Data Structure

Antonio Vega

Vectors

Data types in R

A **vector** is an ordered sequence of data. `R` has many types of data, for example:

- `logical`: logical (`TRUE` or `FALSE`)
- `integer`: whole numbers, $\mathbb{Z}$
- `numeric`: real numbers, $\mathbb{R}$
- `complex`: complex numbers, $\mathbb{C}$
- `character`: words

In the vectors of `R`, all their objects must be of the same type: all numbers, all words, etc. When we want to use vectors formed by objects of different types, we will have to use **generalized lists**, 'lists' that we will see at the end of the topic.

Basic

- `c()`: to define a vector
- `scan()`: to define a vector
- `fix(x)`: to visually modify the vector *x*
- `rep(a, n)`: to define a constant vector that contains the data *a* repeated *n* times

```r
c(1,2,3)
```

```
[1] 1 2 3
```

```r
rep("Tono",5)
```

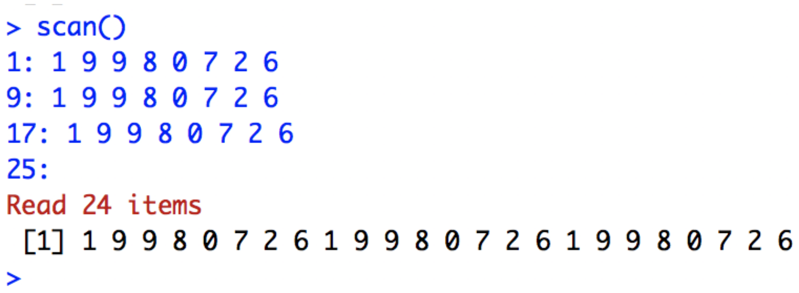
```
[1] "Tono" "Tono" "Tono" "Tono" "Tono"
```

scan() function

This functions also work with URL with text files

Example

This is an example of how to create a vector that contains 3 copies of 1 9 9 8 0 7 2 6 with the scan function:



```
> scan()
1: 1 9 9 8 0 7 2 6
9: 1 9 9 8 0 7 2 6
17: 1 9 9 8 0 7 2 6
25:
Read 24 items
 [1] 1 9 9 8 0 7 2 6 1 9 9 8 0 7 2 6 1 9 9 8 0 7 2 6
>
```

Progressions and Sequences

An arithmetic progression is a succession of numbers such that the **difference**, d, of any pair of successive terms in the sequence is constant.

$$a_n = a_1 + (n-1)\ cdotd$$

- `seq (a, b, by=d)`: to generate an arithmetic progression of difference d that starts in *to* up get to b
- `seq (a, b, length.out=n)`: define arithmetic progression of length n ranging from a to b with difference d. Therefore $d = (b-a)/(n-1)$
- `seq (a, by= d, length.out=n)`: define the arithmetic progression of length n and difference d starting in a
- `a:b`: define the sequence of consecutive numbers ** $(\mathbb{Z})$ between two numbers a and b

Example of sequence

- This is an example of a sequence 5, 8.5, 12, 15.5, 19, 22.5, 26, 29.5, 33, 36.5, 40, 43.5, 47, 50.5, 54, 57.5
- If we want a sequence of **n** lengh we can use 5, 14.1666667, 23.3333333, 32.5, 41.6666667, 50.8333333, 60
- Ifwe want to start in n number and have a determinate lenght with n step we use 5, 8, 11, 14, 17, 20, 23

Sequences

Exercise

- Numbers sequence from 1 ot 20: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- First 20 pair numbers: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40
- 30 equidistant numbers bettwen 17 and 98 with 4 significative decimals: 17, 19.7931034, 22.5862069, 25.3793103, 28.1724138, 30.9655172, 33.7586207, 36.5517241, 39.3448276, 42.137931, 44.9310345, 47.7241379, 50.5172414, 53.3103448, 56.1034483, 58.8965517, 61.6896552, 64.4827586, 67.2758621, 70.0689655, 72.862069, 75.6551724, 78.4482759, 81.2413793, 84.0344828, 86.8275862, 89.6206897, 92.4137931, 95.2068966, 98

Example of vector

This is another example to generate a vector, c is also named as concatenation:

```
c(rep(pi, 5), 5:10, -7) -> x
x
```

```
 [1]  3.141593  3.141593  3.141593  3.141593  3.141593  5.0
 [8]  7.000000  8.000000  9.000000 10.000000 -7.000000
```

```
c(0, x, 10, x, 20)
```

```
 [1]  0.000000  3.141593  3.141593  3.141593  3.141593  3.1
 [8]  6.000000  7.000000  8.000000  9.000000 10.000000 -7.0
[15]  3.141593  3.141593  3.141593  3.141593  3.141593  5.0
[22]  7.000000  8.000000  9.000000 10.000000 -7.000000 20.0
```

Functions

When we want to apply a function to each of the elements of a data vector, the `sapply` function saves us having to program with loops in `R`:

- "sapply (vector_name, FUN = function_name)": to apply this function to all vector elements
- `sqrt(x)`: Calculate a new vector with the square roots of each of the elements of the vector x

```r
x <- 1:10
sapply(x, FUN = function(element){sqrt(element)})
```

```
 [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490
 [8] 2.828427 3.000000 3.162278
```