Chapitre 3 : Introduction to R

Armelle de le Court

20/04/2021

1: Creating objects in R

```
3+5
```

```
## [1] 8
```

```
# we can simply type math in the console
```

when we need to assign values to objects, <- is the assignment operator. (shortcut is alt + -)

```
weight_kg <- 55
weight_kg
```

```
## [1] 55
```

Naming variables, names cannot start with numbers, and there are some names already use for fundamental functions in R. Its also best to avoid dots.

When assigning a value to an object, r does not print anything, but we can force R to print the value by using () or by typing the object name

```
weight_kg <- 55 # doesnt print
(weight_kg <- 55) #it now has been printed
```

```
## [1] 55
```

now we can do arithmetic with our variable, for example, convert weight into pounds.

```
weight_lb <- 2.2*weight_kg
weight_lb
```

```
## [1] 121
```

2: Functions and their arguments

many functions are predefined or can be made avaible by importing R packages.

```
round(3.14159)
```

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

```
# automatically, it rounds to the nearest whole number.
# to see mode digits :
round(3.14159, digits = 2)
```

```
## [1] 3.14
```

```
# OR
round(3.14159, 2)
```

```
## [1] 3.14
```

3: Vectors and data types

We can assign a series of values to a vector using the c() function.

```
weight_g <- c(50,60,65,83)
weight_g
```

```
## [1] 50 60 65 83
```

```
# with characters it works too
molecules <- c("dna","rna","protein")
molecules
```

```
## [1] "dna"     "rna"     "protein"
```

Functions to inspect content of a vector

```
length(weight_g)
```

```
## [1] 4
```

```
length(molecules)
```

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

```
# class to know if either its a numeric or character vector
class(weight_g)
```

```
## [1] "numeric"
```

```
class(molecules)
```

```
## [1] "character"
```

```
# str for an overview of the structure of an object
str(weight_g)
```

```
##  num [1:4] 50 60 65 83
```

```
str(molecules)
```

```
##  chr [1:3] "dna" "rna" "protein"
```

We can use c() again to add other elements to our vector

```
weight_g <- c(weight_g, 90)
weight_g <- c(30, weight_g)
```

4: Subsetting vectors

If we want to extract values from a vector, we need to use brackets [] (shortcut is alt+maj+5)

```
molecules <- c("dna", "rna", "peptide", "protein")
molecules[2]
```

```
## [1] "rna"
```

```
# we extracted only the second value
molecules[c(3,2)]
```

```
## [1] "peptide" "rna"
```

```
# to extract more than one value, we need c()
```

We can repeat indices to create an object with more elements than the original one

```
more_molecules <- molecules[c(1, 2, 3, 2, 1, 4)]
more_molecules
```

```
## [1] "dna"     "rna"     "peptide" "rna"     "dna"     "protein"
```

it is also possible to get all the elements of a vector expect some specified using negative indices.

```
molecules[-1] # all but the first one
```

```
## [1] "rna"     "peptide" "protein"
```

```
molecules[-c(1,3)] # all but the first and the third one
```

```
## [1] "rna"     "protein"
```

5: Conditional subsetting

We can subset by using logical vector, TRUE select the element with the same index and FALSE will not.

```
weight_g <- c(21, 34, 39, 54, 55)
weight_g[c(TRUE, FALSE, TRUE, TRUE, FALSE)]
```

```
## [1] 21 39 54
```

but we usually dont type them but hand

```
# weight>50
# returns logical with TRUE for the indice higher than 50
```

to select only values above 50

```
weight_g[weight_g>50]
```

```
## [1] 54 55
```

"&" means that both conditions are true

"|" means that at least one condition is true (shortcut is alt+maj+l)

```
weight_g[weight_g < 30 | weight_g > 50]
```

```
## [1] 21 54 55
```

```
weight_g[weight_g >= 30 & weight_g == 21]
```

```
## numeric(0)
```

If we search fo a certains strings in a vector, we can use %‰in%‰, it will test if any of the elements of a search vector are found.

```
molecules <- c("dan", "rna", "protein", "peptide")
molecules[molecules == "rna" | molecules == "dna"]
```

```
## [1] "rna"
```

```
molecules %in% c("rna", "dna", "metabolite", "peptide", "glycerol")
```

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

```
molecules[molecules %in% c("rna", "dna", "metabolite", "peptide", "glycerol")]
```

```
## [1] "rna"     "peptide"
```

6: Names

We can anme each elements of a vector using the names() function.

```
x <- c(1, 5, 3, 5, 10)
names(x) <- c("A", "B", "C", "D", "E")
names(x)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
# we now have names
```

