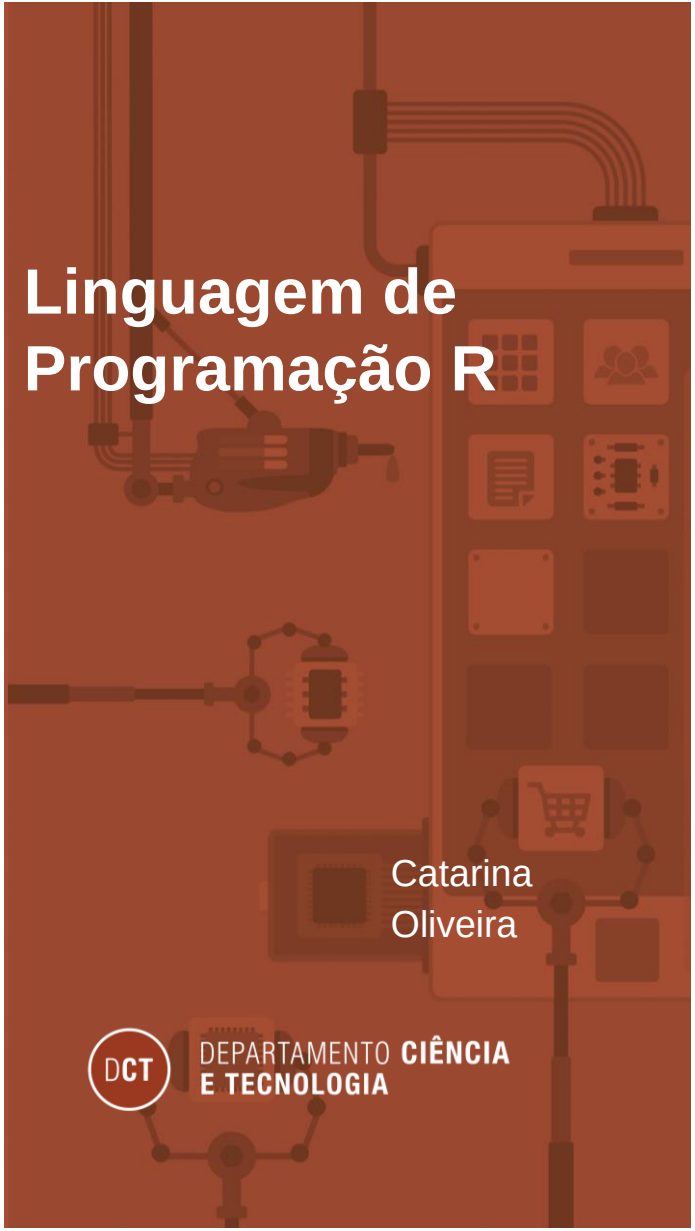


UNIVERSIDADE PORTUCALENSE

CONTEÚDO

R

O R é uma linguagem e ambiente para computação estatística e elaboração de gráficos

Inclui:

- Habilidade de tratar e guardar dados eficazmente
- Um conjunto de operadores para efetuar cálculos em vetores e matrizes
- Um conjunto grande e integrado de ferramentas de análise de dados
- Capacidades gráficas para análise e visualização de dados
- Estruturas condicionais, repetitivas
- Possibilidade de definir funções
- Capacidade de input e output

Instalação do ambiente

Windows

- R: https://cran.r-project.org/bin/windows/
- Rstudio: https://download1.rstudio.org/desktop/windows/RStudio-1.3.1093.exe

