# Lab1 : Introduction to R and RStudio

Ramnivas Singh

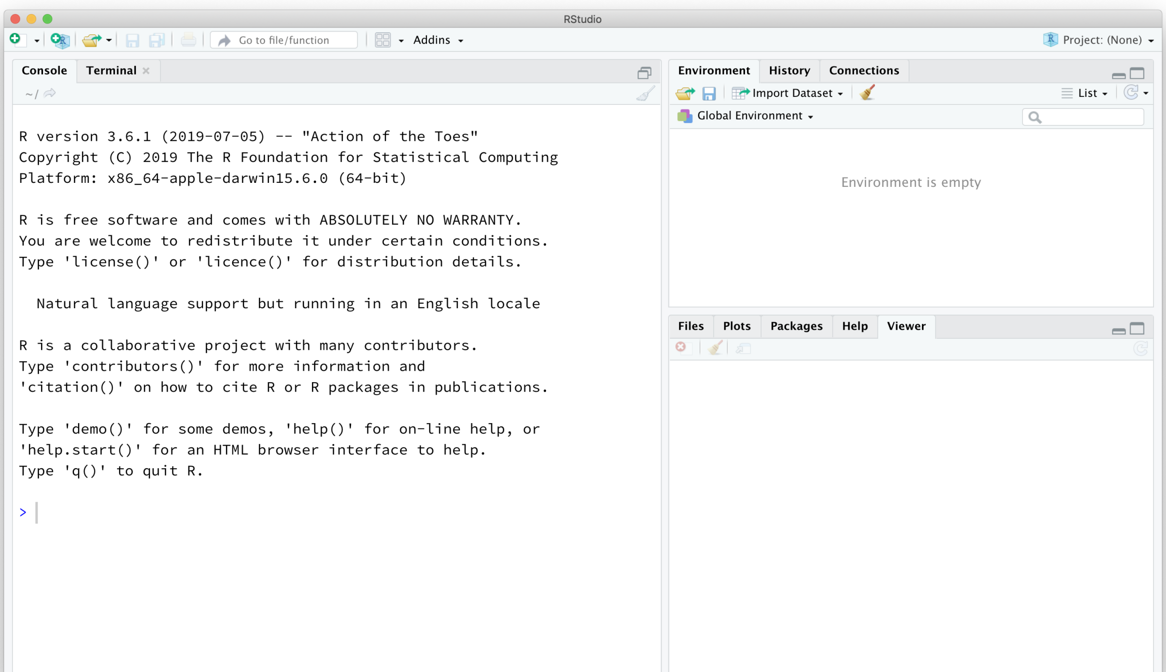
02/07/2021

## The RStudio Interface

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 course and to analyze real data and come to informed conclusions. To clarify 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 launch RStudio. You should see a window that looks like the image shown below.




The panel on the lower 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 informationis 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.

The panel in the upper right contains your *environment* 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. This is also where you can browse your files, access help, manage packages, etc.

### 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:

- The suite of **tidyverse** packages: for data wrangling and data visualization
- **openintro**: for data and custom functions with the OpenIntro resources

If these packages are not already available in your R environment, install them by typing the following three lines of code into the console of your RStudio session, pressing the enter/return key after each one. Note that you can check to see which packages (and which versions) are installed by inspecting the *Packages* tab in the lower right panel of RStudio.

```
install.packages("tidyverse")
install.packages("openintro")
```

You may need to select a server from which to download; any of them will work. Next, you need to load these packages in your working environment. We do this with the `library` function. Run the following three lines in your console.

```
library(tidyverse)
library(openintro)
```

You only need to *install* packages once, but you need to *load* them each time you relaunch RStudio.

The Tidyverse packages share common philosophies and are designed to work together. You can find more about the packages in the tidyverse at tidyverse.org.

### Creating a reproducible lab report

We will be using R Markdown to create reproducible lab reports. See the following videos describing why and how:

**Why use R Markdown for Lab Reports?**

**Using R Markdown for Lab Reports in RStudio**

In a nuthshell, in RStudio, go to New File -> R Markdown… Then, choose From Template and then choose `Lab Report for OpenIntro Statistics Lab 1` from the list of templates.

Going forward you should refrain from typing your code directly in the console, and instead type any code (final correct answer, or anything you're just trying out) in the R Markdown file and run the chunk using either the Run button on the chunk (green sideways triangle) or by highlighting the code and clicking Run on the top right corner of the R Markdown editor. If at any point you need to start over, you can Run All Chunks above the chunk you're working in by clicking on the down arrow in the code chunk.

## Dr. Arbuthnot's Baptism Records

To get started, let's take a peek at the data.

```
source('more/arbuthnot.r')
```

You can run the command by

- clicking on the green arrow at the top right of the code chunk in the R Markdown (Rmd) file, or
- putting your cursor on this line, and clicking the **Run** button on the upper right corner of the pane, or
- holding `Ctrl-Shift-Enter`, or
- typing the code in the console.

