Programming for Data Science

Vectors in R language

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica

October 2021

Vectors in R

- an ordered list of homogeneous elements;
- Vectors are the simplest type of object in R;
  There are 3 main types of vectors:
  - Numeric vectors;
  - Character vectors;
  - Logical vectors.
- To create a numeric vector x consisting of 6 numbers, 1.4, 6, 23.1, 65.43, 2.7, 55 use:

$$> x = c(1.4, 6, 23.1, 65.43, 2.7, 55)$$

or

$$> x <- c(1.4, 6, 23.1, 65.43, 2.7, 55)$$

or

$$> assign("x", c(1.4, 6, 23.1, 65.43, 2.7, 55))$$

Numeric vectors in R

- To print the contents of x:

  $> x$
  [1]1.4 6 23.1 65.43 2.7 55

  symbol [1] in front of the result is the index of the first element in the vector x.

- To access a particular element of x:

  $> x[1]$
  [1]1.4

  $> x[6]$
  [1]55

  $> x[c(1,6)]$
  [1]1.4 55

  $> x[-c(1,5)]$ **Operator - means: select all the elements except those ....**
  [1]6 23.1 65.43 2.7

Numeric vectors in R

- To modify a particular vector element:

  $> x[2] = 5$ **to modify the 2nd element of x in 5**
  [1]1.4 5 23.1 65.43 2.7 55

  $> x[4] = 5$
  [1]1.4 5 23.1 5 2.7 55

- To modify more than one vector elements:

  $> x[c(2, 4)] = c(6, 65.43)$
  [1]1.4 6 23.1 65.43 2.7 55

  $> y = x$
  $> y[y < 3] = 1$
  $> y$
  [1]1 6 23.1 65.43 1 55

Numeric vectors in R

- A vector can be used to do further assignments:

  $> y = c(x, 2, 3, x[c(1, 3)])$

  vector y with 10 entries is created:

  $> y$
  [1]1.4 6 23.1 65.43 2.7 55 2 3 1.4 23.1

- Operation are performed on each single element:

  $> x/10$
  [1]0.14 0.6 2.31 6.543 0.27 5.5

- Short vectors are "recycled" to match long ones (if it is possible):

  $> v = x[c(1, 2)] + y$ $x[c(1, 2)]$ **is repeated 5 times**
  $> v$
  [1]2.8 12 24.5 71.43 4.1 61 3.4 9 2.829.1

Numeric vectors in R

- Short vectors are "recycled" to match long ones (if it is possible)

  $> v = x + y$
  Warning message:
  In $x + y$: longer object length is not a multiple of shorter object length

- Some functions take vectors of values and produce results of the same length:
  sin, cos, tan, asin, acos, atan, log, exp, ...

  $> log(x)$
  [1]0.3364722 1.7917595 3.1398326 4.1809809 0.9932518 4.0073332

- Some functions return a single value:
  sum, mean, max, min, prod, ...

$> length(x)$
[1]6
$> sum(x)$
[1]153.63
$> sum(x)/lenght(x)$
[1]25.605

$> mean(x)$
[1]25.605
$> max(x)$
[1]65.43
$> min(x)$
[1]1.4

Numeric vectors in R

- Some special functions are:

  sort, cumsum, cumprod, pmax, pmin, range...

  $> x$
  [1]1.4 6 23.1 65.43 2.7 55

  $> sort(x)$
  [1]1.40 2.70 6.00 23.10 55.00 65.43

  $> cumsum(x)$ **cumulative sums**
  [1]1.40 7.40 30.50 95.93 98.63 153.63

  $> y = c(2, 3, 5, 6, 100, 9)$
  $> pmax(x, y)$ **max among 2 or more vector/scalar**
  [1]2 6 23.1 65.43 100 55

  $> pmin(x, y)$
  [1]1.40 3 5 6 2.7 9

  $> range(x)$
  [1]1.40 65.43

How to generate sequences in R

- In R it is possible to generate sequences of numbers
  - using operator ":"
    $> 1 : 5$
    [1]1 2 3 4 5
  - using function seq()
    $> seq(1, 5)$
    [1]1 2 3 4 5
    $> seq(from = 1, to = 5)$
    [1]1 2 3 4 5
    We can also specify a step size (using by=value) or a length (using length=value) for the sequence.
    $> seq(1, 5, by = 0.5)$
    [1]1 1.5 2 2.5 3 3.5 4 4.5 5
    $> seq(from = 1, to = 5, length = 9)$
    [1]1 1.5 2 2.5 3 3.5 4 4.5 5
  - using function rep()
    $> rep(x, 3)$
    [1]1.40 6.00 23.10 65.43 2.70 55.00 1.40 6.00 23.10 65.43 2.70 55.00
    [13]1.40 6.00 23.10 65.43 2.70 55.00

Character vector in R

- A string is identify by " "
- A string vector is defined as well as number vector by c() operator
  $> x = c(\text{“}ROMA\text{”}, \text{“}MILANO\text{”}, \text{“}TORINO\text{”})$
- several functions in R to manipulate character vectors.
  paste, as.character, is.character, strsplit, substr. . .

