



A Gentle Introduction to R

Jonathan Whiteley

2023-08-10

Prerequisites

- Access to a copy of the ¹ software
 - ▶ i.e., a “binary executable”
 - ▶ Go to *www.r-project.org* to get a copy, or ask your system administrator.
- No previous experience with R or programming required.

¹The R logo () is © 2016 The R Foundation and used as-is under the terms of the **CC-BY-SA 4.0** license

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.

Answer in the chat:

What emoji best describes your current mood or state of mind?

Introductions

- Name
- Pronouns
- Job Title, role
- Have you used R before?
- Have you used a programming language before?
- *optional*: a hobby or activity you enjoy?

Icebreaker activity

What is this?

1–3 word description, for example:

- “This is grey”
- “This looks uncomfortable”

On your turn:


- 1 Previous person's name
- 2 Their answer to the question
- 3 Your name
- 4 Your answer
- 5 Name of the person to go next



Figure 1: What is this?

© John Speirs/Comedywildlifephoto.com

Learning Objectives

- Get familiar with the  *interface*
- Use technical *terms* for R concepts
- Enter *commands*
 - ▶ use R interactively: understand input & output
 - ▶ use some common *functions*
- Get familiar with 'R objects'
 - ▶ store & retrieve values
- Understand *Errors*, *Warnings*, and *Messages*
- How to get Help

Why is it named 'R'?

- 1 R started as an *open-source* implementation of the S statistical computing language (S-PLUS)²
 - ▶ S was created at Bell Laboratories in 1976³
 - ▶ R was based on the S syntax (mostly v3), but works very differently “under the hood”.
- 2 R was created by Ross Ihaka and Robert Gentleman — aka “R & R”⁴ — at the University of Auckland in the early 1990s.

*Read more about the history of R on Wikipedia*⁵

²<https://www.r-project.org/about.html>

³[https://en.wikipedia.org/wiki/S_\(programming_language\)](https://en.wikipedia.org/wiki/S_(programming_language))

⁴<https://www.r-project.org/contributors.html>

⁵[https://en.wikipedia.org/wiki/R_\(programming_language\)#History](https://en.wikipedia.org/wiki/R_(programming_language)#History)

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 the 'base R GUI'

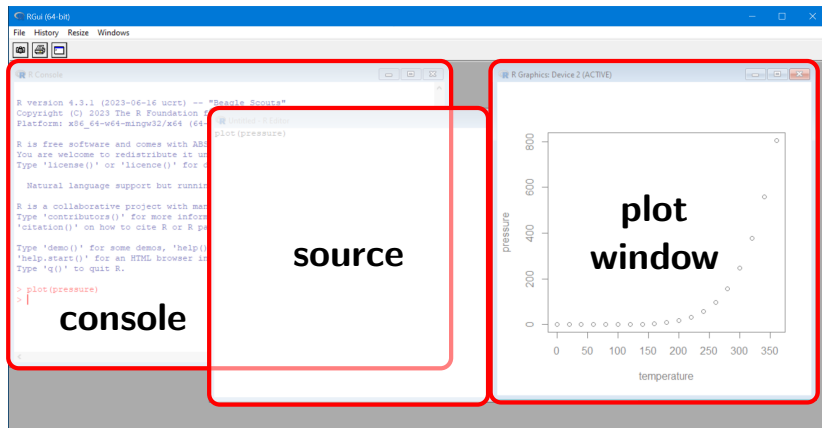


Figure 2: Screenshot of the R GUI in Windows.

A quick tour of RStudio

The RStudio GUI has 4 ‘panes’ that contain ‘tabs’.

