

Data Scientist - Introdução ao R

Soraia Pereira^a e Tiago Marques^{a,b}

^a CEAUL e FCUL, Universidade de Lisboa

^b CREEM, University of St Andrews, e Dept de Biologia Animal, FCUL



FCUL, 28 de Fevereiro de 2020

Programa

- ▶ Introdução ao R e ao RStudio
- ▶ Relatórios dinâmicos em R Markdown
- ▶ Importar dados para o R
- ▶ Manipulação de dados em R

House Keeping

Recursos do curso disponíveis na pasta

<https://tinyurl.com/CEAULGADESCursoRM1>

O curso decorre entre as 18:30 e as 22:30



Entre as 20:30 e as 20:45 faremos uma pausa para café.

The leaRning cuRve

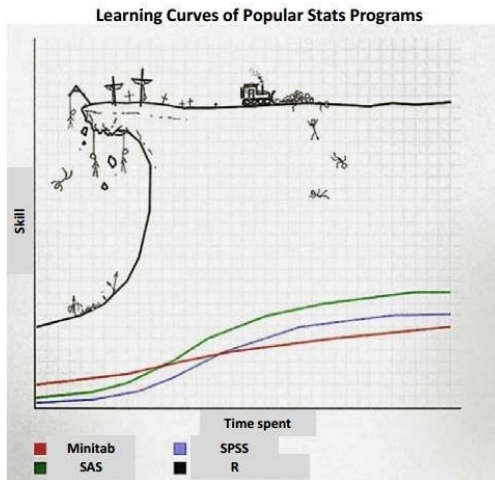


Figure: Why am I doing this?

O R é uma linguagem de programação e ambiente direcionados para computação estatística e gráfica.

Porquê usar o R?

- ▶ O R tornou-se a linguagem standard para criação de software estatístico. Consequentemente, os novos métodos estatísticos são na maior parte dos casos primeiro implementados nesta linguagem.
- ▶ Existem muitas ferramentas estatísticas disponíveis no R e milhares de packages adicionais que vão além das funcionalidades básicas.
- ▶ Permite criar gráficos estatísticos com baixo esforço.
- ▶ O software é de alta qualidade.
- ▶ É relativamente fácil de usar (para uma linguagem de programação)
- ▶ É grátis e está disponível para toda a comunidade.

Documentação útil

- ▶ O site oficial do R disponibiliza documentação para os utilizadores:
<https://cran.r-project.org>
Ver secção "documentation" do lado esquerdo da página.
- ▶ O site oficial do RStudio disponibiliza Cheat Sheets
<https://www.rstudio.com/resources/cheatsheets/>
- ▶ R for Data Science / Garrett Golemund and Hadley Wickham
<https://r4ds.had.co.nz/index.html>
- ▶ Help online do R. Para aceder basta seleccionar "Help" no painel inferior direito do RStudio.

WELCOME!

Follow @rbloggers 79.1k

Here you will find daily news and tutorials about R, contributed by hundreds of bloggers.

There are many ways to follow us -

By e-mail:

Your e-mail here

Subscribe

53,400 readers

BY FEEDBURNER

On Facebook:



R blog...
77 mil gostos

Gostar da Página

18 amigos gostam disto



If you are an R blogger yourself you are invited to add your own R content feed to this site (Non-English R bloggers should add themselves - here)

On the relationship of the sample size and the correlation

January 27, 2020

By mrajter




Read more »

This has bugged me for some time now. There is a "common knowledge" that the correlation size is dependent on the variability, i.e. higher the variability – higher the correlation. However, when you put this into practice, there seems to be a confusion on what this really means. To analyse this I have divided this ...
Continue reading On...

Programming with data.table

January 26, 2020

By HighlandR

 Programming with data.table getting started multiple bare variable names in data.table functions - Flexible functions in data.table I'm

MCMC, with common misunderstandin gs

January 26, 2020

By xi'an

SEARCH R-BLOGGERS

Search..

Go

MOST VISITED ARTICLES OF THE WEEK

1. Top 10 Most Valuable Data Science Skills in 2020
2. 5 Ways to Subset a Data Frame in R
3. How to write the first for loop in R
4. RStudio Projects and Working Directories: A Beginner's Guide
5. Explanatory Model Analysis with modelStudio
6. R – Sorting a data frame by the contents of a column
7. In-depth introduction to machine learning in 15 hours of expert videos
8. Installing R packages
9. Date Formats in R

SPONSORS

Submit your abstract for the Enterprise Applications of the R Language Conference (EARL) before 31/3/20

London 8-10 September, 2020



RStudio é um ambiente de desenvolvimento integrado para o R (IDE). Embora não seja obrigatório recorrer a um IDE para editar código em R, este ambiente traz muitas vantagens.

