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 texbook 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.

Go ahead and sign on to RStudio at https://vm-manage.oit.duke.edu/containers. You should see a window that looks like the image shown below.

The panel in the upper right contains your *workspace* as well as a history of the commands that you've previously entered. Any plots that you generate will show up in the panel in the lower right corner.

The panel 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 of R 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.

Creating a reproducible lab report

We will be using a markdown language, R Markdown, to type up the lab report. This allows you to complete your lab entirely in RStudio as well as ensuring reproducibility of your analysis and results. To help get you started we are providing a template for you. Use the following code to download this template:

```
download.file("http://stat.duke.edu/courses/Summer17/sta101.001-2/uploads/rmd/lab1.Rmd", destfile = "la
```

You will see a new file called `lab1.Rmd` in the Files tab on the pane in the bottom right corner of your RStudio window. We will refer to this as your "R markdown file" or "your report". Click on the file name to open the file. All you need to do to complete the lab is to type up your **brief** answers and the R code (when necessary) in the spaces provided in the document. Earlier in the lab spaces are provided for you to enter R code chunks. Later in the lab you'll need to figure out whether code is needed to answer a particular question, and if so a new chunk can be inserted by clicking on the *Insert Chunk* button (dropdown menu under *Chunks* on the upper right corner of your markdown document).

Before you keep going type your team name, the name of the team member who is the "author" for the day, and the "discussants", who are the other team members present in lab today. Then click on *Knit HTML* and you'll see your document in a new pop-up window.

R Packages

R is an open-source programming language, meaning that users can contribute packages that make our lives easier, and we can use them for free. For this lab, and many others in the future, we will use the following R packages:

- `dplyr`: for data wrangling
- `ggplot2`: for data visualization

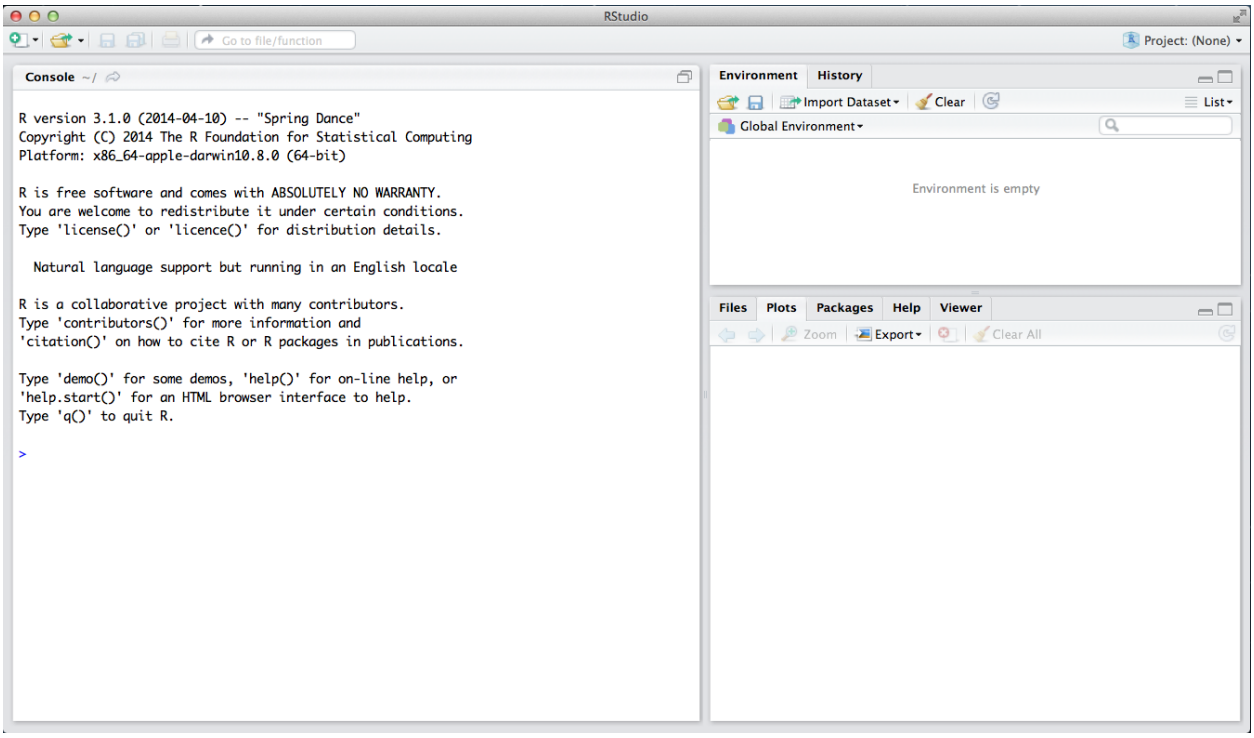

Figure 1: rinterface

These packages have already been installed for you, however you will still need to load them in your working environment. To do so, type the following in the console:

```
library(dplyr)
library(ggplot2)
```

Note that these two lines of code also appear on top of your R Markdown document. We need to load the packages both in the console and in your R Markdown environment since these two environments work independently of each other.

