Intro to R - Basics - 2

One should look for what is and not what he thinks should be – Albert Einstein

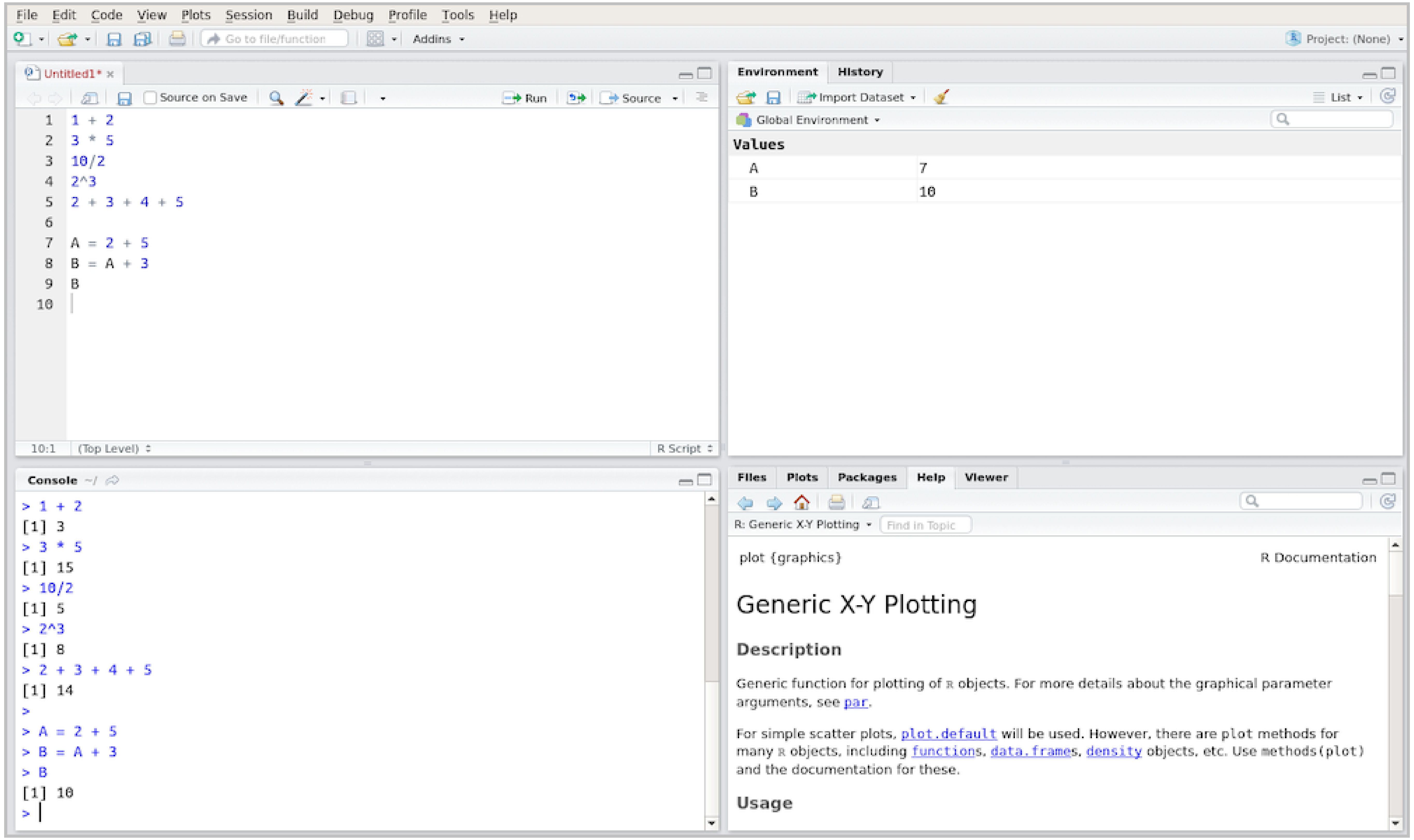
Module completion checklist

Objective	Complete
Perform basic calculations in R	
Work with variables in R	
Good coding practices for clarity and reproducibility	

RStudio overview

- A default RStudio layout includes 4 panes:
  - **Top left** pane is used as a `Script` pane where you can write and run your code as well as open other R scripts
  - **Bottom left** pane has a `Console`, which shows the output of running R commands
  - **Top right** is a helper pane that shows your `Environment` or `History`
  - **Bottom right** is another helper pane that shows `Files`, static `Plots` and interactive plots through `Viewer`, `Help`, and `Packages`
- As you work in R, you might decide to change the layout by going to **View > Panes > Pane Layout**