Porquê usar o RStudio?

- ▶ Torna mais acessível a utilização da linguagem R.
- ▶ Inclui um editor de código, juntamente com ferramentas de visualização.
- ▶ Inclui complementos úteis e gratuitos, como por exemplo, a integração com o package `knitr` que pode ser usado com o package `RMarkdown` para gerar relatórios dinâmicos em HTML, PDF, Word...

RStudio

Ver também o *hands on tutorial* que vos é oferecido na pasta com material para os participantes!

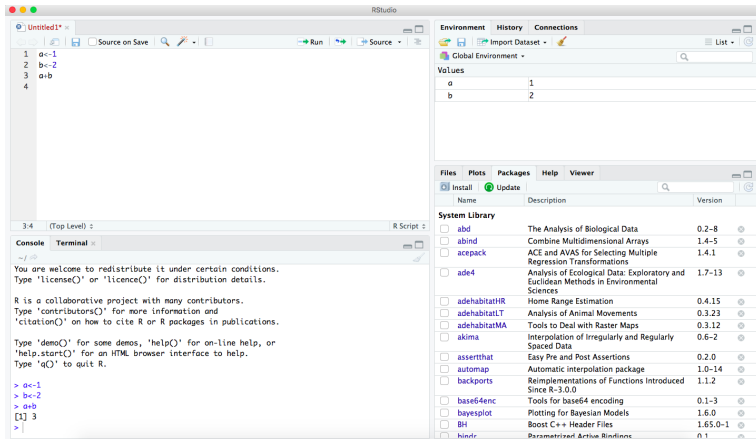
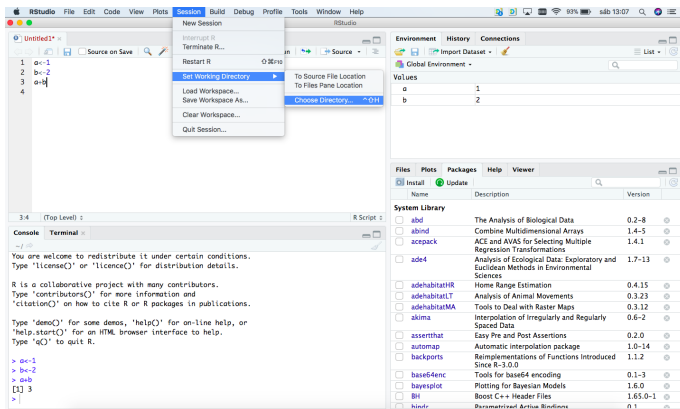


Figure: RStudio

Diretoria de trabalho

Saber em que diretoria estamos a trabalhar é fundamental para evitar error. Definir a diretoria de trabalho deve ser um dos primeiros passos na utilização do R. É na diretoria definida que serão guardados por defeito todos os *scripts* e ficheiros de *output*.



Operações simples

- ▶ As operações aritméticas básicas podem ser executadas em R utilizando os sinais comuns desse tipo de operações.
- ▶ Embora o sinal "=" possa ser utilizado para atribuição de valores às variáveis, é recomendado o operador "<=", pois evita possíveis conflitos com o operador de comparação "==".
- ▶ Ao contrário de alguns softwares estatísticos como o SPSS, o R não requer a declaração do tipo de variáveis antes da sua utilização.
- ▶ Permite manusear muitos tipos de variáveis (numéricas, caracteres, datas, TRUE/FALSE...). Além disso, as variáveis do R permitem armazenar objectos do R, tais como funções ou gráficos. Para verificar o tipo de variável, basta usar a função `class()`.

Operações simples

The screenshot displays the RStudio IDE interface. The top menu bar includes RStudio, File, Edit, Code, View, Plots, Session, Build, Window, and Help. The main editor window shows an R script with the following code:

```

1 a<-1
2 b<-2
3 a*b
4
5 (a*b+b)/a
6
7 class(a)
8
9 x<-\"Hello world\"
10
11 class(x)
12 |

```

The console on the left shows the execution output:

```

> a<-1
> b<-2
> a*b
[1] 3
> (a*b+b)/a
[1] 4
> class(a)
[1] \"numeric\"
> x<-\"Hello world\"
> class(x)
[1] \"character\"

```

The right sidebar contains the Environment pane, which shows the following variables:

Variable	Value
a	1
b	2
x	\"Hello world\"

Below the Environment pane is the Plots pane, and at the bottom is the Help pane, which displays the documentation for the `sum` function, titled "Sum of Vector Elements".