Going forward you will be asked to load any relevant packages at the beginning of each lab, and the code for loading these packages will also be included on top of your R Markdown document as well.

The Data: Dr. Arbuthnot's Baptism Records

To get you started, run the following command from your markdown file.

```
load(url("https://stat.duke.edu/~mc301/data/arbuthnot.RData"))
```

To do so, you can simply put your cursor on that line, and hit the **Run** button on the upper right corner of the pane.

This command instructs R to load some data: the Arbuthnot baptism 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, and sometimes you create them yourself as the byproduct of a computation or some analysis you have performed.

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 baptism 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
```

However printing the whole dataset in the console is not that useful. One advantage of RStudio is that it comes with a built-in data viewer. Click on the name `arbuthnot` in the *Environment* pane (upper right window) that lists the objects in your workspace. This will bring up an alternative display of the data set in the *Data Viewer* (upper left window). You can close the data viewer by clicking on the x in the upper lefthand corner.

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 baptized that year, respectively. Use the scrollbar on the right side of the console 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 generally be helpful. R has stored Arbuthnot's data in a kind of spreadsheet or table called a *data frame*.

You can see the dimensions of this data frame by typing:

```
dim(arbuthnot)
```

This command should output `[1] 82 3`, indicating that there are 82 rows and 3 columns (we'll get to what the `[1]` means in a bit), 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 data frame 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 math class; 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 data frame.

Some Exploration

Let's start to examine the data a little more closely. We can access the data in a single column of a data frame separately using a command like

```
arbuthnot$boys
```

This command will only show the number of boys baptized each year. The dollar sign basically says "go to the data frame that comes before me, and find the variable that comes after me".

1. What command would you use to extract just the counts of girls baptized? Try it! (Enter your answer in your R markdown document and run the entire report by hitting **Knit HTML**. Voila! The R output you need is already in your report. Let's make this the first and last time we print out an entire dataset in our lab reports!)

Notice that the way R has printed these data is different. When we looked at the complete data frame, we saw 82 rows, one on each line of the display. These data are no longer structured in a table with other variables, so they are displayed one right after another. Objects that print out in this way are called vectors; they represent a set of numbers. R has added numbers in [brackets] along the left side of the printout to indicate locations within the vector. For example, 5218 follows [1], indicating that 5218 is the first entry in the vector. And if [43] starts a line, then that would mean the first number on that line would represent the 43rd entry in the vector.

R has some powerful functions for making graphics. We can create a simple plot of the number of girls baptized per year with the command

```
qplot(x = year, y = girls, data = arbuthnot)
```

The `qplot()` function (meaning "quick plot") looks at the type of data you have provided it and makes the decision to visualize it with a scatterplot. The plot should appear under the *Plots* tab of the lower right panel of RStudio. Notice that the command above again looks like a function, this time with three arguments separated by commas. The first two arguments in the `qplot()` function specify the variables for the x-axis and the y-axis and the third provides the name of the data set where they can be found. If we wanted to connect the data points with lines, we could add a fourth argument to specify the geometry that we'd like.

```
qplot(x = year, y = girls, data = arbuthnot, geom = "line")
```

You might wonder how you are supposed to know that it was possible to add that fourth argument. Thankfully, R documents all of its functions extensively. To read what a function 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.

```
?qplot
```

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

2. Is there an apparent trend in the number of girls baptized over the years? How would you describe it?

Now, suppose we want to plot the total number of baptisms. 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 baptisms in 1629. We could repeat this once for each year, but there is a faster way. If we add the vector for baptisms for boys to that of girls, R will compute all sums simultaneously.

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

What you will see are 82 numbers (in that packed display, because we aren't looking at a data frame here), each one representing the sum we're after. Take a look at a few of them and verify that they are right.

We'll be using this new vector to generate some plots, so we'll want to save it as a permanent column in our data frame.

```
arbuthnot <- arbuthnot %>%
  mutate(total = boys + girls)
```

What in the world is going on here? The `%>%` operator is called the **piping** operator. Basically, it takes the output of the current line and pipes it into the following line of code.

A note on piping: Note that we can read these three lines of code as the following:

"Take the ***`arbuthnot`*** *dataset and* ***pipe*** *it into the* ***`mutate`*** *function. Using this mutate a new variable called* ***`total`*** *that is the sum of the variables called* ***`boys`*** *and* ***`girls`****. Then assign this new resulting dataset to the object called* ***`arbuthnot`****, i.e. overwrite the old* ***`arbuthnot`*** *dataset with the new one containing the new variable."*

This is essentially equivalent to going through each row and adding up the boys and girls counts for that year and recording that value in a new column called total.

Where is the new variable? When you make changes to variables in your dataset, click on the name of the dataset again to update it in the data viewer.

You'll see that there is now a new column called `total` that has been tacked on to the data frame. The special symbol `<-` performs an *assignment*, taking the output of one line of code and saving it into an object

in your workspace. In this case, you already have an object called `arbuthnot`, so this command updates that data set with the new mutated column.

We can make a plot of the total number of baptisms per year with the command

