



# Introdução ao ambiente R

Luciana Souto Mofatto

# R

O Rato Roeu a Roupa do Rei de Roma.

A Rainha de Raiva Rasgou o Resto.



# O que é R?

*R project for Statistical Computing* (<http://www.r-project.org/>)

- Software livre
- Multiplataforma
- Linguagem de programação
- Computação de dados estatísticos
- Criação de gráficos



# **Conceitos Básicos**

**Tipos de dados, operadores,  
comparações, condicionais...**

# Tipos de dados

- Números (*numeric*)

```
> class(1)  
[1] "numeric"
```

- Número Complexo (*complex*)

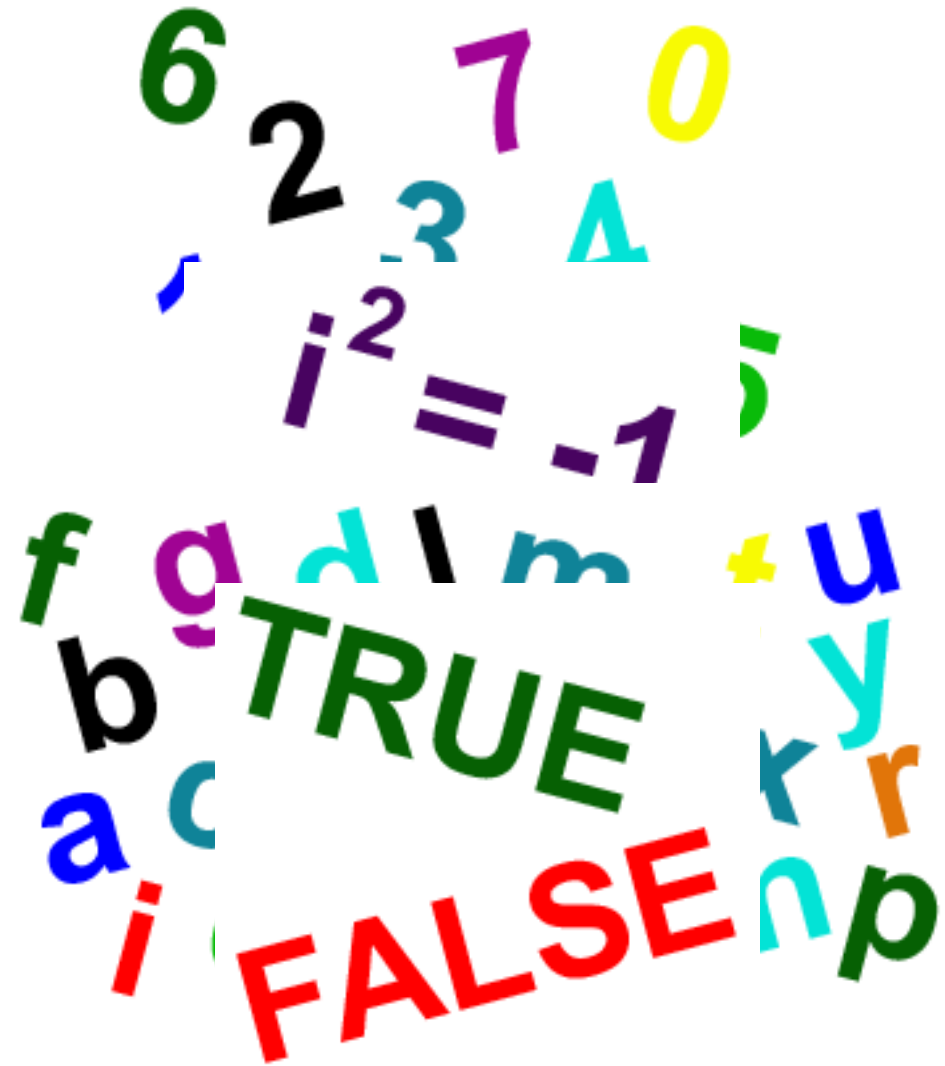
```
> class(1i)  
[1] "complex"
```

- Caracteres (*character*)

