Chapter 2

R as a Calculator

R can be used as a calculator. This chapter (a) outlines rules associated with the order in which R executes simple arithmetic operations, (b) introduces a function that controls the number of digits that R uses to display the results of a calculation, (c) shows how R handles an extreme calculation like $1/0$, and (d) introduces the integer-divide and modulo functions.

In the examples given throughout this text, R commands that you type will be set in `monospace` font following the > prompt. The response from R will be given on the next line or lines, also set in monospace font. This is exactly what you will see in the R environment.

2.1 Order of operations

We begin with simple arithmetic operations. The order that arithmetic operations are performed is consistent with convention. The PEMDAS convention implies that parentheses get highest priority, followed by exponentiation, followed by multiplication and division, followed by addition and subtraction. The first example of using R in the calculator mode is given below.

```
> 2 + 9 * 4             # PEMDAS; place spaces around operators
[1] 38
```

No equal sign is necessary; pressing the return key serves that role. The response from R is `[1] 38`, which means that R performed the multiplication before the addition per the PEMDAS convention, giving a result of 38. The `[1]` that appears before the 38 indicates that the result is a vector of length one. More details concerning vectors will be given in Chapter 4. The `[1]` can be ignored for now. The text after the `#` is a comment and has been added to highlight the concept being presented. It is good programming practice to surround the operators with a single space for readability. This is not necessary, however, and it is perfectly acceptable to key in the R command as

```
> 2+9*4                 # no spaces around operators; it still works
[1] 38
```

But you could imagine how difficult this would be to read if it was 30 or 40 characters long.

The exponentiation operator is the caret symbol, `^`, as illustrated with

```
> 4 + 3 / 10 ^ 2        # exponentiation first, then division, then addition
[1] 4.03
```

Parentheses can be used to alter the order of operations.

```
> 4 + (3 / 10) ^ 2      # division first, then exponentiation, then addition
[1] 4.09
> (4 + 3) / 10 ^ 2      # addition first, then exponentiation, then division
[1] 0.07
> (4 + 3 / 10) ^ 2      # division first, then addition, then exponentiation
[1] 18.49
> ((4 + 3) / 10) ^ 2    # addition first, then division, then exponentiation
[1] 0.49
```

If you accidentally press the return key prior to completing a command, the usual greater than prompt will be replaced by a + prompt, which is R asking you to continue the command.

```
> 2 * 3 -               # + prompt for more input
+ 7                     # completed command 2 * 3 - 7
[1] -1
```

An R command that traverses several lines in this fashion is appropriate for a particularly-long calculation. Now consider computing $1/4$.

```
> 1 / 4                 # the display suppresses trailing zeros
[1] 0.25
```

R calculates one-fourth and displays it as the decimal equivalent 0.25, suppressing the trailing zeros. Now consider computing $1/3$.

```
> 1 / 3                 # default seven display digits
[1] 0.3333333
```

R must make a decision on how many digits to display. The default in R is to display seven digits. This can be altered, however, as will be shown subsequently.

R does not include commas to separate groups of three digits when displaying the result.

```
> 8 * 10 ^ 3            # exponentiation first
[1] 8000
```

Now consider the calculation of a large number.

```
> 1111111 * 1111111     # no commas on constants; result in scientific notation
[1] 1.234568e+12
```

The result of this calculation is expressed in scientific notation: $1.234568 \cdot 10^{12}$. With only seven digits displayed, it is not possible to know the exact value of the product.

2.2 Number of digits displayed

By default, R displays seven digits, but more digits are stored. As an illustration, the built-in constant π is designated with `pi`.

```
> pi                    # built in constant
[1] 3.141593
```