Statistics 13, Lab 1

Getting Started

This first lab session is nothing more than an introduction: We will help you navigate the Statistics Department's (all Mac) computing facility and we will get you started with R. Specifically, you will activate your account, "log in" to one of the lab computers, and work with two data sets that are related to our first lecture. There is an assignment associated with this first meeting, but it is not particularly challenging. Your TA will also provide you with some advice about where to get R and RStudio and how to install them, should you choose to do some of your lab work at home. (Which we sincerely hope you will!)

Keep in mind, however, that you are required to attend lab sessions. Your TA will be taking attendance. You should use this time each week to start your lab assignment. It gives your TA an opportunity to go over the assignment, explain what he expects from you and answer any questions you might have. If you finish your work during the hour, great. Some labs, however, may take a little longer and you can either return to our lab during one of the open computing periods or you can finish up at home.

The Mac

Once you have logged into a lab computer, you'll be greeted with the Mac desktop. Most of your work will involve one of the programs listed in the "dock" at the bottom of the screen. The Computing Staff have installed a number of programs in your dock. For those of you who aren't Mac users, Safari is a web browser and Appleworks is like MS Office (if you place your mouse over the icons in the dock, you'll see the names of the applications). You can start any of these applications by clicking on the icon in the dock.

Launching RStudio and loading data

Recall from class that we will be using RStudio as our main portal to the R computing environment. We start RStudio by clicking on its icon in the dock (your TA will point it out). An interface similar to the one on the next page should appear. There are three major regions to the RStudio interface. You interact with the R environment by typing commands into the "R Console," the large region at the left. The two regions on the right give you access to your "workspace," the history of commands you've typed in previously and any plots you've made.

There was a lot of R-specific jargon in that last paragraph. Let's explain things a little. The main Console window of the RStudio application is the large white area that initially displays some information about R itself (like the version you're working with and how to get help). After all this chatter you should see the "R prompt," a > sign. As it's name suggests, this prompt is really a request, a request for a command.

Initially, interacting with R is all about typing commands and interpreting the output. These commands and their syntax have evolved over decades (literally) and now provide what many users feel is a fairly natural
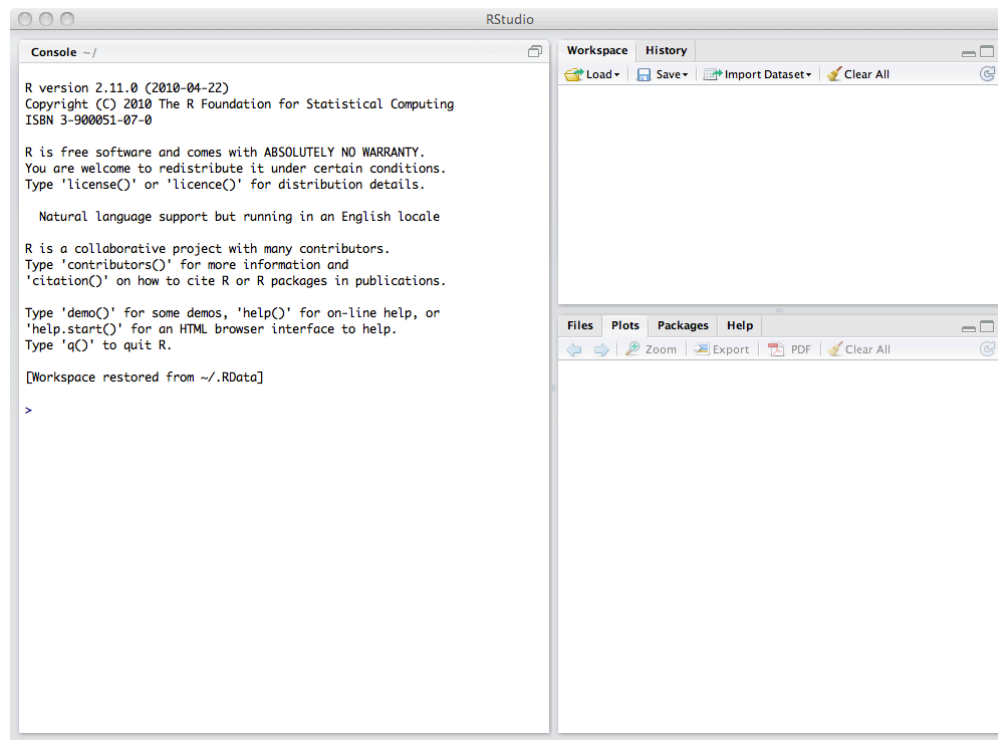
Figure 1: The RStudio window.

