

# R Programming: A Crash Course

Mestrado em Análise e Engenharia de Big Data  
Mestrado em Matemática e Aplicações



Faculdade de Ciências e Tecnologia  
Universidade Nova de Lisboa

2019/20

- Este é um (mini)curso sobre a linguagem de programação R pensado e desenhado para quem nunca teve contato, ou teve ainda pouco contato, com o R
- Não é (nem se pretende que seja, seria impossível!) exaustivo
- Tem como objetivo principal dotar os alunos de ferramentas iniciais básicas necessárias ao arranque na utilização do R
- O curso não se esgota em si mesmo! Requer trabalho autónomo e continuado para maior eficiência de trabalho!

# Introdução

- Trata-se de uma linguagem de programação e simultaneamente de um ambiente para computação estatística, cálculo e visualização gráfica de dados. Permite manipular e analisar dados de forma muito eficiente. Uma das suas grandes virtudes é a capacidade gráfica permitindo uma sofisticada visualização gráfica dos dados.
- Criado em 1995, por **Ross Ihaka** e **Robert Gentleman**, Department of Statistics of the University of Auckland, Auckland, New Zealand com base na linguagem S desenvolvida em meados dos anos 70, nos Bell Labs (actualmente Lucent Technologies) por Rick Becker, John Chambers e Allan Wilks. Actualmente é mantido por um grupo alargado de investigadores — R Core Development Team.
- Open source e gratuito
- A instalação do R inclui um conjunto base de packages - livrarias com funções e bases de dados. É possível adicionar novos packages a este conjunto base. Qualquer pessoa pode criar um package e submetê-lo ao portal do R para que aí seja disponibilizado para toda a comunidade científica.

- Manuais (vêm com a instalação do R):

- 1 *An Introduction to R*
- 2 *R Installation and Administration*
- 3 *R Data Import/Export*
- 4 *Writing R extensions*
- 5 *R Language Definition*
- 6 *Sweave User Manual*

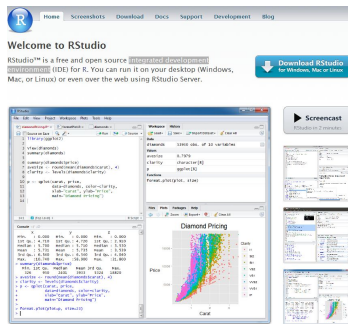
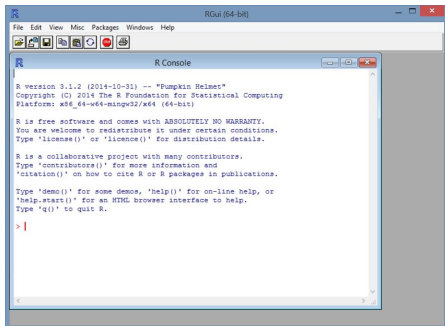
- Outra documentação de distribuição gratuita:

- 1 *Using R for Data Analysis and Graphics - Introduction, Examples and Commentary*, John Maindonald
- 2 *Simple R*, John Verzani
- 3 *Practical Regression and Anova using R*, Julian Faraway
- 4 *An Introduction to R: Software for Statistical Modelling and Computing*, Petra Kuhnert and Bill Venables
- 5 *R for Beginners*, Emmanuel Paradis
- 6 *Gráficos Estadísticos con R*, Juan Carlos Correa and Nelfi González
- 7 *R reference card*, Tom Short
- 8 *The R Inferno*, Patrick Burns

- Livros: Books related to R

- 1 Crawley (2014). *The R book*. John Wiley & Sons
- 2 Torgo (2009). *A linguagem R - Programação para a análise de dados*. Escolar Editora

# Interagir com o R: RGui vs. RStudio



- Um ambiente de desenvolvimento integrado (*integrated development environment*) para o R

Website: [www.rstudio.org](http://www.rstudio.org)



Welcome to RStudio

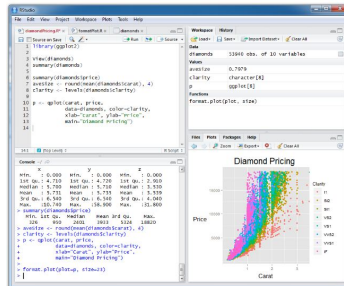
RStudio™ is a free and open source **integrated development environment** (IDE) for R. You can run it on your desktop (Windows, Mac, or Linux) or even over the web using RStudio Server.