This command instructs R to load some data: the Arbuthnot baptism counts for boys and girls. You should see that the environment 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 the work of 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. Once again, we can view the data by typing its name into the console.

```
arbuthnot
```

```
##    year boys girls
## 1  1629 5218  4683
## 2  1630 4858  4457
## 3  1631 4422  4102
## 4  1632 4994  4590
## 5  1633 5158  4839
## 6  1634 5035  4820
## 7  1635 5106  4928
## 8  1636 4917  4605
## 9  1637 4703  4457
## 10 1638 5359  4952
## 11 1639 5366  4784
## 12 1640 5518  5332
## 13 1641 5470  5200
## 14 1642 5460  4910
## 15 1643 4793  4617
## 16 1644 4107  3997
## 17 1645 4047  3919
## 18 1646 3768  3395
## 19 1647 3796  3536
## 20 1648 3363  3181
## 21 1649 3079  2746
## 22 1650 2890  2722
## 23 1651 3231  2840
## 24 1652 3220  2908
## 25 1653 3196  2959
```

```
## 26 1654 3441  3179
## 27 1655 3655  3349
## 28 1656 3668  3382
## 29 1657 3396  3289
## 30 1658 3157  3013
## 31 1659 3209  2781
## 32 1660 3724  3247
## 33 1661 4748  4107
## 34 1662 5216  4803
## 35 1663 5411  4881
## 36 1664 6041  5681
## 37 1665 5114  4858
## 38 1666 4678  4319
## 39 1667 5616  5322
## 40 1668 6073  5560
## 41 1669 6506  5829
## 42 1670 6278  5719
## 43 1671 6449  6061
## 44 1672 6443  6120
## 45 1673 6073  5822
## 46 1674 6113  5738
## 47 1675 6058  5717
## 48 1676 6552  5847
## 49 1677 6423  6203
## 50 1678 6568  6033
## 51 1679 6247  6041
## 52 1680 6548  6299
## 53 1681 6822  6533
## 54 1682 6909  6744
## 55 1683 7577  7158
## 56 1684 7575  7127
## 57 1685 7484  7246
## 58 1686 7575  7119
## 59 1687 7737  7214
## 60 1688 7487  7101
## 61 1689 7604  7167
## 62 1690 7909  7302
## 63 1691 7662  7392
## 64 1692 7602  7316
## 65 1693 7676  7483
## 66 1694 6985  6647
## 67 1695 7263  6713
## 68 1696 7632  7229
## 69 1697 8062  7767
## 70 1698 8426  7626
## 71 1699 7911  7452
## 72 1700 7578  7061
## 73 1701 8102  7514
## 74 1702 8031  7656
## 75 1703 7765  7683
## 76 1704 6113  5738
## 77 1705 8366  7779
## 78 1706 7952  7417
## 79 1707 8379  7687
```

```
## 80 1708 8239  7623
## 81 1709 7840  7380
## 82 1710 7640  7288
```

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 environment. 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 as well as the names of the variables and the first few observations by typing:

```
glimpse(arbuthnot)
```

```
## Rows: 82
## Columns: 3
## $ year  <int> 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1...
## $ boys  <int> 5218, 4858, 4422, 4994, 5158, 5035, 5106, 4917, 4703, 5359, 5...
## $ girls <int> 4683, 4457, 4102, 4590, 4839, 4820, 4928, 4605, 4457, 4952, 4...
```

It is better practice to type this command into your console, since it is not necessary code to include in your solution file.

This command should output the following

Rows: 82 Columns: 3 $ year 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1… $ boys 5218, 4858, 4422, 4994, 5158, 5035, 5106, 4917, 4703, 5359, 5… $ girls 4683, 4457, 4102, 4590, 4839, 4820, 4928, 4605, 4457, 4952, 4…

We can see that there are 82 observations and 3 variables in this dataset. The variable names are `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 `glimpse` command, for example, 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
```

```
##  [1] 5218 4858 4422 4994 5158 5035 5106 4917 4703 5359 5366 5518 5470 5460 4793
## [16] 4107 4047 3768 3796 3363 3079 2890 3231 3220 3196 3441 3655 3668 3396 3157
```

```
## [31] 3209 3724 4748 5216 5411 6041 5114 4678 5616 6073 6506 6278 6449 6443 6073
## [46] 6113 6058 6552 6423 6568 6247 6548 6822 6909 7577 7575 7484 7575 7737 7487
## [61] 7604 7909 7662 7602 7676 6985 7263 7632 8062 8426 7911 7578 8102 8031 7765
## [76] 6113 8366 7952 8379 8239 7840 7640
```

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!

```
arbuthnot$girls
```

```
##  [1] 4683 4457 4102 4590 4839 4820 4928 4605 4457 4952 4784 5332 5200 4910 4617
## [16] 3997 3919 3395 3536 3181 2746 2722 2840 2908 2959 3179 3349 3382 3289 3013
## [31] 2781 3247 4107 4803 4881 5681 4858 4319 5322 5560 5829 5719 6061 6120 5822
## [46] 5738 5717 5847 6203 6033 6041 6299 6533 6744 7158 7127 7246 7119 7214 7101
## [61] 7167 7302 7392 7316 7483 6647 6713 7229 7767 7626 7452 7061 7514 7656 7683
## [76] 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.

