects
- Save the entire workspace (not covered here).
- Rmarkdown or knitr (not covered here).

Saving R objects (Optional)

- `save()` will save a particular object.
- Saved objects can be “restored” (in a new R session) using `load()`.
- I sometimes use this approach after “cleaning” a large dataset or for saving a results in a data.frame.

Working Directory (Optional)

- The “working directory” is the default directory for R (used for saving).
- Use `getwd()` to find the location and `setwd()` to set the location.
- Specifying the working directory is not necessary but can be helpful if you are working on several different projects.

R Packages

- R packages contain additional functions (and data).
- Before using a package for the first time, you may need to install it. This step only needs to be done once! If a package is already installed, you will still need to load it during the R session when you want to use it.
- From RStudio, use the "Packages" tab to install and load packages. Packages can also be loaded using the `library()` or `require()` functions.

Importing Data

- To import CSV files (comma separated values), we will use `read.csv()`.
- The file.choose() function allows files to be chosen interactively (allowing the user to "browse" to find a file.)
- Ex: `rice <- read.csv(file.choose())`
- Another approach is to specify the file path location.
- Ex: `rice <- read.csv("C:/...STAT512/Rice.csv")`

More about Importing Data

- Full list of options (skipping rows, comment characters, etc) are given in the help page: `?read.csv`.
- `read.table()` is a (related) alternative to `read.csv()`. Note: The defaults are different for `read.csv()` and `read.table()`.
- Excel files can be imported (and exported) to R. This can be done using the XLConnect package.