Introduction to Probability and Statistics

2023/24

Stephen Connor

Table of contents

Overview

Welcome to IPS!

This web site is used to provide some of the course materials, and should be used alongside the module's Moodle page. All of the written assignment submission points can be found on Moodle, along with the quizzes for completion as you work through the computer labs.

You only *need* to use this site to access the computer lab material. However, you will also be able to access copies of the written assignments here in **html** format, in case you find that more accessible than the **pdf** files which will be available on Moodle.

> **Note**
>
> You can access the pdf version of any page of this site by clicking on the pdf icon in the left-hand menu. You can also choose to view the page in **dark mode**, if that's more comfortable.

Computer labs

The goal of these labs is to introduce you to, and build up your proficiency with, R and RStudio. You'll be using these throughout the course, both to learn the statistical concepts discussed in the lectures 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;
- RStudio is a convenient interface.

The R language is the standard statistical tool used by most statisticians at universities. One reason data scientists and statisticians like to use R is that all known statistical techniques are available in R. Whenever someone develops a new statistical technique, one of the first things they do is produce an R package so that the technique becomes available in R. The reason they do this for R rather than for one of the commercial alternatives is that R is open source and freely available to all, and of course that the previous methods on which the new method builds are already available in R.

Feeling comfortable using R is not only important for this module and any further statistics modules you may take at the Department of Mathematics of the University of York, it can also be an important factor for your future career (see the article "R skills attract the highest salaries". Even though R is specially designed for statistics, it is consistently in the list of the top ten most important programming languages compiled by the IEEE spectrum magazine.

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.

Assessment

> **!** Important
>
> The five main labs (imaginatively named "Lab 1" to "Lab 5") **count for credit**: your best 4 out of 5 will marks will count for 20% of the module mark.

Each lab will have an accompanying **Moodle quiz**. As you work through each lab you will find places where you are asked to perform a calculation and then enter your mark in the appropriate quiz.

> Warning
>
> The online quizzes will give you immediate feedback and allow you to try again if you get an answer wrong. However there will be a **20% deduction for each wrong**

attempt at a part of a question. You get to do the quizzes at your own time and can revisit them again and again during the period while they are open.

The **Intro lab** does *not* count for credit, but you should attempt this in the first week of the semester to make sure that:

- you can successfully access R
- you know how to enter answers in the accompanying Moodle quiz.

Schedule

(Each link will only work once the relevant lab has been released.)

Lab	Hand-out date	Quiz due date
Intro Lab (not for assessment)	Thursday 28 Sep (Week 1)	—
Lab 1	Thursday 5 Oct (Week 2)	Monday 9 Oct (Week 3)
Lab 2	Thursday 19 Oct (Week 4)	Monday 23 Nov (Week 5)
Lab 3	Thursday 9 Nov (Week 6)	Monday 13 Nov (Week 7)
Lab 4	Thursday 23 Nov (Week 8)	Monday 27 Nov (Week 9)
Lab 5	Thursday 7 Dec (Week 10)	Monday 11 Dec (Week 11)

Intro Lab: Meeting R and RStudio

> This tutorial is adapted from OpenIntro and is released under a Creative Commons Attribution-ShareAlike 3.0 Unported license. This lab was adapted for OpenIntro by Andrew Bray and Mine Çetinkaya-Rundel from a lab written by Mark Hansen of UCLA Statistics; it was extended for the University of York by Gustav Delius, and subsequently by Stephen Connor.

In this introduction we begin with the fundamental building blocks of R and RStudio: the interface, reading in data, and basic commands.

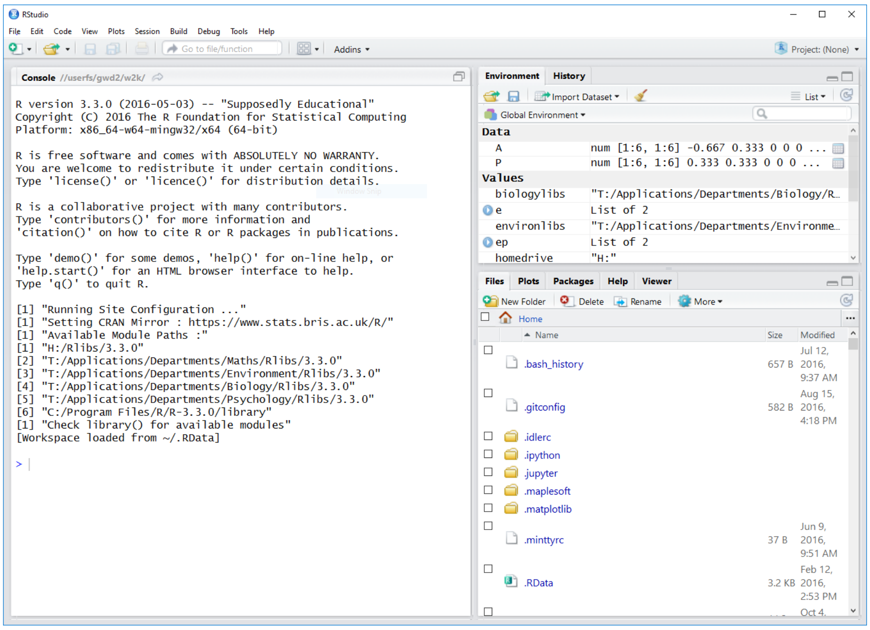
The first step is to open RStudio.