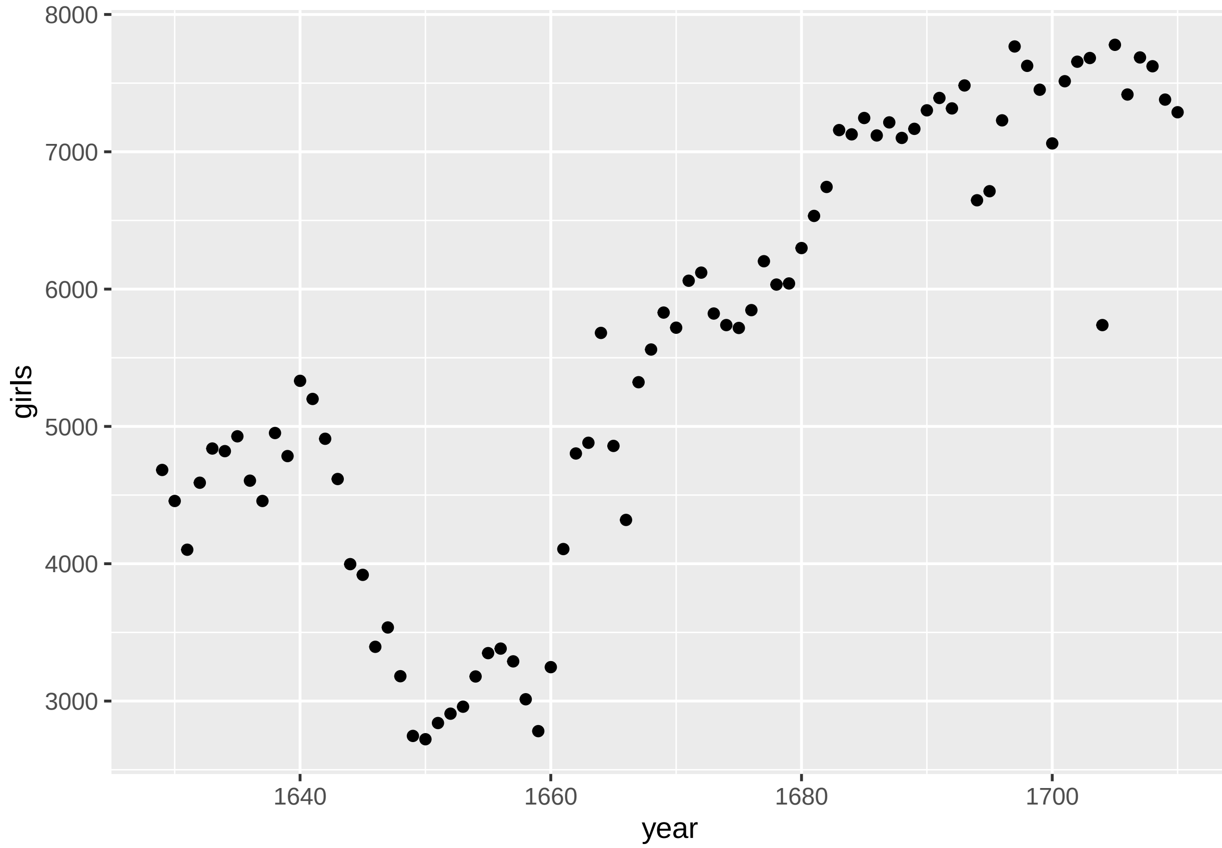
## Adding total to the data frame

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

### Data visualization

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

```
ggplot(data = arbuthnot, aes(x = year, y = girls)) +
  geom_point()
```



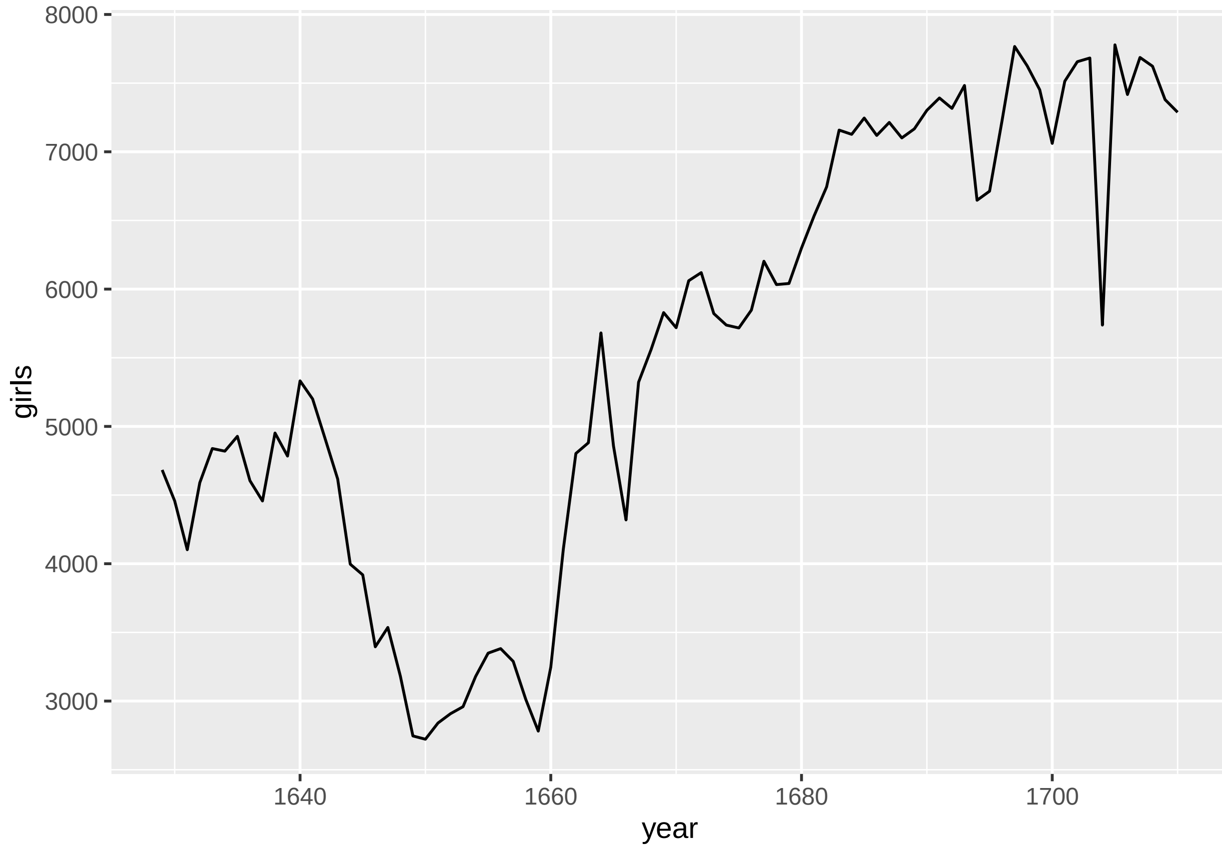

We use the `ggplot()` function to build plots. If you run the plotting code in your console, you should see the plot appear under the *Plots* tab of the lower right panel of RStudio. Notice that the command above again looks like a function, this time with arguments separated by commas.