RStudio overview



```
1 + 2
3 * 5
10/2
2^3
2 + 3 + 4 + 5

A = 2 + 5
B = A + 3
B

```

```
> 1 + 2
[1] 3
> 3 * 5
[1] 15
> 10/2
[1] 5
> 2^3
[1] 8
> 2 + 3 + 4 + 5
[1] 14
>
> A = 2 + 5
> B = A + 3
> B
[1] 10
> 
```

```
Values
  A          7
  B          10
```

plot {graphics} R Documentation

Generic X-Y Plotting

Description

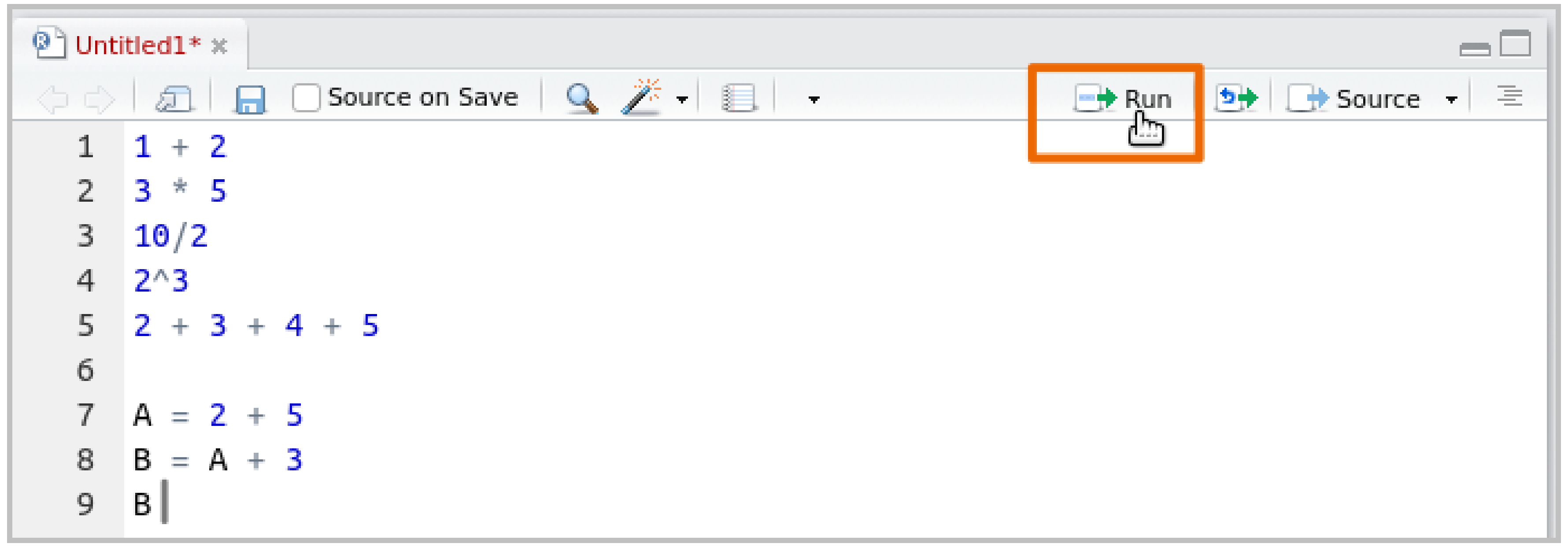
Generic function for plotting of R objects. For more details about the graphical parameter arguments, see par.

For simple scatter plots, plot.default will be used. However, there are plot methods for many R objects, including functions, data.frames, density objects, etc. Use methods(plot) and the documentation for these.