Vetores

Vector é uma estrutura de dados do R que permite guardar elementos do mesmo tipo (e.g. numérico ou *string*).

```
> a<-c(2,7,3,1)
> a
[1] 2 7 3 1
> b<-c(1,8,2,4)
> b
[1] 1 8 2 4
> a+b
[1] 3 15 5 5
> a<-c("vermelho","azul","verde")
> a
[1] "vermelho" "azul"      "verde"
```

Data frames

As data frames são as estruturas de dados do R mais úteis e utilizadas. Podem ser vistas como tabelas, em que cada coluna pode ter uma nome, e podem incluir colunas com diferentes tipos de dados.

```
> x<-c("Soraia","Tiago","João","Mónica")
> y<-c(20,15,16,12)
> z<-c(2,1.5,1,1.2)
> pascoa<-data.frame(x,y,z)
> names(pascoa)<-c("Nome", "Nr.chocolates.páscoa", "Aumento.quilos")
> pascoa
```

	Nome	Nr.chocolates.páscoa	Aumento.quilos
1	Soraia	20	2.0
2	Tiago	15	1.5
3	João	16	1.0
4	Mónica	12	1.2

Data frames

Podemos aceder facilmente a uma coluna em particular a partir do operador \$, e a um valor individual da tabela usando a notação [linha,coluna].

```
> pascoa$Nr.chocolates.pascoa  
[1] 20 15 16 12  
> pascoa$Aumento.quilos  
[1] 2.0 1.5 1.0 1.2  
> pascoa[2,3]  
[1] 1.5
```

Ciclos - for/while

No R os ciclos são implementados com recurso ao operador `for` ou `while`.

```
> for (i in 1:3)
+ {
+   y<-i+1
+   print(y)
+ }
[1] 2
[1] 3
[1] 4
> i<-1
> while(i<=3)
+ {
+   y<-i+1
+   print(y)
+   i<-i+1
+ }
[1] 2
[1] 3
[1] 4
```


Condições - if

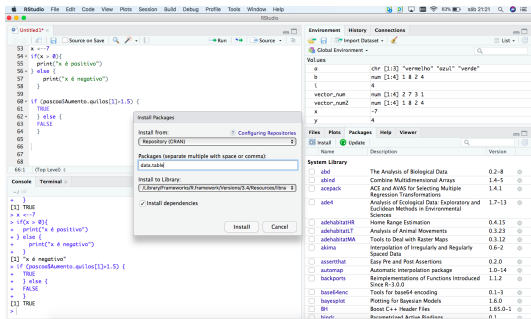
A estrutura para definição de condições no R é a seguinte:

```
if(condicao){afirmacao1} else {afirmacao2}
```

```
> x <--7
> if(x > 0){
+   print("x é positivo")
+ } else {
+   print("x é negativo")
+ }
[1] "x é negativo"
> if (pascoa$Aumento.quilos[1]>1.5) {
+   TRUE
+ } else {
+   FALSE
+ }
[1] TRUE
```

Instalação de packages

A instalação de um package pode ser feita selecionando "Packages" no painel inferior direito, seguido de "Install" e indicar o nome do package pretendido. Exemplo de instalação do package `data.table`:



Nota: Após a instalação, deverá usar-se o comando `library()`. No exemplo, deverá usar `library(data.table)`. Alternativamente, poderá seleccionar o package respetivo na lista de packages do painel inferior direito.

Importação e exportação de dados

- ▶ O R permite importar dados de ficheiros externos com vários formatos, nomeadamente ASCII (.txt e .dat), Excel (.csv e .xlsx), SPSS (.sav), SAS (.SAS)...
- ▶ Uma das funções mais utilizadas para fazer importação de ficheiros ASCII é a `read.table()`. Esta função faz a importação para um objecto `data.frame`. Por exemplo, para importar o ficheiro "dados.txt", deverá usar-se o comando `dados<-read.table("dados.txt")`
- ▶ Existem algumas variantes do `read.table()`, tal como `read.csv()`. Para mais informações sobre esta função, poderá recorrer ao help online do R.
- ▶ A exportação de dados poderá ser feita usando as funções `write.table` ou `write.csv()` (respetivamente para os formatos .txt e .csv). Por exemplo, `write.table(df, "file.txt")` ou `write.csv(df, "file.csv")`.
- ▶ Para leitura e escrita de dados com a estrutura dos objectos R deverá usar-se respetivamente `load("file.RData")` e `save(df, file = "file.Rdata")`.