Mac

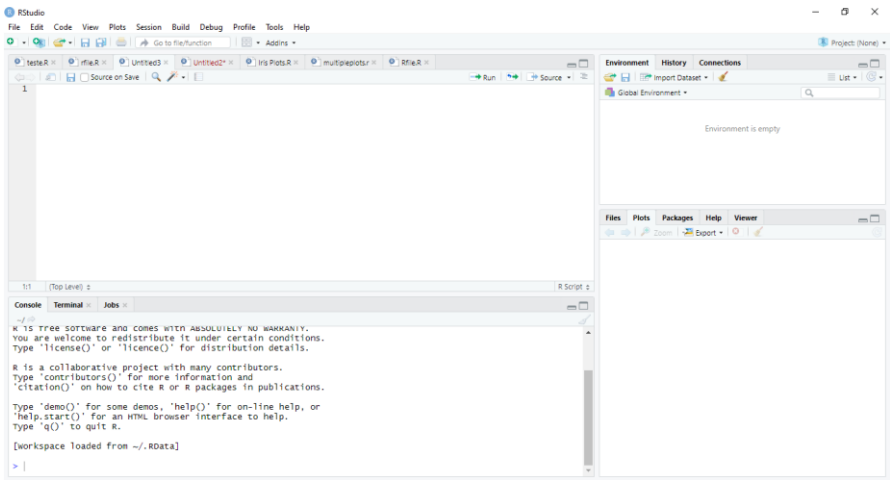
- R: https://cran.r-project.org/bin/macosx/
- Rstudio: https://download1.rstudio.org/desktop/macos/RStudio-1.3.1093.dmg

Linux

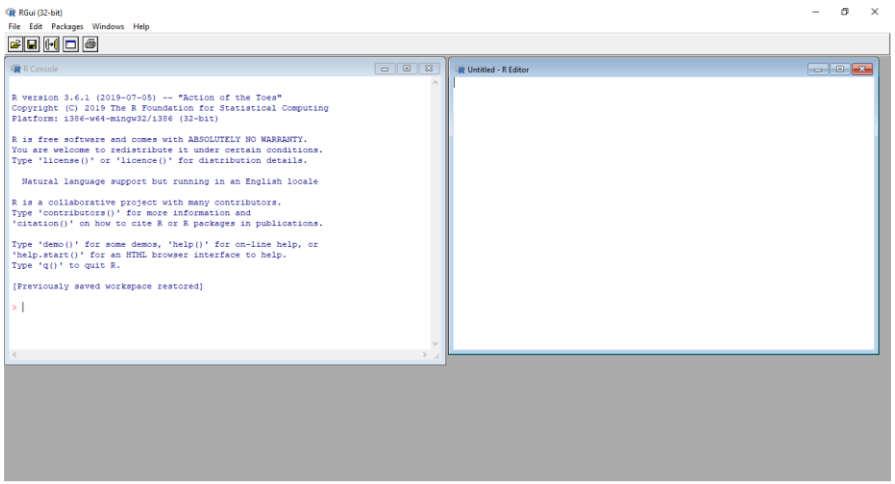
- R: https://cran.r-project.org/bin/linux/
- Rstudio (depende da versão do Linux)

Ambiente de desenvolvimento

RStudio



R GUI



Comandos úteis

Workspace

```
getwd ()                    # mostra a pasta em que estamos a trabalhar
setwd (" path ")            # define a pasta em que estamos a trabalhar
```

Variáveis

```
ls ()                       # mostra a lista de objetos (variáveis, funções)
rm( object_name )           # remove do ambiente o objeto chamado object_name
```

Exemplos e ajuda

```
example ( package_name )    # mostra exemplos do pacote chamado package_name
help ( package_name )       # mostra ajuda do pacote chamado package_name
?nome_funcao                # mostra ajuda para a função chamada nome_funcao
```

Cheat Sheet básica: https://rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf

Cheat Sheets: https://rstudio.com/resources/cheatsheets/

Variáveis

Variável: posição de memória usada para guardar informação. A informação pode ser alterada

Exemplos

- Nomes de variáveis válidos

```
division
Quadrado
.sub.multiplication    # começar com ponto seguido de letra. Variável invisível para o ls()
accumulative_sum
Sum5
```

- Nomes de variáveis inválidos

```
tot@l                  # utilização de carateres especiais
5um                    # começar com um número
_fine                  # começar com underscore
FALSE                  # palavra reservada
.0three                # começar com ponto seguido de número
```

Constantes

Constante: posição de memória usada para guardar informação. A informação **não** pode ser alterada

Exemplos

- Numéricas

```
typeof (2)
typeof (2L)          # L - long (inteiros grandes)
typeof (2i)          # números complexos
```

- Carateres (indiferente usar plicas ou aspas)

```
typeof ('example')
typeof ("2")
```

Operadores

O R conta com diversos operadores para executar diferentes operações

Tipos:

