

A Gentle Introduction to R

Jonathan Whiteley


2023-08-06

Pop Quiz

We will review these *at the end*, so you can see how much you have learned.

- What does 'CRAN' stand for?
- Why is it named 'R'?
- How can you use R *interactively*?
- How do you find out what a function does & how to use it?
- How do you store values to re-use later?
- True or False: Warnings can be ignored, but an Error means I made a mistake.
- True or False: Error messages will tell me how to fix the problem.

Learning Objectives

- Get familiar with the  *interface*
- Use technical *terms* for R concepts
- Enter *commands*
 - ▶ input & output: using R interactively
 - ▶ use some common *functions*
- Understand *Errors & Warnings*
- How to get Help

Why is it named ?

- R started as an *open-source* implementation of the S statistical computing language (S-PLUS)
 - ▶ S was created at Bell Laboratories in 1976
 - ▶ R was based on the S syntax (mostly v3), but works very differently “under the hood”.
- R was created by Ross Ihaka and Robert Gentleman at the University of Auckland in the early 1990s.

The Interface

- ‘base R’ has a slightly different interface for each **O**perating **S**ystem (OS)
 - ▶ GUI = **G**raphical **U**ser **I**nterface
- R can also run inside of a terminal (no GUI) or other software (different GUI).

Integrated **D**evelopment **E**nvironment (IDE)

- An IDE is like an extra interface layer on top of ‘base R’
- IDEs often add convenient tools to make writing code easier (e.g., syntax highlighting), and for developing larger projects with multiple files.
- **RStudio** is one of the most popular cross-platform IDEs for R.
 - ▶ RStudio is available in open source (free/libre) and commercial^a editions.

^afor organizations not able to use software licensed with AGPL

A quick tour of 'base R'



A quick tour of RStudio

- Regardless of the interface, you interact with R primarily using a *command line*
- “Question-and-Answer Model”
 - ▶ You ask R to do something (a *command*), and R tells you the answer (*result*).
- Instructions are given to R using the *R language*.

The command-line

- The command *prompt* normally looks like this:

```
>
```

(the colour varies depending on the interface)

- ▶ This is R's way of saying "I am ready to accept new commands".
- ▶ Type a new command on the line after this prompt (i.e., *input*).
- Press **return/enter** to *run* the current *command*
- If you can still edit the command next to the prompt, then it has not been submitted to R to execute (it is still waiting for input).
- If the last prompt is not empty (i.e., there is text beside it) *and* you cannot edit what is beside the prompt, it means R is still running the last command and is not ready to accept a new command yet.
 - ▶ Wait for a new empty prompt to appear before entering the next command.

The command-line (cont'd)

- If the prompt looks like this:

```
+
```

it means the last command was *incomplete* and R is waiting for more input.

R will not do anything until the command is completed or cancelled.

- ▶ This usually means you forgot a closing
quote `"`, parenthesis `(`, bracket `[`, or brace `{`
- You can *cancel* the current command at any time by pressing escape
(`esc`)

In this presentation,

- *commands* that can be entered in the *command-line* look like this:

```
Input (commands)
```

- ▶ You can try these yourself!

- Expected output (results) look like this:

```
## Output (results)
```

`demo(graphics)`

- some plots and graphs that can be made with R

`demo(image)`

- image-like graphics and maps that can be produced with R

`demo(lm.glm)`

- a demonstration of linear modelling & GLMs

`demo()`

- a list of available demos

`help.start()`

- ← A great place to start, especially if you are comfortable reading documentation for a programming language. More on this later.
-

Note

R will not only show the output, but also *the code used to produce it*.

R is a show-off (alt)

`demo(graphics)`

`demo(image)`

`demo(lm.glm)`

`demo()`

`help.start()`

- some plots and graphs that can be made with R
- image-like graphics and maps that can be made with R
- a demonstration of linear modelling & GLMs
- a list of available demos

↑
A great place to start,
especially if you are
comfortable reading
documentation for a
programming language.
More on this later.

Note

R will not only show the output, but also
the code used to produce it.

```
1 + 1
```

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

```
2 * 2
```

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

```
2 ^ 3
```

```
## [1] 8
```

```
10 - 1
```

```
## [1] 9
```

```
8 / 2
```

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

```
sqrt(9)
```

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

- These are *expressions*
- *Expressions* are *evaluated*, and the *value* (result) is *returned* (sometimes *invisibly*)

- With the cursor next to the empty prompt (`>`), use the up & down **arrow keys** (`↑↓`) to re-produce previous commands
- This lets you “scroll through your *command history*”
- Press **up** (`↑`) once, and you get the last command you entered without having to copy & paste

Symbolic *variables*

- You can store values (*objects*) in symbolic variables (*names*) using an *assignment operator*

`<-` assign the *value* on the **right** to the *name* on the **left**

- Names can include:

letters	a-z A-Z
numbers	0-9
periods	.
underscores	_

```
A <- 10
B <- 10 * 10
A_log <- log(A)
B.seq <- 1:B

assign('x', 3)
```

- Names *should begin with a letter*

Retrieve values

When a variable *name* is evaluated, it returns the stored *value*.

A

```
## [1] 10
```

A_log

```
## [1] 2.303
```

B.seq

B

```
## [1] 100
```

x

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

```
##      [1]      1      2      3      4      5      6      7      8      9     10     11     12
##    [13]     13     14     15     16     17     18     19     20     21     22     23     24
##    [25]     25     26     27     28     29     30     31     32     33     34     35     36
##    [37]     37     38     39     40     41     42     43     44     45     46     47     48
##    [49]     49     50     51     52     53     54     55     56     57     58     59     60
##    [61]     61     62     63     64     65     66     67     68     69     70     71     72
##    [73]     73     74     75     76     77     78     79     80     81     82     83     84
##    [85]     85     86     87     88     89     90     91     92     93     94     95     96
##    [97]     97     98     99    100
```

Built-in variables

Some words and letters already have values in R
and should **never be used as variable names**.

```
pi
```

```
## [1] 3.142
```

```
version
```

```
## ... information about  
## this version of R ...
```

```
letters
```

```
## [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m"  
## [14] "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```
LETTERS
```

```
## [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M"  
## [14] "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

Reserved words

Some words and letters already have special meaning in the R language (*keywords*) and should **never be used as variable names**.



References & More Information

```
help.start()
```

Available from the screen above (offline):

- An Introduction to R
- The R Language Definition

Online:

- RStudio Education (education.rstudio.com)
 - ▶ tutorials, workshop materials, and other resources.
-  Manuals (<https://cran.r-project.org/manuals.html>)
-  Contributed Documentation
 - ▶ e.g., <http://cran.r-project.org/doc/contrib/usingR.pdf>
- Internet search
 - ▶ Stack Overflow (stackoverflow.com)
 - ▶ Cookbook for R (www.cookbook-r.com)