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.

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.

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("more/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 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:

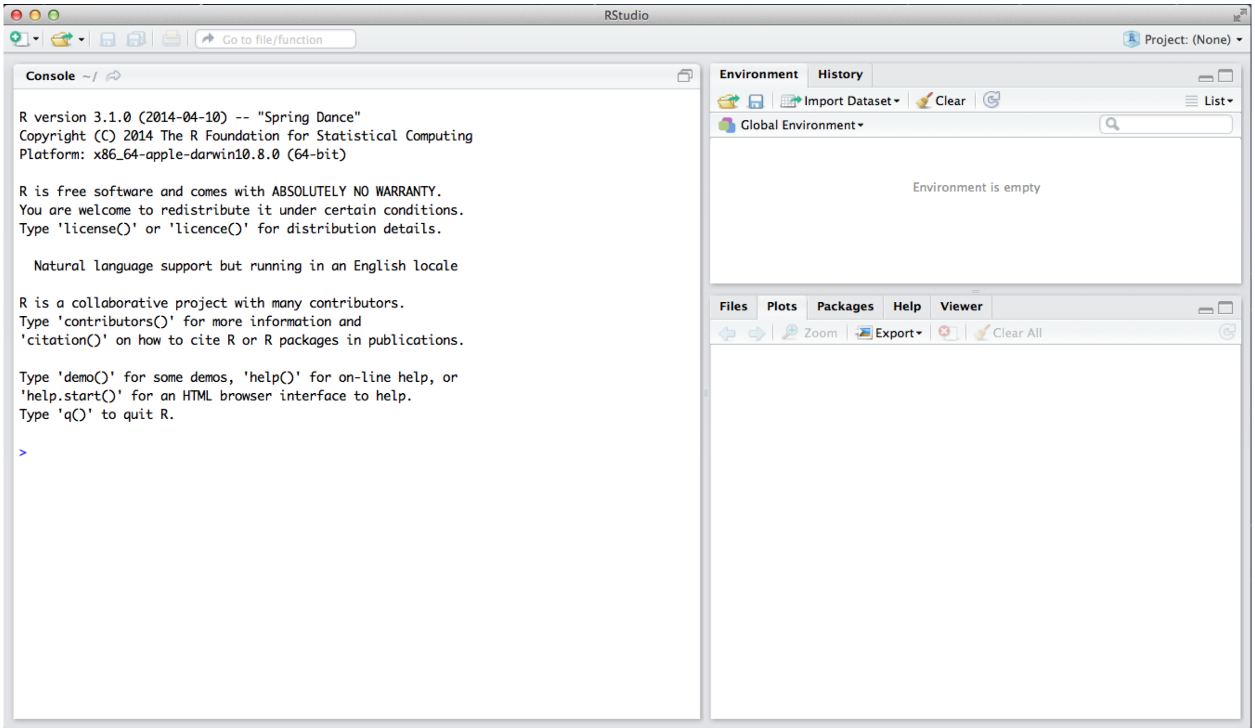


Figure 1: rinterface

```
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.

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.

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.

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

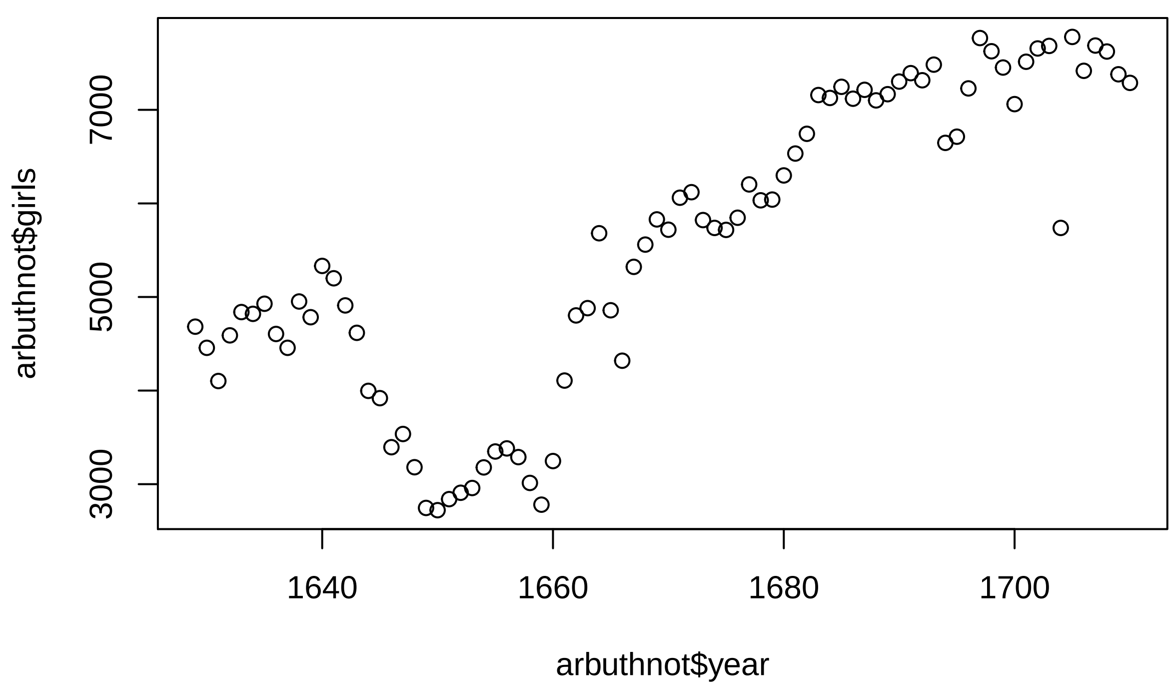
```
arbuthnot$girls
```

```