```
qplot(x = year, y = total, data = arbuthnot, geom = "line")
```

Similarly to how we computed the total number of births, we can compute the ratio of the number of boys to the number of girls baptized in 1629 with

```
5218 / 4683
```

or we can act on the complete columns with the expression

```
arbuthnot <- arbuthnot %>%
  mutate(boy_to_girl_ratio = boys / girls)
```

We can also compute the proportion of newborns that are boys in 1629

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

or this may also be computed for all years simultaneously and append it to the dataset:

```
arbuthnot <- arbuthnot %>%
  mutate(boy_ratio = boys / total)
```

Note that we are using the new `total` variable we created earlier in our calculations.

3. Now, generate a plot of the proportion of boys born over time. What do you see?

Tip: 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 panel. 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 in each year with the expression

```
arbuthnot <- arbuthnot %>%
  mutate(more_boys = boys > girls)
```

This command add a new variable to the `arbuthnot` dataframe containing the values of either `TRUE` if that year had more boys than girls, or `FALSE` if that year did not (the answer may surprise you). This variable contains different kind of data than we have considered so far. All other columns in the `arbuthnot` data frame have values are numerical (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 lab, so let's stop here. To exit RStudio just close your browser window. And then sign back in to RStudio at

https://vm-manage.oit.duke.edu/containers

You'll find your R session is just as you left it.

On Your Own

In the previous few pages, you recreated some of the displays and preliminary analysis of Arbuthnot's baptism 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.

```
load(url("https://stat.duke.edu/~mc301/data/present.RData"))
```

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

- What years are included in this data set? What are the dimensions of the data frame? What are the variable (column) names?
- How do these counts compare to Arbuthnot's? Are they on a similar scale?
- Make a plot that displays the proportion of boys born over time. What do you see? Does Arbuthnot's observation about boys being born in greater proportion than girls hold up in the U.S.? Include the plot in your response. *Hint:* You should be able to reuse your code from Ex 3 above, just replace the dataframe name.
- In what year did we see the most total number of births in the U.S.? *Hint:* First calculate the totals and save it as a new variable. Then, sort your dataset in descending order based on the total column. You can do this interactively in the data viewer by clicking on the arrows next to the variable names. To include the sorted result in your report you will need to use two new functions: `arrange` (for sorting). We can arrange the data in a descenting order with another function: `descr` (for descending order). Sample code provided below.

```
present %>%
  arrange(desc(total))
```

These data come from reports by the Centers for Disease Control listed in the references section. If you would like to read more abour an analysis of sex ratios at birth in the United states, check out this report.

Resources for learning R and working in RStudio

That was a short introduction to R and RStudio, but we will provide you with more functions and a more complete sense of the language as the course progresses.

In this course we will be using R packages called `dplyr` for data wrangling and `ggplot2` for data visualization. If you are googling for R code, make sure to also include these package names in your search query. For example, instead of googling "scatterplot in R", google "scatterplot in R with ggplot2".

These cheathseets may come in handy throughout the semester:

- RMarkdown cheatsheet
- Data wrangling cheatsheet
- Data visualization cheatsheet

Note that some of the code on these cheatsheets may be too advanced for this course, however majority of it will become useful throughout the semester.

References

Martin, Joyce A, National Center for Health Statistics (US), and others. 2003. "Births: Final Data for 2002." *National Vital Statistics Reports* 52 (10). US Department of Health; Human Services, Centers for Disease

Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr52/nvsr52_10.pdf.

———. 2005. "Births: Final Data for 2003." *National Vital Statistics Reports* 52 (10). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr54/nvsr54_02.pdf.

———. 2006. "Births: Final Data for 2004." *National Vital Statistics Reports* 55 (1). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr55/nvsr55_01.pdf.

———. 2007. "Births: Final Data for 2005." *National Vital Statistics Reports* 56 (06). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr56/nvsr56_06.pdf.

———. 2009. "Births: Final Data for 2006." *National Vital Statistics Reports* 57 (07). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr57/nvsr57_07.pdf.

———. 2010a. "Births: Final Data for 2007." *National Vital Statistics Reports* 58 (24). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr58/nvsr58_24.pdf.

———. 2010b. "Births: Final Data for 2008." *National Vital Statistics Reports* 59 (01). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr59/nvsr59_01.pdf.

———. 2011. "Births: Final Data for 2009." *National Vital Statistics Reports* 60 (01). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr60/nvsr60_01.pdf.

———. 2012a. "Births: Final Data for 2010." *National Vital Statistics Reports* 61 (01). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr61/nvsr61_01.pdf.

———. 2012b. "Births: Final Data for 2011." *National Vital Statistics Reports* 61 (01). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr61/nvsr61_01.pdf.

———. 2013. "Births: Final Data for 2012." *National Vital Statistics Reports* 62 (09). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr62/nvsr62_09.pdf.

———. 2015. "Births: Final Data for 2013." *National Vital Statistics Reports* 64 (01). US Department of Health; Human Services, Centers for Disease Control; Prevention, National Center for Health Statistics. http://www.cdc.gov/nchs/data/nvsr/nvsr64/nvsr64_01.pdf.