INTRODUCING R IN AN APPLIED STATISTICS COURSE FOR NONMAJORS

Lindsey Bell and Keshav Jagannathan
Coastal Carolina University: Department of Mathematics and Statistics
P.O. Box 261954 Conway, SC 29528
lbell2@coastal.edu and kjaganna@coastal.edu

1 Introduction

Technology makes the collection of data easier than it has ever been before. Anyone can create an online survey, mobile phone apps measure and monitor almost anything imaginable, and social media provides a wealth of information. Due to the enormous volume and variety of data available, the ability to analyze and interpret this data is a coveted skill. In some cases, the enrollment in upper-level applied statistics courses has been increasing as more students and teachers recognize the importance of statistical literacy. Many of these students are not mathematics or statistics majors, but seek to learn statistics as it can be applied to their field of study. Part of this application is the use of statistical software. However, learning statistical software for the first time alongside new statistical concepts can be overwhelming for many students. In this paper we provide several tools for introducing statistical software in such a classroom setting.

To introduce and describe these concepts we use the statistical software, R [6]. R is the open source sister software of S-Plus [5]. It is an extremely powerful and flexible software with tools available for graphing, summarizing data, and performing statistical inference. Our medium-sized institution uses R across all applied statistics courses to provide continuity for the students. Additionally, it is a popular tool in a variety of disciplines. For example, computational biologists use it in conjunction with the BioConductor Project(www.bioconductor.org) [1], many social scientists encourage its use [3], and teachers can use it to supplement mathematical statistics courses [4]. Even though we focus on R, many of the concepts introduced in this paper can be applied to various software packages.

2 Methods

2.1 Tools for Inside the Classroom

In a culture that heavily relies on technology, we expect our students to be fairly proficient. However, experience has shown that they often need more guidance than expected. Being "app" savvy does not always translate into general computer literacy. First, we provide suggestions for introducing R within the classroom environment.

We discuss downloading the software with students and introducing them to more user friendly interfaces such as R Commander. A list of nine must cover topics is provided. Finally, a beginner's tutorial for practicing these concepts is provided.

2.1.1 Getting Started

Many students will have no difficulties downloading and installing R software on their own. However, experience has shown that still other users do indeed have difficulties. If possible, download the software with students during class. If this is not possible, consider including the following information in your class introduction to R.

The software can be downloaded from `www.r-project.org`. The website has limited aesthetic appeal and some students wonder if they are in the right place to begin with. Once confident that they are in the right place, the students often wonder about selecting a CRAN Mirror (Comprehensive R Archive Network). Explain that they will need to select a location for downloading the software. In fact, thousands of packages are stored on these CRAN Mirrors and students will need to select one each time a new package is obtained. But, what is a package? Students should know that the initial download has a base set of statistical functions. More specialized functions are grouped in packages stored on the CRAN Mirrors. Applied statistics courses usually cover some topics that require functions beyond the base package.

Once the software has been downloaded, the user interface may be the next hurdle encountered by students. The user interface for R is command line based. Most students have no experience using such an interface and need to be walked through the idea and provided with functions that will be used. Furthermore, there are many opportunities for making syntactic errors in this environment. Another alternative is to incorporate a more user friendly environment such as R Commander(`www.rcommander.com`) [2]. R Commander is a graphical user interface that runs on R but provides familiar drop down menus for common statistical functions. It is installed and loaded in the same manner as other packages mentioned above. In addition, there are several integrated development environments (IDEs) that streamline the use of R and even work in conjunction with R Commander. One such IDE is RStudio(`www.rstudio.com`).

2.1.2 Must Cover Topics

There are common basic topics encountered across applied statistics courses that incorporate software. In no particular order, we briefly identify nine such topics that should be addressed at the beginning of the course. Some of the comments here may seem trivial to teachers, but experience has shown that they should be addressed with the students. We introduce them in the context of R with no additional interface capabilities. Of course the topics would be appropriate for any software, but the details should be adjusted accordingly.

1. **Types of Data**
   In an applied statistics course we will usually use numeric (real numbers), character (text), or logical (true/false) data modes. Data in any of these modes are stored as one of the following data types.

   - *Vectors*
     These are sets of items of the same mode and have a length attribute (one dimension).