way to access data and organize, describe and invoke statistical computations. To get you started, enter the following command at the R prompt (your text should appear blue as you type; you can also just "yank and put" the text from this PDF file).

```
source("http://www.stat.ucla.edu/~cocteau/stat13/data/arbuthnot.R")
```

This command instructed R to venture out to my corner of the web and fetch some data, the Arbuthnot christening counts for boys and girls. You should see that the "workspace" area in the upper righthand corner of the RStudio window now lists a data set called `arbuthnot` that has 82 observations on 3 variables. We'll come back to this window in a moment but for now you can also see the data you have to work with by typing the following command in the main window on the left at the $>$ prompt.

```
ls()
```

This lists the objects in your "workspace." As you interact with R, you will create a series of objects. Sometimes you load them as we have done here, sometimes you create them yourself as the byproduct of a computation or some analysis you have performed. In response to your `ls()` command, R should have indicated that you have an item called `arbuthnot`. Note that because you are accessing data from the web, this command (and the entire assignment) will work in our computer lab, in the library, or in your dorm room, anywhere you have access to the Internet.

2

Some exploration

The name `arbuthnot` that should have appeared in your R Console refers to an object in your workspace. This object holds the christening data. As with any object, you can see its contents in the Console window by typing its name at the command prompt and hitting return. For example, try typing the following.

```
arbuthnot
```

What you should see are four columns of numbers, each row representing a different year: The first entry in each row is simply the row number (an index we can use to access the data from individual years if we want), the second is the year, and the third and fourth are the numbers of boys and girls christened that year, respectively. Use the scrollbar on the righthand side of the window to examine the complete data set. Does this match Arbuthnot's original tables given in lecture? What about the mysterious drop we saw at 1704, is that included in these data or has it been corrected?

Note that the row numbers in the first column are not part of Arbuthnot's data. R adds them as part of its printout to help you make visual comparisons. You can think of them as the index that you see on the lefthand side of a spreadsheet. In fact, the comparison to a spreadsheet will be helpful generally. R has stored Arbuthnot's data in a kind of spreadsheet or table (in R parlance, the object is called a `data frame`). You can see the size of this table by asking R for its dimensions:

```
dim(arbuthnot)
```

With luck, you will see that the table has 82 rows and 3 columns. You can see the names of these columns (or variables) by typing:

```
names(arbuthnot)
```

You should see that the table contains the columns year, boys, and girls.

At this point, you might notice that many of the commands in R look a lot like functions from mathematics; that is, invoking R commands means supplying a function some number of arguments. The `dim` and `names` commands, for example, each took a single argument, the name of a table. In the future we will use the terms "command" and "function" interchangeably.

One advantage of RStudio is that it comes with a built-in data viewer. Double click on the name "arbuthnot" in the upper righthand window that lists the objects in your workspace. You'll see an alternative display of the Arbuthnot counts. Notice also that this action has invoked the command `View(arbuthnot)` in the R Console. You can close the data viewer by clicking on the "x" in the upper lefthand corner.

Alright, so far we've accomplished little more than Arbuthnot did in his examination of his printed tables (but 300 years later). Let's start to examine the data a little more closely. We can access the data in each column of a table separately using a command like

```
arbuthnot$boys
```

This should only show the number of boys christened each year. How would you extract just the counts of girls christened? Notice that the way R has printed these data is different. When we looked at the complete table, we saw 82 rows, one on each line of the display. Now, instead of wasting space and printing out one value per line (each being the number of boys christened in a year), the 82 values are packed from left to right so as to fit as many as possible in the display window. R has added numbers in []'s at the left to indicate the row or index of the leftmost data element in this packed display. (You will likely see [1] to indicate that 5218 is the first entry, then [14] to indicate that 5460 is the fourteenth entry, and so on.)

So it's sensible to ask why this happened, and whether you think it's a good idea. We don't want to make too much of how R prints out data, but it's worth thinking about. When you ask to print a table (as you did when you typed arbuthnot), you really do want to be able to make comparisons along rows; say, to identify the years where girls outnumbered boys. When you ask for a single column of data (as you did when you typed `arbuthnot$boys`), you are probably going to examine these values as a group and so a more compact display is convenient. This is probably your first example of R trying to be "smart" as you work with data. It's a trivial example, no doubt, but we wanted to have you ask questions early.

