

Chapter 2

Basic arithmetic in R

- Arithmetic is performed as one would expect:

```
> 3^2 * exp(2) - 1
```

```
[1] 65.5015
```

```
> 67/(log(2) + 1)
```

```
[1] 39.57128
```

```
> 2 * pi
```

```
[1] 6.283185
```

- R can do complex arithmetic; try, for example, $(1 - 2i) * (-3 - 1i)$

You must tell R you wish to do complex arithmetic; compare the output of

```
> sqrt(-1)
```

```
[1] NaN
```

with the output of

```
sqrt(-1 + 0i)
```

```
[1] 0+1i
```

- R is case-sensitive. This means that `abc`, `ABC` and `aBc` are different.

- All commands must be followed by parentheses; usually, these parentheses contain arguments to the function. But, as the example above for `q()` shows, even if there are no arguments to the function the parentheses are necessary (otherwise we see the function definition, which is scary as well as not what we want).
- Round brackets are used for function arguments; square brackets are used for indexing vectors and matrices.
- R works happily with *variable* names. Variables must start with a letter, but can contain numbers and dots (and, more recently, underscores). It is common in R to use the period in variable names; for example, `model1.residual` is a valid variable name. Numbers can be used in variable names, except as the first character. The following are all valid variable names: `x3`, `x2.2`, `D.2.x.2`
- Variables can be defined using the assignment operator `<-`

```
> radius <- 3
> circumference <- 2 * pi * radius
> area <- pi * radius^2
> area

[1] 28.27433

> circumference

[1] 18.84956
```

Note that spacing is not important: `x <- 2` is the same as `x <-2`. However, use spaces wisely for readability¹.

- Assignment is also allowed using `=`, though `<-` is preferred. All these are the same
 - `x <- 3` (Best)
 - `x = 3` (OK)
 - `3 -> x` (OK)
 - Not so many years ago, this syntax was also valid: `x _ 3` which is very hard to read. This syntax is being phased out (thankfully).

¹But note: `x<-3` can be interpreted two ways: `x < (-3)` or `x <- (3)`. So use spaces wisely and carefully!

You can even do things like this: `x <- 2 -> y` (but this looks awful) and `x <- y <- z <- 2`.

- Strings are usually given in double quotes (though single quotes work in more recent versions too); for example: `plot.title <- "Title for my plot"`
- Comments are indicated by the `#` character: Any text following this character is ignored by R. This is useful for explaining what your commands do.

```
area <- pi * radius^2    # Circle area
```

- Two or more commands can be entered on a line at once if separated by a semi-colon; try entering this all on one line in R:
`length <- 2; width <- 5; length * width`
- In the command window, a `+` character means R is waiting for you to *complete* an instructions that is unfinished (for example, brackets are unmatched).
- Commands not completed are automatically continued on to the next line with the `+` prompt. For example, try typing the incomplete command `5 + log(` and then pressing RETURN. The command prompt will change to `+` when you should type 1) so R will have a complete command; it will then give you an answer.
- R is a vector-based language, and so instructions generally work with vectors as easily as scalars:

```
> x <- seq(-3, 3, by = 0.25)
> x
```

```
[1] -3.00 -2.75 -2.50 -2.25 -2.00 -1.75 -1.50 -1.25 -1.00 -0.75 -0.50
[12] -0.25  0.00  0.25  0.50  0.75  1.00  1.25  1.50  1.75  2.00  2.25
[23]  2.50  2.75  3.00
```

```
> x^2
```

```
[1] 9.0000 7.5625 6.2500 5.0625 4.0000 3.0625 2.2500 1.5625 1.0000
[10] 0.5625 0.2500 0.0625 0.0000 0.0625 0.2500 0.5625 1.0000 1.5625
[19] 2.2500 3.0625 4.0000 5.0625 6.2500 7.5625 9.0000
```

```
> x + 10
```

```
[1] 7.00 7.25 7.50 7.75 8.00 8.25 8.50 8.75 9.00 9.25 9.50
[12] 9.75 10.00 10.25 10.50 10.75 11.00 11.25 11.50 11.75 12.00 12.25
[23] 12.50 12.75 13.00
```

- A list of the current variables can be found using `objects()` or `ls()`. It is good practice to remove unwanted variables using `rm(variable.name)`.
- When calling functions, the names of the input variable can be used explicitly (this is unlike MATLAB). For example, `plot(x, y, xlab="The x-axis label")` explicitly assigns a value to `xlab` wherever it appears in the definition of the function `plot`. More on this in Section 3.3.2.