$> paste(\text{“}HOME\text{”}, \text{“}WHILE\text{”}, \text{“}DOG\text{”}, sep = \text{“ : ”})$
$[1]\text{“}HOME : WHILE : DOG\text{”}$ **Concatenate char vectors**

$> x = 1 : 5$
$> is.character(x)$ **test if an object is of type character**
$[1] FALSE$

$> is.character(as.character(x))$
$[1] TRUE$

$> Y = paste(\text{“}HOME\text{”}, \text{“}WHILE\text{”}, \text{“}DOG\text{”}, sep = \text{“ : ”})$
$> strsplit(Y, split = \text{“}O\text{”})$ **split the elements of Y into sub-strings w.r.t split string**
$[[1]]$
$[1]\text{“}H\text{”}\text{“}ME : WHILE : D\text{”}\text{“}G\text{”}$

$> substr(Y, 5, 10)$ **Extract or replace sub-strings in a character vector.**
$[1]\text{” : }WHILE\text{”}$

Logical vector in R

- A logical vector is a vector whose elements are TRUE, FALSE or NA.
- it is generated by conditions:

  $> x$

  [1]1.4 6 23.1 65.43 2.7 55

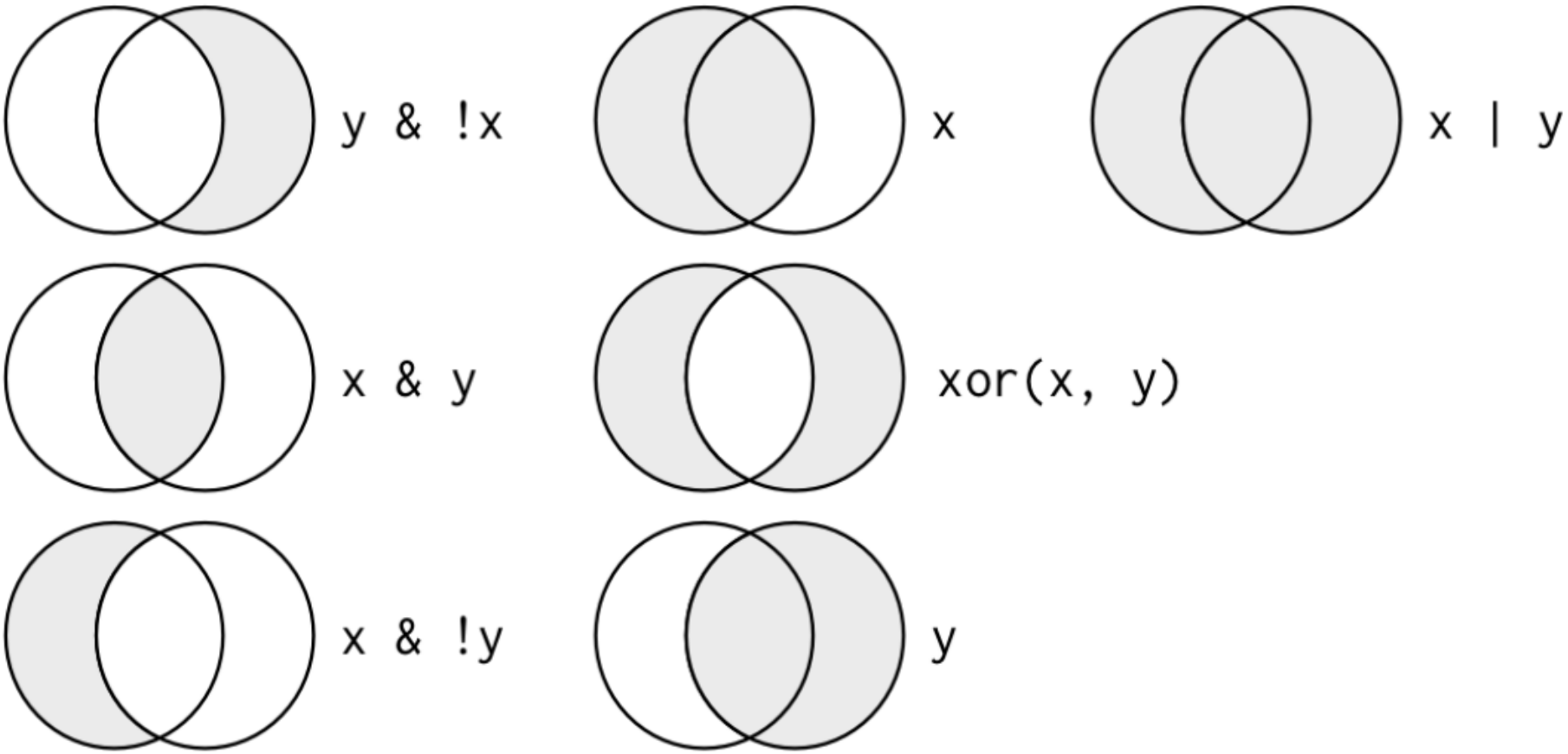
  $> logic = x > 34$

  [1]***FALSE FALSE FALSE TRUE FALSE TRUE***

  It compares each element of x with 34. It returns a vector the same length as x, with a value TRUE when the condition is met and FALSE when it is not.

- logical operators are >, >=, <, <=, ==, !=, &, |.

Logical operator



Logical operator

- Operator == can be used on floating-point numbers:

  $> sqrt(2)\hat{}2 == 2$

  [1] *FALSE*

  $> 1/49 * 49 == 1$

  [1] *FALSE*

- Computers use finite precision arithmetic so remember that every number you see is an approximation (use **options(digits=20)**).

- Instead of relying on ==, use near() in library **dplyr** :

  $> near(sqrt(2)\hat{}2, 2)$

  [1] *TRUE*

Factor in R

- A factor is a special type of vector used to a vector of data, usually taking a small number of distinct values. To store in statistical modelling data as factors insures that will be treated not as continuous variables but as categorical variable.
  - it is internally stored as a vector of integer values with a corresponding set of character values to use when the factor is displayed (an efficient way);