   - *Lists*
     Lists are similar to vectors, but may have items of different modes. For example, a list may contain both numerical and text values.

   - *Matrices*
     Matrices contain items of one mode arranged in rows and columns. Thus, there are two dimension attributes.

   - *Data Frames*
     These are similar to matrices except that they may contain more than one data mode. Most data sets used in applied statistics courses will be of this data type.

   - *Changing Data Modes and Types*
     Sometimes it is necessary to change the type and/or mode of data. There are several functions in R to accomplish such as task including `as.vector`, `as.matrix`, `as.numeric`, and `as.factor`. Consider an experiment run at three different temperatures ($10\,^{\circ}\mathrm{C}$, $20\,^{\circ}\mathrm{C}$, $30\,^{\circ}\mathrm{C}$). To read the temperatures as factor levels rather than numeric values, one might use the command `temp = as.factor(temp0)`.

2. **Reading in Data**
   There are too many methods for reading in data to mention them all here. Many of the methods depend on the type of data file to be read. Here we mention the two most commonly used methods in an applied statistics course.

   - *From the clipboard*
     Perhaps one of the easiest ways to read data into R is to copy it directly from an Excel file in conjunction with the `read.delim` command.

   - *From a file*
     The `read.delim` command is used again. However, the file path is provided rather than reading the data from the clipboard. Note that this command creates a dataset of the data frame type in R.

   - *Attaching the data*
     When data is read in as a data frame using the commands above, the user may then `attach()` the data. This allows for easier access of objects (variables) within the data frame. If you use this option, it is important to `detach` the data when finished. Other options include specifying the data frame of reference

within an R function via `dat=` or using the `dataframe$variable` notation to extract a variable.

- *Example*
  Suppose we wish to enter data about cereal from an Excel file. An excerpt of the data along with annotated R code and output are given next.

target	shelf	calories	carbs	fat	sugars
child	middle	120	12	2	12
child	middle	110	12	1	13
...	...	...	...	...	...
adult	top	110	17	0	3

```
# First, highlight and copy the data in Excel
> cereal = read.delim("clipboard",header=T)
#Take a look to ensure it read properly
> head(cereal)
  target  shelf calories carbs fat sugars
1  child middle      120    12   2     12
2  child middle      110    12   1     13
3  child middle      110    13   1     12
4  child middle      110    11   0     14
5  adult bottom      110    22   0      3
6  adult bottom      100    21   0      2

#Average by referencing calories within the data frame
> mean(cereal$calories)
[1] 113.0435

#Cannot find calories without referencing the data frame
> mean(calories)
Error in mean(calories) : object 'calories' not found

#Attach to directly access objects within the data frame
> attach(cereal)
> mean(calories)
[1] 113.0435
```

3. **Function and Subsetting Notation**
   In most programs there are subtle differences in syntax that many students do not pickup on right away. Additionally, some students have used other programming languages before (e.g. MATLAB) which may have opposite meanings for the same notation. Be sure to point out that R uses parentheses for calling a function and brackets for subsetting. (MATLAB reverses the use). A subtle, yet important example is the following.

```
> 2(4 + 5)
Error: attempt to apply non-function

> 2*(4+5)
[1] 18
```

From experience, this is one of the most common causes of errors by students beginning to use R. It would be a good idea to explain this example along with the error message. Parentheses denote calling a function, but 2 is a numeric value, not a function.

4. **Variable Naming**
   First, draw attention to the fact that R is case sensitive. Names should begin with letters or dots and may included numbers and underscores. It is possible to name things simply `Bob` or `Suzy`, however students should be encouraged to adopt consistent naming schemes that are appropriate to the problem. Names should be short and give an indication as to what the object contains. For example, a dataset with information about players in the NBA may be called `NBA.dat` rather than `Bob`. When starting a new problem about cereal sold in supermarkets, the name `NBA.dat` should not be carried over. Consider `cereal.dat` instead.

5. **Graphing**
   R boasts powerful graphing capabilities. Students should be introduced to several basic graphs at the outset to boost confidence and demonstrate basic concepts. Introductory graphs include histograms, boxplots, comparative boxplots, and scatterplots. Show students how to manage details such as labeling the axes and including a main title. Students also enjoy changing colors, line types, scales and orientation. Finally, demonstrate how to save the graphs for including in reports.