With `ggplot()`:

- The first argument is always the dataset.
- Next, you provide the variables from the dataset to be assigned to `aes`thetic elements of the plot, e.g. the x and the y axes.
- Finally, you use another layer, separated by a `+` to specify the `geom`etric object for the plot. Since we want to scatterplot, we use `geom_point()`.

For instance, if you wanted to visualize the above plot using a line graph, you would replace `geom_point()` with `geom_line()`.

```
ggplot(data = arbuthnot, aes(x = year, y = girls)) +
  geom_line()
```



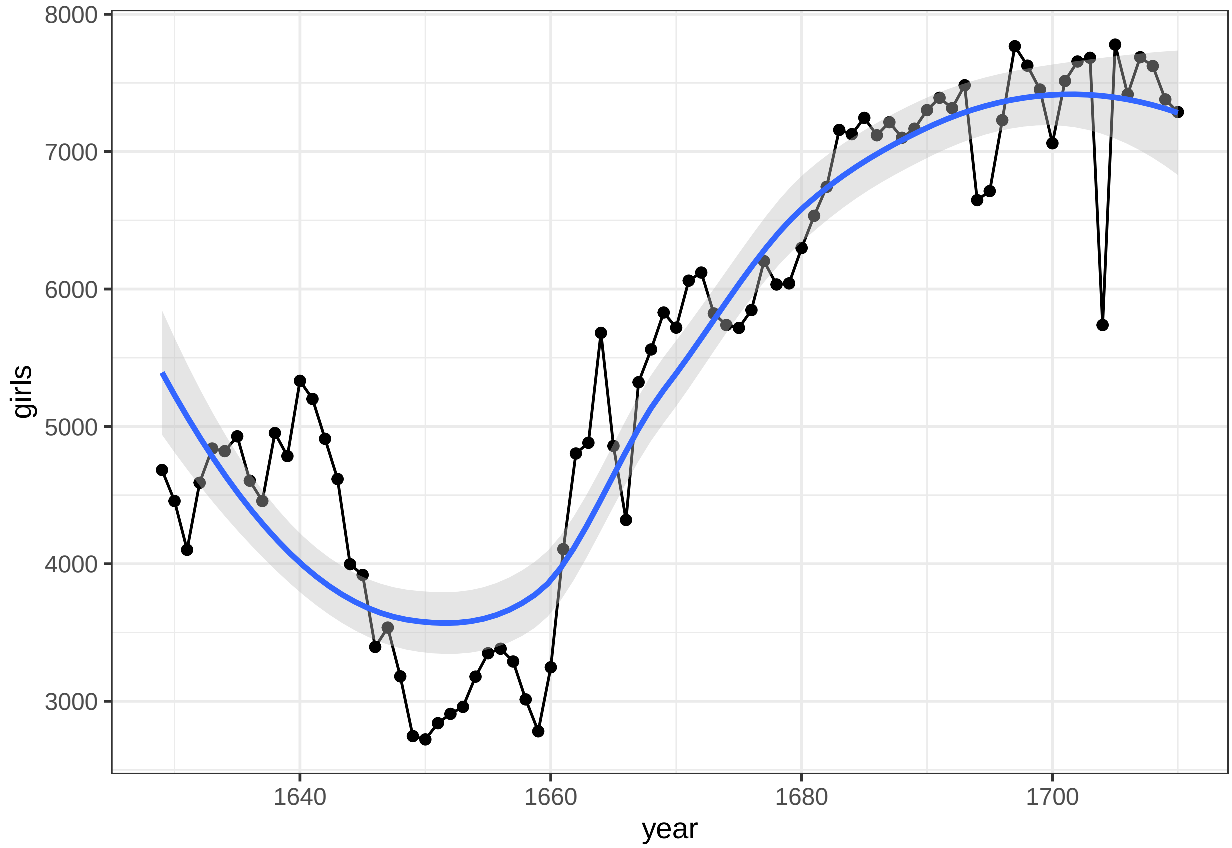

You might wonder how you are supposed to know the syntax for the `ggplot` function. Thankfully, R documents all of its functions extensively. To learn what a function does and its arguments that are available to you, just type in a question mark followed by the name of the function that you're interested in. Try the following in your console:

```
#?ggplot
```

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? (To ensure that your lab report is comprehensive, be sure to include the code needed to make the plot as well as your written interpretation.)

```
ggplot(arbuthnot, aes(x=year,y=girls))+geom_point()+
  geom_line()+stat_smooth(fill="grey")+theme_bw()
```




Interpretation:
1. Trend was decreased to baptize the girls between 1640 and 1660
2. Between 1660 and 1680, trend to baptize the girls was increased
3. Between 1680 and 1700, trend was increased but slow
4. Overall trend to baptize the girls was increased

**R as a big calculator**

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 to that of girls, R will compute all sums simultaneously.

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

```
##  [1]  9901  9315  8524  9584  9997  9855 10034  9522  9160 10311 10150 10850
## [13] 10670 10370  9410  8104  7966  7163  7332  6544  5825  5612  6071  6128
## [25]  6155  6620  7004  7050  6685  6170  5990  6971  8855 10019 10292 11722
## [37]  9972  8997 10938 11633 12335 11997 12510 12563 11895 11851 11775 12399
## [49] 12626 12601 12288 12847 13355 13653 14735 14702 14730 14694 14951 14588
```

```
## [61] 14771 15211 15054 14918 15159 13632 13976 14861 15829 16052 15363 14639
## [73] 15616 15687 15448 11851 16145 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.

### Adding a new variable to the data frame

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)
```

The `%>%` operator is called the **piping** operator. It takes the output of the previous expression and pipes it into the first argument of the function in the following one. To continue our analogy with mathematical functions, `x %>% f(y)` is equivalent to `f(x, y)`.