##  [1] 4683 4457 4102 4590 4839 4820 4928 4605 4457 4952 4784 5332 5200 4910
## [15] 4617 3997 3919 3395 3536 3181 2746 2722 2840 2908 2959 3179 3349 3382
## [29] 3289 3013 2781 3247 4107 4803 4881 5681 4858 4319 5322 5560 5829 5719
## [43] 6061 6120 5822 5738 5717 5847 6203 6033 6041 6299 6533 6744 7158 7127
## [57] 7246 7119 7214 7101 7167 7302 7392 7316 7483 6647 6713 7229 7767 7626
## [71] 7452 7061 7514 7656 7683 5738 7779 7417 7687 7623 7380 7288
```

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.

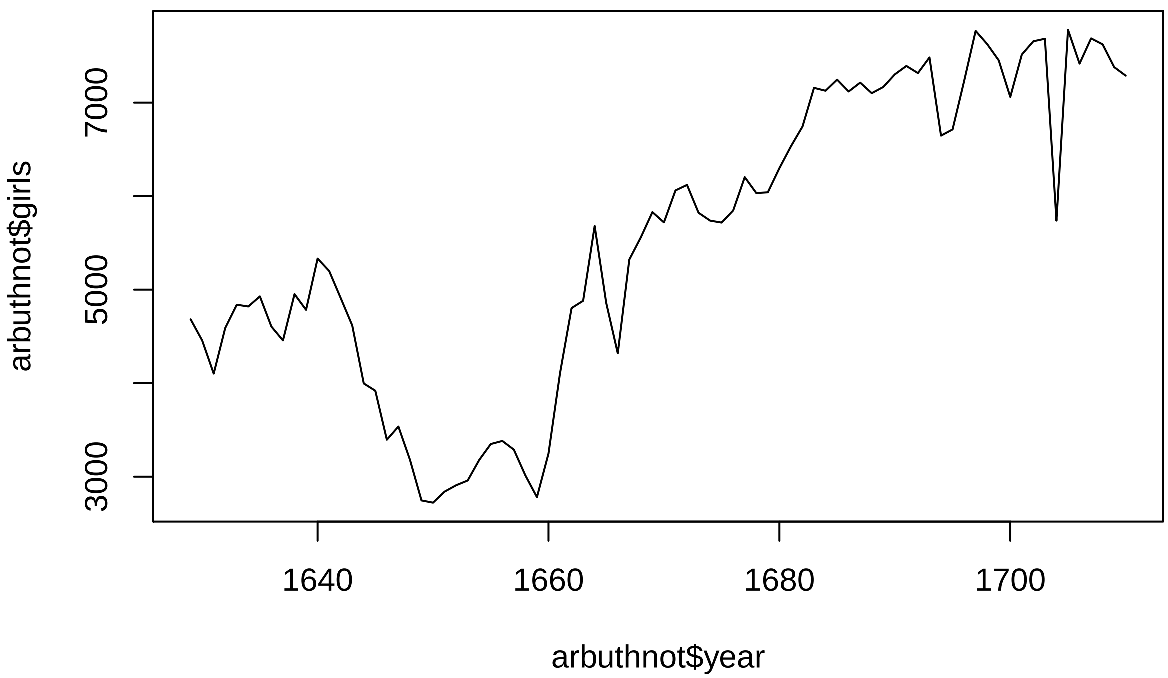
R has some powerful functions for making graphics. We can create a simple plot of the number of girls baptized per year with the command