6. **Summary Statistics**
   Statistical analysis often begins with examining plots and summary statistics. This is a great way to teach students some basics of the software and integrate new knowledge with existing knowledge. Give the students opportunities for success early on by demonstrating several functions to obtain summary statistics. These may include, but are not limited to: `mean, sd, summary, table, and cor`. Include problems finding descriptive statistics for subsets of variables as well. Examples of this are provided in the tutorial.

7. **Commenting and Annotating**
   In R, the # symbol is used for commenting. Demonstrate how to save commands in a script and annotate them with comments. Emphasize that this is important for remembering what each line or section of code is used for. When providing code in class notes, include proper use of commenting as an example to the students and to increase their understanding of the code itself.

8. **Expectations for Reporting Results**
   Students should gain practice in scientific and professional writing before entering the workplace. Assignments are an excellent place to practice this and get feedback. However, the expectations need to be made clear to students. Consider providing an example assignment and creating a list of do's and don'ts for submitting assignments. You may wish to include some of the following in your list of expectations:

   - DO begin with your data entry steps. If you were required to create the data set, show it. If there are small errors in your results, this enables the teacher to find the data entry error.
   - DO show your R code and appropriate output.
   - DO NOT include error messages and unused output.

9. **Differences Between Mac and PC**
   Be prepared to encounter small differences between R implementation on a Mac and PC. A couple of important differences are given below.

   - The data entry step using the clipboard is quite different for a Mac and PC. Students should be aware of this from the start.
     PC command: `read.delim(''clipboard'', sep=',', header=T)`
     Equivalent Mac command: `read.delim(pipe(''pbpaste''), sep=',', header=T)`
   - The Mac version of R contains helpful guidance when calling functions. It recognizes the function a user is typing and shows the syntax for that function at the bottom of the screen. The PC installation cannot boast this useful feature.
   - The interface for installing packages is a little different between the two operating systems.

2.1.3 Tutorial

According to Eglen, "the best way that the students learn a programming language is by actually using the language on problem sets." [1] Many teachers and students alike would agree with this statement. After downloading R and demonstrating the basic set-up, it would be beneficial to provide the students with a tutorial to get familiar with the software. Students can mimic the commands given in the tutorial and explore functionality by changing inputs such as titles and colors in the graphics. Completing an assignment individually will help the students become more familiar with the software while providing them with detailed guidelines for doing so. We provide a sample beginners tutorial in the appendix. This tutorial highlights many of the must cover topics discussed previously.

2.2 Tools for Outside the Classroom

Next, resources for outside the classroom are provided. One valuable tool is to teach students how to interpret error messages provided by the statistical software. If they are unable to understand the message, it is encouraged that they search the error online. This simple step is often not thought of by students, but it is valuable in developing problem solving skills. Another tool to provide students with is a course specific R guide created by the teacher. This takes little effort on the teacher's part, but has proven useful to the students. Finally, a list of books and online resources is provided at the introductory level.

2.2.1 Teaching to Troubleshoot

All of us have encountered errors when using command line based software. These errors are inconveniences for seasoned programmers, but are usually quickly remedied. However, such errors can be especially frustrating for beginners as they may spend hours debugging only to find that a parenthesis was misplaced. At this point the student has lost sight of the original statistical problem and is often turned off from further work. On the other hand, some students give up much more quickly and do not even attempt to find the problem. They turn to the teacher for help immediately. To both ends, it is extremely important to teach students to efficiently be their own problem solvers. First encourage the students to actually read the error messages and decipher their meaning. Give them examples and explain the thought process every time you obtain an error when working with the program in class.

One common error that students encounter is `Error:object 'something' not found`. This error could occur for several reasons, though it usually results from misspelling that "something" or using different capitalization. It may also result when attempting to reference a variable that is in an unattached data set. Students may use the function `objects()` to view the list of objects in the current workspace. This may be enlightening as to why the object is not found.

