Lab 1.2: Introduction to R and RStudio

Example 1 We will work with data on annual rainfall in inches for various cities throughout the world.

Algiers	30	Lagos	72
Athens	16	La Paz	23
Beirut	35	Lima	2
Berlin	23	London	23
Bogota	42	Madrid	17
Bombay	71	Moscow	25
Cairo	1	Oslo	27
Dublin	30	Paris	22
Geneva	34	Rome	30
Havana	48	Vienna	26

Open RStudio. We will enter the numbers in the Console (Quadrant III) using the R function “c”. The standard file format for statistical data is that each column is a variable and each row is a case. Here the variable is rainfall and the cases are the cities. Sometimes I like to think of the name of the “c” function as representing “column”. You can use it to enter a column of data, i.e., a single variable. Use it like this: at the R prompt “>” type

```
rainfall = c(30, 16, 35, 23,..., 30, 26)
```

Of course, you must type in the rest of the data where I typed "...".Hit RETURN at the end of the line. Nothing happens. If in doubt, R is silent. To check to see if you succeeded, just type rainfall at the R prompt. R should tell you what is in the variable "rainfall".

```
rainfall
[1] 30 16 35 23 42 71 1 30 34 48 72 23 2 23 17 25 27 22 30 26
```

Ignore the [1] for now. There are many slick ways to get data into R but for now just typing it in will do. Another option is a simple data editor available in some versions of R. Type

```
data.entry(rainfall)
```

to see if your version includes this feature. If it does, this is a good time to edit any typos in your data entry. The following worked in the Windows version. Double click on a cell to edit it. Hit RETURN when done with that cell. When done with all cells, right click on the Data Editor and chose Close. The Data Editor only edits data already entered into R. You can trick it into creating a new column of data. Let's say we also have snowfall data. Type

```
snowfall = c(1)
data.entry(snowfall)
```

The "1" in the first command is just a placeholder. Any number will do. When the Data Editor opens, replace the 1 with actual data. Once you have the data in R, you can get a variety of summary statistics and displays. Try some of the following.

```
mean(rainfall)
[1] 29.85
```

```
median(rainfall)
[1] 26.5
```

If you get different numbers, proofread your data for typos. If you do not have a Data Editor, you can fix one number at a time from the command line. Let's say that for the sixth city, Bombay, you typed 17 instead of 71.

```
rainfall[6]=71
```

will fix this.

```
mode(rainfall)
[1] "numeric"
```

This is probably not what we had in mind for the mode. R is telling us that numerical data is what is stored in rainfall. If you really want the mode, the table command will work for data sets for which a mode is reasonable.

```
table(rainfall)
rainfall
 1  2 16 17 22 23 25 26 27 30 34 35 42 48 71 72
 1  1  1  1  1  3  1  1  1  3  1  1  1  1  1  1
```

This tells us 23 and 30 are tied for mode with three occurrences each.

```
sd(rainfall)
[1] 18.07084
max(rainfall)
[1] 72
min(rainfall)
[1] 1
range(rainfall)
[1]  1 72
```

Oh, my! I work so hard to convince my students that the range is one number! Use diff on the output of the previous command.

```
diff(range(rainfall))
[1] 71
```

```
fivenum(rainfall)
[1]  1.0 22.5 26.5 34.5 72.0
```

```
lentgth(rainfall)
Error: couldn't find function "lentgth"
```

```
length(rainfall)
[1] 20
```

So, the range is 71. Many people include a sixth number in the five number summary: the number of observations, n. This is returned by the length function in R. (If you mistype something, R will give an error message. Most are much more cryptic than this one.) We know that different textbooks use different definitions of the second and fourth numbers in the five number summary. Here's how R does it:

```
fivenum(c(1,2,3,4,5))
[1] 1 2 3 4 5
```

```
fivenum(c(1,4,5,6))
[1] 1.0 2.5 4.5 5.5 6.0
```

```
stem(rainfall)
```

The decimal point is 1 digit(s) to the right of the |

```
  0 | 1267
  2 | 233356700045
  4 | 28
  6 | 12
```

I believe that every software lesson should include analysis of a dataset that shows how the software can actually be used to find out something useful about the data. Here we will also learn something useful about stem and leaf plots. They can be made on a variety of scales. The one chosen by R is a bit odd. If we type

```
?stem
```

we get help on the stem command. (How help appears depends on the platform. In RStudio, it arrives in Quadrant IV.) In this case, the help included this information:

'`stem`' produces a stem-and-leaf plot of the values in 'x'. The parameter '`scale`' can be used to expand the scale of the plot. A value of '`scale= 2`' will cause the plot to be roughly twice as long as the default.