Usage

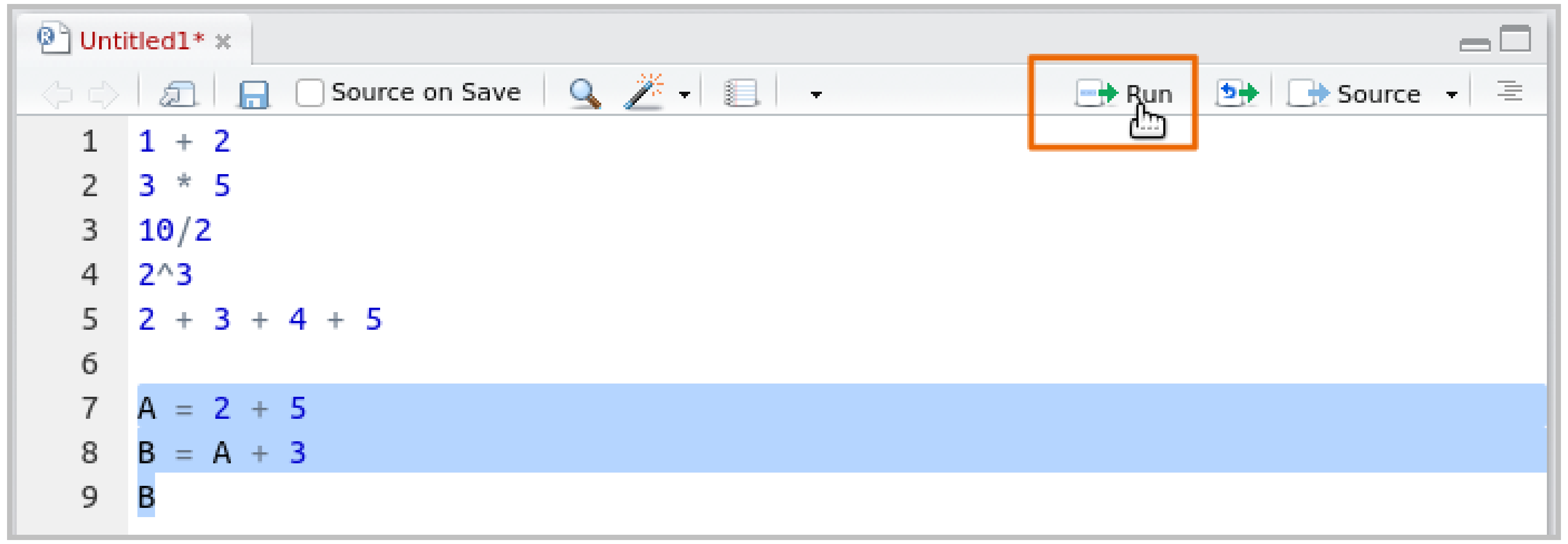
Executing commands in R

- Code is **executed** when you press `Run` in the top right corner of the script window
- R runs the line of code where your cursor is located
- You can highlight multiple lines to run at once
- An equivalent command from keyboard to `Run` button is a `Ctrl + Enter` (on PC) or `Command + Enter` (on Mac)



```
1 + 2
3 * 5
10/2
2^3
2 + 3 + 4 + 5

A = 2 + 5
B = A + 3
B
```



```
1 + 2
3 * 5
10/2
2^3
2 + 3 + 4 + 5

A = 2 + 5
B = A + 3
B
```

Basic calculations and operations

- To build some fluency in using R, let’s start with some basic calculations

- **Adding (+)**

```
# Add whole numbers.
1 + 2
```

```
[1] 3
```

```
# Add numbers with decimals.
3.23 + 4.65
```

```
[1] 7.88
```

- **Subtracting (-)**

```
# Subtract whole numbers.
10 - 7
```

```
[1] 3
```

```
# Subtract numbers with decimals.
3.23 - 4.65
```

```
[1] -1.42
```

Basic calculations and operations (cont'd)

- **Multiplying (*)**

```
# Multiply whole numbers.
1 * 2
```

```
[1] 2
```

```
# Multiply numbers with decimals.
3.23 * 4.65
```

```
[1] 15.0195
```

- **Dividing (/)**

```
# Divide whole numbers.
9 / 3
```

```
[1] 3
```

```
# Divide numbers with decimals.
3.23 / 4.65
```

```
[1] 0.6946237
```

Basic calculations and operations (cont'd)

- **Square roots (sqrt())**

```
# Take square root of a number with.
sqrt(100)
```

```
[1] 10
```

```
# Take square root of an expression.
sqrt(7 * 5)
```

```
[1] 5.91608
```

- **Exponents (^ or **)**

```
# Raise number to a power with `^`.
9 ^ 3
```

```
[1] 729
```

```
# Raise number to a power with `**`.
9 ** 3
```

```
[1] 729
```

```
# Raise expression to a power.
(3.23 / 4.65)^2
```

```
[1] 0.482502
```

Basic calculations and operations (cont'd)

- **Get remainder from division (%%)**

```
# Get remainder from division.
7 %% 3
```

```
[1] 1
```

```
# Get remainder from division.
4 %% 2
```

```
[1] 0
```