Another common class of errors are syntactic in nature. These usually result from failure to close quotes, parentheses, or brackets, and omitting commas. The error message generated by R often reads `Error:unexpected symbol in ' '` and stops the error message near the problem area. Students may also encounter the + notation when an enclosure has not been matched on the right hand side. Explain how to escape and rerun the line with matching enclosures. Examples follow.

```
#Comma missing between calories and fat
> plot(calories fat, main ='Calories vs Fat')
Error: unexpected symbol in "plot(calories fat"

#Comma missing between fat and main
```

```
> plot(calories, fat main =‘Calories vs Fat’)
Error: unexpected symbol in "plot(calories, fat main"

#Failure to close quote. Need to escape line and amend.
> plot(calories, fat, main =‘Calories vs Fat)
+

#Correct syntax
> plot(calories, fat, main =‘Calories vs Fat’)
```

A third type of error encountered in R deals with attempts to apply non-functions or functions that cannot be found. An example that occurs using mathematical operations was given in section 2.1.2. Similar to the `object not found` errors, these errors may be caused by a misspelling of the function name. For example,

```
# Misspelling of ’plot’
> polt(calories, fat, main =‘Calories vs Fat’)
Error: could not find function "polt"
```

The three common error types mentioned here are certainly not exhaustive, but will cover nearly all errors encountered by students in an undergraduate applied statistics course. Taking the time to describe these types of errors will greatly reduce the time students take to debug their own code and make them less reliant on the teacher. Encourage students to search the error messages online as well. There are countless forums for R and it is likely that someone has posted questions about the error message before. Finally, encourage students to use the R Manual. Show them how to use the `?` or `help()` commands. Students need to know that much of the information given on the help pages may not apply to what they are doing and that the examples at the bottom are often helpful.

Patrick Burns’ “The R Inferno” is a humorous and comprehensive look at errors encountered in R. This would be an appropriate resource for a teacher, but not the introductory student. There are also tools to aid in debugging including the built-in `debug()`, `trace()`, and `browse()` functions or tools in IDE’s such as RStudio. However, these too may be inappropriate for the introductory student.

2.2.2 Course Specific Reference Packet

Students who are new to R often do not know what functions are available, the purpose of those functions, or the syntax. Ideally they would be introduced to necessary functions in the class notes and lecture. Even if this is the case, many students still have trouble organizing their resources. Based on student feedback and suggestions, we have found it helpful to create a course specific R guide to organize commands used in a particular course. It is suggested that these guides organize commands corresponding to the textbook chapter with which they are introduced. Formatting may be similar to the R Manual, but altered to a level appropriate for students in the course. Consider including the following with each function in the reference packet: command name, purpose of the command with respect to the course, list of inputs, description of outputs, and an example. While this is clearly an added commitment on the teacher's part, the resource has proven invaluable to students juggling both tasks of learning statistical software and statistical concepts.

2.2.3 Books and Online Resources

After teaching students to troubleshoot, you can provide them with extra resources for the task. One simple suggestion is to create a message board environment on your course website solely for troubleshooting. Because students encounter similar problems, those that have found the solution are able help others when they experience the same problem. This will create a collaborative backdrop for the classroom and increase debugging skills. In addition to students helping one another, a non exhaustive list of beginner resources is provided below.

- A Beginner's Guide to R, by Zurr, Ieno, and Meesters.
- R in Action: Data Analysis and Graphics with R, by Kabacoff.
- Similar to Kabacoff's book: `www.statmethods.net`
- Using R for Introductory Statistics, by Verzani. This book may be found here `cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf`
- Introductory Statistics with R, by Peter Dalgaard.

3 Conclusion

It is exciting to see the variety of students entering applied statistics classes. Across disciplines, they understand the value in learning statistics and statistical software. These students will increase statistical literacy, be able to add software expertise to their CV, and be more prepared for graduate school. It is our desire and goal for them to thrive in this endeavor. There are several things we can do in and out of the classroom to help them succeed. One class period may be used to introduce R by walking through the installation and must cover topics. The students are then prepared to complete the tutorial outside of class time. A portion of next class period can be used to discuss the tutorial and provide teachable moments regarding troubleshooting. These techniques are desirable because they take up little class time and teach students to be their own problem solvers. The tools and confidence we equip our students with during the first days of class will carry them far during the remainder of the semester and beyond.

References

[1] Stephen J. Eglen. A quick guide to teaching R programming to computational biology students. *PLOS Computational Biology*, August 2009.

[2] John Fox. The R Commander: A basic statistics graphical user interface to R. *Journal of Statistical Software*, 14(9):1–42, 2005.