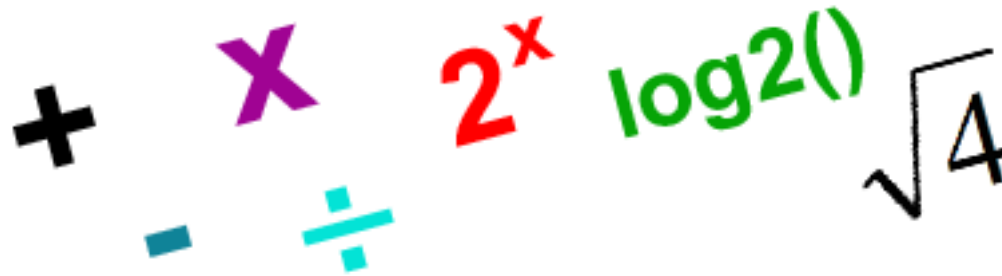
```
> class("aula")  
[1] "character"
```

- Lógico (*logical*)

```
> class(TRUE)  
[1] "logical"
```



# Operadores



## - Soma (+)

```
> 1 + 1  
[1] 2
```

## - Subtração (-)

```
> 4 - 2  
[1] 2
```

## - Multiplicação (\*)

```
> 2*2  
[1] 4
```

## - Divisão (/)

```
> 4/2  
[1] 2
```

## - Exponencial (^)

```
> 2^2  
[1] 4
```

## - Raiz quadrada (sqrt)

```
> sqrt(4)  
[1] 2
```

## - Logaritmo (log2, log10)

```
> log2(20)  
[1] 4.321928
```

```
> log10(20)  
[1] 1.30103
```

# Atribuição de valores e variáveis

- Variável: objeto que é capaz de receber um valor ou uma expressão.

- Símbolos de atribuição: **=** ou **<-**

- Exemplos:

```
> x = 1  
> x  
[1] 1
```

```
> y <- 2  
> y  
[1] 2
```

# Prática 1:

- Encontrar a solução para a seguinte equação:

$$x = \frac{(100^2 + \sqrt{4000} - \log_2(345)).250}{230}$$



# Solução – Prática 1:

```
> x = ((100^2 + sqrt(4000) - log2(345)) * 250) / 230
```

```
> x
```

```
[1] 10929.15
```

```
> 100^2
```

```
[1] 10000
```

```
> sqrt(4000)
```

```
[1] 63.24555
```

```
> log2(345)
```

```
[1] 8.430453
```

```
> 10000 + 63.24555 - 8.430453
```

```
[1] 10054.82
```

```
> 10054.82 * 250
```

```
[1] 2513705
```

```
> 2513705/230
```

```
[1] 10929.15
```

# Comparação



## - Igual (==)

```
> x = 1  
> y <- 1  
> x == y  
[1] TRUE
```

## - Menor (<)

```
> x < y  
[1] FALSE
```

## - Menor ou igual (<=)

```
> x <= y  
[1] TRUE
```

## - Maior (>)

```
> x > y  
[1] FALSE
```

## - Maior ou igual (>=)

```
> x >= y  
[1] TRUE
```

## - Diferente (!=)

```
> x != y  
[1] FALSE
```

## - E (&)

```
> (x > 0) & (x <= 1)  
[1] TRUE
```

## - Ou (|)

```
> (y > 0) | (y < 1)  
[1] TRUE
```

# Prática 2

Sabendo que:

$$\begin{aligned}x &= 15; \\y &= \log_{10}(235); \\z &= (25)^3\end{aligned}$$

Quais expressões abaixo são verdadeiras ou falsas?

a)  $(x + z) \geq y$  ?

b)  $(y + z) \leq x^2$  ?

c)  $\log_2(z) == (x + y)$  ?

# Solução – Prática 2a:

```
> x = 15
> y = log10(235)
> z = 25^3

> # a) (x + z) >= y

> (x + z) >= y
[1] TRUE

> x + z
[1] 15640
> y
[1] 2.371068
```

# Solução – Prática 2b:

```
> x = 15
> y = log10(235)
> z = 25^3

> # b) (y + z) <= x^2

> (y + z) <= x^2
[1] FALSE
> y + z
[1] 15627.37
> x^2
[1] 225

> # C) z == (x + y)
```

# Solução – Prática 2c:

```
> x = 15
> y = log10(235)
> z = 25^3

> # c) log2(z) == (x + y)

> log2(z) == (x + y)
[1] FALSE

> log2(z)
[1] 13.93157
> (x + y)
[1] 17.37107
```

# Tabela Verdade

## Comparação por “E”

Condição 1	Condição 2	(Condição 1) E (Condição 2)
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

## Comparação por “OU”

Condição 1	Condição 2	(Condição 1) OU (Condição 2)
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

# Condicional

- Se (*if*)

```
> x = 5  
> if(x > 1){  
+ x  
+ }  
[1] 5
```

- Senão (*else*)

```
> x <- 2  
> y <- 1  
> if( x < y ){  
+ x  
+ } else {  
+ y  
+ }  
[1] 1
```

*if* (condição) {

...