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

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

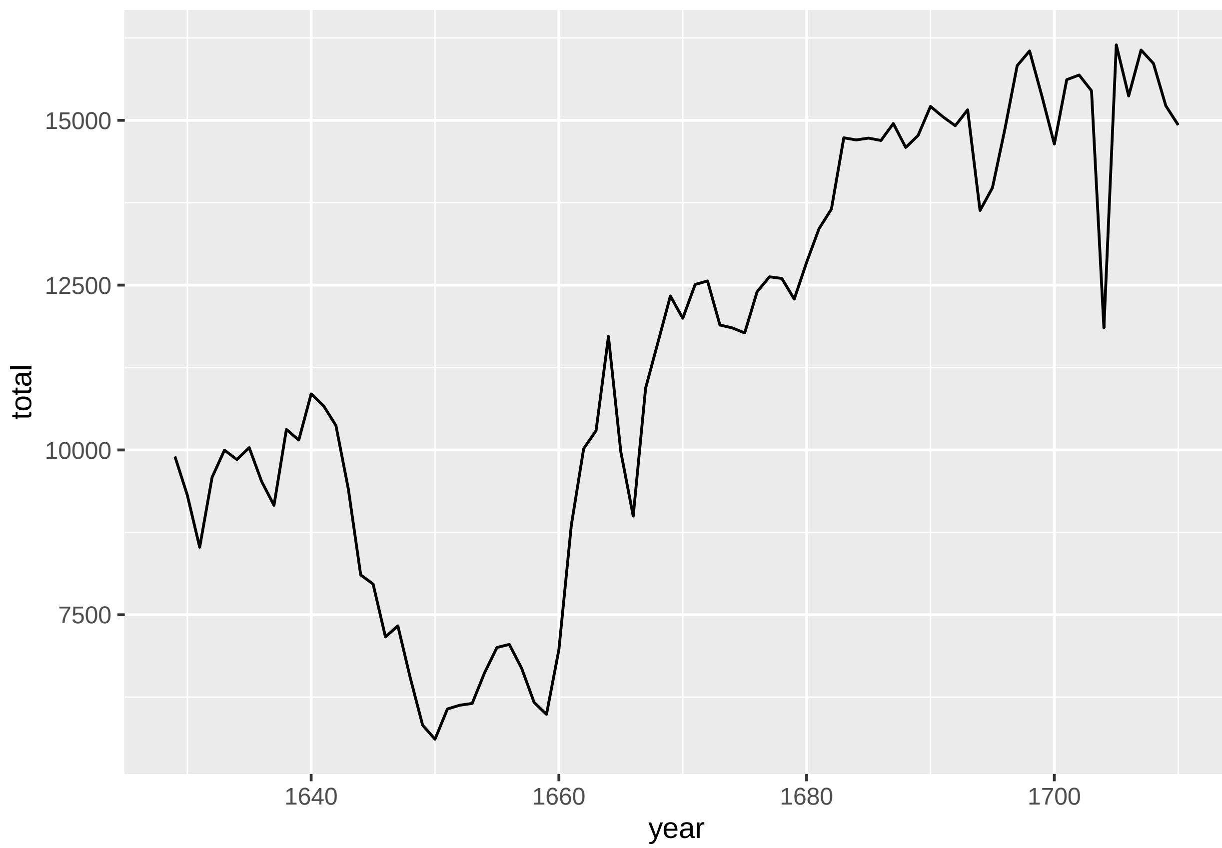
This is equivalent to going through each row and adding up the `boys` and `girls` counts for that year and recording that value in a new column called `total`.

**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 onto 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 environment. In this case, you already have an object called `arbuthnot`, so this command updatesthat data set with the new mutated column.

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

```
ggplot(data = arbuthnot, aes(x = year, y = total)) +
  geom_line()
```




