

Installing R

To install R, go to <https://cloud.r-project.org> and choose your operating system, download and install.

Installing RStudio

I recommend that you use RStudio to work with R. RStudio is a free integrated development environment (IDE) for R. You can use RStudio to enter code, read in datasets, store your work and make plots – all within a single environment.

R and RStudio are installed separately.

To install RStudio, go to <https://posit.co/download/rstudio-desktop/> and download/install the version of RStudio for your operating system.

Using R as a calculator

Run RStudio. The RStudio window you get is normally divided into four: a window where you can write and edit code (initially labelled “Untitled1”, normally top left); the main console window where you can type code, or cut and paste it (bottom left); a window with environment/history tabs (top right); a window with files/plots/... tabs (bottom right), the plot tab is where your plots will appear.

Alternatively run R instead of RStudio.

You should have a “Console” window which contains a welcome message, and underneath the welcome message is a line like the following:

```
>
```

The > is the command prompt, indicating that R is ready for you to type a command. For example, typing 8+5 and then pressing the Enter key, you should see the following:

```
> 8+5
[1] 13
>
```

Don't type the > character, just type 8+5 then press Enter. R has computed 8+5. The last > indicates that R is ready for the next command. The first element of the answer is labelled [1] even when, as above, there is only one element.

In everything that follows, don't type the > character at the beginning of a line, R prints that to indicate that it is ready for new input from you.

To set up a vector x containing the elements 1.4, 3.5, 7.2, type the following, then press Enter.

```
x <- c(1.4, 3.5, 7.2)
```

The symbol <- (i.e. “less than”, followed by “minus”) is the assignment operator in R. The above sets x equal to the vector (1.4, 3.5, 7.2). To check this, type x then press Enter (this prints x) and you should see

```
> x
[1] 1.4 3.5 7.2
```

You can also use = instead of <- for assignment, so x = c(1.4, 3.5, 7.2) has the same effect.

To generate a vector of integers from 1 to 5, a shortcut is to use 1:5 instead of c(1, 2, 3, 4, 5).

```
y <- 1:5
```

The operation y + 10 adds 10 to each component of y, and similarly y / 8 divides each component of y by 8. Subtraction (e.g. y - 20) and multiplication (e.g. 4 * y) work similarly. For example

```
> y + 10
[1] 11 12 13 14 15
```

The above does not change the value of y . To define z to be $y + 10$ use

```
z <- y + 10
```

and to check the answer

```
> z
[1] 11 12 13 14 15
```

Plots and data

For a simple scatterplot of y against x we can use the following. Note `plot(x, y)` and `plot(y ~ x)` do the same thing.

```
x <- 1:5
y <- c(0, 4, 3, 7, 8)
plot(x, y)
plot(y ~ x)
```

The trees data (which is built-in to R) consists of three measurements – Girth, Height and Volume – for 31 trees. “Girth” is the tree diameter.

```
> trees
  Girth Height Volume
1   8.3    70  10.3
2   8.6    65  10.3
3   8.8    63  10.2
...
```

To see each measurement separately:

```
> trees$Height
[1] 70 65 63 72 81 ...
> trees$Volume
[1] 10.3 10.3 10.2 16.4 18.8 ...
```

How does the volume of wood in a tree depend on Height, or on Girth?

```
plot(Volume ~ Height, data=trees)
# this plot command is nicer than plot(trees$Volume ~ trees$Height)
# though both versions produce the same plot
```

```
plot(Volume ~ Girth, data=trees)
```

Probability

If $x = (x_1, \dots, x_n)$ then the *cumulative sum vector* is the vector $(x_1, x_1 + x_2, x_1 + x_2 + x_3, \dots, \sum_{i=1}^n x_i)$. Similarly $(x_1, x_1 x_2, x_1 x_2 x_3, \dots, \prod_{i=1}^n x_i)$ is the *cumulative product vector*. The R functions are `cumsum()` and `cumprod()`.

```
# random walk
x <- sample(c(1,-1), 100, replace=TRUE)
s <- cumsum(x)
plot(s, type="l")
```

```
# birthday problem
1 - cumprod((365:343)/365)
plot(1 - cumprod((365:266)/365), xlab="n", ylab="probability")
```

Python

You could use Python instead of R. However if you continue with statistics in Part A, then R will be recommended there (and not Python). So if you are doing Maths & Stats or possibly interested in doing more statistics next year, then I would recommend you try R.

We'll assume that calculations in Python are familiar from Computational Mathematics. But scatterplots of data, such as the trees data above, may not be. To try a couple of plots, first download a copy of the trees data from

<http://www.stats.ox.ac.uk/~laws/data/trees.txt>

and save it in a new folder. You'll need to change to that directory, then read in the data, look at it, and draw a couple of plots:

```
import os
path = [customise for your setting]

# set your working directory
os.chdir(os.path.join(path, folder))

# get your current working directory
os.getcwd()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#### Plots and data ####
trees = pd.read_table("trees.txt", delimiter = ",", index_col = 0)

trees
trees.head()
trees['Height']
trees['Volume']

# plot Volume vs Height
plt.scatter(trees['Height'], trees['Volume'])
plt.xlabel("Height")
plt.ylabel("Volume")
plt.show()

# plot Volume vs Girth
plt.scatter(trees['Girth'], trees['Volume'])
plt.xlabel("Girth")
plt.ylabel("Volume")
plt.show()

#### Probability ####

# random walk
x = np.random.choice([-1, 1], 100, replace = True)
s = np.cumsum(x)
plt.plot(s)
plt.show()

# birthday problem
1 - np.cumprod(np.linspace(365, 343, dtype = int) / 365)
plt.plot(1 - np.cumprod(np.linspace(365, 266, dtype = int) / 365))
plt.xlabel("n")
```

```
plt.ylabel("probability")  
plt.show()
```