} *else* {

...

}



# Prática 3

Sabendo que:

$$\begin{aligned}x &= 26^3 - (12 * 10)^2; \\ y &= \log_2(234.5) + (23 * 16) / 100; \\ z &= 100 - 9^2 + 12^3\end{aligned}$$

Quais expressões são verdadeiras?

- a) Se  $((x > y) \text{ AND } (z \leq x)) \{ x = 1 \}$  senão  $\{ x = 0 \}$ ?
- b) Se  $((x + z) > 10 \text{ OR } (x + y) == (z - x)) \{ x = 1 \}$  senão  $\{ x = 0 \}$ ?
- c) Se  $((z \geq x) \text{ AND } (y \geq x)) \{ x = 1 \}$  senão  $\{ x = 0 \}$ ?

# Solução – Prática 3a:

```
> x = 26^3 - (12*10)^2;  
> y = log2(234.5) + (23*16)/100;  
> z = 100 - 9^2 + 12^3
```

```
> # a) (x > y) && (z <= x)
```

```
> (x > y) && (z <= x)  
[1] TRUE
```

```
> (x > y)  
[1] TRUE  
> (z <= x)  
[1] TRUE
```

```
> x  
[1] 3176  
> y  
[1] 11.55344  
> z  
[1] 1747
```

```
> if((x > y) && (z <= x)) {  
+ x = 1  
+ } else {  
+ x = 0  
+ }  
>  
> x  
[1] 1
```

# Solução – Prática 3b:

```
> x = 26^3 - (12*10)^2;  
> y = log2(234.5) + (23*16)/100;  
> z = 100 - 9^2 + 12^3
```

```
> # b) ((x + z) > 10) || ((x + y)  
== (z - x))
```

```
> ((x + z) > 10) || ((x + y) ==  
(z - x))
```

```
[1] TRUE
```

```
> x + z > 10
```

```
[1] TRUE
```

```
> x + z
```

```
[1] 4923
```

```
> ((x + y) == (z - x))
```

```
[1] FALSE
```

```
> x + y
```

```
[1] 3187.553
```

```
> z - x
```

```
[1] -1429
```

```
> if(((x + z) > 10) || ((x + y)  
== (z - x))){
```

```
+ x = 1
```

```
+ } else {
```

```
+ x = 0
```

```
+ }
```

```
> x
```

```
[1] 1
```

# Solução – Prática 3c:

```
> x = 26^3 - (12*10)^2;  
> y = log2(234.5) + (23*16)/100;  
> z = 100 - 9^2 + 12^3
```

```
> # c) (z >= x) && (y >= x)  
> (z >= x) && (y >= x)  
[1] FALSE
```

```
> (z >= x)  
[1] FALSE  
> (y >= x)  
[1] FALSE
```

```
> z  
[1] 1747  
> x  
[1] 3176  
> y  
[1] 11.55344
```

```
> if((z >= x) && (y >= x)){  
+ x = 1  
+ } else {  
+ x = 0  
+ }  
> x  
[1] 0
```

# **Conceitos Intermediários**

**Objetos e suas funções,  
iterações**



# Objetos - Vector

Vector: lista de elementos. Tipos: *numeric, character, logical*

```
> a = c(1, 2, 3, 4, 5, 6)
> b = c("one", "two", "three", "four", "five", "six")
> c = c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE)
> d = vector(mode = "numeric", length = 10)
```

Aceita operações de:

- Soma, subtração, divisão, multiplicação...