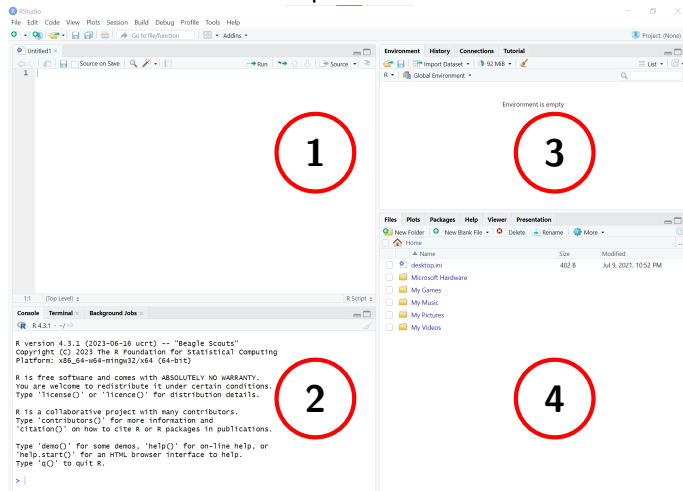


Figure 3: Screenshot of RStudio (default layout).

left:

- 1 top: **Source**^a
- 2 bottom: **Console, Terminal, ...**

right:

- 3 top: **Environment, History, ...**
- 4 bottom: **Files, Plots, Help, ...**

^aempty until you create or open a file

- Regardless of the GUI, you interact with R primarily using a *command line*
 - ▶ aka a command line interface (cli)
 - ▶ the command line is usually in the *console*
- “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 *console* is a window or pane where you will find:

- The *command line*
 - ▶ where you will enter commands for R to run
- Results of commands and other output
- Messages, *Warnings*, and **Errors**

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 (continued)

- 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`)

Input & Output

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)

R offers suggestions

Read the opening message carefully.

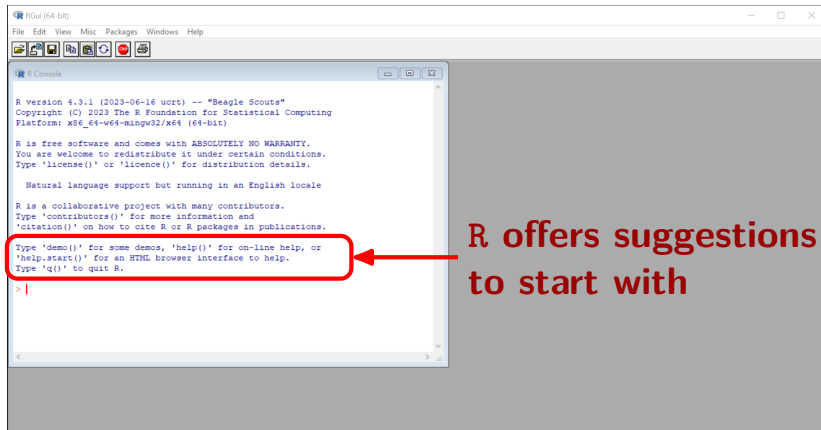


Figure 4: R offers suggestions of commands to **Type** in the console when it starts.

`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)
```

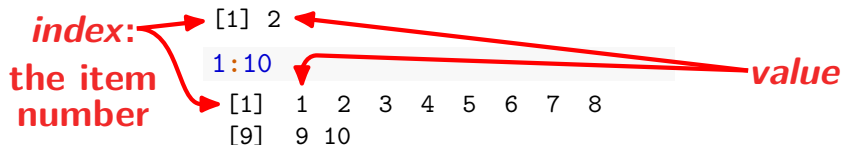
```
[1] 3
```

- 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.

Vectors

- The most basic kind of *object* in R is a *vector*
- Think of a vector as a list of related values (data), which are *all the same type*
- A single value is an “*atomic vector*” (a vector with a length of 1)



Vectors

- Vectors can be used in calculations
- Operations are applied to each item (*element-wise*)

```
sum( c(1, 2, 3, 4, 5) )  
1:10 +2  
1:5 * 5:1
```

- Vectors can be used to plot data in a graph

```
plot( rnorm(1000) )  
hist( rnorm(1000) )
```

Some data types (of *atomic vectors*)

numeric

- Includes *integers*, *real* (decimal / *double*), and *complex* numbers.
- 1.23

```
class(1.23)
class('hello')
class("1.23")
class(FALSE)
```

character (string)

- in single ' or double " quotes.
- 'hello world'
- "1.23"

```
typeof(1.23)
typeof(1:10)
```

```
as.character(c(1,2,NA,4))
```

logical

- TRUE or FALSE

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	B
[1] 10	[1] 100
A_log	x
[1] 2.303	[1] 3
B.seq	

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

Quiz Review



References & More Information

```
help.start()
```

Accessible 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)