swirl Lesson 1: Basic Building Blocks

Raphael Carvalho

08/05/2019

Basic Building blocks

- In its simplest form, R can be used as an interactive calculator. Type 5 + 7 and press Enter.

```
5 + 7
```

```
## [1] 12
```

- To assign the result of 5 + 7 to a new variable called x, you type x <- 5 + 7. This can be read as 'x gets 5 plus 7'. Give it a try now.

```
x <- 5 + 7
```

- To view the contents of the variable x, just type x and press Enter. Try it now.

```
x
```

```
## [1] 12
```

- Now, store the result of x - 3 in a new variable called y.

```
y <- x - 3
```

- What is the value of y? Type y to find out.

```
y
```

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

- The easiest way to create a vector is with the c() function, which stands for 'concatenate' or 'combine'. To create a vector containing the numbers 1.1, 9, and 3.14, type c(1.1, 9, 3.14). Try it now and store the result in a variable called z

```
z <- c(1.1, 9, 3.14)
```

- Anytime you have questions about a particular function, you can access R's built-in help files via the ? command. For example, if you want more information on the c() function, type ?c without the parentheses that normally follow a function name. Give it a try.

```
?c
```

- Type z to view its contents. Notice that there are no commas separating the values in the output.

```
z
```

```
## [1] 1.10 9.00 3.14
```

- You can combine vectors to make a new vector. Create a new vector that contains z, 555, then z again in that order. Don't assign this vector to a new variable, so that we can just see the result immediately.

```
c(z, 555, z)
```

```
## [1]   1.10   9.00   3.14 555.00   1.10   9.00   3.14
```

- Numeric vectors can be used in arithmetic expressions. Type the following to see what happens: z * 2 + 100.

```
z * 2 + 100
```

```
## [1] 102.20 118.00 106.28
```

- Take the square root of z - 1 and assign it to a new variable called my_sqrt.

```
my_sqrt <- sqrt(z-1)
```

- Before we view the contents of the my_sqrt variable, what do you think it contains?

2: a vector of length 3

- Print the contents of my_sqrt.

```
my_sqrt
```

```
## [1] 0.3162278 2.8284271 1.4628739
```

- Now, create a new variable called my_div that gets the value of z divided by my_sqrt.

```
my_div <- z/my_sqrt
```

- Which statement do you think is true?

3: The first element of my_div is equal to the first element of z divided by the first element of my_sqrt, and so on...

- Go ahead and print the contents of my_div.

```
my_div
```

```
## [1] 3.478505 3.181981 2.146460
```

- To see another example of how this vector 'recycling' works, try adding c(1, 2, 3, 4) and c(0, 10). Don't worry about saving the result in a new variable.

```
c(1, 2, 3, 4) + c(0, 10)
```

```
## [1]  1 12  3 14
```

- Try c(1, 2, 3, 4) + c(0, 10, 100) for an example.

```
 c(1, 2, 3, 4) + c(0, 10, 100)
```

```
## Warning in c(1, 2, 3, 4) + c(0, 10, 100): comprimento do objeto maior não é
## múltiplo do comprimento do objeto menor

## [1]   1  12 103   4
```

- In many programming environments, the up arrow will cycle through previous commands. Try hitting the up arrow on your keyboard until you get to this command (z * 2 + 100), then change 100 to 1000 and hit Enter. If the up arrow doesn't work for you, just type the corrected command.

```
z * 2 + 1000
```

```
## [1] 1002.20 1018.00 1006.28
```

- You can type the first two letters of the variable name, then hit the Tab key (possibly more than once). Most programming environments will provide a list of variables that you've created that begin with 'my'. This is called auto-completion and can be quite handy when you have many variables in your workspace. Give it a try. (If auto-completion doesn't work for you, just type my_div and press Enter.)

```
my_div
```

```
## [1] 3.478505 3.181981 2.146460
```