Let's try it!

```
stem(rainfall, scale=0.5)

The decimal point is 2 digit(s) to the right of the |

  0 | 00222222333333344
  0 | 577
```

```
stem(rainfall, scale=2)
The decimal point is 1 digit(s) to the right of the |

  0 | 12
  1 | 67
  2 | 2333567
  3 | 00045
  4 | 28
  5 |
  6 |
  7 | 12
```

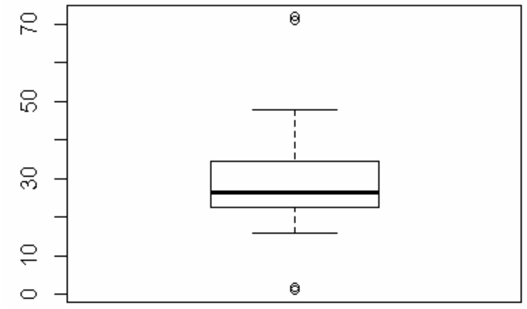
```
stem(rainfall, scale=5)

The decimal point is 1 digit(s) to the right of the |

  0 | 12
  0 |
  1 |
  1 | 67
  2 | 2333
  2 | 567
  3 | 0004
  3 | 5
  4 | 2
  4 | 8
  5 |
  5 |
  6 |
  6 |
  7 | 12
```

```
stem(rainfall, scale=10)

The decimal point is at the |

   0 | 0
   2 | 0
   4 |
   6 |
   8 |
  10 |
  12 |
  14 |
  16 | 00
  18 |
  20 |
  22 | 0000
  24 | 0
  26 | 00
  28 |
  30 | 000
  32 |
  34 | 00
  36 |
  38 |
  40 |
  42 | 0
  44 |
  46 |
  48 | 0
  50 |
  52 |
  54 |
  56 |
  58 |
  60 |
  62 |
  64 |
  66 |
  68 |
  70 | 0
  72 | 0
```

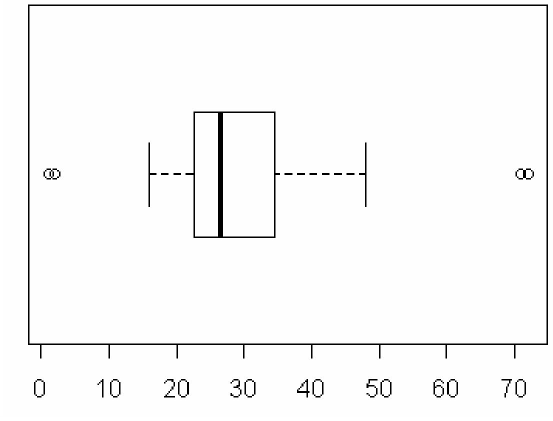
The stem and leaf with scale=0.5 is also odd. scale=2 is probably what you or your students would have done. It is better than the previous two scales because it reveals the two potential high outliers. Using scale=5 is even better because it shows the two potential low outliers. Are they really outliers? On my R system

```
boxplot(rainfall)
```

caused a boxplot to pop up in a graphics window. Go to Export in Quad IV and copy to clipboard.



```
boxplot(rainfall, horizontal=TRUE)
```



That's better.
For data such as this, which appears skewed toward high values, a transformation is often appropriate.

```
stem(log(rainfall))
```

The decimal point is at the |

```
  0 | 07
  1 |
  2 | 88
  3 | 11112334445679
  4 | 33
```

```
stem(log(rainfall), scale=2)
```

The decimal point is at the |

```
  0 | 0
  0 | 7
  1 |
  1 |
  2 |
  2 | 88
  3 | 1111233444
  3 | 5679
  4 | 33
```

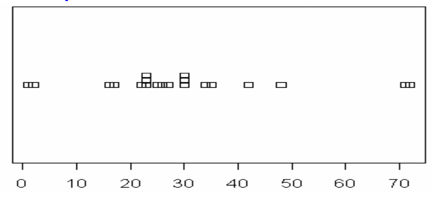
On a logarithmic scale, only the low outliers appear!

Discussion Topic

If you were in international agribusiness, which rainfall would be more of an outlier for your purposes, one inch per year or 72 inches per year?

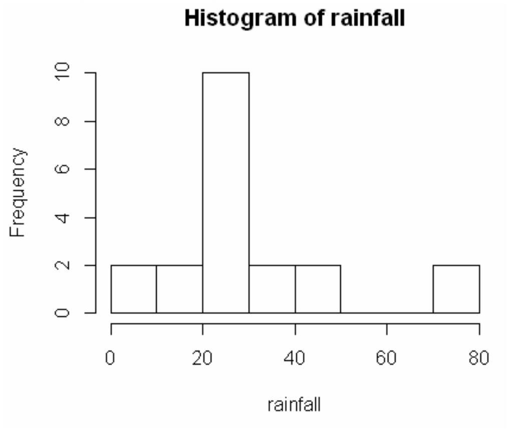
You can also get other displays from R. Here is something akin to what I would call a dotplot:

```
stripchart(rainfall, method = "stack")
```



And a histogram:

```
hist(rainfall)
```



Moral

If we were to examine this data by hand, we might be tempted to do a single display simply because making more is extra work. A stem and leaf is a good choice if we are doing things by hand. Were we to chose the same scale R did, we would not have learned much about this data. However, using the computer we were able to change the scale and select the best scale for our data. In the process we discovered two high outliers and then two low outliers. A boxplot, a logarithmic transformation, and thought about the meaning of the data all confirmed the significance of the low outliers.