```
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 **l**ine.

```
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
```

```
## starting httpd help server ... done
```

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?

   Zach's Response: Yes, overall, there is an apparent trend of an increase in the number of girls baptized over the years. More specifically, after an initial increase from 1629 to around 1640, there appears to be a decrease in the number of girls baptized around 1640. This decrease continues until about 1660. From 1660 to 1710 there is a large increase in the number of girls baptized.

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
```

```
## [1] 9901
```

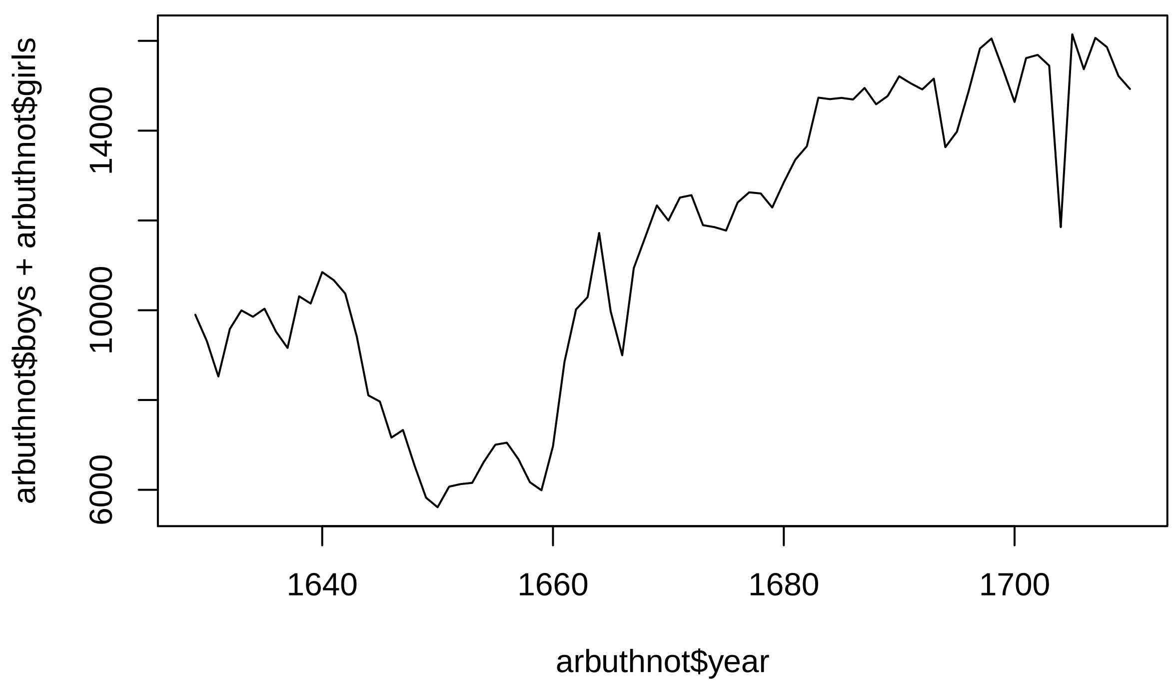
to see the total number of baptisms in 1629. We could repeat this once for each year, but there is a faster way. If we add the vector for baptisms for boys and girls, R will compute all sums simultaneously.

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