- **Perform integer division (%/%)**

```
# Perform integer division.
7 %/% 3
```

```
[1] 2
```

```
# Perform integer division.
4 %/% 2
```

```
[1] 2
```

Module completion checklist

Objective	Complete
Perform basic calculations in R	✔
Work with variables in R	
Good coding practices for clarity and reproducibility	

Variables

- In R, a variable is a piece of information to be stored, referenced, and used by a program
- Variables can be assigned any expression, not just numbers
- We can set variables by setting numbers equal to letters or terms using R's assignment operators, $<-$ and $=$

```
# Define a variable using `<-`
# as an assignment operator.
A <- 3
A
```

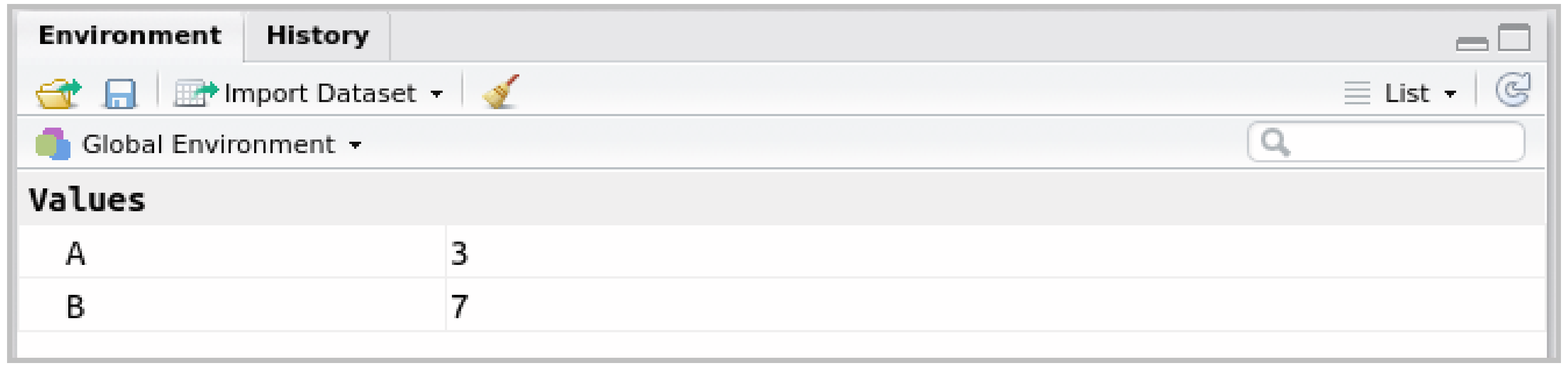
```
[1] 3
```

```
# Define a variable using `=`
# as an assignment operator.
B = 2 + 5
B
```

```
[1] 7
```

Variables (contd)

- When a variable is named (instantiated), R stores it in its environment
- R uses the values stored within its environment for all calculations within that session
- **Tip:** Check the `Environment` pane for all current objects in your coding session



Values

A	3
B	7

Operations with variables

Adding

- We can add variables

```
# Add 2 variables.
C = A + B
C
```

```
[1] 10
```

```
# Add a variable and a number.
D = C + 5
D
```

```
[1] 15
```

Subtracting

- We can subtract variables

```
# Subtract 2 variables from each other.
D - C
```

```
[1] 5
```

```
# Subtract a variable from number.
33 - D
```

```
[1] 18
```

```
# Or a number from a variable.
D - 33
```

```
[1] -18
```

- The same stands for **all** other arithmetic operations

Comparison

- Comparison operators in R are used to compare variables with values or variables to other variables.

```
# Check variables are equal.
A == B
```

```
[1] FALSE
```

```
# Check if variables are not equal.
A != B
```

```
[1] TRUE
```

```
# Check if one is greater than the other.
A > B
```

```
[1] FALSE
```

```
# Check if one is greater than or equal to 5.
A >= 5
```

```
[1] FALSE
```

```
# Check if one is less than or equal to 3.
A <= 3
```

```
[1] TRUE
```

```
# Check if one is smaller than the other.
A < B
```

```
[1] TRUE
```

Variable value re-assignment

- We can also name variables more descriptively, using strings of text

```
# 1. Create a variable and assign 67 to it.
this_variable = 67
this_variable
```

```
[1] 67
```

```
# 2. Create another variable and assign -54.
that_variable = -54
that_variable
```

```
[1] -54
```

```
# 3. Calculate their sum.
this_variable + that_variable
```

```
[1] 13
```