- Atribuição
- Aritméticos
- Relacionais
- Lógicos

Operadores de atribuição

Usados para atribuir valores a variáveis

Operator	Description
<−	Leftwards assignment
<<−	Leftwards assignment (global assignments — global variables)
=	Leftwards assignment
−>	Rightwards assignment (rarely used)
−>>	Rightwards assignment (rarely used)

Variáveis globais: declaradas fora de qualquer função e podem ser acedidas em qualquer função do programa

Variáveis locais: declaradas dentro de uma função, e apenas podem ser usadas dentro dessa função

Exemplos

```
x <- 20
x = 30
5 -> x
```

Operadores aritméticos

Usados para operações aritméticas

Operator	Description
+	Addition
−	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

Exemplos

```
x <- 20
y <- 15
x + y
x - y
x * y
y / x
y %/% x
y %% x
y ^ x
```

Operadores relacionais

São usados para fazer comparações entre valores

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
! =	Not equal to

Exemplos

```
x <- 20
y <- 15
x < y
x > y
x <= 15
```

Operadores Lógicos

São usados para efetuar operações lógicas

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

Operador lógico "element-wise": combina cada elemento do primeiro vetor com o elemento correspondente do segundo vetor e devolve o output

Operador lógico: utiliza o primeiro elemento de cada vetor

Exemplos

```
x <- c(TRUE ,FALSE ,0 ,3)
y <- c(FALSE ,TRUE ,FALSE , TRUE )
!x
x & y
x || y
```

Precedência e associatividade

Operator	Description	Associativity
^	Exponent	Right to Left
$-x, +x$	Unary minus, Unary plus	Left to Right
%%	Modulus	Left to Right
$*, /$	Multiplication, Division	Left to Right
$+, -$	Addition, Subtraction	Left to Right
$<, >, <=, >=, ==, !=$	Comparisions	Left to Right
!	Logical NOT	Left to Right
&, &&	Logical AND	Left to Right
|, ||	Logical OR	Left to Right
$->, ->>$	Rightward assignment	Left to Right
$<-, <<-$	Leftward assignment	Right to Left
$=$	Leftward assignment	Right to Left

Exemplos

Precedência

```
3 + 4 / 2
(3 + 4) / 2
```

Associatividade

```
3 / 4 / 2
3 / (4 / 2)
```

Objetos complexos

O R conta com diferentes tipos de objetos complexos

Categorias

- Vetor
- Matriz
- Lista
- *Data frame*
- *Factor*

Vetor

Um vetor é um tipo básico de estrutura de dados que contém elementos do mesmo tipo. Os tipos de dados podem ser lógicos, inteiros, reais, caracteres, etc.

Os vetores são criados utilizando a função c()

O tamanho de um vetor pode ser obtido utilizando a função length()

O tipo de um vetor pode ser obtido usando a função typeof()

Exemplos com vetores

Criação de vetores usando a função c()

```
x <- c(1, 5, 4, 9, 0)
x <- c(1, 5.4 , TRUE , " hello ")
```

Criação de vetores usando o operador ":"

```
x <- 1:7
y <- 2:-2
```

Criação de vetores usando a função seq()

```
seq (1, 3, by =0.2)          # pelo tamanho do intervalo
seq (1, 5, length.out =4)    # pelo tamanho do output
```

Funções para vetores

```
length(x)                    # mostra o nº de elementos do vetor
sort(x, decreasing = TRUE)   # por ordem decrescente. Para crescente usar decreasing = TRUE
unique(x)                    # mostra valores únicos do vetor
table(x)                     # calcula frequências dos elementos do vetor
```

Exemplos com vetores

Leitura usando vetor lógico como índice

```
x <- seq (-3, 3, 2)
x[c(TRUE , FALSE , FALSE , TRUE )]
x[x < 0]
x[x > 0]
```

Leitura usando vetor de carateres como índice

```
x <- c(first=3, second=0, third=9)
names (x)
x["second"]
x[c(" first ", " third ")]
```

Exemplos com vetores

Modificar um vetor

```
x <- seq (-3, 2, 1)
x [2] <- 0
x[x <0] <- 5
x <- x [1:4]
```

Eliminar um vetor

```
rm(x) # remove variable
```

Matriz

Uma matriz é uma estrutura de dados de duas dimensões

A matriz é semelhante ao vetor, mas contém ainda o atributo dimensão que pode ser verificado com a função attributes()