# Objetos - Vector

Mais operações:

```
> x = c(9,8,7,6,5,4,3,2,1,0)
> length(x) #tamanho do vector
[1] 10
```

```
> x[4] # imprime o quarto elemento do vector
[1] 6
```

```
> x[1:3] # imprime na tela os elementos 1 a 3
[1] 9 8 7
```

```
> head(x) # imprime os 6 primeiros elementos do vector
[1] 9 8 7 6 5 4
```

```
> tail(x) # imprime os 6 últimos elementos do vector
[1] 5 4 3 2 1 0
```



# Iteração (loop)

## *For:*

```
> x = c(9,8,7,6,5,4,3,2,1,0)
> a = vector(mode =
"numeric",length=length(x))

> for(i in 1:length(x)){
+ a[i]=x[i]+ 1
+ }
>
> a
[1] 10  9  8  7  6  5  4  3  2  1
```

## *While:*

```
> x = c(9,8,7,6,5,4,3,2,1,0)
> a = vector(mode =
"numeric",length=length(x))
>
> i = 1

> while(i <= length(x)){
+ a[i] = x[i] + 1
+ i = i + 1
+ }
>
> a
[1] 10  9  8  7  6  5  4  3  2  1
```

# Prática 4

Sabendo que:

$$a = c(10,2,3,14,34,200,40)$$
$$b = c(0,23,65,93,72,10,11,90)$$

Calcule:

- a) Quais os tamanhos de cada *vector*?
- b) Quais os valores das somas dos primeiros 4 elementos de cada *vector*?
- c) E se somar os primeiros 4 elementos de a com os 4 últimos elementos de b, qual seria o valor?

# Solução – Prática 4a e 4b

```
> a = c(10,2,3,14,34,200,40)
> b = c(0,23,65,93,72,10,11,90)

> # a) length(a); length(b)
> length(a)
[1] 7
> length(b)
[1] 8
> # b) soma dos primeiros 4 elementos de cada vector
> soma_a = 0
> for(i in 1:4){
+ soma_a = soma_a + a[i]
+}
> soma_a
[1] 29
> soma_b = 0
> for(i in 1:){
+ soma_b = soma_b + b[i]
+}
> soma_b
[1] 181
```

# Solução – Prática 4c

```
> a = c(10,2,3,14,34,200,40)
> b = c(0,23,65,93,72,10,11,90)
```

```
> # b) soma dos primeiros 4 elementos de a e 4 ultimos de b
> sum(a[1:4]) + sum(b[5:8])
[1] 212
```

```
> sum(a[1:4])
[1] 29
```

```
> sum(b[5:8])
[1] 183
```

```
> 29 + 183
[1] 212
```

# Objetos - Matrix

Matriz: são estruturas bi-dimensionais utilizadas para armazenar dados de um tipo.

*matriz[ linha, coluna]*

```
> w = cbind(a,b,c) #criacao de uma matriz a partir de 3 vectors
```

```
> w
```

```
      a      b      c
[1,] "1" "one" "TRUE"
[2,] "2" "two" "FALSE"
[3,] "3" "three" "TRUE"
[4,] "4" "four" "FALSE"
[5,] "5" "five" "TRUE"
[6,] "6" "six" "FALSE"
```

```
> w[, "a"]
```

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

```
> w[1,]
```

```
      a      b      c
"1" "one" "TRUE"
```

# Prática 5

Considere a matriz de 3 colunas e 7 linhas:

```
m = cbind(a = c(100,122,33,54,64,20,4),  
          b = c(40,123,55,67,21,18,12),  
          c = c(12,21,44,23,87,50,123))
```

Calcule:

A média e a mediana dos elementos da coluna “a”, “b” e “c” e concatene em uma nova matriz.

(Dica: utilize os comandos *mean* e *median* para média e mediana)

# Solução – Prática 5

```
> m = cbind(a = c(100,122,33,54,64,20,4),  
+ b = c(40,123,55,67,21,18,12),  
+ c = c(12,21,44,23,87,50,123))  
>  
> media = c(mean(m[, "a"]), mean(m[, "b"]), mean(m[, "c"]))  
> mediana = c(median(m[, "a"]), median(m[, "b"]), median(m[, "c"]))  
>  
> d = cbind(media, mediana)  
> d
```

	media	mediana
[1,]	56.71429	54
[2,]	48.00000	40
[3,]	51.42857	44

# Objetos - List

Lista: conjunto ordenado de objetos, os quais podem conter valores de diferentes tipos de dados ou outros objetos.

```
>
> class(list_samples)
[1] "list" list_samples = list( name = "Sample01", group = "Control",
no.replicates = 3, time = c( 1, 2, 3))
```

## Operações de List:

```
> list_samples
$name
[1] "Sample01"
$group
[1] "Control"
$no.replicates
[1] 3
$time
[1] 1 2 3
```