- Ambiente *user-frendly*

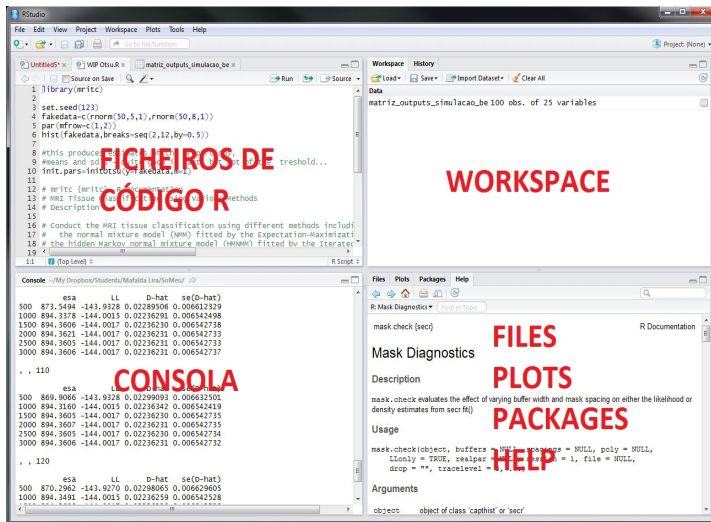
- Facilita a interação com o R, nomeadamente à custa de inúmeros atalhos e *syntax highlighting*

- Numa única aplicação acesso a uma panóplia de opções como os ficheiros de código, a linha de comandos, as figuras, a ajuda, etc.



**Screenshot**  
RStudio in 2 minutes

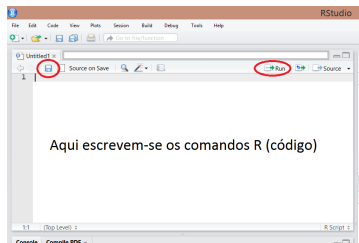






# Ficheiros de código R (*R Scripts*)

- Para criar um novo ficheiro R: *File -> New File -> R Script*

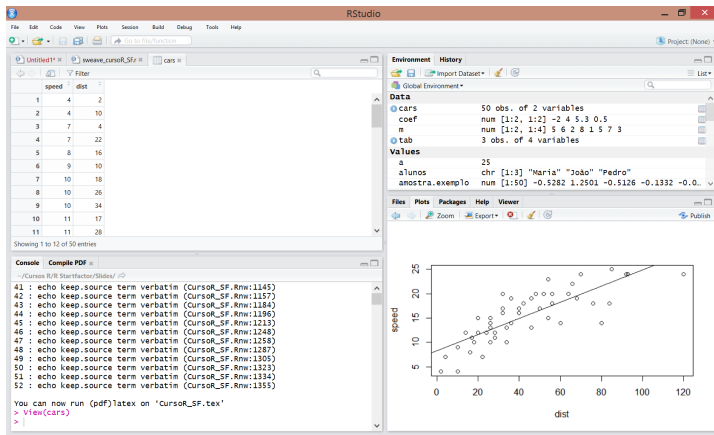


- **Todo** o código deve ser escrito neste ficheiro!
- Para executar o código (posicionar cursor na linha ou selecionar código): *Run* ou *Ctrl+Enter*
- Todo o código deve ser extensamente comentado! Usando o símbolo cardinal, tudo o que é escrito à frente não é lido como código R

```
> 3+3 # Isto é uma continha!
```

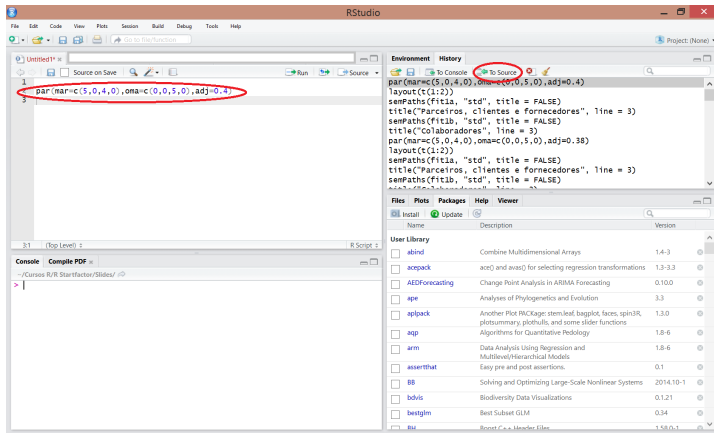
- Para guardar o ficheiro (extensão *.R*): *File -> Save as/Save* ou botão de acesso direto.