- We can also re-assign values, variables, and expressions to variables we've already used

```
# 4. Re-assign a value to `this_variable`.
this_variable = 35
this_variable
```

```
[1] 35
```

```
# 5. Add two variables and store the result
#    in `that_variable`.
that_variable = this_variable + that_variable
that_variable
```

```
[1] -19
```

- Just be sure to keep track and not to **overwrite** something you didn't intend to

Module completion checklist

Objective	Complete
Perform basic calculations in R	✔
Work with variables in R	✔
Good coding practices for clarity and reproducibility	

Good coding practices: comments

- It’s good practice to leave detailed comments in your code if you intend to collaborate with others
- A **hashmark (#)** is used to add a comment and annotate your code
- Comments can also help with orientation when revisiting previously written code in the future

```
# This is an example of a comment in R.

A = 2 + 5 #<- you can also add comments at the
end of line
B = A + 3
```

Good coding practices: spacing

- When calling a function or indexing an object, it is a good practice to use a space after each “,” , “<-”, and “=”
- Doing so makes your code more readable, whereas not doing so will make it more cramped and difficult to skim
- Again, both snippets below will run, but one is much easier for human eyes

```r
# This code works but it is difficult to understand
my_object=array(5:20,c(7,8))
my_object<-array(5:20,c(7,8))
my_object[,c(1,2)]


# A clear way to make code readable
my_object = array(5:20, c(7, 8))
my_object <- array(5:20, c(7, 8))
my_object[, c(1, 2)]
```

Good coding practices: naming conventions

- Function and variable names can use letters, digits, period (`.`) and underscore (_)
- They **must** start with a letter or a period, and if you begin with a period, you cannot follow with a digit
- All the examples below are valid names, but not all are easy to read

```
# Good
my_variable_name <- -5

# Bad
myvariablename <- -5

# Good
MyFunctionName()
my_function_name()

# Bad
myfunctionname()
```