  - Factor's levels is always a character values;
- a factor is created as follows:

  $> f = factor(rep(c(\text{“}Control\text{”}, \text{“}Treated\text{”}), c(3,4)))$
  [1] *Control Control Control Treated Treated Treated Treated*
  Levels: Control Treated

- main factor operators:

  $> levels(f)$ **it returns the levels of a factor**

  $> summary(f)$ **it returns the frequencies associated with each level**

  $> str(f)$ **it returns a compact visualization of the factor**

Exercises on Vectors

1. Create a vector x with the following entries:
   $$3\ 4\ 1\ 1\ 2\ 1\ 6$$
   Check which elements of x are lower and equal to 2.
   Modify x so that all of the 1 values are changed to 0 values.

2. Create a vector y containing the elements of x that are greater than 2;

3. Create a sequence of numbers from 1 to 20 in steps of 0.25 and store in k.
   Change the elements in positions 4 and 5 with values 11 and 12;

4. Concatenate x and y into a vector called Vec;

5. Display all objects in the workspace and then remove Vec.

Exercises on Vectors

- Create a vector x with the following entries:

  3 4 1 1 2 1 6

  Check which elements of x are lower and equal to 2.
  Modify x so that all of the 1 values are changed to 0 values.

  $> x = c(3, 4, 1, 1, 2, 1, 6)$

  $> x <= 2$

  $> x[x == 1] = 0$

Exercises on Vectors

- Create a vector y containing the elements of x that are greater than 2;

  $> y = x[x > 2]$

  $> y$

  [1]3 4 6

Exercises on Vectors

- Create a sequence of numbers from 1 to 20 in steps of 0.25 and store in k. Change the elements in positions 4 and 5 with values 11 and 12

  $> k = seq(1, 20, by = 0.25)$

  $> k[c(4, 5)] = c(11, 12)$

Exercises on Vectors

- Concatenate x and y into a vector called Vec:

  $> Vec = c(x, y)$

  $> Vec$

  [1]3 4 1 1 2 1 6 3 4 6

Exercises on Vectors

- Display all objects in the workspace and then remove Vec.

  $> ls()$

  [1]"Vec" "x" "y" "z"

  $> rm(Vec)$

  $> rm(list = ls())$ **To remove all variables**