- Ambiente com todos os objetos (vetores, matrizes de dados, funções, etc) criados numa sessão de trabalho



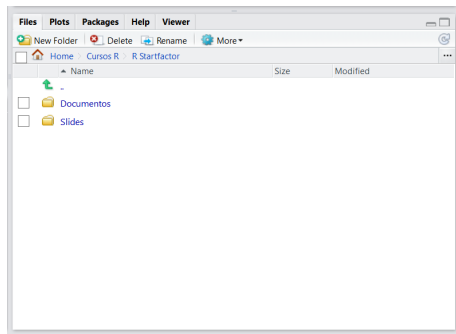
- Permite editar e visualizar dados
- Pode (ou não) salvar-se (extensão .RData) e carregar-se posteriormente

- Armazena história de comandos

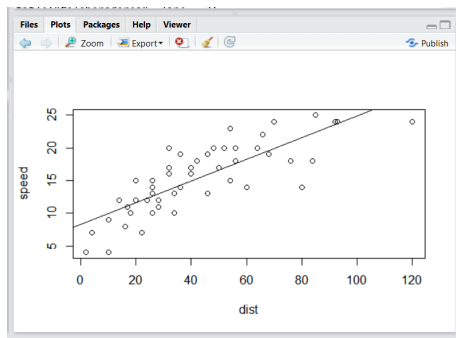


- Inclui atalho (*To Source*) para copiar código da história de comandos para o script

- Permite gerir ficheiros e diretorias de trabalho.
- Para fixar uma diretoria de trabalho: seleccionar (ou criando) a diretoria onde se quer trabalhar, clicar em *More -> Set as working directory*



- Janela gráfica, com setas de navegação



- Um package não é mais do que um grupo de objectos (funções, dados, etc.) prontos as ser usados pelo utilizador.
- Uma instalação do R vem com uma base de packages instaladas.
- Sempre que necessário podem instalar-se novos packages
- A instalação de um novo package pode fazer-se usando o botão *install* ou a função `install.packages()`

```
> install.packages(nortest)
```

- Para usar os packages não incluídos na base do R é necessário torna-los ativos, usando a função `library()`.

```
> library(nortest)
```

- No caso de se saber o nome da função que se pretende usar, a obtenção de toda a informação sobre a sua utilização é feita simplesmente precedendo de um ponto de interrogação do nome da função

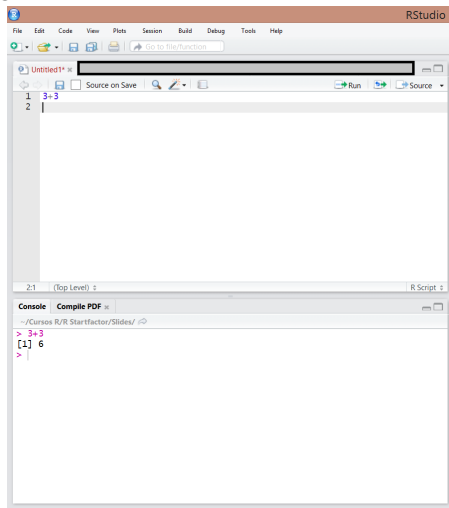
```
> ?glm
```

- Quando se pretende saber quais as funções disponíveis num determinado âmbito ou relacionadas com um conceito específico, pode obter-se ajuda através do comando `help.search()`

```
> help.search("median")
```

- A execução deste comando fornece a listagem completa das funções que contêm referências ao texto usado como argumento na função `help.search`.