# Objetos - List

## Acesso aos dados:

```
> ### Nome dos campos
> list_samples$name
[1] "Sample01"

> list_samples$group
[1] "Control"

> list_samples$no.replicates
[1] 3

> list_samples$time # 3 elementos
[1] 1 2 3
> list_samples$time[1]
[1] 1
> list_samples$time[2]
[1] 2
> list_samples$time[3]
[1] 3
```

## Ou:

```
> ### Index dos campos
> list_samples[[1]]
[1] "Sample01"

> list_samples[[2]]
[1] "Control"

> list_samples[[3]]
[1] 3

> list_samples[[4]] # 3 elementos
[1] 1 2 3
> list_samples[[4]][1]
[1] 1
> list_samples[[4]][2]
[1] 2
> list_samples[[4]][3]
[1] 3
```

# Prática 6

Faça listas contendo dados de 2 experimentos:

- a) Lista 1 = experimento da amostra 1 do grupo “controle” com 3 réplicas biológicas e tempos de 24 e 72 horas;
- b) Lista 2 = experimento da amostra 2 do grupo “teste” com somente uma réplica e tempos de 1, 2 e 3 horas.

# Solução – Prática 6

```
> list1 = list(name="Amostra1",  
group="Controle",nro.replicas=3,  
time=c(24,72))
```

```
>
```

```
> list1
```

```
$name
```

```
[1] "Amostra1"
```

```
$group
```

```
[1] "Controle"
```

```
$nro.replicas
```

```
[1] 3
```

```
$time
```

```
[1] 24 72
```

```
> list2 =
```

```
list(name="Amostra2",group="Teste",  
nro.replicas=1,time=c(1,2,3))
```

```
> list2
```

```
$name
```

```
[1] "Amostra2"
```

```
$group
```

```
[1] "Teste"
```

```
$nro.replicas
```

```
[1] 1
```

```
$time
```

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

# Objetos - Data frame

Permite vários tipos de dados, contando que todas colunas tenham o mesmo número de linhas.

```
> df = data.frame(  
  gene_id = c("MT-CO1", "COL1A2", "COL1A1", "FTH1", "FN1", "MT-  
TF", "VIM", "EEF1A1", "THBS1", "GREM1"),  
  read_count_s1 = c(10, 2, 15, 13, 19, 22, 0, 0, 0, 200),  
  read_count_s2 = c(50, 0, 0, 0, 0, 111, 12, 9, 10, 300))  
  
> class(df)  
[1] "data.frame"
```

# Objetos - Data frame

## Operações de um data frame:

```
> row.names(df) = df$gene_id #dar nome às linhas de um data.frame
```

```
> head(df) # imprime as 6 primeiras linhas do vector
```

	gene_id	read_count_s1	read_count_s2
MT-CO1	MT-CO1	10	50
COL1A2	COL1A2	2	0
COL1A1	COL1A1	15	0
FTH1	FTH1	13	0
FN1	FN1	19	0
MT-TF	MT-TF	22	111

# Objetos - Data frame

Mais operações de um data frame:

```
> df["FN1",] # imprime somente a linha chamada "C"  
      gene_id read_count_s1 read_count_s2  
FN1      FN1           19           0
```

```
> df[2:5,] # imprime as linhas 2 a 5  
      gene_id read_count_s1 read_count_s2  
COL1A2 COL1A2           2           0  
COL1A1 COL1A1          15           0  
FTH1    FTH1          13           0  
FN1     FN1          19           0
```

```
> dim(df) # imprime as dimensões de um data.frame  
[1] 10 3
```

# Prática 7

Considere:

```
genes = c("a","b","c","d");  
read_count_amostra1 = c(100,120,0,12);  
read_count_amostra2 = c(12,90,20,0);  
read_count_amostra3 = c(23,45,67,33)
```

Criar um *dataframe* que contenha os dados acima, também calculando os valores de soma dos *read counts* para cada gene.