[3] John Fox and Robert Andersen. Using the R statistical computing environment to teach social statistics courses. Department of Sociology, McMaster University, January 2005.

[4] Jane M. Horgan. Introducing undergraduates to probability using the open-source programming language R. International Association of Statistical Education (IASE), July 2010.

[5] Insightful Corporation, Seattle, WA, USA. *S-Plus 8 for Windows User's Guide*, May 2007.

[6] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

R Tutorial

This tutorial serves as a beginner's introduction to R. We walk through basic steps for reading in data, plotting, and obtaining descriptive statistics. BE VERY CAREFUL to type all commands presented here exactly as they appear. Small differences in syntax, spelling, or capitalization will cause errors! Please type your solutions and organize them in a clear manner. For each part, you should provide, (1) the R code, (2) the R-output, and (3) answers to questions posed for that part.

---

1. **Data Entry**

   (a) We will be using the cereal.xlsx data set posted on the course site. It contains information about breakfast cereals, including the target audience, the shelf position, calorie count, fat content (g), and sugar content (g). Highlight and copy the data from this file (`ctrl+c`), including the variable names. In R, type the following command

   For a PC: `cereal.dat = read.delim('clipboard', header=T)`

   For a Mac: `cereal.dat = read.delim(pipe('pbpaste'), header=T)`

   - You have defined a data set called `cereal.dat`. Notice the name is appropriate for the data topic and the `.dat` portion reminds you that this is a data set.
   - The `'clipboard'` (or `'pbpaste'`) portion tells R to read the information you stored on the clipboard by highlighting and copying the data in Excel.
   - The `header=T` portion tells R that the first row contains titles for each column.

   When you press enter, nothing should appear. The data set has simply been stored as `cereal.dat`. To see if the data was read in properly, type the following command and hit enter.

   ```
   head(cereal.dat)
   ```

   Does this display the entire data set? If not, what portion is displayed? What happens if you enter `head(cereal.dat,2)` into the command line? What do you think the command `tail(cereal.dat,3)` would do? Try it out and provide the output to confirm your answer.

(b) Now that we have successfully read the data, we would like to access individual variables in the data set.

- Suppose we want to access the variable called "sugars". Type `sugars` into the R Console and report the resulting error. REMEMBER THIS ERROR! This is a common frustration in R and the solutions are provided in the next two steps.
- This error occurred because R does not view each variable as individual objects. They are located within the data frame `cereal.dat`. To access variables in the data frame, use the `$` syntax. Try the following command and report your results.

  ```
  cereal.dat$sugars
  ```

- Using the `$` syntax for accessing variables in a data frame requires a lot of typing. An alternative is to `attach` the data frame so that each variable is its own object. Try the following.

  ```
  attach(cereal.dat)
  ```

  Nothing should appear when you press enter, but now try typing `sugars` and report what appears.

Takeaway:

- *Don't forget to attach your data set!*
- *Name data sets appropriately.*

Note: Data can also be entered by hand. See the class notes for a detailed example.

2. **Subsetting**
In some cases, you may wish to access select portions of a dataset. For example, perhaps you would like to only consider cereals whose target audience is children. This is called subsetting and the syntax requires square brackets to accomplish the task. Let's explore this with the following steps.

(a) If we would like to look at cereals targeted at children, we need to look at the variable 'target'. In the R Console type the following and provide the output. Explain what you see.

```
target==‘child’
```

Notice that we used a single equal to sign for defining/naming our data set. Now, it is important to use two equal to signs. This tells R that we are checking a condition (Is the target audience 'child'?) rather than assigning something a name. It is also important to include quotes around 'child' since this is a possible value of a variable rather than a stand-alone object.

(b) Suppose we are interested in the amount of sugar in children's cereal. We can do this by subsetting `sugars` with square brackets where the target audience is a child. Try the following and provide the output. Explain what you see.

```
sugars[target==‘child’]
```

(c) Suppose we are not only interested in the amount of sugar in children's cereals, but all the other variables as well. We can obtain a subset of the entire data frame. First, notice that `sugars` only has one dimension (length of 23 observations). However, the data frame `cereal.dat` has two dimensions, 23 observations and 6 variables. In the square bracket notation, we now need two pieces of information. For example, type the following and report your output for each.

```
cereal.dat[1,1]
cereal.dat[1,2]
cereal.dat[2,1]
cereal.dat[ ,1]
cereal.dat[1, ]
```

What does the first entry in the bracket notation provide? Row or column information? What does the second entry indicate? What happens when the first entry is blank? What happens when the second entry is blank?

