Aula 1.2 - classes básicas de objetos e vetores

Elder Sodré

Monday, February 23, 2015

### Classes básicas de objetos

O R possui 5 classes básicas de objetos:

* character (caracter; texto)
* numeric (números reais)
* integer (números inteiros)
* complex (números complexos)
* logic (lógica; verdadeiro ou falso)

Textos devem estar sempre envolvidos por aspas; caso contrário o R os tratará como objetos.

a <- 23  
a

## [1] 23

"a"

## [1] "a"

b <- "a"  
b

## [1] "a"

class(b)

## [1] "character"

Na maioria das vezes, R trata os números como números reais.

c <- 20  
d <- 14.7  
  
class(c)

## [1] "numeric"

class(d)

## [1] "numeric"

Para dizer explicitamente que você quer um número inteiro, basta acrescentar o sufixo L

e <- 34L  
class(e)

## [1] "integer"

Números complexos são formados adicionando-se i a um número real

f <- 12+3i  
class(f)

## [1] "complex"

Um número especial é o Inf (infinito). Inf pode ser usado em cálculos e aparecer como resposta em cálculos.

1/0

## [1] Inf

1/Inf

## [1] 0

Inf+Inf

## [1] Inf

Outro número especial é NaN (not a number; não é um número). Este número aparece quando divide-se 0 por 0.

0/0

## [1] NaN

Objetos de lógica são representados por TRUE/FALSE, ou, resumidamente T/F.

TRUE

## [1] TRUE

FALSE

## [1] FALSE

T

## [1] TRUE

F

## [1] FALSE

Lógicas aparecem como resultados de operadores lógicos, como > e ==

45>9 #Maior que

## [1] TRUE

23<10+3 #Menor que

## [1] FALSE

2>=2 #Maior ou igual a

## [1] TRUE

8<=2\*10 #Menor ou igual a

## [1] TRUE

10+3!=14-1 #Diferente de

## [1] FALSE

4+2==7 #Igual a

## [1] FALSE

IMPORTANTE! Deve-se usar == para representar o operador lógico. O sinal de igual sozinho (=) tem outra função no R.

### Vetores

Vetor é um tipo de objeto composto por uma sequência de elementos. Todos os elementos de um vetor precisam ser do mesmo tipo. A exceção desta regra é um tipo de vetor chamado lista. Elementos da lista podem ser de qualquer tipo (inclusive outra lista).

#### Criando vetores

Como juntar dois ou mais objetos formando um vetor?

Com a função c(), onde c vem se concatenar, ou combinar

a <- c(3,4.5,0)   
b <- c("Elder","é muito","legal","!")  
c <- c(T,T,F,T)

Objetos também podem ser concatenados em um vetor.

d <- c(-2,a,67,8,a)  
print(d)

## [1] -2.0 3.0 4.5 0.0 67.0 8.0 3.0 4.5 0.0

Lembra que todos os elementos de um vetor precisam ser do mesmo tipo? O que acontece quando junto elementos de tipos diferentes em um vetor? Vamos ver!

v1 <- c(4.7,10L) #Numeric + Integer  
v2 <- c(4.6,9-2i) #Numeric + complex  
v3 <- c(1,F) #Numeric + Logic  
v4 <- c(8L,"Elder") #Integer + Character  
v5 <- c(TRUE, "olá") #Logic + Character  
v6 <- c(FALSE, TRUE, 4) #Logic + Numeric  
  
class(v1)

## [1] "numeric"

class(v2)

## [1] "complex"

class(v3)

## [1] "numeric"

class(v4)

## [1] "character"

class(v5)

## [1] "character"

class(v6)

## [1] "numeric"

v1

## [1] 4.7 10.0

v2

## [1] 4.6+0i 9.0-2i

v3

## [1] 1 0

v4

## [1] "8" "Elder"

v5

## [1] "TRUE" "olá"

v6

## [1] 0 1 4

Quando tentamos juntar objetos de tipos diferentes no mesmo vetor, o R usa de "coerção" para transformar os objetos para que fiquem do mesmo tipo. O R usa esta ordem automática de coerção:

logic-->integer-->numeric-->complex-->character

Para alterar o tipo de um vetor, basta usar funções começadas com as. : as.integer() as.numeric() as.complex() as.character()

OBS: Funções começadas com is. servem para perguntar se o vetor é daquele determinado tipo is.integer() is.numeric() is.complex() is.character()

Uma forma comum de se criar um vetor é com o uso de dois pontos(:). Desta forma cria-se uma sequência do primeiro ao último número.