# Solução – Prática 7

```
> genes = c("a","b","c","d");
> read_count_amostra1 = c(100,120,0,12);
> read_count_amostra2 = c(12,90,20,0);
> read_count_amostra3 = c(23,45,67,33)
> soma_rc = 0

> for(i in 1:length(genes)){
+ soma_rc[i] = read_count_amostra1[i] + read_count_amostra2[i] +
read_count_amostra3[i]
+ }

> soma_rc
[1] 135 255 87 45

> df =
data.frame(read_count_amostra1,read_count_amostra2,read_count_amostra
3,soma_rc, row.names=genes)
```



# Solução – Prática 7

```
> df =  
data.frame(read_count_amostra1, read_count_amostra2, read_count_amostra  
3, soma_rc, row.names=genes)
```

```
> df  
  read_count_amostra1 read_count_amostra2 read_count_amostra3 soma_rc  
a                   100                   12                   23      135  
b                   120                   90                   45      255  
c                     0                   20                   67       87  
d                     12                    0                   33       45
```

# **Conceitos Avançados**

**Manipulação de arquivos**



# PC and Pixel

BY: THACH BUI  
& GEOFF JOHNSON



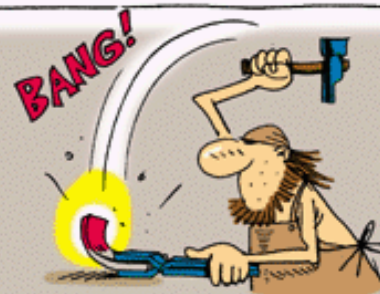
Pixelandpc@aol.com



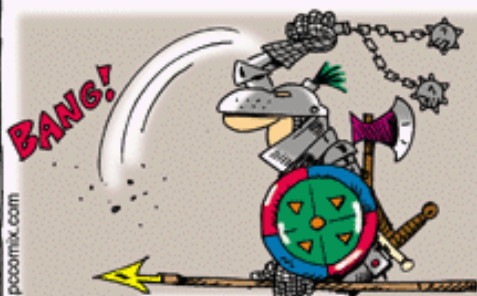
STONE AGE



BRONZE AGE



IRON AGE



DARK AGE



MODERN AGE



COMPUTER AGE

www.pccomix.com

Distributed by the Washington Post Writers Group

By: T. Bui & G. Johnson ©2000 Thach Bui

TAK  
7-25

# Importar e Manipular Arquivos

R pode importar arquivos de tabelas Excel, txt (“\t”), csv (“,”).

```
> getwd() #mostra o diretorio de trabalho atual
[1] "/Users/Teste"
> setwd("/Users/Teste/tables") #altera o diretorio de trabalho
> table01 = read.table( file = "text_table01.txt", sep = "\t",
header = TRUE)
> row.names(table01) = table01$Alunos
> table01 = table01[,2:4]
> names(table01)
[1] "Disciplina_A" "Disciplina_B" "Disciplina_C"
> head(table01)
```

	Disciplina_A	Disciplina_B	Disciplina_C
Aluno01	A	10.0	D
Aluno02	A	7.5	B
Aluno03	B	8.5	A
Aluno04	B	4.5	A
Aluno05	A	2.0	B
Aluno06	D	5.0	A

# Importar e Manipular Arquivos

```
> # Comando que cria um subconjunto com notas = A na Disciplina A
> notas_A = subset(table01, Disciplina_A = "A")
> head(notas_A)
```

	Disciplina_A	Disciplina_B	Disciplina_C
Aluno01	A	10.0	D
Aluno02	A	7.5	B
Aluno05	A	2.0	B
Aluno08	A	6.7	A
Aluno13	A	9.0	C

```
> class(notas_A)
[1] "data.frame"

> # Comando que calcula a média de notas da Disciplina B
> mean(table01$Disciplina_B)
[1] 7.3

> # Comando para salvar uma tabela em um arquivo formato texto
> write.table(notas_A, file="text_table01_notas_A_discA.txt",
row.names=TRUE)
```

# Prática 8

Antes de começar:

- Criar um dataframe:

```
> genes =  
c("GAPDH", "COL12A1", "ACTB", "ANXA2", "SOD2", "MMP2", "FSTL1", "IGFBP4", "A  
HNAK", "COL8A1")  
> rc_sample1 = c(1200, 3425, 23987, 567, 789, 342, 555, 678, 444, 231)  
> rc_sample2 = c(5050, 978, 349, 954, 565, 4787, 344, 66, 737, 100)  
> rc_sample3 = c(230, 4646, 5858, 3432, 567, 8382, 453, 2020, 100, 102)  
  
> df = data.frame(rc_sample1, rc_sample2, rc_sample3, row.names=genes)
```

-Salvar o dataframe em um arquivo:

```
> write.table(df, file = "arquivo_pratica8.txt", sep="\t")
```

# Prática 8

- Abrir o arquivo “arquivo\_pratica8.txt”, calcular a média e a mediana por gene.
- Criar outro *dataframe* contendo:
  - Todos os read counts por gene
  - Todas as médias por gene
  - Todas as medianas por gene
- Salvar o novo dataframe como “arquivo2\_pratica8.txt” separado por tabulação.