```
##  [1]  9901  9315  8524  9584  9997  9855 10034  9522  9160 10311 10150
## [12] 10850 10670 10370  9410  8104  7966  7163  7332  6544  5825  5612
## [23]  6071  6128  6155  6620  7004  7050  6685  6170  5990  6971  8855
## [34] 10019 10292 11722  9972  8997 10938 11633 12335 11997 12510 12563
## [45] 11895 11851 11775 12399 12626 12601 12288 12847 13355 13653 14735
## [56] 14702 14730 14694 14951 14588 14771 15211 15054 14918 15159 13632
## [67] 13976 14861 15829 16052 15363 14639 15616 15687 15448 11851 16145
## [78] 15369 16066 15862 15220 14928
```

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
```

```
## [1] 1.114243
```

or we can act on the complete vectors with the expression

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

```
##  [1] 1.114243 1.089971 1.078011 1.088017 1.065923 1.044606 1.036120
##  [8] 1.067752 1.055194 1.082189 1.121656 1.034884 1.051923 1.112016
## [15] 1.038120 1.027521 1.032661 1.109867 1.073529 1.057215 1.121267
## [22] 1.061719 1.137676 1.107290 1.080095 1.082416 1.091371 1.084565
## [29] 1.032533 1.047793 1.153901 1.146905 1.156075 1.085988 1.108584
## [36] 1.063369 1.052697 1.083121 1.055242 1.092266 1.116143 1.097744
## [43] 1.064016 1.052778 1.043112 1.065354 1.059647 1.120575 1.035467
## [50] 1.088679 1.034100 1.039530 1.044237 1.024466 1.058536 1.062860
## [57] 1.032846 1.064054 1.072498 1.054359 1.060974 1.083128 1.036526
## [64] 1.039092 1.025792 1.050850 1.081931 1.055748 1.037981 1.104904
## [71] 1.061594 1.073219 1.078254 1.048981 1.010673 1.065354 1.075460
## [78] 1.072132 1.090022 1.080808 1.062331 1.048299
```

The proportion of newborns that are boys

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

```
## [1] 0.5270175
```

or this may also be computed for all years simultaneously:

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

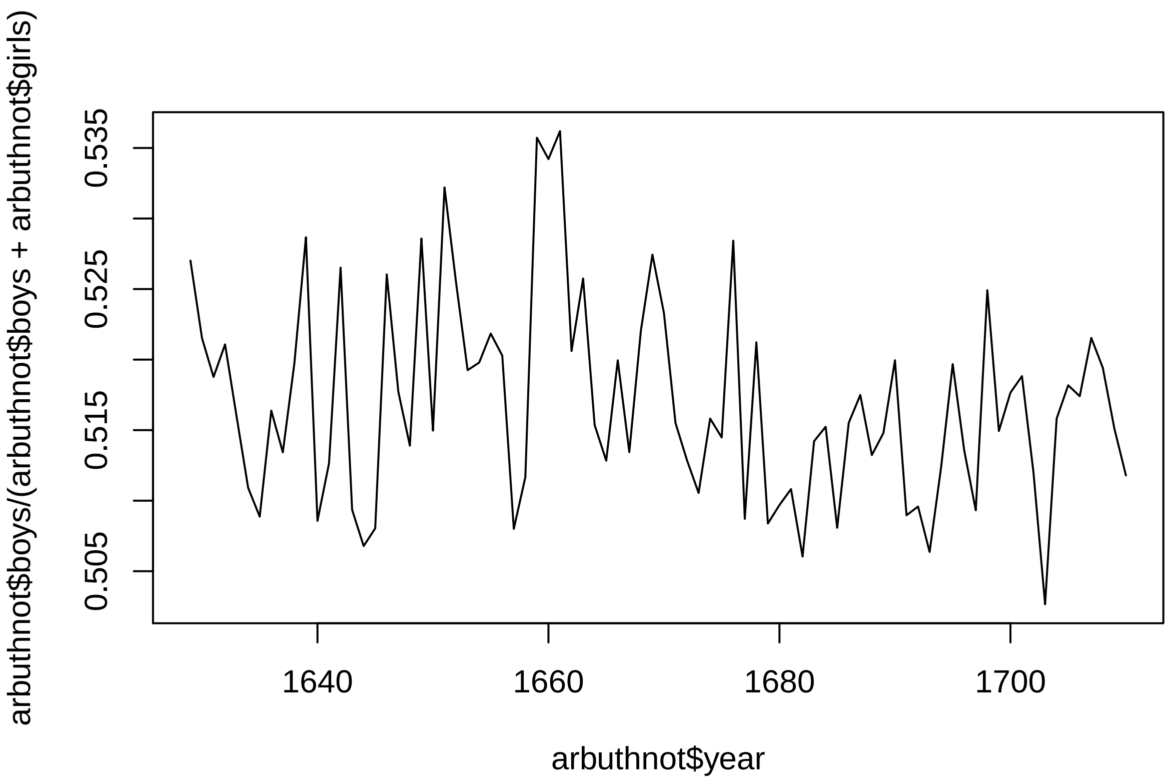
```
##  [1] 0.5270175 0.5215244 0.5187705 0.5210768 0.5159548 0.5109082 0.5088698
##  [8] 0.5163831 0.5134279 0.5197362 0.5286700 0.5085714 0.5126523 0.5265188
## [15] 0.5093518 0.5067868 0.5080341 0.5260366 0.5177305 0.5139059 0.5285837
## [22] 0.5149679 0.5322023 0.5254569 0.5192526 0.5197885 0.5218447 0.5202837
## [29] 0.5080030 0.5116694 0.5357262 0.5342132 0.5361942 0.5206108 0.5257482
## [36] 0.5153557 0.5128359 0.5199511 0.5134394 0.5220493 0.5274422 0.5232975
## [43] 0.5155076 0.5128552 0.5105507 0.5158214 0.5144798 0.5284297 0.5087122
## [50] 0.5212285 0.5083822 0.5096910 0.5108199 0.5060426 0.5142178 0.5152360
## [57] 0.5080788 0.5155165 0.5174905 0.5132301 0.5147925 0.5199527 0.5089677
## [64] 0.5095857 0.5063659 0.5123973 0.5196766 0.5135590 0.5093183 0.5249190
## [71] 0.5149385 0.5176583 0.5188268 0.5119526 0.5026541 0.5158214 0.5181790
## [78] 0.5174052 0.5215362 0.5194175 0.5151117 0.5117899
```

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.

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. This will save you a lot of typing in the future.

   Zach's Response: There appears to be a slight decrease in the proportion of boys being baptized over time (see plot below).

```
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
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [29] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [43] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [57] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [71] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

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.

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.

```
source("more/present.R")
```

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 and what are the variable or column names?

The years included in the dataset are from 1940 to 2002.

```
range(present$year)
```

```
## [1] 1940 2002
```

The dimensions of the data frame are 63 rows and 3 columns.

```
dim(present)
```

```
## [1] 63  3
```

The variable names are "year", "boys", and "girls".