In class, we considered simple plots (line graphs) of the data; and R has some fairly powerful tools for making graphics. We can get a simple plot of the number of girls christened per year with the command

```
plot(arbuthnot$year,arbuthnot$girls)
```

By default, R creates a scatter plot, meaning the points are indicated by (in this case) open circles. The plot itsef should appear under the "Plots" tab of the lower righthand window in RStudio. Notice that the command above again looks like a function, this time with two arguments specifying data for the x- and y-axes, respectively. If we wanted to connect the data points with lines, we could add an argument to the command:

```
plot(arbuthnot$year,arbuthnot$girls,type="l")
```

I admit that you would not know to use the argument type to change the type of plot, but thankfully R documents all of its functions extensively. To see what a command does you just type in a question mark followed by the name of the function you're interested in. Try the following.

```
?plot
```

Now, suppose we want to plot the total number of christenings? To compute this, we could use the fact that R is really just a big calculator. We can type in mathematical expressions like

```
5218+4683
```

to see the total number of christenings in 1629. To repeat this 82 times would be, well, painful. Here is where R starts to get interesting: R can perform this sum down entire columns at once.

```
arbuthnot$boys+arbuthnot$girls
```

What you should see are 82 numbers (in that packed display, because we aren't looking at a table here), each

one representing the sum we're after. Have a look at a few of them and verify that they are right. Therefore, we can make a plot of the total number of christenings per year with the command

```
plot(arbuthnot$year,arbuthnot$boys+arbuthnot$girls,type="l")
```

Similarly, we can compute the ratio of the number of boys to the number of girls christened in 1629 with

```
5218/4683
```

or we can act on the complete columns with the expression

```
arbuthnot$boys/arbuthnot$girls
```

The proportion of the babies that are boys in 1629 is

```
5218/(5218+4683)
```

or

```
arbuthnot$boys/(arbuthnot$boys+arbuthnot$girls)
```

Now, make a plot of the ratio or the proportion. What do you see? Hint: If you use the up and down arrow keys, you can scroll through your previous commands, your so-called command history. This will save you a lot of typing in the future.

Finally, in addition to simple mathematical operators like subtraction and division, you can ask R to make comparisons like greather-than, ">", less-than, "<", and equality, "==". For example, we can ask if boys outnumber girls with the expression

```
arbuthnot$boys > arbuthnot$girls
```

which should return 82 values TRUE or FALSE. While we won't dwell on it now, these are a different kind of data than we've been looking at so far. In the arbuthnot table, our values have been integers (the year, the number of boys and girls). Here, we've asked R to create logical data, data where the values are TRUE and FALSE. In general, data analysis will involve many different kinds of data types; and one reason for using R is that it is able to represent and compute with many of them.

This seems like a fair bit for your first hour, so let's stop here. To exit R you type the quit command.

```
q()
```

You will be prompted about saving your workspace. If you enter 'y', R will save the history of your commands and all the objects in your workspace so that when you startup RStudio again and type ls() you will see arbuthnot and you will have access to the commands you typed in your previous session. (Go ahead, start up RStudio again and make sure.)

Your Assignment

In the previous few pages, you recreated some of the displays and some of the preliminary analysis of Arbuthnot's christening data. Your assignment involves repeating these steps, but for present day birth records. As we mentioned in class, the system of vital statistics in the United States is fairly sophisticated now, and almost entirely electronic. Load up the present day data with the following command.

```
source("http://www.stat.ucla.edu/~cocteau/stat13/data/present.R")
```

The data are stored in an R object called `present`.

1. What years are included in this data set? What are the dimensions of the table and what are the variable or column names?

2. How do these numbers compare to Arbuthnot's? Similar scale? Larger? A lot larger?

3. Does Arbuthnot's observation about boys being born in greater proportion than girls hold up in the U.S.?

4. Make a plot of the ratio of the number of boys born per year to the number of girls in the U.S. What do you see?

5. In what year did we see the most total number of births in the U.S.?

That was a rather short introduction to R and RStudio, but we will provide you with more commands and a more complete sense of the language as the quarter progresses. Feel free to browse around the websites for R `www.r-project.org` and RStudio `www.rstudio.org` if you're interested in learning more.