A Very Basic Introduction to R – Part VII

Vectors

A numeric vector is a list of numbers. The c() function is used to collect things together into a vector. We can type

```
> c(0, -2, 5, -3)

[1]  0 -2  5 -3
```

We can assign this to a named object. For example,

```
> temperatures <- c(0, -2, 5, -3)
```

To see the contents of `temperatures`, type

```
> temperatures

[1]  0 -2  5 -3
```

Vectors can be joined together (i.e. *concatenated*) with the `c` function. For example, note what happens when we type

```
> temp2 <- c(3, 2, 17, temperatures)
> temp3 <- c(temperatures, temp2)
> temp3

 [1]  0 -2  5 -3  3  2 17  0 -2  5 -3
```

Patterned Vectors

Patterned vectors can be produced using the `seq()` function. For example, the sequence of odd numbers less than or equal to 21 can be obtained using

```
> seq(1, 21, by=2)

 [1]  1  3  5  7  9 11 13 15 17 19 21
```

Notice the use of `by=2` here. The default value of `by` is 1.

Repeated patterns are obtained using `rep()`. To repeat the value 3, 12 times, use

```
> rep(3, 12)
```

```
 [1] 3 3 3 3 3 3 3 3 3 3 3 3
```

Here are some other examples.

```
> # repeat the pattern 2 4 ... 20,
> # twice
> rep(seq(2, 20, by=2), 2)

 [1]  2  4  6  8 10 12 14 16 18 20  2  4  6  8 10 12 14 16 18 20

> # repeat 1, 3 times and  4, twice
> rep(c(1, 4), c(3, 2))

[1] 1 1 1 4 4

> # repeat 1 and 4, 3 times
> rep(c(1, 4), each=3)

[1] 1 1 1 4 4 4

> # repeat each value twice
> rep(seq(2, 20, 2), rep(2, 10))

 [1]  2  2  4  4  6  6  8  8 10 10 12 12 14 14 16 16 18 18 20 20
```

Extracting elements from vectors

A way to display only some elements of a vector is to use square brackets to extract just that element:

To print the third element of `temp2`, type

```
> temp2[3]

[1] 17
```

We can extract more than one element at a time. For example,

```
> temp2[c(1, 3)]

[1]  3 17

> y <- temp3[1:5]
> y

[1]  0 -2  5 -3  3
```

Negative indices can be used to avoid certain elements. For example, we can select all but the second element of `x` as follows:

```
> temp2[-2]

[1]  3 17  0 -2  5 -3

> temp3[-c(2,3)] # avoid elements 2 and 3

[1]  0 -3  3  2 17  0 -2  5 -3

> temp3[-(2:5)] # avoid elements 2,3,4,5

[1]  0  2 17  0 -2  5 -3
```

Using a zero index returns nothing. This is not something that one would usually type, but it may be useful in more complicated expressions.

```
> temp3[c(0, 3, 6)]

[1] 5 2
```

Do not mix positive and negative indices. To see what happens, consider

```
> temp3[c(-2, 3)]

Error in temp3[c(-2, 3)] : only 0's may be mixed with negative subscripts
```

The problem is that it is not clear what is to be extracted: do we want the third element of `x` before or after removing the second one?

Exercises

1. Write down the R code required to produce the following sequences.

```
 [1] 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5
```

```
[1]  1  4  7 10 13 16 19
```

```
 [1]  1  5  1  5  1  5  1  5  1  5  1  5  1  5  1  5  1  5  1  5
[19]  1  5  1  5  1  5  1  4  7 10 13 16 19
```

```
 [1]  2  4  6  8 10  7  9 11 13 15 12 14 16 18 20 17 19 21
[19] 23 25 22 24 26 28 30 27 29 31 33 35 32 34 36 38 40 37
[37] 39 41 43 45 42 44 46 48 50 47 49 51 53 55
```

2. Construct the vector `z` as follows:

```
> z <- rep(seq(2,10), seq(2,10))
```

   (a) Identify the 33rd and 39th elements of `z`.

   (b) Use the `sum()` function to add up the first 17 elements of `z`.

   (c) Add up all elements of `z` except for the 20th.

3. Construct the vector `y` as follows:

```
> set.seed(12432)
> y <- rnorm(250, sd=7)
```

   (a) Identify the 201st and 242nd elements of `y`.

   (b) Find the average of all elements of `y` using the `mean()` function.

   (c) Find the average of all elements of `y` except for the first 2 elements.