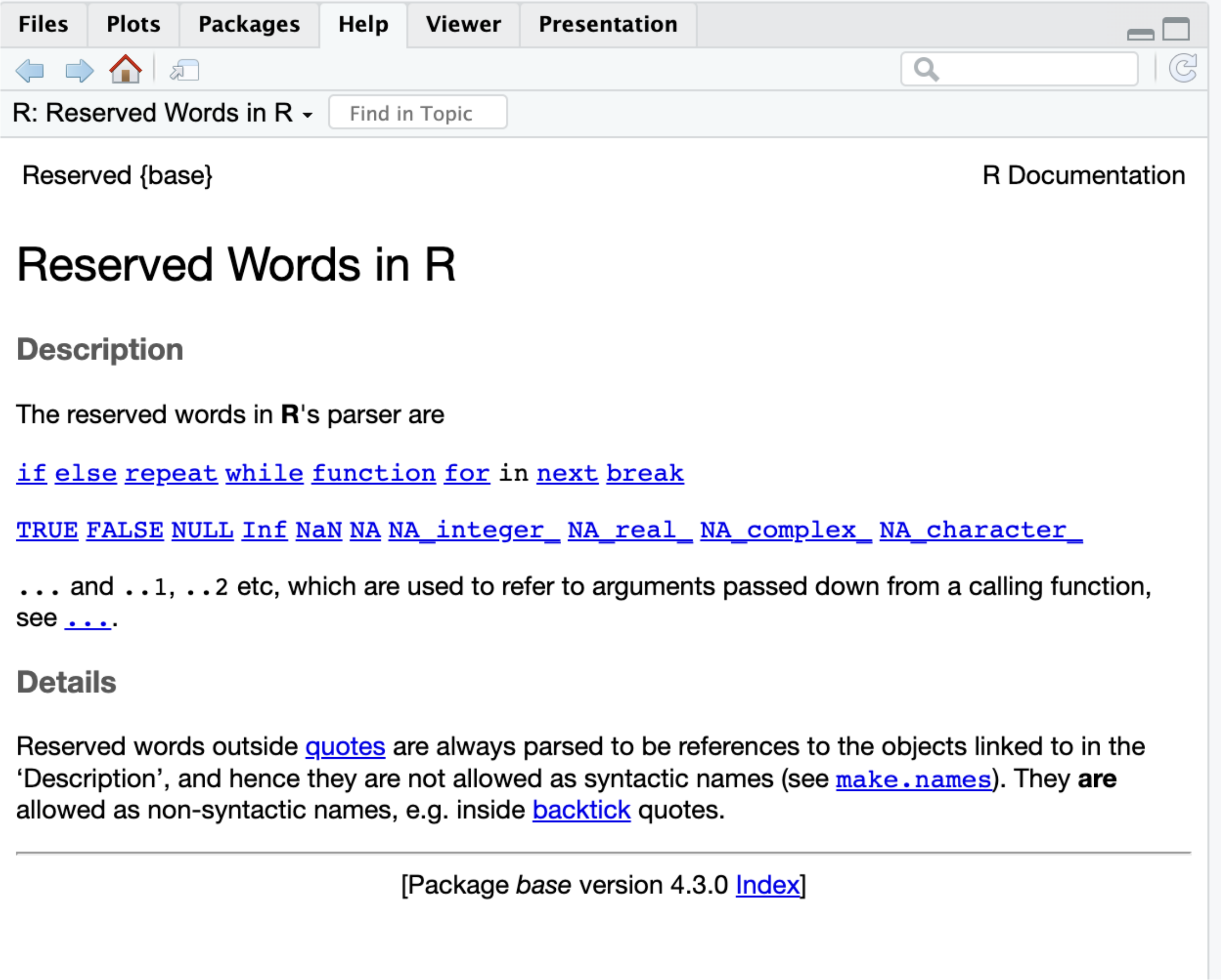
- Above all else, **be consistent**

Good coding practices: reserved words

- Reserved words **cannot** be used as names and can be viewed by running `?reserved` in R

```
?reserved
```

- You can also access a list of reserved words using the documentation in the `Help` pane



Reserved {base} R Documentation

Reserved Words in R

Description

The reserved words in **R**'s parser are

`if` `else` `repeat` `while` `function` `for` `in` `next` `break`

`TRUE` `FALSE` `NULL` `Inf` `NaN` `NA` `NA_integer_` `NA_real_` `NA_complex_` `NA_character_`

`...` and `..1`, `..2` etc, which are used to refer to arguments passed down from a calling function, see `...`.

Details

Reserved words outside quotes are always parsed to be references to the objects linked to in the 'Description', and hence they are not allowed as syntactic names (see `make.names`). They **are** allowed as non-syntactic names, e.g. inside backtick quotes.

[Package *base* version 4.3.0 Index]

Good coding practices: libraries first

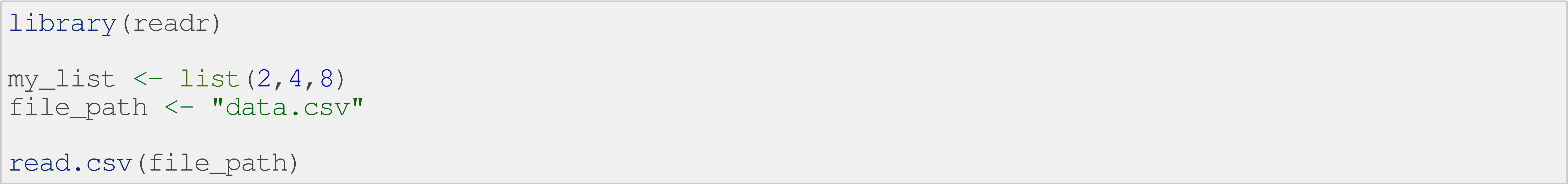
- A library is a collection of functions, data, and other compiled code that will let you perform certain operations
- Code should always **begin** by stating which libraries to import
- The code below is messy because it tries to load the data without importing the library `tidyverse` that will be used on it

```
file_path <- "data.csv"
read.csv(file_path)
library(tidyverse)
```

```
Error in file(file, "rt") : cannot open the connection Calls: <Anonymous> ... withVisible( -
> eval -> -> read.csv -> read.table -> file
```

Good coding practices: libraries first (cont’d)

- Your code will often depend on those libraries being in place to run properly
- For this reason, we often call libraries **dependencies**, which should be accounted for in code before any tasks are performed
- The code below begins by importing the library, then specifying variables, and finally performing a certain operation



```
library(readr)

my_list <- list(2,4,8)
file_path <- "data.csv"

read.csv(file_path)
```

Knowledge check

Exercise

You are now ready to try Tasks 1-12 in the Exercise for this topic

Module completion checklist

Objective	Complete
Use RStudio to write and execute R code	
Perform basic calculations in R	✔
Work with variables in R	✔
Good coding practices for clarity and reproducibility	✔

Basics: Topic summary

In this part of the course, we have covered:

- Overview of Data Science and its tools
- R and RStudio as tools in data analysis and their features
- Basic calculations in R

Congratulations on completing this module!