Save Workspace Image?

When you leave R, you will be asked if you want to **`Save Workspace Image`**. *You do!* This will mean that all the data you worked hard to type in or import will be there waiting for you the next time you start R.

Example 2. To get you started, enter the following command at the R prompt (i.e. right after `>` on the console). 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 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. 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 Baptism 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 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
```

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 dataset.

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 `[1] "year"  "boys"  "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. One advantage of RStudio is that it comes with a built-in data viewer. Click on the name arbuthnot in the upper right window (Quardant I) that lists the objects in your workspace. This will bring up an alternative display of the Arbuthnot counts in the upper left window (Quadrant III). 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 a single column of a data frame separately using a command like

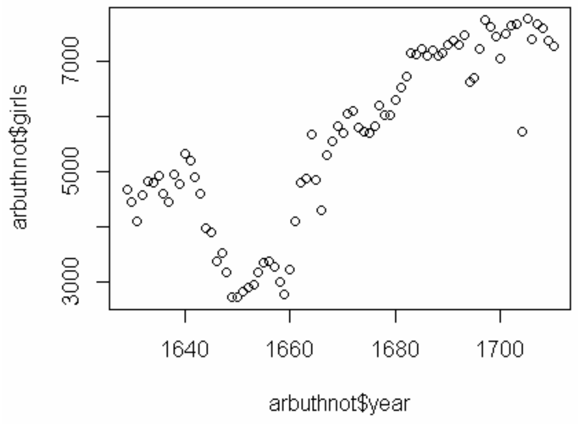
```
arbuthnot$boys
```

This command will only show the number of boys baptized each year.

`Exercise 1` What command would you use to extract just the counts of girls baptized? Try it!

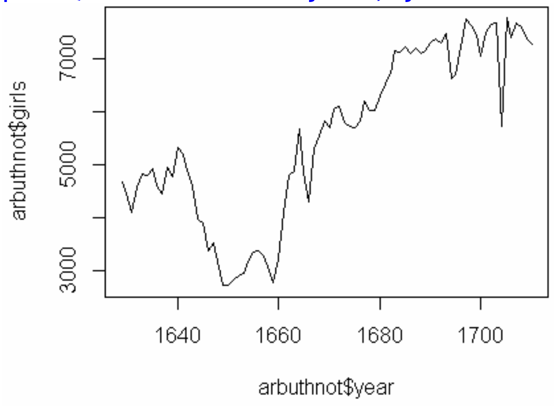
Notice that the way R has printed these data is different. When we looked at the complete data frame, we saw 82 rows, one on each line of the display. These data are no longer structured in a table with other variables, so they are displayed one right after another. Objects that print out in this way 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 `[34]` starts a line, then that would mean the first number on that line would represent the 34th 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

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



By default, R creates a scatterplot with each x,y pair indicated by an open circle. The plot itself 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 two arguments separated by a comma. The first argument in the plot function specifies the variable 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 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.

```
?plot
```

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.

Exercise 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 and 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. Therefore, we can make a plot of the total number of baptisms 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 baptized 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

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

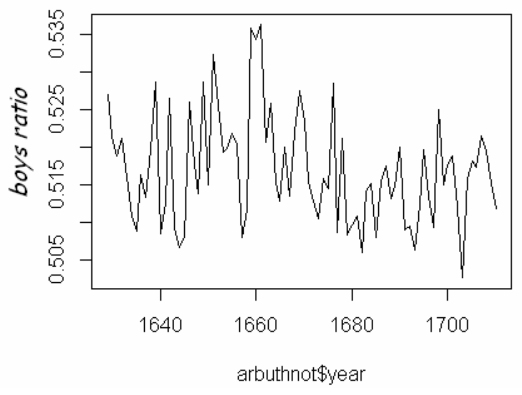
or this may also be computed for all years simultaneously:

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

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.

Exercise 3 Now, make a plot of the proportion of boys 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 (Quad1). This will save you a lot of typing in the future.

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



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$boys > arbuthnot$girls
```

This command returns 82 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 output shows a different kind of data than we have considered so far. In the arbuthnot data frame our 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 you can click the “x” in the upper right corner of the whole window. You will be prompted to save your workspace. If you click “save”, RStudio will save the history of your commands and all the objects in your workspace so that the next time you launch RStudio, you will see arbuthnot and you will have access to the commands you typed in your previous session. For now, click “save”, then start up RStudio again.

Lab 1: On Your Own **Name**_______________________ **Score**______

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.

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

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

1. What years are included in this data set? What are the dimensions of the data frame and what are the variable or column names? `1940 - 2002`

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 data set. What do you see?

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 commands.

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 functions 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://rstudio.org if you're interested in learning more, or find more labs for practice at http://openintro.org .