Using this information, we can subset the data set for children's cereals with the following.

```
cereal.dat[target==‘child’, ]
```

Provide the output.

Takeaway:

- *Subsetting uses square brackets and double equal signs to check a condition.*
- *Vectors only have one index.*
- *Data frames have two indices. The first indicates the row and the second indicates the column.*

3. **Graphing**
R has many powerful graphing capabilities. The options for adding colors, titles, and labels are similar for each type of graph, so we will use boxplots as an example. Boxplots are used extensively in this course. They are useful for summarizing the five number summary and provide a display of the measures of center and spread in data.

(a) Suppose we wish to create a boxplot of the sugar content in our cereals. Try the following and provide the resulting plot.

```
boxplot(sugars)
```

Notice that the plot is plain and does not inlcude labels.

(b) To include labels on the plot (which is ALWAYS preferred) add the following options and provide the resulting graph.

```
boxplot(sugars, xlab = ‘Sugar’, ylab=‘Sugar Content (g)’,
        main = ‘Boxplot of Sugar Content’)
```

(c) You can also add color to your plots. The site `www.stat.columbia.edu/~tzheng/files/Rcolor.pdf` has an extensive list of color options in R. This is also posted on the course website for quick reference. Try adding the following color, or chose one of your liking, and provide the output.

```
boxplot(sugars, xlab = ‘Sugar’, ylab=‘Sugar Content (g)’,
     main = ‘Boxplot of Sugar Content’, col=‘cyan4’)
```

(d) Now suppose we would like to study the sugar content for different target audiences. We could do this by creating a boxplot for each target audience. Use the following code to create such a plot. Provide the plot and comment on what you see. Is the median sugar content similar in the two groups? Is the variability (spread) similar in the two groups?

```
boxplot(sugars~target, xlab = ‘Target Audience’, ylab=‘Sugar
     Content (g)’, main = ‘Boxplot of Sugar Content’,
             col=c(‘cyan4’, ‘goldenrod’))
```

4. **Summary Statistics**
R has many built-in functions to calculate basic summary information for variables. Here we look at a few of those functions for both categorical and quantitative variables.

(a) We have two categorical variables in our data, target audience and shelf position. Suppose we want to find out how many cereals are geared for children and how many are for adults. We can do this by creating a `table`. Try the following and provide the output. Who is the mode, children or adults?

```
table(target)
```

(b) To make this more interesting, we can examine where the cereals are placed in the store according to target audience. Try the following and report your results. Are the results surprising? Explain what you notice.

```
table(target,shelf)
```

(c) We can calculate summary statistics for quantitative variables such as the mean, standard deviation, and five number summary. Let's calculate these statistics for `sugars`. Try the following and provide your output.

```
mean(sugars)
sd(sugars)
summary(sugars)
```

(d) Finally, suppose we wish to summarize the sugar content in children's cereals only. Find the mean, sd, and five number summary using the commands above along with your knowledge of subsetting. Repeat for adult cereals. Report the results. Compare and contrast the two groups.

Cereal Data

target	shelf	calories	carbs	fat	sugars
child	middle	120	12	2	12
child	middle	110	12	1	13
child	middle	110	13	1	12
child	middle	110	11	0	14
adult	bottom	110	22	0	3
adult	bottom	100	21	0	2
adult	middle	120	15	1	9
child	middle	110	9	1	15
adult	bottom	100	15	1	6
adult	top	110	10	3	7
adult	top	110	17	0	3
child	bottom	110	11.5	1	10
adult	top	140	21	2	7
adult	top	100	20	0	3
adult	top	140	15	1	14
adult	bottom	100	17	1	3
adult	top	130	13.5	2	10
child	middle	100	12	2	6
adult	middle	100	16	1	3
adult	top	100	14	1	6
adult	top	150	16	3	11
adult	bottom	110	17	2	1
child	bottom	110	16	0	3