When a vector has names, we can access elements by their names.

```
x[c(1,3)]
```

```
## A C
## 1 3
```

```
# is the same as
x[c("A","C")]
```

```
## A C
## 1 3
```

7: Missing data

If the data includes NA, most of the functions will return NA. So we need to ignore them.

```
heights <- c(2, 4, 4, NA, 6)
mean(heights)
```

```
## [1] NA
```

```
# It returns NA bc one value is "NA"
mean(heights, na.rm = TRUE)
```

```
## [1] 4
```

```
# now the NA value is ignored
```

to extract elements which are not missing values

```
heights[!is.na(heights)]
```

```
## [1] 2 4 4 6
```

8: Generating vectors

There exist functions to generate vectors of different type. For a vecor of numerics:

```
numeric(3)
```

```
## [1] 0 0 0
```

```
numeric(10)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

Replicate elements

For exemple, if we want to repeat "-1" 5 times :

```
rep(-1,5)
```

```
## [1] -1 -1 -1 -1 -1
```

To repeat 1,2,3 five times :

```
rep(c(1,2,3),5)
```

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

To repeat 1,2 an 3 five times but with 5 1s, 5 2s and 5 3s in that order:

```
rep(c(1,2,3),each=5)
```

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

```
#OR
sort(rep(c(1,2,3),5))
```

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

Sequence generation

```
seq(from = 1, to = 20, by = 2)
```

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

default value of "by" is 1

```
seq(1,5,1)
```

```
## [1] 1 2 3 4 5
```

```
# is the same as
seq(1,5)
```

```
## [1] 1 2 3 4 5
```

```
# is the same as
1:5
```

```
## [1] 1 2 3 4 5
```

If we want a length of 3

```
seq(from = 1, to = 20, length.out = 3)
```

```
## [1]  1.0 10.5 20.0
```

Random samples and permutations

```
sample(1:10) # for numbers
```

```
##  [1]  4 10  2  8  1  6  5  3  9  7
```

```
sample(letters,5) # for letters
```

```
## [1] "m" "k" "y" "c" "l"
```

If we want an output larger than the input vector, we need to add "replace = TRUE"

```
sample(1:5, 10, replace = TRUE)
```

```
##  [1] 2 2 4 3 4 1 2 3 5 4
```

same set.seed repeat always the same random draw

```
set.seed(123)
sample(1:10)
```

```
##  [1]  3 10  2  8  6  9  1  7  5  4
```

```
set.seed(123)
sample(1:10)
```

```
##  [1]  3 10  2  8  6  9  1  7  5  4
```

rnorm

```
rnorm(5,2,2)
```

```
## [1] -0.5301225  0.6262943  1.1086761  4.4481636  2.7196277
```

```
# rnorm(n,mean,sd)
```

9: Additional exercises

Question 1

```
# create vectors x and y 1 to 10 and 10 to 1
x <- 1:10
y <- 10:1

#types ?
class(x)
```

```
## [1] "integer"
```

```
class(y) #theyre both integers
```

```
## [1] "integer"
```

```
#sum of each? identical ?

sumx<- sum(x)
```

```r
sumy <- sum(y)
identical(sumx,sumy) #its true so sum of x and y are the same
```

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

```r
#swap values of x and y
y2 <- y
y <- x
x <- y2
x
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```r
y #i swapped x and y valus
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Question 2

```r
# vector x with number 20 to 2, retrieve (=recup) elements larger than 5 or equal to 15

x <- (20:2)
x[x>5 | x<=15 ]
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2
```

```r
#remove the fisrt 8 elemnts from x and store in x2

x2 <- x[-c(1:8)]
x2
```

```
##  [1] 12 11 10  9  8  7  6  5  4  3  2
```

Question 3

```r
#counting from Monday to Friday how many molds you see in your cell cultures.
#vector "molds" with results 1, 2 , 5, 8, 10

molds <- c(1,2,5,8,10)
names(molds) <- c("monday","tuesday","wednesday","thursday","friday")

#extract the number of molds identified on Wednesday.
molds["wednesday"]
```

```
## wednesday
##         5
```

Question 4

```r
# mean of a random distribution N(15, 1) of size 100 and store it in variable m1
m1<- sample(15:1, 100, replace=TRUE)
mm1<- mean(m1)
```

```r
#mean of a random distribution N(0, 1) of size 100 and store it in variable m2
m2 <- sample(0:1, 100, replace=TRUE)
mm2 <- mean(m2)

#mean of another random distribution N(15, 1) of size 1000 and store it in variable m3
m3 <- sample(15:1, 1000, replace= TRUE)
mm3 <- mean(m3)

#which one of m1 and m2 will be larger
mm1>mm2 #TRUE so mm1 is larger than mm2
```