- Onde aparece o código executado e resultados obtidos



- Na consola aparece a prompt (>) seguida do código executado
- Note-se que sempre que o código está incompleto ou está escrito usando várias linhas, aparece o simbolo (+) no lugar da prompt



## Fundamentos da linguagem

- O R é uma linguagem de programação orientada a objetos, isto é, toda a informação — números, texto, vectores, funções,... — é organizada na forma de um objecto (entidade com *identidade própria* ou *self*).
- Para armazenar/criar um objecto com um determinado **nome** usa-se o *operador de atribuição* (`<-`) na forma **object\_name <- content**

```
> a <- 25  
> b <- 5  
> Total <- a+b
```

- Para conhecer o conteúdo de um objecto basta digitar a sua designação e executar essa linha de código

```
> Total  
[1] 30
```

- A listagem dos objectos guardados em memória pode obter-se usando as funções **ls()** ou **objects()**.

```
> ls()  
[1] "a"      "b"      "Total"
```

- Para remover objectos da memória tem-se a função **rm()**. Para remover todos **rm(list = ls())**

Os tipos básicos de objectos do R incluem:

- **Vectores:** Estruturas de dados que permitem armazenar um conjunto de valores (numéricos ou não), sob um mesmo nome.
- **Arrays** (caso particular, **Matrizes**): Estruturas multidimensionais, indexadas em função de 2 (neste caso, têm-se as matrizes) ou mais índices.
- **Factores:** Vetores de armazenamento de variáveis qualitativas.
- **Listas:** Colecção de objectos de diferentes tipos.
- **Data Frames:** Estruturas tipo-matriz. Análogo a bases de dados.
- **Funções:** Estruturas organizadas do R que permitem incorporar um conjunto de instruções que serão executadas conjuntamente: *Built-in* vs. *user-defined*.

## Estrutura das funções:

```
function_name(arg1 = value1, arg2 = value2,...)
```

- ▶ Os vários argumentos (caso existam vários) são sempre separados por vírgulas;
- ▶ Os argumentos podem ser explicitados pela sua designação:

```
> rep(x = 5, times = 3)
```

ou pela sua ordem:

```
> rep(5,3)
```

- ▶ Explicitar o nome dos argumentos permite ignorar a sua ordem.
- ▶ Quando não se usa o nome dos argumentos tem de se respeitar a sua ordem dentro da função.

**Modo** Representa a forma como o objeto é armazenado no R (atributo intrínseco). Os modos mais comuns (mutuamente exclusivos) são *numeric*, *complex*, *logical*, *character*. Obtém-se usando a função `mode()`

```
> mode(Total)
[1] "numeric"
> c<-"Grupo controle"
> mode(c)
[1] "character"
```

**Classe** A classe de um objeto define o seu "comportamento" (atributo extrínseco). As classes incluem o modo e ainda as classes *matrix*, *dataframe*, *factor*, *array*, *list*. Obtém-se usando a função `class()`

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

**Tamanho** Dá informação sobre número de elementos num vetor ou lista e sobre o número de colunas numa data frame ou matriz. Obtém-se usando a função `length()` para vetores/fatores e `dim()` para arrays/matrizes

```
> length(Total)
[1] 1
```

- A função `c()` permite criar vetores separando os elementos por vírgulas

```
> x<-c(7, 3.1, 8, 18, 12.5)
> x
[1] 7.0 3.1 8.0 18.0 12.5
```

Alguns casos particulares:

- Sequências: Função `seq(from,to)` e operador `from:to`.

```
> seq(1,5)
[1] 1 2 3 4 5
> 1:5
[1] 1 2 3 4 5
```

- Repetições: `rep(x,times)`

```
> rep(3,4)
[1] 3 3 3 3
```

- Para criar matrizes (caso particular das arrays) usa-se a função específica `matrix(x,nrow,ncol)`

```
> m<-matrix(c(5,6,2,8),2,2)
> m
      [,1] [,2]
[1,]    5    2
[2,]    6    8
```

- As funções matemáticas são genericamente aplicáveis a matrizes (*regra da reciclagem*)
- Operações com matrizes (álgebra linear):

```
> n<-matrix(c(1,2,3,4),2,2)
> m*n #produto elemento a elemento
> t(m) #transposta
> t(m)%*%n #produto matricial
> crossprod(m,n) #produto matricial
> diag(m) #diagonal
> diag(diag(m)) #matriz diagonal
```