1:10

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

8:287

## [1] 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
## [18] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41  
## [35] 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58  
## [52] 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75  
## [69] 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92  
## [86] 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109  
## [103] 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126  
## [120] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
## [137] 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160  
## [154] 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177  
## [171] 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194  
## [188] 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211  
## [205] 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228  
## [222] 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245  
## [239] 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262  
## [256] 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279  
## [273] 280 281 282 283 284 285 286 287

10:-10

## [1] 10 9 8 7 6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6  
## [18] -7 -8 -9 -10

7.8:16.2

## [1] 7.8 8.8 9.8 10.8 11.8 12.8 13.8 14.8 15.8

A função seq() é semelhante, mas ela permite que você determine o intervalo entre os números na sequência.

seq(1,10) #resultado idêntico a 1:10

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

seq(3,-3,by=-0.5)

## [1] 3.0 2.5 2.0 1.5 1.0 0.5 0.0 -0.5 -1.0 -1.5 -2.0 -2.5 -3.0

seq(1,2,by=0.03)

## [1] 1.00 1.03 1.06 1.09 1.12 1.15 1.18 1.21 1.24 1.27 1.30 1.33 1.36 1.39  
## [15] 1.42 1.45 1.48 1.51 1.54 1.57 1.60 1.63 1.66 1.69 1.72 1.75 1.78 1.81  
## [29] 1.84 1.87 1.90 1.93 1.96 1.99

A função rep() cria um vetor através da repetição de um vetor ou número.

rep(0,5)

## [1] 0 0 0 0 0

rep(7,3)

## [1] 7 7 7

t <- c(1,2,3)  
rep(t,3)

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

rep(t,each=3)

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

rep(c(T,F),5)

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

Outra função para criar um vetor é vector.

vector() #Vetor vazio.

## logical(0)

Também é possível com esta função especificar o tipo de vetor (logical é o default) e o comprimento.

vector(length=6)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE

vector(mode="numeric",length = 10)

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

Duas funções pra criar números aleatórios: runif(distribuição uniforme) e rnorm(distribuição normal):

runif(n=20,min=3,max=5) #Cria 20 números aleatórios com distribuição normal, entre 3 e 5

## [1] 4.147 3.881 4.212 3.317 4.538 3.312 4.612 3.550 3.879 3.987 4.287  
## [12] 3.332 3.164 4.365 4.036 4.827 3.156 4.427 4.403 3.080

rnorm(n=10,mean=0,sd=0.5) #Cria um vetor com dez números aleatórios, de média 0 e desvio padrão 0.5

## [1] -0.65694 -0.23510 0.61587 -0.14237 0.49256 -0.45364 -0.05945  
## [8] -0.29833 0.07266 -0.41171

##### Algumas funções interessantes para vetores numéricos:

x <- rnorm(n=100,mean=30,sd=5)  
min(x) #Valor mínimo

## [1] 19.49

max(x) #Valor máximo

## [1] 43.19

mean(x) #Média

## [1] 30.47

sd(x) #Desvio padrão

## [1] 5.061

Mais algumas funções úteis:

Três funções úteis para vetores são class(), length() e names()

vet1 <- c("a","b","c")  
vet2 <- c(9,12,90,0,5)  
  
class(vet1)

## [1] "character"

class(vet2) #Pergunta a classe do vetor

## [1] "numeric"

length(vet1)

## [1] 3

length(vet2) #Pergunta o comprimento do vetor (número de elementos)

## [1] 5

Ao criar um vetor com a função c(), pode-se dar nomes aos elementos:

notas <- c(Juca=9.5,Juju=6.8,Zeca=5,Mina=10,Rosinha=9)  
notas

## Juca Juju Zeca Mina Rosinha   
## 9.5 6.8 5.0 10.0 9.0

names(notas)

## [1] "Juca" "Juju" "Zeca" "Mina" "Rosinha"

As funções length() e names() também podem ser usadas para alterar o nome e o comprimento do vetor.

length(vet1) <- 10  
vet1

## [1] "a" "b" "c" NA NA NA NA NA NA NA

length(vet2) <- 3  
vet2

## [1] 9 12 90

names(vet2) <- c("A","B","C")  
vet2

## A B C   
## 9 12 90

##### Operações com vetores

Vamos olhar as operações abaixo:

1:4+c(10,5,0,-5)

## [1] 11 7 3 -1

Quando os dois vetores na operação não possuem o mesmo comprimento, o R "recicla", ou seja, repete, o vetor menor até que o comprimento do vetor maior seja alcançado.

rep(3,4)\*c(10,0)

## [1] 30 0 30 0

10:15/2

## [1] 5.0 5.5 6.0 6.5 7.0 7.5

#A expressão acima, para o R, é igual a:  
10:15/c(2,2,2,2,2,2)

## [1] 5.0 5.5 6.0 6.5 7.0 7.5

Se o comprimento do vetor maior não for múltiplo do vetor menor, o R dá uma mensagem de erro mas faz o cálculo mesmo assim:

c(1,5,3,6,7,4,5)-c(12,11)

## Warning: comprimento do objeto maior não é múltiplo do comprimento do  
## objeto menor

## [1] -11 -6 -9 -5 -5 -7 -7