Workshop R

JEST

29/03/2018

Introdução ao R

O R é um sistema de computação científico e estatístico, programável e que permite o tratamento de vários tipos de dados. Esta linguagem contém um conjunto de ferramentas que nos permite efetuar cálculos, armazenamento, processamento, analise e visualização de dados. Hoje em dia os *Datasets* surgem de várias fontes de informação nos mais diversos formatos e o primeiro passo é tornar essa informação analizável para poder ser processada. O R é uma linguagem bastante poderosa para tratamento estatístico de dados e por isso mesmo é importante consolidar os conceitos básicos para poder seguir para os mais avançados.

Atribuição de valores a um objecto

A operação mais básica que se pode realizar sobre um objecto em R é a atribuição. Para esse efeito utiliza-se o operador "<-"

```
x <-  3
print(x)
```

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

```
x #podemos também imprimir variaveis sem utilizar a função print();
```

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

```
y <- 'variavel'
print(y)
```

```
## [1] "variavel"
```

```
xfloat <-  3.2
print(xfloat)
```

```
## [1] 3.2
```

```
bol <-  TRUE
bol
```

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

Operações Aritméticas

```
x <- 9
y <- 2
x + y     # adição
```

```
## [1] 11
```

```
x - y     # subtração
```

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

```
x * y     # multiplicação
```

```
## [1] 18
```

```
x / y     # divisção
```

```
## [1] 4.5
```

```
x ^ y     # (ou x ** y)     x elevado a y
```

```
## [1] 81
```

```
x %% y    # resto da divisão
```

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

```
x %/% y   # divisão arredondada por defeito
```

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

Operadores lógicos

O símbolo & performa a operação AND entre os dois operandos.

```
TRUE & TRUE
```

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

```
TRUE & FALSE
```

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

```
FALSE & TRUE
```

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

```
FALSE & FALSE
```

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

O símbolo | performa a operação OR entre os dois operandos.

```
TRUE | TRUE
```

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

```
TRUE | FALSE
```

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

```
FALSE | TRUE
```

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

```
FALSE | FALSE
```

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

O símbolo "!" contraria a expressço à sua frente.

```
!(FALSE & FALSE)
```

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

Operadores relacionais

Para comparar dois valores faz-se uso dos operadores relacionais. Esta operação devolve como output um dos valores lógicos, TRUE se é verdadeira, ou FALSE se é falsa.

```
x == y #para ver se os valores são iguais
```

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

```
x != y #para ver se são diferentes
```

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

```
x < y  #para ver se x é menor que y
```

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

```
x > y  #para ver se x é maior que y
```

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

```
x <= y #para ver se x é menor ou igual a y
```

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

```
x >= y #para ver se x é maior ou igual a y
```

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

Vetores

O R é orientado aos vectores que constituem objectos que permitem guardar uma sequência de valores. Existem vários tipos de vectores: Numéricos, Lógicos e de Strings.

Vetores Numericos

A função c() permite juntar (concatenar) um vetor com novos elementos (ou com outros vetores).

```
primeiro <- c(1,2,3)
segundo <- c(4,5,6)
vecFinal <- c(primeiro,segundo)
vecFinal
```

```
## [1] 1 2 3 4 5 6
```

Outra forma de criar um vector numéricos por ser através da geração de uma sequência de números.

```
1:10
```

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

```
20:10
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10
```

```
seq(1,2,0.2) #por definição a ordem dos argumentos de entrada na função seq é: from, to, by. Se especif
```

```
## [1] 1.0 1.2 1.4 1.6 1.8 2.0
```

```
v <- rep(0,10); #Esta é uma maneira possivel de criar um vetor de zeros. No primeiro argumento indicamo
v
```

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

É possível realizar operações aritméticas sobre vetores de forma bastante simples.

```
contasAritmeticas <- 2:8
2*contasAritmeticas+3    #As operações são feitas para todos os elementos do vetor
```

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

Operações aritméticas entre vetores também é possivel. Um operação aritmética entre dois vectores realiza a operação entre cada par de valores nas mesmas posições de ambos os vetores. Se os vetores não tiverem o mesmo tamanho, o mais pequeno será reciclado, isto é, será repetido enquanto for necessário. Quando o tamanho do vector não for multiplo do menor, deverá aparecer uma mensagem de erro.

```
maior <- 1:10
menor <- 1:5
maior+menor
```

```
##  [1]  2  4  6  8 10  7  9 11 13 15
```

Acesso aos elementos de vectores

Para consultarmos um elemento de um vetor, escrememos o nome do vector e entre parênteses retos, o indice. Assim se quisermos aceder a um elemtento do vetor X na posiçao i, escrevemos X[i].

```
X <- 1:10
X[3]  #aceder ao elemento na posição 3 do vetor
```