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

Question 5

```r
# simulate a set of 100 students voting (randomly) for 1, 2 or 3
std <- sample(1:3, 100, replace=TRUE)

#values as a table
table(std)
```

```
## std
##  1  2  3
## 29 32 39
```

```r
#number of stds with more than 1
#?
```

Question 6

```r
v1 <- c(1, 2, 3, "4")
v2 <- c(45, 23, TRUE, 21, 12, 34)
v3 <- c(v1, v2)

#
class(v3)
```

```
## [1] "character"
```

```r
length(v3)
```

```
## [1] 10
```

```r
names(v3) <- c(letters[1:10])
v3
```

```
##    a    b    c    d    e    f    g    h    i    j
##  "1"  "2"  "3"  "4" "45" "23"  "1" "21" "12" "34"
```

```r
# create v4 containing "2"   "1"   "NEW" "3"   "4" with v1
v4 <- c("2","1","NEW","3","4")
v4
```

```
## [1] "2"   "1"   "NEW" "3"   "4"
```

```r
#round pi o 2 decimals
round(pi, digits = 2)
```

```
## [1] 3.14
```

Question 7

```r
p1 <- c(1, 1, 1)
names(p1) <- c("A34", "D3", "F12")
p2 <- c(2, 2, 2, 2)
names(p2) <- c("W4", "A21", "K7", "K8")
p3 <- c(3, 3, 3, 3, 3, 3, 3)
names(p3) <- c("D1", "D2", "A10", "D5", "D15", "A16", "B22")

# command would you use to identify the number of respective answers
length(p1)
```

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

```r
length(p2)
```

```
## [1] 4
```

```r
length(p3)
```

```
## [1] 7
```

```r
#Concatenate all answers into a single vector p4.
p4 <- c(p1,p2,p3)
p4
```

```
## A34  D3 F12  W4 A21  K7  K8  D1  D2 A10  D5 D15 A16 B22
##   1   1   1   2   2   2   2   3   3   3   3   3   3   3
```

```r
#get the vote for student D2 from vector p4
p4["D2"]
```

```
## D2
##  3
```

Question 8

```r
test <- c(student1 = 12, student2 = 11, student3 = 4, student4 = 6, student5 = 7,
  student6 = 8.5, student7 = 13.5, student8 = 5.5, student9 = 13.5,
  student10 = 2.5, student11 = 17, student12 = 18, student13 = 15,
  student14 = 8, student15 = 7, student16 = 12, student17 = 18.5,
  student18 = 7.5, student19 = 13.5, student20 = 6, student21 = 9,
  student22 = 16, student23 = 8.5, student24 = 9, student25 = NA,
  student26 = NA, student27 = 14, student28 = 16.5, student29 = 12,
  student30 = NA, student31 = 12.5, student32 = 3, student33 = NA,
  student34 = 17, student35 = 16, student36 = 9, student37 = 6,
  student38 = 7, student39 = 8.5, student40 = 8.5, student41 = 8,
  student42 = 16.5, student43 = 4.5, student44 = NA, student45 = 8,
  student46 = 8, student47 = 7.5, student48 = 8.5, student49 = 2,
  student50 = 14, student51 = 6.5, student52 = 12, student53 = 16.5,
```

```
  student54 = 7, student55 = 9.5, student56 = 12, student57 = 8.5,
  student58 = 15.5, student59 = 9, student60 = 13.5, student61 = 18,
  student62 = 12.5, student63 = 19.5, student64 = 13, student65 = 17.5,
  student66 = 8.5, student67 = 9, student68 = 7, student69 = 12.5,
  student70 = NA, student71 = 19, student72 = 11.5, student73 = 9,
  student74 = 9.5, student75 = 12, student76 = 11, student77 = 12,
  student78 = 14, student79 = 17, student80 = 8.5, student81 = 10,
  student82 = 10, student83 = NA, student84 = 10.5, student85 = 14,
  student86 = 7.5, student87 = 4, student88 = 9, student89 = 6.5,
  student90 = 10.5, student91 = 9.5, student92 = 13, student93 = 11.5,
  student94 = NA, student95 = 6, student96 = 12.5, student97 = 11.5,
  student98 = 4, student99 = 11.5, student100 = 8)

#number of students that have a mark > 10?

test1 <- test[!is.na(test)]
more_than_10 <- test1[test1>10]
length(more_than_10)
```

```
## [1] 45
```

```
# number of students that have a mark greater than the average score

mean(test1)
```

```
## [1] 10.6087
```

```
more_than_avg <- test1[test1>10.6087]
length(more_than_avg)
```

```
## [1] 43
```