Exemplos com matrizes

Criar uma matriz usando a função matrix()

```
a <- matrix (1:9 , nrow = 3, ncol = 3)
b <- matrix (1:9 , nrow = 3)
c <- matrix (1:9 , nrow =3, byrow = TRUE )

# Change column and row names
x <- matrix (1:9 , nrow = 3, dimnames = list (c("X","Y","Z"), c("A","B","C")))
colnames (x)
rownames (x)
colnames (x) <- c("C1","C2","C3")
rownames (x) <- c("R1","R2","R3")
```

Criação de matriz usando as funções cbind() e rbind()

```
a <- cbind (c(1 ,2 ,3) ,c(4 ,5 ,6))
b <- rbind (c(1 ,2 ,3) ,c(4 ,5 ,6))
```

Criação de matriz usando a função dim()

```
x <- c(1 ,2 ,3 ,4 ,5 ,6)
dim(x) <- c(2 ,3)
```

Exemplos com matrizes

Leitura usando uma matriz de inteiros como índice

```
x <- matrix (1:9 , nrow = 3, ncol = 3)

x[1 ,]                    # select first row
x[, 1]                    # select first column
x[,]                      # leaving row as well as column field blank will select entire matrix
x[-1 ,]                   # select all rows except first

x[c(1 ,2) ,c(2 ,3)]       # select rows 1 & 2 and columns 2 & 3
x[c(1 ,2) ,]              # leaving column field blank will select entire columns
```

Exemplos com matrizes

Leitura usando uma matriz de valores lógicos como índice

```
x[c(TRUE ,FALSE , TRUE ),c(TRUE ,TRUE , FALSE )]
x[c(TRUE , FALSE, TRUE ),c(2 ,3)]
x[x >5]                  # select elements greater than 5
x[x%%2 == 0]             # select even elements
```

Leitura usando uma matriz de carateres como índice

```
colnames (x) <- c("A","B","C")
x[,"A"]
x[,c("A","C")]
x[2:3 ,c("A","C")]
```

Exemplos com matrizes

Modificar uma matriz

```
x[2 ,2] <- 10;          # modify a single element
x[x <5] <- 0;           # modify elements less than 5
t(x)                    # transpose a matrix
cbind (x, c(1, 2, 3))   # add column
rbind (x, c(1 ,2 ,3))   # add row
x <- x [1:2 ,];         # remove the third row
```

Eliminar uma matriz

```
rm(x) # remove variable
```

Lista

Estrutura de dados com elementos de vários tipos de dados

Exemplos:

Criar uma lista utilizando a função list()

```
x <- list ("a" = 2.5 , "b" = TRUE , "c" = 1:3)
str(x)                          # check list structure
x <- list (2.5 , TRUE ,1:3)     # tags are optional
```

Lista

Ler elementos de uma lista

```
x[c (1:2) ] # index using integer vector
x[ -2] # using negative integer to exclude second component
x[c(T,F,F)] # index using logical vector
x <- list (" name " = " John ", " age" = 19 , " speaks " = c(" English "," French "))
x[c(" age"," speaks ")] # index using character vector
x[" age"]
typeof (x[" age"]) # single [ returns a list
x[[" age"]] # double [[ returns the content
typeof (x[[" age" ]])
x$name # same as x[[" name "]]
x$a # partial matching , same as x$ag or x $age
x[["a"]] # cannot do partial match with [[

# indexing can be done recursively
x$ speaks [1]
x[[" speaks " ]][2]
```

Lista

Modificar uma lista

```
x[[" name "]] <- " Clair "
x[[" married "]] <- FALSE
```

Eliminar uma lista

```
rm (x)
```

Data frame

Um *data frame* é uma estrutura de dados bidimensional.

