Week 1 - R Basics - Vectors

Sally Longmore

22/01/2020

The simplest and most common data structure in R is the vector.

Vectors come in two different flavours
- atomic vectors: Contains exactly one datatype - lists: Contains multiple data types

Atomic vectors can include only one datatype, but that datatype can include - logical - character - integer - complex

Logical vectors contain the values TRUE, FALSE and NA (for 'not available'. These values are generated as the result of logical 'conditions'.

Create a numeric vector that contains the values 0.5, 55, -10 and 6 and assign it to the variable name num_vect

```
num_vect <- c(0.5, 55, -10, 6)
```

Now create a variable called tf that gets the result of num_vect < 1.

```
tf <- num_vect < 1
tf
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

This returns a vector of logical values indicating which elements are less than 1

The statement num_vect < 1 is a condition and tf fells us whether wach corresponding element of our numeric vector num_vect satisifies this condition.

The frist element of num_vect is 0.5, which is less than 1 and therefore the statement 0.5 < 1 is TRUE.
The second element of num_vect is 55, which is greater than 1, so the statement 55 < 1 is FALSE.
The same logic applies to the third and fourth elements.

Lets try num_vect >= 6 without assigning to a new variable

```
num_vect >= 6
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

In this example we are checking if each element is greater than or equal to 6.
Since only 55 and 6 are greater than or equal to 6, only the second and fourth elements are TRUE, the rest are FALSE.

The <, >, ==, <=, >= and **!=** are logical operators.

We can have two logical expressions by using the **OR** operator | which is the pipe and the **AND** operator which is the ampersand.

Lastly, ! is the negation of the expression and is TRUE when A is FALSE.

```
(3 > 5) & (4 == 4)
```

```
## [1] FALSE
```

```
(TRUE == TRUE) | (TRUE == FALSE)
```

```
## [1] TRUE
```

```
((111 >= 111) | !(TRUE)) & ((4 + 1) == 5)
```

```
## [1] TRUE
```

Character Vectors

Character vectors use double quotes " to distinguish character objects.

Create a character vector that contains the following words: “My”, “name”, “is”.

```
my_char <- c("My", "name", "is")
my_char
```

```
## [1] "My"   "name" "is"
```

my_char is a character vector with a lenght of 3.

We can join the elements of my_char together in one continous string using the paste() function. The example below will join the elements using a space in between.

```
paste(my_char, collapse = " ")
```

```
## [1] "My name is"
```

We can concatenate to my_char using the c() function like this

```
my_name <- c(my_char, "Sally")
my_name
```

```
## [1] "My"    "name"  "is"    "Sally"
```

Now if we use the paste() function we can join the works in my_name together.

```
paste(my_name, collapse = " ")
```

```
## [1] "My name is Sally"
```

Paste can also joing character vectors that are lenght of 1.

```
paste("Hello", "world!", sep = " ")
```

```
## [1] "Hello world!"
```

For a slightly more compleicated exmaple, we can join two vectors, each of length 3. Use paste() to join the integer vector 1:3 with the character vector “X”, “Y”, “Z” using a seperator of "". This will joing each of the elements in the vectors, and return a character vector of lenght 3.

’

```
paste(1:3, c("X", "Y", "Z"), sep = "")
```

```
## [1] "1X" "2Y" "3Z"
```

Vector Recycling

When concatenating vectors of different lenghts it is called vector recycling. The shorter vector will be reused or recycled. LETTERS is a predefined variable in R containing a character vector of all 26 characters in the english alphabet.

```r
paste(LETTERS, 1:4, sep = "-")
```

```
##  [1] "A-1" "B-2" "C-3" "D-4" "E-1" "F-2" "G-3" "H-4" "I-1" "J-2" "K-3" "L-4"
## [13] "M-1" "N-2" "O-3" "P-4" "Q-1" "R-2" "S-3" "T-4" "U-1" "V-2" "W-3" "X-4"
## [25] "Y-1" "Z-2"
```

Since the character vector LETTERS is longer than the numeric vector 1:4, R simply recycles the numeric vector until it matches the length of the character vector LETTERS.

It is also worth nothing that the numeric vector 1:4 get 'coerced' into a character vector by the paste() function.