```
names(present)
```

```
## [1] "year"  "boys"  "girls"
```

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

The counts in the "present" data set are much higher than those in Arbuthnot's data set. Although we are looking at different years between the two data sets, the number of births for boys and girls are much higher than the number of baptisms in Arbuthnot's data set. They are not on a similar scale.

- Make a plot that displays the boy-to-girl ratio for every year in the data set. 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.

When looking at the plot (see below) and ratio of boys to girls born in the U.s. every year since 1940, Arbuthnot's observation that boys are being born in greater proportion than girls does hold up in the U.S. for the time period between 1940 and 2002. I created an extra column/variable in the present dataset to calculate the boy-to-girl ratio by year:

```
# created a column in the present dataset to calculate the
# boy-to-girl ratio for every year in the dataset.
present$boyprop <- (present$boys / present$girls)
present
```

```
##    year    boys   girls  boyprop
## 1  1940 1211684 1148715 1.054817
## 2  1941 1289734 1223693 1.053969
## 3  1942 1444365 1364631 1.058429
## 4  1943 1508959 1427901 1.056767
## 5  1944 1435301 1359499 1.055757
## 6  1945 1404587 1330869 1.055391
## 7  1946 1691220 1597452 1.058698
## 8  1947 1899876 1800064 1.055449
## 9  1948 1813852 1721216 1.053820
## 10 1949 1826352 1733177 1.053760
## 11 1950 1823555 1730594 1.053716
## 12 1951 1923020 1827830 1.052078
## 13 1952 1971262 1875724 1.050934
## 14 1953 2001798 1900322 1.053399
## 15 1954 2059068 1958294 1.051460
## 16 1955 2073719 1973576 1.050742
## 17 1956 2133588 2029502 1.051286
## 18 1957 2179960 2074824 1.050672
## 19 1958 2152546 2051266 1.049374
## 20 1959 2173638 2071158 1.049480
## 21 1960 2179708 2078142 1.048873
## 22 1961 2186274 2082052 1.050057
## 23 1962 2132466 2034896 1.047948
## 24 1963 2101632 1996388 1.052717
## 25 1964 2060162 1967328 1.047188
## 26 1965 1927054 1833304 1.051137
## 27 1966 1845862 1760412 1.048540
## 28 1967 1803388 1717571 1.049964
## 29 1968 1796326 1705238 1.053417
## 30 1969 1846572 1753634 1.052997
## 31 1970 1915378 1816008 1.054719
## 32 1971 1822910 1733060 1.051845
## 33 1972 1669927 1588484 1.051271
```

```
## 34 1973 1608326 1528639 1.052129
## 35 1974 1622114 1537844 1.054797
## 36 1975 1613135 1531063 1.053605
## 37 1976 1624436 1543352 1.052538
## 38 1977 1705916 1620716 1.052569
## 39 1978 1709394 1623885 1.052657
## 40 1979 1791267 1703131 1.051749
## 41 1980 1852616 1759642 1.052837
## 42 1981 1860272 1768966 1.051615
## 43 1982 1885676 1794861 1.050597
## 44 1983 1865553 1773380 1.051976
## 45 1984 1879490 1789651 1.050199
## 46 1985 1927983 1832578 1.052061
## 47 1986 1924868 1831679 1.050876
## 48 1987 1951153 1858241 1.050000
## 49 1988 2002424 1907086 1.049991
## 50 1989 2069490 1971468 1.049720
## 51 1990 2129495 2028717 1.049676
## 52 1991 2101518 2009389 1.045849
## 53 1992 2082097 1982917 1.050017
## 54 1993 2048861 1951379 1.049955
## 55 1994 2022589 1930178 1.047877
## 56 1995 1996355 1903234 1.048928
## 57 1996 1990480 1901014 1.047062
## 58 1997 1985596 1895298 1.047643
## 59 1998 2016205 1925348 1.047190
## 60 1999 2026854 1932563 1.048791
## 61 2000 2076969 1981845 1.047998
## 62 2001 2057922 1968011 1.045686
## 63 2002 2057979 1963747 1.047986
```