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

```
5218 / 4683
```

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

or you can act on the complete columns with the expression

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

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

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

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

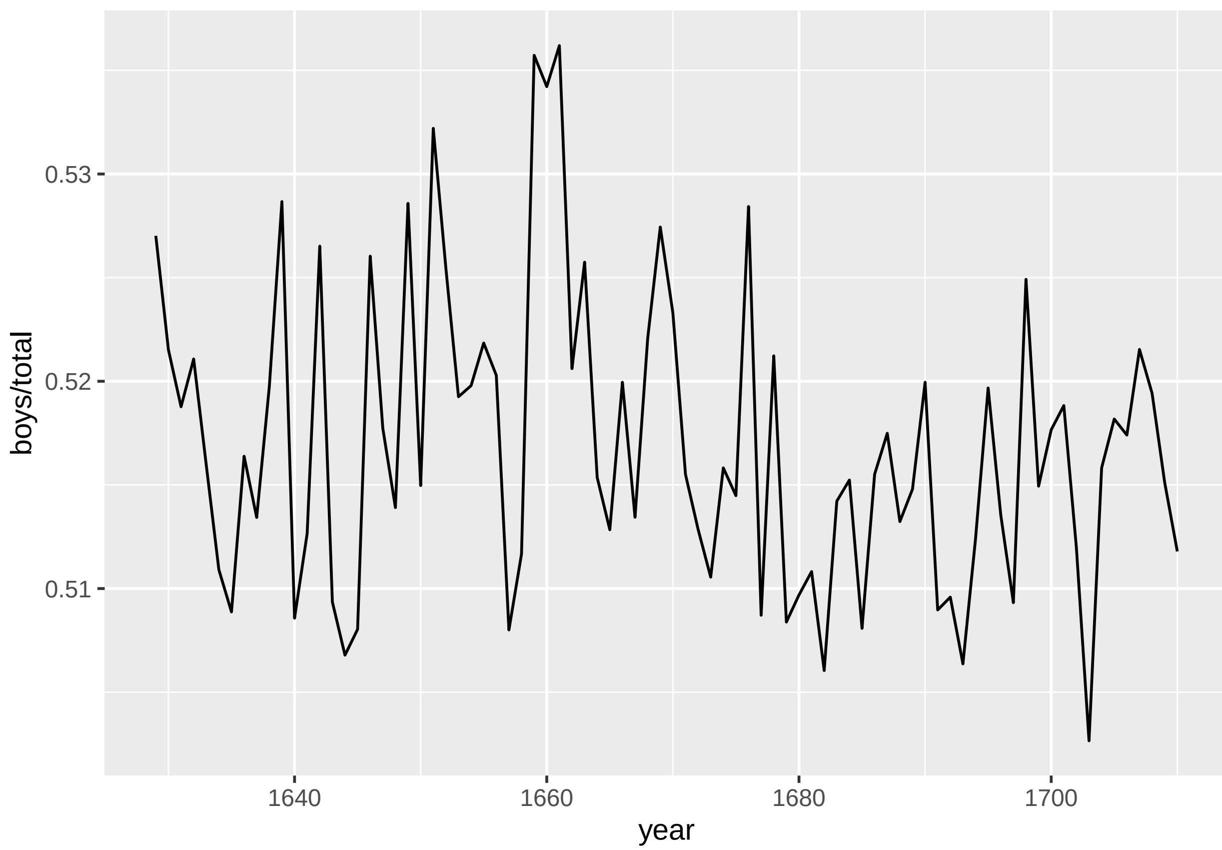
or you can compute this for all years simultaneously and append it to the dataset

```
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?

```
ggplot(data = arbuthnot, aes(x = year, y = boys/total)) + geom_line()
```




**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 the number of births of boys outnumber that of girls in each year with the expression

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

This command adds 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 a different kind of data than we have encountered so far. All other columns in the `arbuthnot` data frame have values that 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.

## More Practice

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

To find the minimum and maximum values of columns, you can use the functions `min` and `max` within a `summarize()` call, which you will learn more about in the following lab. Here's an example of how to find the minimum and maximum amount of boy births in a year:

```
arbuthnot %>%
  summarize(min = min(boys), max = max(boys))
```

```
##    min  max
## 1 2890 8426
```

```
source('more/present.r')
```

1. What years are included in this data set? What are the dimensions of the data frame? What are the variable (column) names?

```
unique(present[c("year")])
```

```
##    year
## 1  1940
## 2  1941
## 3  1942
## 4  1943
## 5  1944
## 6  1945
## 7  1946
## 8  1947
## 9  1948
## 10 1949
## 11 1950
## 12 1951
## 13 1952
## 14 1953
## 15 1954
## 16 1955
## 17 1956
## 18 1957
## 19 1958
## 20 1959
## 21 1960
## 22 1961
## 23 1962
## 24 1963
## 25 1964
## 26 1965
## 27 1966
## 28 1967
```

```
## 29 1968
## 30 1969
## 31 1970
## 32 1971
## 33 1972
## 34 1973
## 35 1974
## 36 1975
## 37 1976
## 38 1977
## 39 1978
## 40 1979
## 41 1980
## 42 1981
## 43 1982
## 44 1983
## 45 1984
## 46 1985
## 47 1986
## 48 1987
## 49 1988
## 50 1989
## 51 1990
## 52 1991
## 53 1992
## 54 1993
## 55 1994
## 56 1995
## 57 1996
## 58 1997
## 59 1998
## 60 1999
## 61 2000
## 62 2001
## 63 2002
```

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

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

```
dim(present)
```

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

```
names(present)
```

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

1. How do these counts compare to Arbuthnot's? Are they of a similar magnitude?

```r
summary(present$boys)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1211684 1799857 1924868 1885600 2058524 2186274
```

```r
summary(arbuthnot$boys)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2890    4759    6073    5907    7576    8426
```

```r
summary(present$girls)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1148715 1711405 1831679 1793915 1965538 2082052
```

```r
summary(arbuthnot$girls)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2722    4457    5718    5535    7150    7779
```

They are of a different magnitude. Comparing the present data from year 1940 to 2002 and Artbuthnot's London birthrate datafrom 1629 to 1710, the scale is quite different:

```r
range(present$boys + present$girls) - range(arbuthnot$boys + arbuthnot$girls)
```

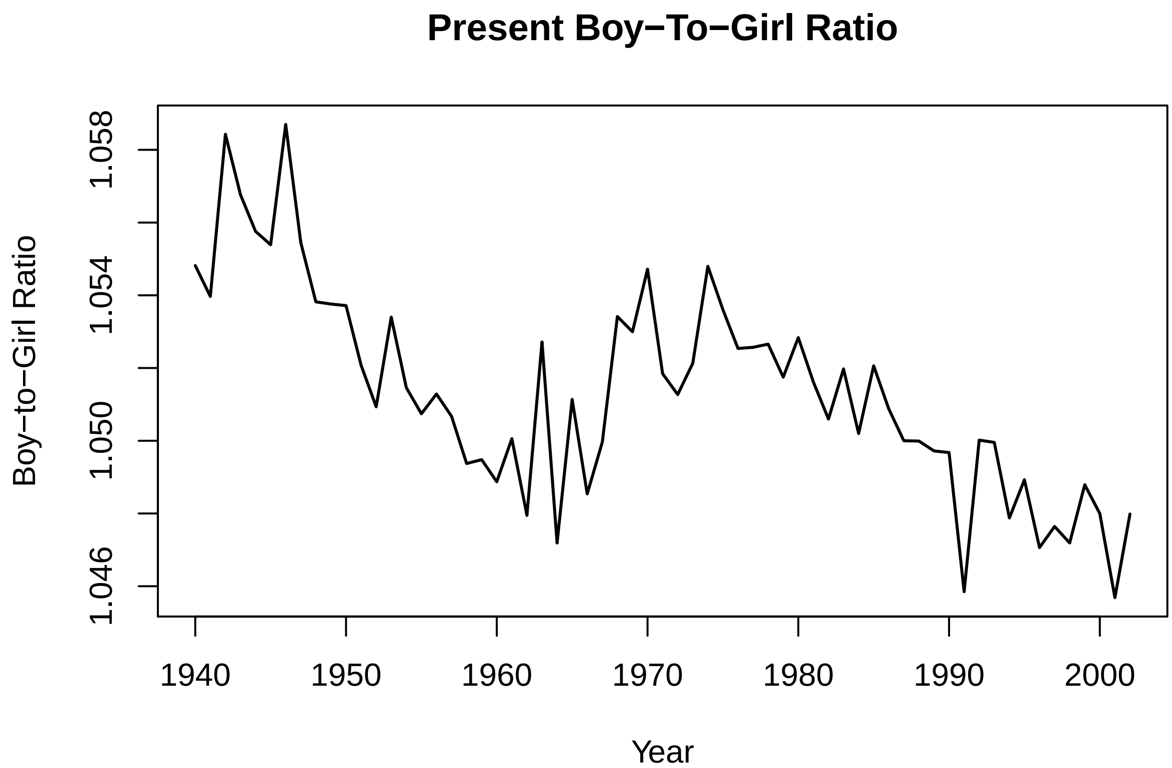
```
## [1] 2354787 4252181
```

1. 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 Exercise 3 above, just replace the dataframe name.
   Boy-to-girl ratio for every year with plot

```r
present$boys/present$girls
```

```
##  [1] 1.054817 1.053969 1.058429 1.056767 1.055757 1.055391 1.058698 1.055449
##  [9] 1.053820 1.053760 1.053716 1.052078 1.050934 1.053399 1.051460 1.050742
## [17] 1.051286 1.050672 1.049374 1.049480 1.048873 1.050057 1.047948 1.052717
## [25] 1.047188 1.051137 1.048540 1.049964 1.053417 1.052997 1.054719 1.051845
## [33] 1.051271 1.052129 1.054797 1.053605 1.052538 1.052569 1.052657 1.051749
## [41] 1.052837 1.051615 1.050597 1.051976 1.050199 1.052061 1.050876 1.050000
## [49] 1.049991 1.049720 1.049676 1.045849 1.050017 1.049955 1.047877 1.048928
## [57] 1.047062 1.047643 1.047190 1.048791 1.047998 1.045686 1.047986
```

```r
plot(x = present$year, y = present$boys /present$girls, type = "l", col = "black", lwd="1.5", main = "P
```

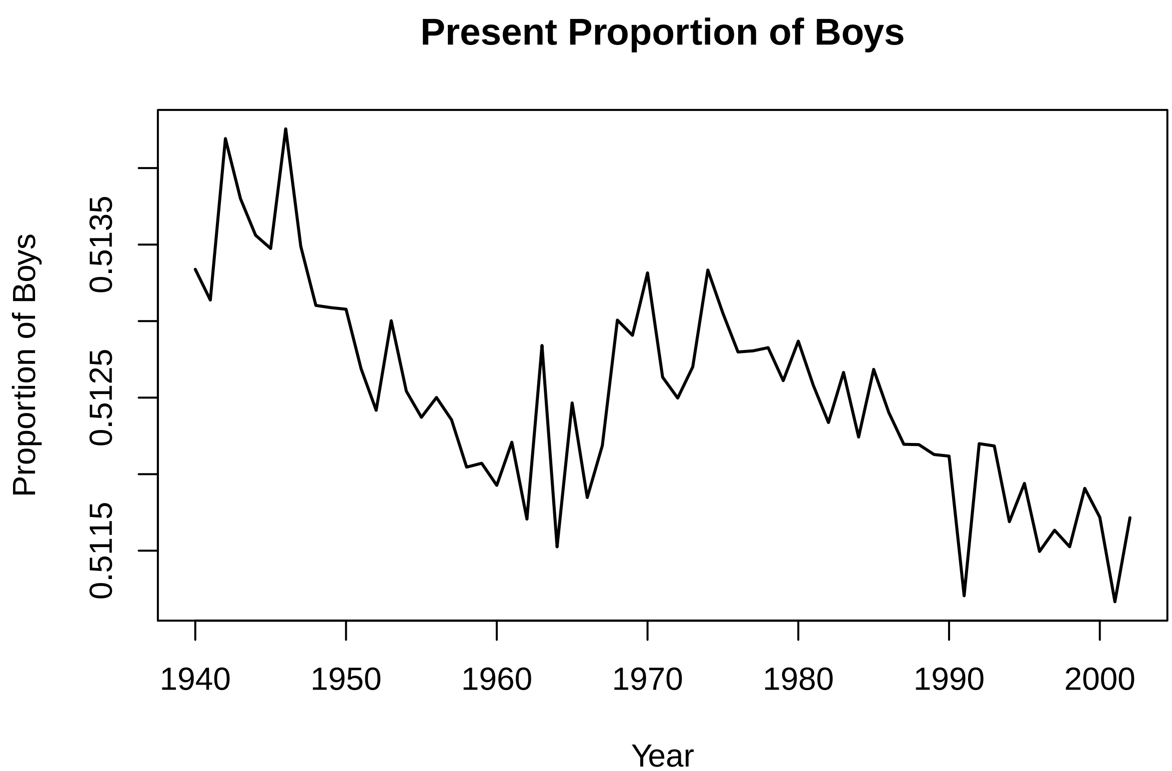



Boy to Girl proportion over the years

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