Missing values

- ▶ Os missings values são representados por NA (Not Available).
- ▶ A função `is.na(x)` retorna um vector lógico com a mesma dimensão de `x`, com valor `TRUE` se e só se o correspondente elemento de `x` é NA.

```
> z<-c(1:4,NA)
> z
[1] 1 2 3 4 NA
> is.na(z)
[1] FALSE FALSE FALSE FALSE TRUE
```

Cheat sheet - Base R

Base R Cheat Sheet

Getting Help

Accessing the help files

?mean

Get help of a particular function.

help.search('weighted mean')

Search the help files for a word or phrase.

help(package = 'dplyr')

Find help for a package.

More about an object

str(iris)

Get a summary of an object's structure.

class(iris)

Find the class an object belongs to.

Using Packages

install.packages('dplyr')

Download and install a package from CRAN.

library(dplyr)

Load the package into the session, making all its functions available to use.

dplyr::select

Use a particular function from a package.

data(iris)

Load a built-in dataset into the environment.

Working Directory

getwd()

Find the current working directory (where inputs are found and outputs are sent).

setwd('C:/file/path')

Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

<code>c(2, 4, 5)</code>	<code>2 4 5</code>	Join elements into a vector
<code>2:6</code>	<code>2 3 4 5 6</code>	An integer sequence
<code>seq(2, 3, by=0.5)</code>	<code>2.0 2.5 3.0</code>	A complex sequence
<code>rep(1:2, times=2)</code>	<code>1 2 1 2 1 2</code>	Repeat a vector
<code>rep(1:2, each=2)</code>	<code>1 1 2 2 2 2</code>	Repeat elements of a vector

Vector Functions

sort(x) Return x sorted.	rev(x) Return x reversed.
table(x) See counts of values.	unique(x) See unique values.

Selecting Vector Elements

By Position

<code>x[4]</code>	The fourth element.
<code>x[-4]</code>	All but the fourth.
<code>x[2:4]</code>	Elements two to four.
<code>x[-(2:4)]</code>	All elements except two to four.
<code>x[c(1, 5)]</code>	Elements one and five.

By Value

<code>x[x == 10]</code>	Elements which are equal to 10.
<code>x[x < 0]</code>	All elements less than zero.
<code>x[x %in% c(1, 2, 5)]</code>	Elements in the set 1, 2, 5.

Named Vectors

<code>x['apple']</code>	Element with name 'apple'.
-------------------------	----------------------------

Programming

For Loop

```
for (variable in sequence){  
  Do something  
}
```

Example

```
for (i in 1:4){  
  j <- i + 10  
  print(j)  
}
```

While Loop

```
while (condition){  
  Do something  
}
```

Example

```
while (i < 5){  
  print(i)  
  i <- i + 1  
}
```

If Statements

```
if (condition){  
  Do something  
} else {  
  Do something different  
}
```

Example

```
if (1 > 3){  
  print('Yes')  
} else {  
  print('No')  
}
```

Functions

```
function_name <- function(var){  
  Do something  
  return(new_variable)  
}
```

Example

```
square <- function(x){  
  squared <- x*x  
  return(squared)  
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Output	Description
<code>df <- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	Read and write a delimited text file.
<code>df <- read.csv('file.csv')</code>	<code>write.csv(df, 'file.csv')</code>	Read and write a comma separated value file. This is a special case of read table/write table.
<code>load('file.Rdata')</code>	<code>save(df, file = 'file.Rdata')</code>	Read and write an R data file, a file type special for R.

Conditions	<code>a == b</code>	Are equal	<code>a > b</code>	Greater than	<code>a >= b</code>	Greater than or equal to	<code>is.na(a)</code>	Is missing
	<code>a != b</code>	Not equal	<code>a < b</code>	Less than	<code>a <= b</code>	Less than or equal to	<code>is.null(a)</code>	Is null

Cheat sheet - Base R

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

as.logical	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE)
as.numeric	1, 0, 1	Integers or floating point numbers
as.character	'1', '0', '1'	Character strings. Generally preferred to factors.
as.factor	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

Maths Functions

log(x)	Natural log.	sum(x)	Sum.
exp(x)	Exponential.	mean(x)	Mean.
max(x)	Largest element.	median(x)	Median.
min(x)	Smallest element.	quantile(x)	Percentage quantiles.
round(x, n)	Round to n decimal places.	rank(x)	Rank of elements.
signif(x, n)	Round to n significant figures.	var(x)	The variance.
cor(x, y)	Correlation.	sd(x)	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```




The Environment

ls()	List all variables in the environment.
rm(x)	Remove x from the environment.
rm(list = ls())	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

Matrices