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

É também possível a indexação ser feita com um vetor sendo que, neste caso, é retornado um vector com os valores do vetor original nas posições do vetor indice.

```
#Utilizando o vector X
X[c(4:9)]
```

```
## [1] 4 5 6 7 8 9
```

```
X[4:9]
```

```
## [1] 4 5 6 7 8 9
```

Para remover um elemento podemos especificar a posição do elemento que queremos eliminar e adicionar o operador '-' imediatamente antes

```
numeros <- 1:10
numeros[-2]
```

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

Atribuir Nomes a vetores

O R permite-nos atribuir nomes a cada uma das posições do vector e depois usar esses nomes para aceder aos seus valores. Para dar nomes aos valores do vetor utiliza-se a função *names()*

```
key <- c('batatas','cenouras','tomates')
values <- c(80,12,20)
names(values) <- key
values
```

```
##  batatas cenouras  tomates
##       80       12       20
```

```r
values['batatas']
```

```
## batatas
##      80
```

Algumas funções que podem ser aplicadas em vetores numéricos

```r
v <- c(-1,-2,3,4)
abs(v) #Valor absoluto (Módulo)
```

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

```r
sqrt(abs(v)) #Raiz quadrada
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

```r
sin(pi/2*v) #seno
```

```
## [1] -1.000000e+00 -1.224647e-16 -1.000000e+00 -2.449294e-16
```

```r
rnd <- c(3,345,5,2221,4.498628,1.987654321,2.123)
round(rnd,2) #Arredonda o valor a um determinado número de casas decimais a definir pelo segundo argume
```

```
## [1]    3.00  345.00    5.00 2221.00    4.50    1.99    2.12
```

```r
v <- c(4,7,2,3,8,3,3,3,1,9,3,2,4)
rev(v) #Inverte o vector
```

```
##  [1] 4 2 3 9 1 3 3 3 8 3 2 7 4
```

```r
sort(v) #Ordena o vetor por ordem crescente
```

```
##  [1] 1 2 2 3 3 3 3 3 4 4 7 8 9
```

```r
sort(v,decreasing <- TRUE) #Ordena o vetor por ordem decrescente
```

```
##  [1] 9 8 7 4 4 3 3 3 3 3 2 2 1
```

```r
order(v) #Retorna a ordem do vetor pelos seus indices
```

```
##  [1]  9  3 12  4  6  7  8 11  1 13  2  5 10
```

```r
v[order(v)] #Ao passarmos o vector ordenado por indices, como indice, o resultado vai ser o vector orde
```

```
##  [1] 1 2 2 3 3 3 3 3 4 4 7 8 9
```

```r
unique(v) #Retorna um vetor apenas com os elementos únicos
```

```
## [1] 4 7 2 3 8 1 9
```

```r
cumsum(v) #Faz a soma cumulativa
```

```
##  [1]  4 11 13 16 24 27 30 33 34 43 46 48 52
```

```r
cumprod(v) #Faz o produto cumulativo
```

```
##  [1]       4      28      56     168    1344    4032   12096   36288
##  [9]   36288  326592  979776 1959552 7838208
```

```r
v <- c(7,2,9,3,12)
sum(v) #Somatório dos valores do vetor
```

```
## [1] 33
```

```r
mean(v) #media dos valores do vetor
```

```
## [1] 6.6
```

```r
min(v) #Menor valor do vetor
```

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

```r
which.min(v ) #Indice do menor valor do vetor
```

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

```r
max(v)#Maior valor do vetor
```

```
## [1] 12
```

```r
which.max(v) #Indice do maior valor do vetor
```

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

```r
which(v>=7) #Indices dos valores maiores ou igual a 7
```

```
## [1] 1 3 5
```

Exercicios

1.1) Defenir um vetor v com os primeiros 30 numeros naturais

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30
```

1.2) Calcular o vetor resultante da soma da constante 5 com cada elemento do vetor v.

```
##  [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## [24] 27 28 29 30 31 32 33
```

1.3) Calcular o vetor 1/v.

```
##  [1] 1.00000000 0.50000000 0.33333333 0.25000000 0.20000000 0.16666667
##  [7] 0.14285714 0.12500000 0.11111111 0.10000000 0.09090909 0.08333333
## [13] 0.07692308 0.07142857 0.06666667 0.06250000 0.05882353 0.05555556
## [19] 0.05263158 0.05000000 0.04761905 0.04545455 0.04347826 0.04166667
## [25] 0.04000000 0.03846154 0.03703704 0.03571429 0.03448276 0.03333333
```

2.1) Definir um vetor com uma sequência de números ímpares menores do que 30

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
```

2.2) Calcular a soma de todos os valores do vetor anteriormente definido

```
## [1] 225
```

2.3) Selecionar os elementos do vetor múltiplos de 3

```
## [1]  3  9 15 21 27
```

Vectores Lógicos

Outro tipo de dados importante no R são os vetores lógicos. Os vetores logicos contêm valores do tipo *boolean* e em alguns casos são uma exelente abordagem para encontrar determinados valores. Vejamos um exemplo.