Dica:

- Calcular a média usando:

```
mean(c(x$rc_sample1, x$rc_sample2, x$rc_sample3))
```

- Calcular mediana usando:

```
median(c(x$rc_sample1, x$rc_sample2, x$rc_sample3))
```

# Solução – Prática 8

```
> tabela = read.delim(file = "arquivo_pratica8.txt",header = TRUE)
> head(tabela)
      rc_sample1 rc_sample2 rc_sample3
GAPDH          1200         5050         230
COL12A1         3425          978        4646
ACTB           23987          349        5858
ANXA2           567           954        3432
SOD2            789           565         567
MMP2            342          4787        8382
> media = 0; mediana = 0;
> for(i in 1:dim(tabela)[1]){
+ media[i] =
mean(c(tabela$rc_sample1[i],tabela$rc_sample2[i],tabela$rc_sample3[i]))
+ mediana[i] =
median(c(tabela$rc_sample1[i],tabela$rc_sample2[i],tabela$rc_sample3[i]))
+ }
> novo_df = data.frame(df,media,mediana)
>
```



# Solução – Prática 8

```
> head(novo_df)
      rc_sample1 rc_sample2 rc_sample3      media mediana
GAPDH          1200       5050        230  2160.0000     1200
COL12A1         3425         978       4646  3016.3333     3425
ACTB          23987         349       5858 10064.6667     5858
ANXA2           567         954       3432  1651.0000      954
SOD2            789         565        567   640.3333      567
MMP2            342       4787       8382  4503.6667     4787
> write.table(novo_df, file="arquivo2_pratica8.txt", sep="\t")
```

# HELP!!!

```
> ?mean
```

```
starting httpd help server ... done
```

```
mean {base}
```

R Documentation

## Arithmetic Mean

### Description

Generic function for the (trimmed) arithmetic mean.

### Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

### Arguments

**x** An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

**trim** the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

**na.rm** a logical value indicating whether NA values should be stripped before the computation proceeds.

**...** further arguments passed to or from other methods.

### Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

<http://www.r-project.org/>



#### About R

[What is R?](#)  
[Contributors](#)  
[Screenshots](#)  
[What's new?](#)

#### Download, Packages

[CRAN](#)

#### R Project

[Foundation](#)  
[Members & Donors](#)  
[Mailing Lists](#)  
[Bug Tracking](#)  
[Developer Page](#)  
[Conferences](#)  
[Search](#)

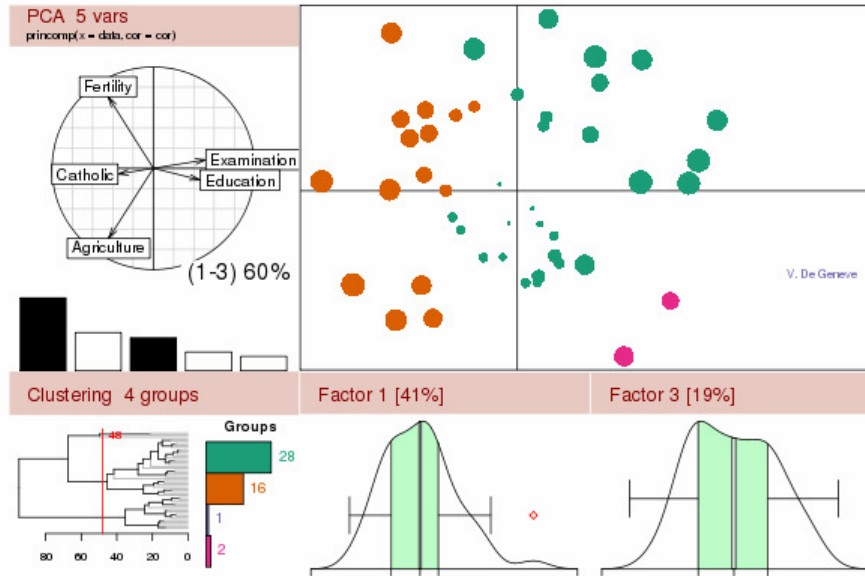
#### Documentation

[Manuals](#)  
[FAQs](#)  
[The R Journal](#)  
[Wiki](#)  
[Books](#)  
[Certification](#)  
[Other](#)