```
m <- matrix(x, nrow = 3, ncol = 3)
# Create a matrix from x.
```

 m[2,]	- Select a row	t(m)	Transpose
 m[, 1]	- Select a column	m %*% n	Matrix Multiplication
 m[2, 3]	- Select an element	solve(m, n)	Find x in: m * x = n

Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
# A list is a collection of elements which can be of different types.
```

l[[2]]	l[[1]]	l\$x	l['y']
Second element of l	New list with only the first element.	Element named x.	New list with only element named y.

Also see the **dplyr** package.

Data Frames

```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
# A special case of a list where all elements are the same length.
```

x	y
1	a
2	b
3	c

Matrix subsetting

df[, 2]	
df[2,]	
df[2, 2]	

nrow(df)
Number of rows.

ncol(df)
Number of columns.

dim(df)
Number of columns and rows.

cbind - Bind columns.

Understanding a data frame

View(df)	See the full data frame.
head(df)	See the first 6 rows.

Strings

Also see the **stringr** package.

paste(x, y, sep = ' ')	Join multiple vectors together.
paste(x, collapse = ' ')	Join elements of a vector together.
grep(pattern, x)	Find regular expression matches in x.
gsub(pattern, replace, x)	Replace matches in x with a string.
toupper(x)	Convert to uppercase.
tolower(x)	Convert to lowercase.
nchar(x)	Number of characters in a string.

Factors

factor(x)	Turn a vector into a factor. Can set the levels of the factor and the order.
cut(x, breaks = 4)	Turn a numeric vector into a factor by 'cutting' into sections.

Statistics

lm(y ~ x, data=df)	Linear model.	t.test(x, y)	Perform a t-test for difference between means.	prop.test	Test for a difference between proportions.
glm(y ~ x, data=df)	Generalised linear model.	summary	Get more detailed information out of a model.	pairwise.t.test	Perform a t-test for paired data.
		aov	Analysis of variance.		

Distributions

Random Variables	Density Function	Cumulative Distribution	Quantile
Normal	rnorm	dnorm	pnorm
Poisson	rpois	dpois	qpois
Binomial	rbinom	dbinom	qbinom
Uniform	runif	dunif	qunif

Plotting

Also see the **ggplot2** package.

plot(x)	Values of x in order.	plot(x, y)	Values of x against y.	hist(x)	Histogram of x.
---------	-----------------------	------------	------------------------	---------	-----------------

Dates

See the **lubridate** package.

Seleção e modificação de subconjuntos de dados

- ▶ Supondo que o objecto x tem missing values e que pretendemos eliminá-los. A função `!is.na()` fornece os índices de x que não são NA (ver exemplo).
- ▶ A seleção de um subconjunto de um vector x pode ser feita facilmente, conforme o exemplo.

```
> z<-c(6,1,4,NA,8,2,NA)
> z
[1] 6 1 4 NA 8 2 NA
> z[!is.na(z)]
[1] 6 1 4 8 2
> z[1:4]
[1] 6 1 4 NA
> z[2]
[1] 1
> z[-5]
[1] 6 1 4 NA 2 NA
```

Funções c, paste, cbind, rbind

- ▶ A função `c()` permite combinar valores ou strings num vector, tal como vimos no exemplo do slide 10. Note-se que todos os elementos de entrada devem ser do mesmo tipo.
- ▶ A função `paste()` faz a concatenação de vectores, depois de conversão a string.
- ▶ As funções `cbind()` e `rbind()` combinam objectos (vector, matriz ou data frame) por coluna e linha, respetivamente.

```
> x<-c("Alexandre","Raquel")
> y<-c(15,12)
> z<-c(2,1.5)
> pascoa2<-data.frame(x,y,z)
> names(pascoa2)<-c("Nome","Nr.chocolates.páscoa","Aumento.quilos")
> pascoa2
```