É um caso especial de lista com todas as componentes com o mesmo tamanho

Cada componente forma uma coluna, e os conteúdos dos componentes formam as linhas

Exemplos:

Criar um data frame usando a função data.frame()

```
x <- data . frame (SN = 1:2 , Age = c (21 ,15) , Name = c("John "," Dora "))

str(x) # structure of x

x <- data . frame ("SN" = 1:2 , " Age" = c (21 ,15) , " Name " = c(" John ", " Dora "),
     stringsAsFactors = FALSE )

str(x) # now the third column is a character vector
```

Data frame

Ler data frames

```
x[" Name "]
x$ Name
x[[" Name "]]
x [[3]]
str( trees ) # access as a matrix
head (trees ,n =3) # access as a matrix
trees [2:3 ,] # select 2nd and 3rd row
trees [ trees $ Height > 82 ,] # selects rows with Height greater than 82
trees [10:12 ,2]
trees [10:12 ,2 , drop = FALSE ]
```

Data frame

Modificar um data frame

```
x[1," Age"] <- 20
cbind (x, State =c("NY","FL"))
```

Eliminar um data frame

```
rm (x)
```

Factors

Um *factor* é uma estrutura utilizada para campos que apenas podem tomar um número finito de valores (categóricos)

Exemplos:

Criar um factor usando a função factor()

```
x <- factor (c(" single ", " married ", " married ", " single "));

x <- factor (c(" single ", " married ", " married ", " single "), levels = c(" single ", "
     married ", " divorced "));

x <- factor (c(" single "," married "," married "," single "))

str(x)
```

Factor

Ler um factor

```
x [3] # access 3rd element
x[c(2, 4)] # access 2nd and 4th element
x[ -1] # access all but 1st element
x[c(TRUE , FALSE , FALSE , TRUE )] # using logical vector
```

Modificar um factor

```
x [2] <- " single "
x [2] <- " divorced " # modify second element ; x
x [3] <- " widowed " # cannot assign values outside levels
```

Eliminar um factor

```
rm (x)
```

Funções predefinidas do R

O R conta com diversas funções predefinidas , que podem ser classificadas nas seguintes categorias:

- Funções numéricas
- Funções de texto
- Funções estatísticas
- Funções de probabilidade
- Funções úteis

Funções numéricas

Function	Description
$abs(x)$	absolute value
$sqrt(x)$	square root
$ceiling(x)$	ceiling of a variable
$floor(x)$	floor of a variable
$trunc(x)$	trunc of a variable
$round(x, digits = n)$	round of a variable
$signif(x, digits = n)$	significant digits of a variable
$cos(x), sin(x), etc.$	trigonometric functions
$log(x)$	natural logarithm
$log10(x)$	logarithm base 10
$exp(x)$	exponent

Exemplos

```
sqrt (2)
cos(pi)
exp (2)
```

Funções de texto

Function	Description
$substr(x, start = n1, stop = n2)$	Extract or replace substrings in a vector
$grep(pattern, x)$	Search for pattern in x.
$sub(pattern, replacement, x)$	Find pattern in x and replace.
$strsplit(x, split)$	Split the elements of character vector.
$paste(..., sep = "")$	concatenate a string
$toupper(x)$	Uppercase
$tolower(x)$	Lowercase

Exemplos

```
x <- " abcdef "
substr (x, 2, 4)
grep ("A", c("b","A","c"))
paste ("x" ,1:3, sep="")
```

Funções estatísticas

Function	Description
$mean(x, trim = 0, na.rm = FALSE)$	mean of object x
$sd(x)$	standard deviation
$median(x)$	median
$quantile(x, probs)$	quantiles
$min(x)$	minimum
$max(x)$	maximum
$sum(x)$	summation
$range(x)$	range
$scale(x, center = TRUE, scale = TRUE)$	column center.

Exemplos

```
x <- c(2 ,5 ,7)
mean (x)
max(x)
```

Funções de probabilidade

Function	Description
$dnorm(x)$	normal density function
$pnorm(q)$	cumulative normal probability for q
$qnorm(p)$	normal quantile
$rnorm(n, m = 0, sd = 1)$	n random normal deviates with mean and sd.
...	...

Exemplos

```
x <- rnorm (50 , m=50 , sd =10)
pnorm (1.96)
```

Funções úteis

Function	Description
seq(*from*, *to*, *by*)	generate a sequence
rep(*x*, *ntimes*)	repeat x N times
cut(*x*, *n*)	divide continuous variable in factor with n levels

Exemplos

```
x <- seq (1 ,10 ,2)
y <- rep (1:3 , 2)
```

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.