# Lab 1: Introduction to R and RStudio

The goal of this lab is to introduce you to R and RStudio, which you'll be using throughout the course both to learn the statistical concepts discussed in the textbook and also to analyze real data and come to informed conclusions. To straighten out which is which: R is the name of the programming language itself and RStudio is a convenient interface.

As the labs progress, you are encouraged to explore beyond what the labs dictate; a willingness to experiment will make you a much better programmer. Before we get to that stage, however, you need to build some basic fluency in R. Today we begin with the fundamental building blocks of R and RStudio: the interface, reading in data, and basic commands.

The Interface

To launch RStudio, go to *http://beta.rstudio.org* and log in using your gmail address. When you first launch RStudio, you will be greeted by an interface that looks similar to this:
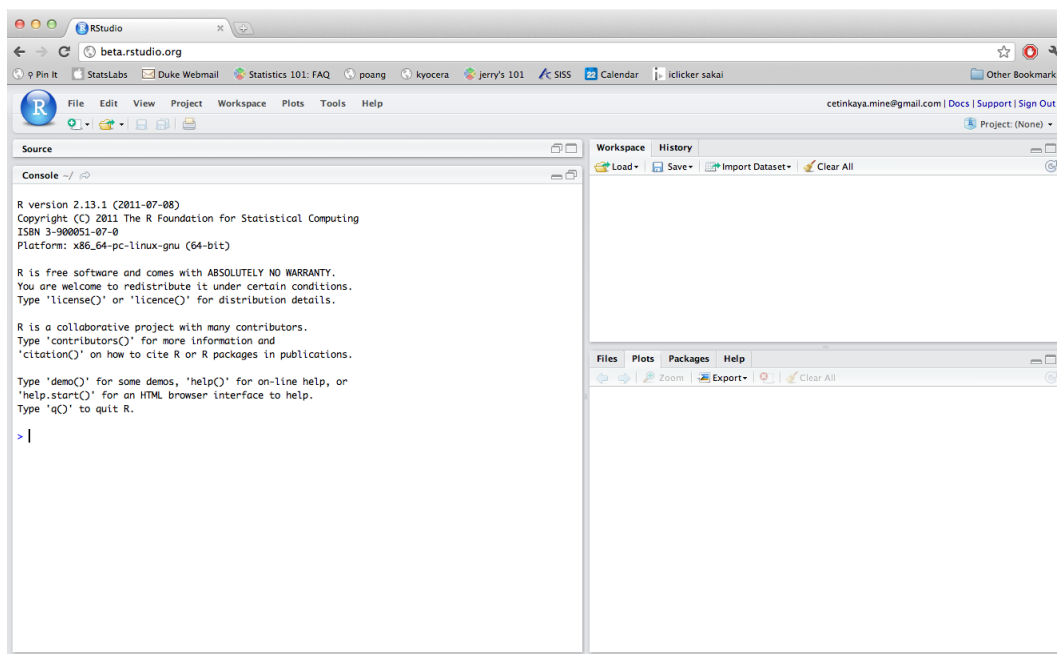
The frame in the upper right contains your *workspace* as well of a history of the commands that you've previously entered. Any plots that you generate will show up in the region in the lower right corner.

The frame on the left is where the action happens. It's called the *console*. Everytime you launch RStudio, it will have the same text at the top of the console telling you the version that you're running. Below that information is the *prompt*. As its 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 way to access data and organize, describe and invoke statistical computations.

To get you started, enter the following command at the R prompt. You can either type it in manually or copy and paste it from this document.

```
source("http://www.openintro.org/stat/data/arbuthnot.R")}
```

This command instructs R to access the OpenIntro website 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. 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. Note that because you are accessing data from the web, this command (and the entire assignment) will work in a computer lab, in the library, or in your dorm room, anywhere you have access to the Internet*.

The Data: Dr. Arbuthnot's Christening Records

The Arbuthnot data set refers to Dr. John Arbuthnot, an 18th century physician, writer, and mathematician. He was interested in the ratio of newborn boys to newborn girls, so he gathered the christening records for children born in London for every year from 1629 to 1710. We can take a look at the data by typing its name into the console.

```
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 right side of the window to examine the complete data set.

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 left side of a spreadsheet. In fact, the comparison to a spreadsheet will be generally helpful. R has stored Arbuthnot's data in a kind of spreadsheet or table called a *dataframe*.

You can see the size of this dataframe by asking R for its dimensions:

```
dim(arbuthnot)
```

---

*If you would like to use RStudio when you don't have Internet access, you can download the RStudio Desktop at *http://rstudio.org/download/desktop*. The desktop application looks and works exactly like the in browser application does.

You will see that the dataframe has 82 rows and 3 columns, just as it says next to the object in your workspace. You can see the names of these columns (or variables) by typing:

```
names(arbuthnot)
```

You should see that the dataframe 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 with some number of arguments. The `dim` and `names` commands, for example, each took a single argument, the name of a dataframe. 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. Click on the name `arbuthnot` in the upper right window that lists the objects in your workspace. You'll see an alternative display of the Arbuthnot counts. You can close the data viewer by clicking on the "x" in the upper lefthand corner.

Some Exploration

Let's start to examine the data a little more closely. We can access the data in each column of a dataframe 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 dataframe, 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 [ ] at the left to indicate the row or index of the leftmost data element in this packed display. To introduce some terms that will come up often, R is storing `arbuthnot$boys` as a *vector* with 82 *elements*.

R has some powerful tools for making graphics. We can get a simple plot of the number of girls christened per year with the command

```
plot(x = arbuthnot\$year, y = arbuthnot\$girls)
```

By default, R creates a scatter plot with each x,y pair indicated by an open circle. The plot itself should appear under the "Plots" tab of the lower right frame in RStudio. Notice that the command above again looks like a function, this time with two arguments separated by a comma. The first specifies data for the x-axis and the second for the y-axis. If we wanted to connect the data points with lines, we could add a third argument, the letter "l" for line.

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

You might wonder how you are supposed to know that it was possible to add that third argument. Thankfully, R documents all of its functions extensively. To read what a command does and learn the arguments that are available to you, just type in a question mark followed by the name of the function that you're interested in. Try the following.

```
?plot
```

Notice that the help file replaces the plot in the lower right frame. You can toggle between plots and help files using the tabs at the top of that frame.

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 painful. Here is where R starts to get interesting: R can perform this sum on entire vectors at once.

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

What you should see are 82 numbers (in that packed display, because we aren't looking at a dataframe here), each one representing the sum we're after. Take 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")
```

This time, note that we left out the names of the first two arguments. We can do this because the help file shows that the default for `plot` is for the first argument to be the x-variable and the second argument to be the y-variable.

Similarly to how we computed the proportion of boys, 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 vectors with the expression

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

The proportion of newborns that are boys in 1629 is

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

or

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

4

Note that with R as with your calculator, you need to be conscious of the order of operations. Here, we want to divide the number of boys by the total number of newborns, so we have to use parentheses. Without them, R will first do the division, then the addition, giving you something that is not a proportion.

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. You can also access it by clicking on the history tab in the upper right frame. 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 greater 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 of either TRUE if that year had more boys than girls, and FALSE if that year did not. These are a different kind of data than we have considered so far. In the `arbuthnot` dataframe our values are integers (the year, the number of boys and girls). Here, we've asked R to create *logical* data, data where the values are either TRUE or 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. The next time you launch RStudio, you will see the commands you typed in your previous session.

On Your Own[*]

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 in the United States. Load up the present day data with the following command.

```
source("http://www.openintro.org/stat/data/present.R")
```

The data are stored in a dataframe called `present`.

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

2. How do these counts compare to Arbuthnot's? Are they on a similar scale?

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

4. Make a plot that displays the boy-to-girl ratio for every year in the dataset. What do you see?

---

[*]Usually this section will be due with the weekly homework assignment. Since we're just getting started, you do not need to turn in your answers this week, however you should still work them out.

5. In what year did we see the most total number of births in the U.S.? You can refer to the help files or the R reference card (*http://cran.r-project.org/doc/contrib/Short-refcard.pdf*) to find helpful command(s).

These data come from a report by the Centers for Disease Control (*http://www.cdc.gov/nchs/data/nvsr/nvsr53/nvsr53_20.pdf*). Check it out if you would like to read more about an analysis of sex ratios at birth in the United States.

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

Submitting your workspace

At the end of each lab session submit your workspace on *Sakai*. To do so, you need to first save your workspace by going to Workspace → Save Workspace As... and save your file as your Net ID. Then go to Sakai and upload your workspace under the corresponding assignment. Make sure to upload your file to the lab assignment for your section. In order to get credit for the lab you must upload your workspace within 10 minutes of the class end time. If you would like to keep working on the lab after that point, you're welcomed to do so, but make sure you upload your workspace first.

Notes

This lab was adapted for OpenIntro by Andrew Bray and Mine Çetinkaya-Rundel from a lab written by Mark Hansen of UCLA Statistics.