	Nome	Nr.chocolates.páscoa	Aumento.quilos
1	Alexandre	15	2.0
2	Raquel	12	1.5

```
> pascoa_novo<-rbind(pascoa,pascoa2)
> pascoa_novo
```

	Nome	Nr.chocolates.páscoa	Aumento.quilos
1	Soraia	20	2.0
2	Tiago	15	1.5
3	João	16	1.0
4	Mónica	12	1.2
5	Alexandre	15	2.0
6	Raquel	12	1.5

Funções sort e summary

- ▶ A função `sort()` ordena um vector ou factor por ordem ascendente (por defeito) ou descendente.
- ▶ A função `summary()` fornece um sumário de um objecto R (vector, matriz, data.frame ou outro).

```
> sort(pascoa_novo$Nr.chocolates.páscoa)
```

```
[1] 12 12 15 15 16 20
```

```
> summary(pascoa_novo)
```

	Nome	Nr.chocolates.páscoa	Aumento.quilos
João	:1	Min. :12.00	Min. :1.000
Mónica	:1	1st Qu.:12.75	1st Qu.:1.275
Soraia	:1	Median :15.00	Median :1.500
Tiago	:1	Mean :15.00	Mean :1.533
Alexandre	:1	3rd Qu.:15.75	3rd Qu.:1.875
Raquel	:1	Max. :20.00	Max. :2.000

Funções table e tapply

- ▶ A função `table()` cria uma tabela de contingência com contagens.
- ▶ A função `tapply()` aplica uma função (por exemplo média ou soma) às células do vector 1 que pertencem a cada grupo do vector 2. No exemplo em baixo, utilizamos esta função para determinar o aumento médio em quilos por cada número de chocolates digeridos.

```
> table(pascoa_novo$Aumento.kilos)

 1 1.2 1.5  2
 1  1  2  2
> tapply(pascoa_novo$Aumento.kilos,pascoa_novo$Nr.chocolates.pascoa,mean)
 12  15  16  20
1.35 1.75 1.00 2.00
```

Função merge

- ▶ A função `merge()` permite juntar dois data frames por nomes de colunas comuns. No exemplo seguinte criámos uma nova data frame com o número de km percorridos a pé. Note-se que a ordem dos nomes está diferente da ordem do primeiro data frame. Utilizámos a função `merge()` para juntar os dois data frames por nome.

```
> x<-c("João","Raquel","Tiago","Alexandre","Soraia","Mónica")
> y<-c(15,10,8,12,10,7)
> km<-data.frame(x,y)
> names(km)<-c("Nome","km.percorridos")
> km
```

	Nome	km.percorridos
1	João	15
2	Raquel	10
3	Tiago	8
4	Alexandre	12
5	Soraia	10
6	Mónica	7

```
> merge(pascoa_novo,km,by="Nome")
```

	Nome	Nr.chocolates.páscoa	Aumento.quilos	km.percorridos
1	Alexandre	15	2.0	12
2	João	16	1.0	15
3	Mónica	12	1.2	7
4	Raquel	12	1.5	10
5	Soraia	20	2.0	10
6	Tiago	15	1.5	8

Escrever as próprias funções

- ▶ O utilizador pode escrever as suas próprias funções. A syntax é a seguinte:

```
name <- function(arg1, arg2, ...) expression
```

- ▶ Exemplo de função para cálculo de variância amostral:

```
> variancia<-function(y){  
+   n<-length(y)  
+   s2<-(1/(n-1))*(sum(y^2)-n*mean(y)^2)  
+   print(s2)  
+ }  
> variancia(c(1,7,2,4,3))  
[1] 5.3  
> var(c(1,7,2,4,3))  
[1] 5.3
```

E agora, mãos na massa

Vamos fazer alguns exercícios práticos com base em dados.

Mas antes, uma nota. Aqui estamos a trabalhar em R base. Mesmo dentro do R existe todo um outro mundo (o tidyverse e o ggplot2 são outras possíveis formas e filosofias de trabalhar dados dentro do R) que vos desafiamos a explorar por vocês!

Uma analogia:

- ▶ Aqui aprendemos a andar de bicicleta, com rodinhas! E quem sabe mesmo a tirar as rodinhas da bicicleta.
- ▶ Mas... os cavaleiros ficam por vossa conta.
- ▶ Depois de aprender a andar de bicicleta, nunca nos esquecemos.
- ▶ Mas se não praticarmos, vamos cair algumas vezes antes de voltar a conseguir andar com confiança!