```r
v <- 10:20
vBoolean <- v>=18
vBoolean
```

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

```r
v[vBoolean]
```

```
## [1] 18 19 20
```

```r
which(vBoolean)
```

```
## [1]  9 10 11
```

Vetores de Strings

Vetores de strings são um tipo de dados em R que nos permite guardar sequencias de caracteres, também designadas por strings.

```r
nomes <- c("Grifo","Joao","Poço","Macario","Leitão","Grifo")
nomes[4]
```

```
## [1] "Macario"
```

```r
nomes[1]==tail(nomes,n=1)
```

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

```r
#Poderiamos ter feito nomes[1]==nomes[6] ou nomes[1]==nomes[length(nomes)]
#A função tail(<vetor>,<n elementos>) vai buscar os n últimos elementos ao vetor indicado
```

Algumas funções que se podem usar para trabalhar com strings

```r
nome <- "Guilherme"
apelido <- "Cruz"
toupper(nome)  #Transforma todos os caracteres da string em letras maiúsculas
```

```
## [1] "GUILHERME"
```

```r
tolower(nome)  #Transforma todos os caracteres da string em letras minúsculas
```

```
## [1] "guilherme"
```

```r
paste(apelido,nomes) #Imprime todas as strings que forem passadas como argumento
```

```
## [1] "Cruz Grifo"   "Cruz Joao"    "Cruz Poço"    "Cruz Macario"
## [5] "Cruz Leitão"  "Cruz Grifo"
```

```r
nchar(apelido) #Retorna o numero de caracteres da strings passada como argumento
```

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

```r
sub("z","zes",apelido) #Substitui a ocurrência passada no primeiro argumento pela string passada no seg
```

```
## [1] "Cruzes"
```

```r
substr(nome,4,7) #Retorna um subtring da string passado no primeiro argumento. O recorte é feito desde
```

```
## [1] "lher"
```

```
str <- "O meu nome é Grifo"
strsplit(str,' ') #"Parte" a string original em várias strings de acordo com o separador indicado no se
```

```
## [[1]]
## [1] "O"     "meu"   "nome"  "é"     "Grifo"
```

```
grep("ifo",nomes) #Permite identificar a ocurrencia de string/substrings no vetor passado como segundo
```

```
## [1] 1 6
```

```
#Explicar a função mapply()
```

Limpeza de dados

Por vezes quando estamos a trabalhar com dados importados, eles não estão trataveis. Valores omissos são muitas das vezes a causa disso mesmo e temos de aprender a lidar com este problema.

```
v <-  c(1,2,NA,NA,NA,7,2,3,NA,3,NA,4,NA,NA,4,6,2,NA,1)
v
```

```
##  [1]  1  2 NA NA NA  7  2  3 NA  3 NA  4 NA NA  4  6  2 NA  1
```

```
v[is.na(v)] #Para mostrar os indices dos valores nulos: which(is.na(v))
```

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

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

```
##  [1] 1 2 7 2 3 3 4 4 6 2 1
```

Exercicios

3) Supondo que os consumos de energia eléctrica do Diogo Poço em 2017 foi: 230, 210, 198, 160, 150, 153, 149, 149, 151, 169, 195, 223 KWh's. Analisar os dados e determinar: 3.1) O consumo total do Diogo em 2017

```
## [1] 2137
```

3.2) O valor mínimo e o valor máximo dos consumos de energia do Diogo. Para o valor máximo calcule o mês em que ocorreu esse gasto.

```
## [1] 149
```

```
## [1] 230
```

4) Atribuir um nome(letras minúsculas) a uma variavél e de seguida substitua a primeira letra no nome de minúscula para maiúscula.

```
## [1] "Nome: macario"
```

```
## [1] "Resultado: Macario"
```

5) Criar três vectores, um para as idades, outro para os nomes e o ultimo para os pesos e de seguida crie um vetor de *strings* com os elementos na seguinte forma: "idade - nome - peso". Para tal utilize a função *paste()* (Nota: Utilizar a documentação se necessário).

```
## [1] "23 - Pedrosa - 70" "28 - Navega - 80"  "21 - Poço - 63"
```