```
##  [1] 0.5133386 0.5131376 0.5141926 0.5138001 0.5135613 0.5134745 0.5142562
##  [8] 0.5134883 0.5131024 0.5130881 0.5130778 0.5126891 0.5124173 0.5130027
## [15] 0.5125423 0.5123716 0.5125011 0.5123550 0.5120462 0.5120713 0.5119269
## [22] 0.5122088 0.5117064 0.5128408 0.5115250 0.5124656 0.5118474 0.5121866
## [29] 0.5130068 0.5129073 0.5133154 0.5126337 0.5124973 0.5127013 0.5133340
## [36] 0.5130513 0.5127982 0.5128057 0.5128266 0.5126110 0.5128692 0.5125792
## [43] 0.5123372 0.5126648 0.5122425 0.5126849 0.5124035 0.5121951 0.5121931
## [50] 0.5121286 0.5121179 0.5112054 0.5121992 0.5121845 0.5116894 0.5119398
## [57] 0.5114951 0.5116337 0.5115255 0.5119072 0.5117182 0.5111665 0.5117154
```

```
plot(x = present$year, y = present$boys /(present$boys +present$girls), type = "l", col = "black", lwd=
```

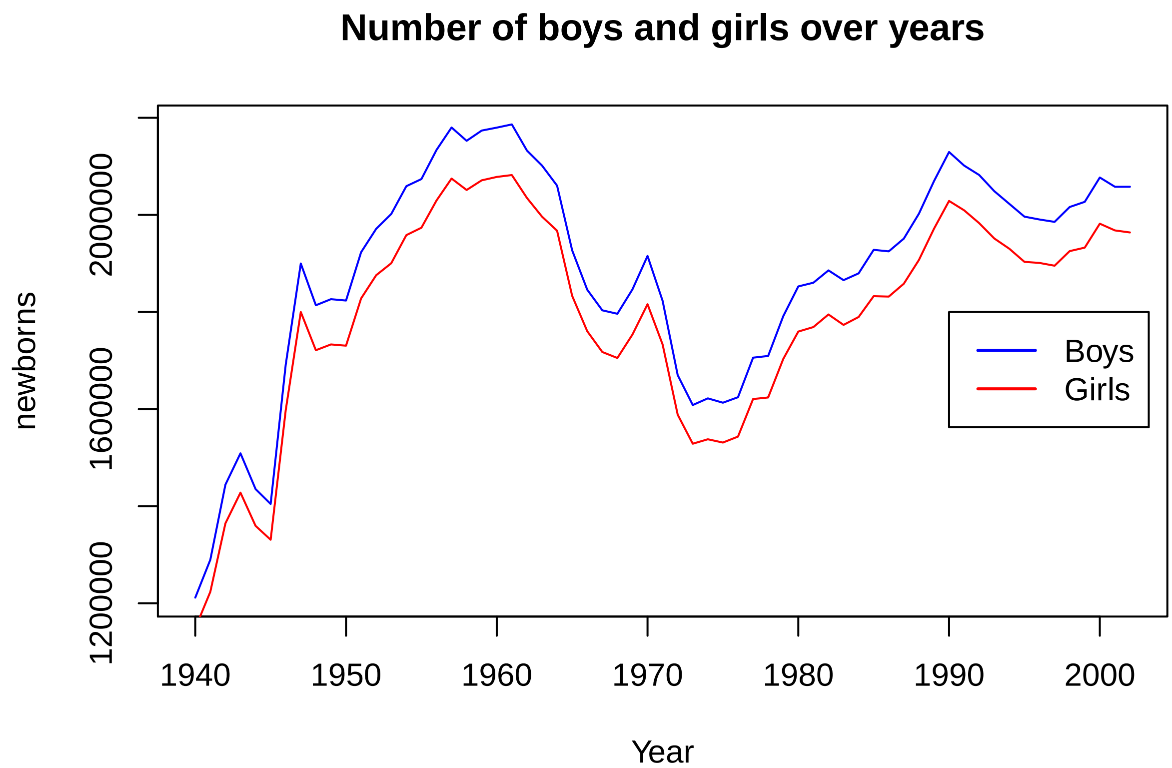



Based on the analysis above, Boy-to-girl ratio as well as proportion are decreasing over the time.Comparison shows that the Arbuthnot's observation about boys being born in greater proportion than girls holds up.

1. 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 descending order with another function: `desc` (for descending order). The sample code is provided below.

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

```
plot(x = present$year, y = present$boys, col="blue", type = "l",
     xlab="Year",ylab="newborns",main="Number of boys and girls over years")
lines(x = present$year, y = present$girls, col="red")
legend(1990, 1800000, c("Boys","Girls"), lty=c(1,1), lwd=c(1.5,1.5), col=c("blue","red"))
```




```
present$year[present$boys + present$girls == max(present$boys+present$girls)]
```

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

These data come from reports by the Centers for Disease Control. You can learn more about them by bringing up the help file using the command `?present`.

### 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 the suite of R packages from the **tidyverse**. The book R For Data Science by Grolemund and Wickham is a fantastic resource for data analysis in R with the tidyverse. If you are googling for R code, make sureto also include these package names in your search query. For example, instead of googling "scatterplot in R", google "scatterplot in R with the tidyverse".

These cheatsheets may come in handy throughout the semester:

- RMarkdown cheatsheet
- Data transformation cheatsheet
- Data visualization cheatsheet

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