- If you are on a campus PC, RStudio is already installed and you can open it from the Windows Start menu. Just start typing 'RStudio' into the search box on the start menu and then click on RStudio when it shows up. (If you get a popup asking you whether you want to upgrade to a newer version of RStudio, simply click the "Ignore update" button.)
- If you would like to work on your own computer, you can download and install R from here and then download and install RStudio from here. Both are free and open-source and available for Windows, Mac and Linux.

Once you've opened RStudio, you should see a window similar to that depicted below.

A good way to work through these labs is is to have this file open on one half of your screen and RStudio on the other half. On a PC you can usually move a window to the left or right half of the screen by holding down the Windows key and pressing the left or right arrow key.

> **Note**
>
> You will see instructions to **Complete quiz questions** as you work thorugh this lab: remember that you should enter your answers in the **Quiz for Intro Lab** on Moodle.



The panel in the upper right of the RStudio window contains your *Environment* as well as a *History* of the commands that you've previously entered. The lower right panel has several tabs, including *Plots* where any plots that you generate will show up.

The panel on the left is where the action happens. It's called the *Console*. Every time you launch RStudio, it will have text at the top of the console giving lots of information that you can mostly ignore, including 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.

> **Tip**
>
> If you're using the html version of this document, then to copy the code you can simply hover your mouse over the box below: you should see a 'Copy to clipboard' symbol appear in the top right corner of the box – click on this, and then paste what you've copied into RStudio.

```r
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 environment area in the upper right hand 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 by-product of a computation or some analysis you have performed.

Note that because it is accessing data on the web, the above command will work in a computer lab, in the library, or at home; just as long as you have access to the internet.

The data: Dr. Arbuthnot's baptism records

The Arbuthnot data set was compiled by 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 and hitting Enter.

```r
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 baptised that year, respectively. Use the scroll bar on the right side of the console window to examine the complete data set.

> Tip
>
> A nice feature of RStudio is that it comes with a built-in data viewer. Click on the name `arbuthnot` in the upper right window that lists the objects in your environment. This will bring up an alternative display of the Arbuthnot counts in the upper left panel of the RStudio window.

Moving back to the console, if we only want to see the first few lines of the data set, we can type

```
head(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
```

Sometimes, as in this example, I'll show you the output of the commands when I run them on my computer, so that you can compare with what you get when you run the commands yourself: any line starting with #> corresponds to code output.

> **Tip**
>
> In the html version of this document, the word `head()` in the code block above is underlined (as is the command `source()` further up the page). Clicking on an R command which is underlined will take you to its online documentation, where you can read more about how to use it.

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)
#> [1] 82  3
```

This indicates 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 Environment tab. You can see the names of these columns (or variables) by typing:

```
names(arbuthnot)
#> [1] "year"  "boys"  "girls"
```

You should see that the data frame contains the columns `year`, `boys`, and `girls`. By this point, you might have noticed that many of the commands in R look a lot like functions; 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 baptised each year.

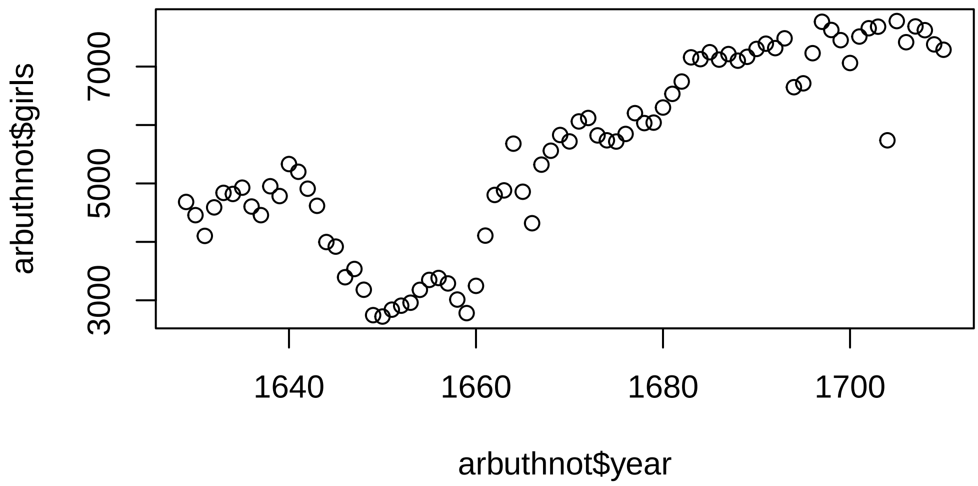
> Your turn
>
> What command would you use to extract just the counts of girls baptised each year? Try it!
> **Now answer quiz question 1.**

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 baptised per year with the command

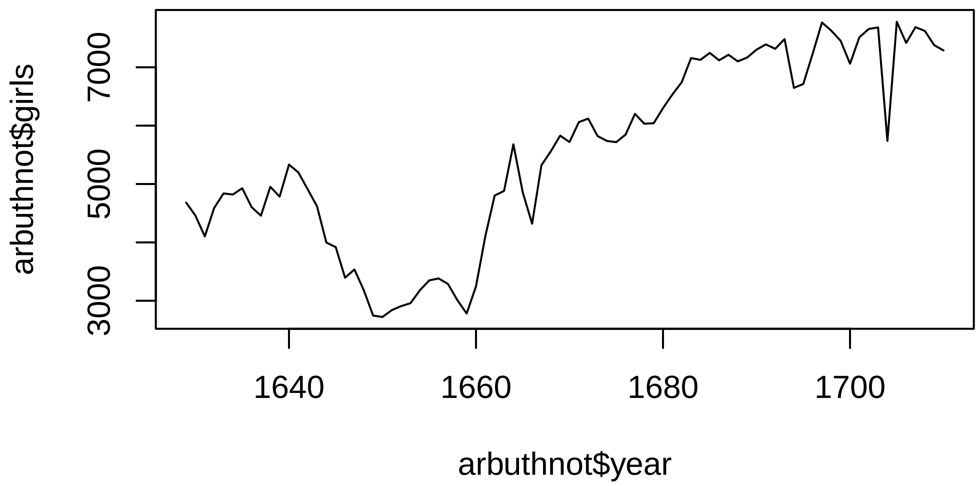
```
plot(x = arbuthnot$year, y = arbuthnot$girls)
```



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: you've already seen that clicking on any of the underlined commands in this page takes you to the relevant entry in the documentation. Another way to read what a function does, and learn the arguments that are available to you, is to just type in a question mark followed by the name of the function that you're interested in. Try the following.

```
?plot
```

Can you figure out how to produce a plot that shows both the points and the lines connecting them?

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.

> Your turn
>
> Is there an apparent trend in the number of girls baptised over the years?
> **Answer quiz question 2.**
> Can you also guess, just by looking at the graph, when the English civil war started?

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.

We can now 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.

Next we calculate the proportion of the baptised children that are boys. We can do this for the year 1629 with the command

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

but 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.

> Your turn
>
> Now, make a plot of the proportion of boys over time. The command for making the plot will be similar to the plot command you used earlier, just with a different expression for the y argument.
> **Now answer quiz question 3.**

> 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.

In addition to simple mathematical operators like subtraction and division, you can ask R to make comparisons like greater than, >, less than, <, and equality, == (note that it has to be a double equal sign, not a single equal sign). 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.

You can count the number of entries for which the condition is TRUE by just summing the entries in the vector

```
sum(arbuthnot$boys > arbuthnot$girls)
```

The reason this works is that R automatically converts TRUE to 1 and FALSE to 0 when asked to do a numerical calculation with these values.

> Your turn
>
> Above you have seen how to calculate the proportion of newborns that are boys.
> You have also learned how to count the number of entries in the data that satisfy a particular condition.
> **Now combine those two to answer quiz question 4.**

A newer data set

In the previous few pages, you recreated some of the displays and preliminary analysis of Arbuthnot's baptism data. To practise your new skills, you will now repeat 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`.

> Your turn
>
> 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?
> 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. What was the largest total number of births in a single year in the U.S. during the

period covered by the dataset? You can refer to the help files or the R reference card to find helpful commands.

Now answer questions 5 and 6 in the quiz.

These data come from a report by the Centers for Disease Control. Check it out if you would like to read more about an analysis of sex ratios at birth in the United States.

To exit RStudio you can click the cross 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.

Written assignments

Assignment sheets will appear on Moodle as **pdf** files. If you would prefer to view an **html** version then you can find links to these below. (Each link will only work once the relevant sheet has been released on Moodle.)

Assignment	Hand-out date	Due date	Solutions
Assignment 1	Tuesday 26 Sep (Week 1)	Thursday 5 Oct (Week 2)	—
Assignment 2	Tuesday 10 Oct (Week 3)	Thursday 19 Oct (Week 4)	—
Assignment 3	Tuesday 24 Oct (Week 5)	Thursday 9 Nov (Week 6)	—
Assignment 4	Tuesday 14 Nov (Week 7)	Thursday 23 Nov (Week 8)	—
Assignment 5	Tuesday 28 Nov (Week 9)	Thursday 7 Dec (Week 10)	—