```r
# used match to isolate the index of the highest proportion of
# boys to girls, then found the year associated with this highest
# proportion. 1946 had the highest proportion of boys to girls
# born in the U.S. in this data set.
present$year[match(max(present$boyprop), present$boyprop)]
```

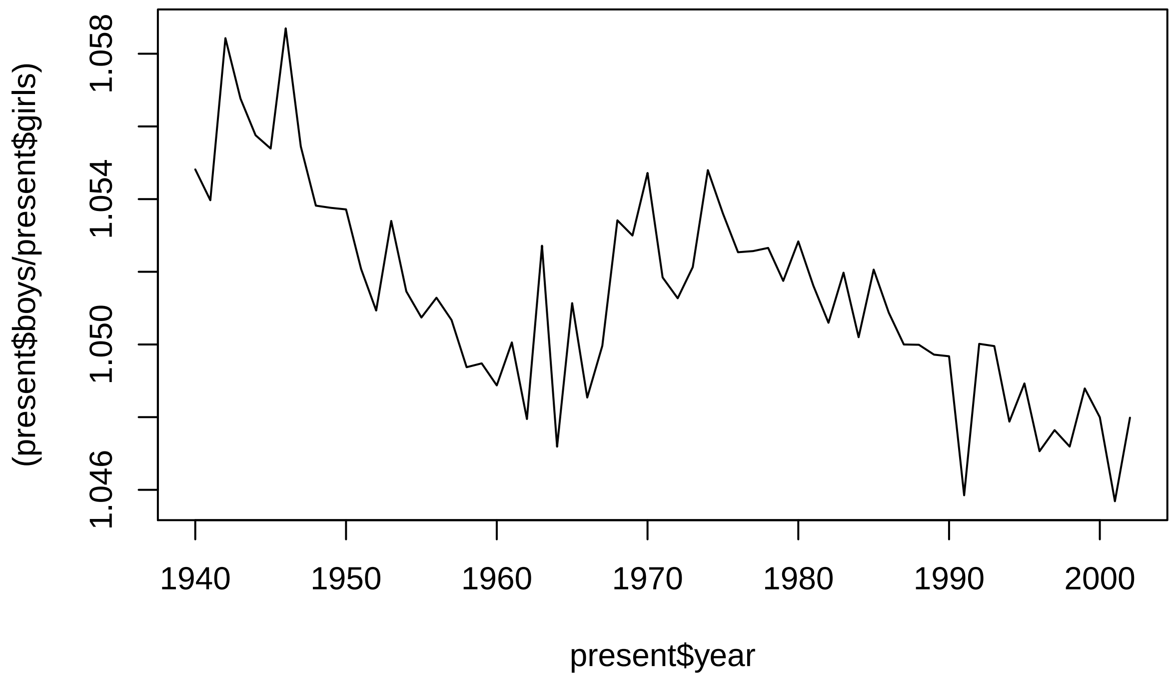
```
## [1] 1946
```

```r
present$year[match(min(present$boyprop), present$boyprop)]
```

```
## [1] 2001
```

We can see that within this data set, even the most equal ratio of boys to girls born in the U.S. was in the year of 2001 - where there were still roughly 104 boys born for every 100 girls born. Although Arbuthnot's observation is supported in this dataset, we do see a slight decline in the ratio of boys born to the girls born in the U.S. during this time period.

```r
plot(present$year, (present$boys / present$girls), type = "l")
```



- 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.

1961 was the year that we saw the most total number of births in the U.S. between 1940 and 2002.

