R Basics "Boot Camp" Topics by Ann Hess

1 Installing R and RStudio

- R is a command line programming language for statistical computing.
- R is free!
- It provides a wide range of packages and functions for statistical analysis.
- Command line programming helps achieve "reproducible research".
- It is open source software.
- It makes great graphics.
- Every time you open R, you will see the following warning: "R is free software and comes with ABSOLUTELY NO WARRANTY."

- To install R, go to www.r-project.org, and click on "download R", choose any CRAN mirror, and download R for your platform (binaries for base distribution).
- To install RStudio, go to www.rstudio.com. Select Products > Open Source > RStudio. We will use the free RStudio Desktop Open Source Edition.

- We will work through RStudio for this course. RStudio is a program that makes it more convenient to work in R.
- R is updated frequently, with a few major releases each year.
- For this course, the current version of R is strongly preferred. But if you already have R installed and your version is LESS than 1 year old, this should be OK. The R version is given at the top of the Console window.
- For the majority of people, the installation process will go smoothly. But if you run into problems, please reach out to the instructor or TA for help.

2 Working in Markdown

- R markdown is a convenient way to create a document including both plain text, R code and output. We will use RMarkdown for examples, HW and exams.

- From RStudio, go to File > New File > R Markdown. Choose Document (default), provide a title and author and choose "Word". This provides a small mock document.
- R code is entered in "chunks" (which appear grey). Any text that appears outside of a code chunk is plain text. (Note: There is an option to include inline R code, but that is beyond today's discussion.)
- As you are working, run code within chunks and check output. Mistakes and errors will happen, fix them as you go.
- There are several ways to run the code: (1) Use ctrl+Enter (Windows) or command+Enter (Mac) to run one or more lines of code. (2) Use the green play button to execute an entire code chunk. (3) Copy/paste code into Console (not preferred).
- Use insert to insert new R code chunks. As you gain experience with markdown, you will find that the use of multiple code chunks can go a long way to creating a "pretty" document. Code chunks are free!
- Save .RMD file as you go and certainly save before exiting R.
- When all code chunks run without errors and you are happy with the result, knit to a document by selecting "Knit". The file will not knit if there are errors within the R code chunks.
- **We recommend knitting to Word**, as this generally causes fewer problems as compared to knitting to pdf.
- Optional: If you want to try knitting to pdf, first install and load the R package "tinytex".

- Spend a moment exploring other helpful tabs/windows including Environment (current objects), History (executed code), Packages and Help.
- We do NOT recommend saving "workspace image" on exit. The idea is if we save our code (via R Markdown), we can easily recreate the results (objects and workspace).

3 Basic Coding (R Example #1)

- We can create objects using assignment operators (= or <-).
- R objects store information.
- R functions do things.

- ALWAYS look at the data!
- From RStudio Environment tab, click on the blue button to the left of the data name. This will provide the number of observations (rows) and variables (columns). It will also provide the exact column names and variable types. This is equivalent using the code `str(InData)`.
- From RStudio Environment tab, click on the data name. This will open a new window showing the data. This is equivalent using the code `View(InData)`.

- Use $ to reference a specific column within a data.frame.
- R requires code to exactly match the column names in the data.frame.
- R is case sensitive.

- Finding help in R is a critical skill.
- When you know the function name, you can search using the Help window. Note that code examples are included for most functions, but you need to scroll down.

4 Importing Data (R Example #2)

- Importing data is a routine first step in the analysis. But unfortunately, can still be frustrating
- Remember, ALWAYS look at the data after importing.

- For this class we will use CSV files (comma separated values) and use read.csv() for importing.
- Import Option #1: Move data file into same folder with .RMD document. It should be the exact same folder, not a sub-folder. If you do this, you should only need to specify the file name.
  `InData <- read.csv("dataname.csv")`
- Import Option #2: Specify full filepath location.
  `InData <- read.csv("filepath/dataname.csv")`
  Replace "filepath" and "dataname" with file specific information.
  - Finding file path for Windows: Navigate to and select the file of interest. Hit shift and right click, choose "Copy as path…". Copy this into the read.csv code above. Note you will need replace \ (backslash) with / (forward slash).
  - Finding file path for Mac: Open finder window and browse to your .csv file. Right click on the file name which reveals a dropdown list for file options. Now hold the "option" button (next to the command button). Notice among the options is to copy the file as Pathname. Paste this into quotes for read.csv. Note: Mac users can shorten pathnames using "~". e.g.
    "~/Dropbox/STAT511/Assignments/Assign1/ex3-30.TXT"
- See R Help document for "point and click" import option #3 and details about textbook datasets.

- Many (but not all) R functions have a data = option. This allows us to avoid $. When available, use the data = option!
  Ex: `boxplot(Y ~ X, data = InData)`

5 Packages and More Markdown (R Example #2 continued)

- R comes with a standard set of packages, but many more (thousands!) are available for download.
- I have heard it said that "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.

- The first time you use a package you need to install it. From the RStudio Packages tab, choose Install. Start to type the package name (case sensitive). Be sure the "Install dependencies" box is checked!
- Every time you want to use a package you need to load it. Within an R code chunk use library() or require() to load packages.
- **Best practice:** Include code to load all required packages in an **early** code chunk.

- All R code chunks need to run without errors before the document will knit.
- Code to import required data needs to be included within the R markdown document (in a code chunk). Order matters: the import code needs to appear before the data is referenced.
- Code to load required packages needs to be included within the R markdown document (in a code chunk). Order matters: the package needs to be loaded before it is used.
- **Knit frequently to avoid last minute problems.**
- To figure out where problems lie, it can be helpful to close R Studio, reopen and rerun the code chunks (in order).
- To learn more about markdown options, see the R Markdown "cheat sheet" and reference guide.

6 Formatting (R Example #3)

- For HW and exams, answers should be organized and labeled such that they can be easily read and understood.

- An easy way to label output is to use plain text (plus multiple code chunks).
- Note that a major benefit of markdown is that it is easy to include both plain text and R code and output. Anything that is not in a code chunk is plain text. There is no need to use print() or paste()!

- A more advanced way to label output is to use inline R code. For example: Mean is `` `r mean(chickwts$weight)` ``.
  This approach is **optional** for this course.
- For publication quality documents (**not** required for this course), most numeric results will be in tables. To accomplish this, tidyverse and kable packages can be handy.