Statistical Computing in R

Owen Ward

2020-10-17

Why do we need to use programming?

- Large data sets.
- Complicated procedures.
- Want to automate these procedures to update as we get new data.
- More reproducible.

What do people use

- Many types of software used for data and data science. Not all are suitable.

What do people use

- Many types of software used for data and data science. Not all are suitable.
- For example, using Excel for data management might be a bad idea. . .

What do people use

- Many types of software used for data and data science. Not all are suitable.
- For example, using Excel for data management might be a bad idea. . .
- The most suitable can depend a lot on what your goals are.

Programming in R

- We will use the R programming language throughout this class.

Programming in R

- We will use the R programming language throughout this class.
- Actively developed open source project.

Programming in R

- We will use the R programming language throughout this class.
- Actively developed open source project.
- Lots of brilliant free resources online.

Programming in R

- We will use the R programming language throughout this class.
- Actively developed open source project.
- Lots of brilliant free resources online.
- Relatively easy to set up (on all platforms).

Setting up RStudio Cloud

- Will aim to use Rstudio Cloud for examples in this class.

Setting up RStudio Cloud

- Will aim to use Rstudio Cloud for examples in this class.
- Works in the browser, don't need to install any software.

Setting up RStudio Cloud

- Will aim to use Rstudio Cloud for examples in this class.
- Works in the browser, don't need to install any software.
- Free!

Setting up RStudio Cloud

- Will aim to use Rstudio Cloud for examples in this class.
- Works in the browser, don't need to install any software.
- Free!
- Somewhat limited computing ability.

Set Up RStudio Cloud

Programming in R

The basic setup

- The basic setup consists of an R console and a blank R script.

The basic setup

- The basic setup consists of an R console and a blank R script.
- Can run code directly in the console, but not saved.

The basic setup

- The basic setup consists of an R console and a blank R script.
- Can run code directly in the console, but not saved.
- If you want to repeat code, write it in a script, save, and run from script.

The basics

- Possibly the simplest way to use R is as a calculator.

```
2 + 2
```

```
## [1] 4
```

```
1 + 3 * (4 * 2) + 10 / 3
```

```
## [1] 28.33333
```

```
pi/2
```

```
## [1] 1.570796
```

The basics

- Possibly the simplest way to use R is as a calculator.
- Everything works as you would expect it to.

```
2 + 2
```

```
## [1] 4
```

```
1 + 3 * (4 * 2) + 10 / 3
```

```
## [1] 28.33333
```

```
pi/2
```

```
## [1] 1.570796
```

Getting help

- R has lots of built functions and help files to understand how they work.

```
sqrt(2)
```

```
## [1] 1.414214
```

```
# ?sqrt
```

Getting help

- R has lots of built functions and help files to understand how they work.
- These are accessed by `?fun_name`

```
sqrt(2)
```

```
## [1] 1.414214
```

```
# ?sqrt
```

Getting help

- `R` has lots of built functions and help files to understand how they work.
- These are accessed by `?fun_name`
- If you want to write comments in R that won't be run, start that line with `#`.

```
sqrt(2)
```

```
## [1] 1.414214
```

```
# ?sqrt
```

Creating objects in R

- Use <- to create an object in R, or assign a new value to an existing object.

```
a <- 2
a * 3
```

```
## [1] 6
```

```
a <- -1
a + 1
```

```
## [1] 0
```

- The name of an object should be informative!

Understanding objects in R

- R has lots of built in object types. You can determine the type of an object using `typeof(obj)`.

```
a <- 1.3
typeof(a)
```

```
## [1] "double"
```

```
b <- TRUE
typeof(b)
```

```
## [1] "logical"
```

```
d <- "some text"
typeof(d)
```

```
## [1] "character"
```

Concatenating

You can combine objects in R using c(), creating a vector. This may make changes if the objects combined have different types.

```
x <- c(1, 2, 3) # we can also do this using c(1:3)
c(x,a)
```

```
## [1] 1.0 2.0 3.0 1.3
```

```
c(x,2)
```

```
## [1] 1 2 3 2
```

Concatenating

```
y <- c(x,b)
y
```

```
## [1] 1 2 3 1
```

```
typeof(y)
```

```
## [1] "double"
```

```
z <- c(x,d)
z
```

```
## [1] "1"         "2"         "3"         "some text"
```

```
typeof(z)
```

```
## [1] "character"
```

Data Types

- Data comes in lots of forms and R has many data types to account for this.
- Vectors, scalars and matrices useful for numeric data in particular.

```
a <- 1 ## scalar
x <- c(1.5, 2.5) ## vector
a * x
```

```
## [1] 1.5 2.5
```

Data Types

```
A <- matrix(c(1, 2, -1, 3), nrow = 2, byrow = TRUE)
A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]   -1    3
```

```
A %*% x
```

```
##      [,1]
## [1,]  6.5
## [2,]  6.0
```

Data Types

```
a*A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]   -1    3
```

```
y <- c(rep(0.5, 3))
y
```

```
## [1] 0.5 0.5 0.5
```

```
A %*% y
```

```
## Error in A %*% y: non-conformable arguments
```

Data Types

- Will see other data types, in particular `list`,`dataframe` and `tibble` when we begin to look at data.

Subsetting Data

We can easily access specific elements, such as the third entry in a vector or a specific element in a matrix.

```
x <- c(4, 5, 6)
x[2]  ## what is the index of the first element?
```

```
## [1] 5
```

```
A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]   -1    3
```

```
A[1, 2]
```

```
## [1] 2
```

More expressive data

Dataframes

- Data that we will analyse in this class will generally be in the form of a `dataframe`, a much more expressive format than what we've seen before.
- Generally a matrix of data, with each row consisting of one observation and each column in that matrix a different variable which is observed.
- For example, each row could be a person,location, etc, with each column being a different variable of interest for each row.

A first data frame

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1
##                   am gear carb
## Mazda RX4          1    4    4
## Mazda RX4 Wag      1    4    4
## Datsun 710         1    4    1
## Hornet 4 Drive     0    3    1
## Hornet Sportabout  0    3    2
## Valiant            0    3    1
```

What is in a data frame

- Actually a `list` object, has to have rectangular structure.
- Can easily view how many rows or columns it has.

```
nrow(mtcars)
```

```
## [1] 32
```

```
ncol(mtcars)
```

```
## [1] 11
```

```
dim(mtcars)
```

```
## [1] 32 11
```

```
length(mtcars) # this shows connection to lists
```

```
## [1] 11
```

More data frames

- Can access a specific column using the $

```
head(mtcars$mpg)
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

Packages

- Perhaps the most powerful feature of `R` is the thousands of packages available.
- Can easily be installed from an online repository (and also other places).
- We will use many different packages throughout this class.

```
install.packages("dplyr")
library(dplyr)
```