#### Misc

[Bioconductor](#)  
[Related Projects](#)  
[User Groups](#)  
[Links](#)

## The R Project for Statistical Computing



### Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).
- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

### News :

- [The R Journal Volume 6/2](#) is available.
- **R version 3.1.2** (Pumpkin Helmet) has been released on 2014-10-31.
- [useR! 2014](#), took place at the University of California, Los Angeles, USA June 30 - July 3, 2014.
- **R version 3.0.3** (Warm Puppy) has been released on 2014-03-06.
- [useR! 2015](#), will take place at the University of Aalborg, Denmark, June 30 - July 3, 2015.

This server is hosted by the [Institute for Statistics and Mathematics](#) of [WU \(Wirtschaftsuniversität Wien\)](#).

<http://www.statmethods.net/interface/index.html>



**Quick-R**  
accessing the power of R

Home | Interface | Input | Manage | Stats | Adv Stats | Graphs | Adv Graphs | Blog

Search

### R Interface

- [Getting Help](#)
- [The Workspace](#)
- [Input/Output](#)
- [Packages](#)
- [Graphic User Interfaces](#)
- [Customizing Startup](#)
- [Publication Quality Output](#)
- [Batch Processing](#)
- [Reusing Results](#)

### R in Action



[R in Action](#) (2nd ed) significantly expands upon this material. Use

## Overview




R is a dialect of the [S language](#). It is a case-sensitive, interpreted language. You can enter commands one at a time at the command prompt (`>`) or run a set of commands from a source file. There is a wide variety of [data types](#), including vectors (numerical, character, logical), matrices, data frames, and lists. Most functionality is provided through built-in and user-created functions and all data objects are kept in memory during an interactive session. Basic functions are available by default. Other functions are contained in [packages](#) that can be attached to a current session as needed.

“ R is a case sensitive language. FOO, Foo, and foo are three different objects!



<http://tryr.codeschool.com/>



1  
2  
3  
4  
5  
6  
7  
8



Try R is Sponsored By:  
**O'REILLY®**

Created By:  
**code school**

R is a tool for statistics and data modeling. The R programming language is elegant, versatile, and has a highly expressive syntax designed around working with data. R is more than that, though — it also includes extremely powerful graphics capabilities. If you want to easily manipulate your data and present it in compelling ways, R is the tool for you.




Share Try R:  Tweet  Like 5.4k

**Start the Course**


### Table of Contents

- 1. **R Syntax:** A gentle introduction to R expressions, variables, and functions
- 2. **Vectors:** Grouping values into vectors, then doing

### Chapter Badges



<https://www.datacamp.com/courses/introduction-to-r/>

 **DataCamp**

Home

Courses

Sign in

Introduction to R

Start Now

42,059 ENROLLED

BEGINNER


ABOUT 4 HOURS

With over 2 million users worldwide R is rapidly becoming the leading programming language in statistics and data science. Every year, the number of R users grows with over 40%, and an increasing number of organizations start to use it in their day-to-day activities.

In this introduction to R, you will master the basics of this beautiful open source language. We'll take you on trips to Las Vegas and galaxies far far away. Basic topics such as factors, lists and data frames will be covered. After finishing this introductory R course, you'll master some very valuable R skills and are ready to undertake your first very own data analysis.

*This course is aimed at people who'd like to start using R for their study assignments, research projects or professional endeavors. No previous experience in programming languages or data science is required.*


COURSE CONTENTS



Chapter 1 - Intro to basics (8 exercises)

In this chapter, you will take your first steps with R. You will learn how to use the console as a calculator and how to assign variables. You will also get to know the basic data types in R. Let us get started!

Course given by

 Jonathan Cornelissen  
DataCamp

Biography

Jonathan Cornelissen is one of the co-founders of DataCamp, and is interested in everything related to data science, R, education and entrepreneurship. He holds a PhD in financial econometrics, and is the author of an R package for quantitative finance. DataCamp is the second education start-up he founded, and the first one that went international. In his spare time, he loves to do some programming, especially in Ruby on Rails.





*That's all Folks!*