```
# create column for total births by year
present$total_births <- present$boys + present$girls
present
```

```
##    year    boys   girls  boyprop total_births
## 1  1940 1211684 1148715 1.054817      2360399
## 2  1941 1289734 1223693 1.053969      2513427
## 3  1942 1444365 1364631 1.058429      2808996
## 4  1943 1508959 1427901 1.056767      2936860
## 5  1944 1435301 1359499 1.055757      2794800
## 6  1945 1404587 1330869 1.055391      2735456
## 7  1946 1691220 1597452 1.058698      3288672
## 8  1947 1899876 1800064 1.055449      3699940
## 9  1948 1813852 1721216 1.053820      3535068
## 10 1949 1826352 1733177 1.053760      3559529
## 11 1950 1823555 1730594 1.053716      3554149
## 12 1951 1923020 1827830 1.052078      3750850
## 13 1952 1971262 1875724 1.050934      3846986
```

```
## 14 1953 2001798 1900322 1.053399      3902120
## 15 1954 2059068 1958294 1.051460      4017362
## 16 1955 2073719 1973576 1.050742      4047295
## 17 1956 2133588 2029502 1.051286      4163090
## 18 1957 2179960 2074824 1.050672      4254784
## 19 1958 2152546 2051266 1.049374      4203812
## 20 1959 2173638 2071158 1.049480      4244796
## 21 1960 2179708 2078142 1.048873      4257850
## 22 1961 2186274 2082052 1.050057      4268326
## 23 1962 2132466 2034896 1.047948      4167362
## 24 1963 2101632 1996388 1.052717      4098020
## 25 1964 2060162 1967328 1.047188      4027490
## 26 1965 1927054 1833304 1.051137      3760358
## 27 1966 1845862 1760412 1.048540      3606274
## 28 1967 1803388 1717571 1.049964      3520959
## 29 1968 1796326 1705238 1.053417      3501564
## 30 1969 1846572 1753634 1.052997      3600206
## 31 1970 1915378 1816008 1.054719      3731386
## 32 1971 1822910 1733060 1.051845      3555970
## 33 1972 1669927 1588484 1.051271      3258411
## 34 1973 1608326 1528639 1.052129      3136965
## 35 1974 1622114 1537844 1.054797      3159958
## 36 1975 1613135 1531063 1.053605      3144198
## 37 1976 1624436 1543352 1.052538      3167788
## 38 1977 1705916 1620716 1.052569      3326632
## 39 1978 1709394 1623885 1.052657      3333279
## 40 1979 1791267 1703131 1.051749      3494398
## 41 1980 1852616 1759642 1.052837      3612258
## 42 1981 1860272 1768966 1.051615      3629238
## 43 1982 1885676 1794861 1.050597      3680537
## 44 1983 1865553 1773380 1.051976      3638933
## 45 1984 1879490 1789651 1.050199      3669141
## 46 1985 1927983 1832578 1.052061      3760561
## 47 1986 1924868 1831679 1.050876      3756547
## 48 1987 1951153 1858241 1.050000      3809394
## 49 1988 2002424 1907086 1.049991      3909510
## 50 1989 2069490 1971468 1.049720      4040958
## 51 1990 2129495 2028717 1.049676      4158212
## 52 1991 2101518 2009389 1.045849      4110907
## 53 1992 2082097 1982917 1.050017      4065014
## 54 1993 2048861 1951379 1.049955      4000240
## 55 1994 2022589 1930178 1.047877      3952767
## 56 1995 1996355 1903234 1.048928      3899589
## 57 1996 1990480 1901014 1.047062      3891494
## 58 1997 1985596 1895298 1.047643      3880894
## 59 1998 2016205 1925348 1.047190      3941553
## 60 1999 2026854 1932563 1.048791      3959417
## 61 2000 2076969 1981845 1.047998      4058814
## 62 2001 2057922 1968011 1.045686      4025933
## 63 2002 2057979 1963747 1.047986      4021726
```

```r
# used match and max again to isolate the year that had the most total
# number of briths in the U.S.
present$year[match(max(present$total_births), present$total_births)]
```

```
## [1] 1961
```

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