- Combinar argumentos por linhas e colunas: `rbind()` e `cbind()`

```
> cbind(m,m)
      [,1] [,2] [,3] [,4]
[1,]    5    2    5    2
[2,]    6    8    6    8
```

- A função `factor(x)` permite que o R reconheça um conjunto de dados como sendo qualitativos

```
> f<-factor(c("m","m","m","f","m","f"))
> f

[1] m m m f m f
Levels: f m
```

- Alternativas:

```
> factor(c(1,1,1,0,1,0),labels = c("f","m"))

[1] m m m f m f
Levels: f m

> factor(c(1,1,1,0,1,0),labels = c("m","f"),levels = c(1,0))

[1] m m m f m f
Levels: m f
```



- Para criar tabelas de dados (tabelas de dupla entrada) organizadas, atribuindo nomes às colunas, usa-se a função `data.frame()`

```
> alunos<-c("Maria","João","Pedro")
> tab<-data.frame(id=c(5,6,7),nomes=alunos,prof=rep("Filo",3))
> tab
```

	id	nomes	prof
1	5	Maria	Filo
2	6	João	Filo
3	7	Pedro	Filo

- Para aceder às componentes de uma data frame pode usar-se a função `attach()`.

```
> attach(tab)
> id
```

[1]	5	6	7
-----	---	---	---

- Não esquecer de usar a função `detach()`

```
> detach(tab)
```

- Veremos mais adiante que existem outras formas de aceder às componentes de uma data frame (indexação)

- A função `list()` permite criar coleções de objetos de diferentes tipos (com atribuição de nomes)

```
> l<-list(id=c(5,6,7),nomes=alunos,prof="Filo")  
> l
```

```
$id  
[1] 5 6 7
```

```
$nomes  
[1] "Maria" "João"  "Pedro"
```

```
$prof  
[1] "Filo"
```

```
> minhalista <- list (a = 1:5, b = "Estudo 1", c = matrix(c(2,4,8,9),2,2))  
> minhalista
```

```
$a  
[1] 1 2 3 4 5
```

```
$b  
[1] "Estudo 1"
```

```
$c  
      [,1] [,2]  
[1,]    2    8  
[2,]    4    9
```

O R usa os seguintes operadores básicos:

- **Aritmética:**  $+$   $-$   $*$   $/$   $^$
- **Relacional:**  $>$   $>=$   $<$   $<=$   $==$   $!=$
- **Lógica:**  $!$ (negação)  $\&$ (conjunção)  $|$ (disjunção)
- **Formulação de modelos:**  $\sim$
- **Indexação:**  $\$$
- **Sequência:**  $:$
- **Atribuição:**  $->$   $<-$   $=$

Alguma notação:

- Infinito:  $\infty = \text{Inf}$ ;  $-\infty = -\text{Inf}$ .
- Indeterminações:  $\infty/\infty = \text{NaN}$  (*Not a Number*).
- *Missing values* são assinalados com **NA** (*Not Available*).

```
> w<-c(7, 3.1, 8, 18, NA)
```

A função **is.na(x)** permite identificar a localização dos NA.

```
> is.na(w)
```

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

## Indexação em vectores

- `x[n]`: n-ésimo elemento
- `x[-n]`: todos os elementos, excepto o n-ésimo
- `x[1:n]`: primeiros n elementos
- `x[c(1,2)]`: elementos específicos
- `x[x>2 & x<4]`: elementos com valor entre 2 e 4
- `x[x %in% c(1:5)]`: elementos com valor pertencente a um conjunto

## Indexação em matrizes e data frames

- `x[i,j]`: elemento na linha *i* e coluna *j*
- `x[i,]`: linha *i*
- `x[,j]`: coluna *j*
- `x[c(1,3),]`: linhas 1 e 3

## Indexação em data frames

- `dataset$x`: coluna *x* da data frame *dataset*

## Indexação em listas

- `x[[n]]`: n-ésimo elemento

# Transformação do tipo de objecto

- Para identificar o tipo/modo de objecto em uso utiliza-se as funções genéricas `is.type()` e `is.mode()`
- Para modificar o tipo/modo de objecto usa-se as funções `as.type()` e `as.mode()`

Type	Função <code>is.type()</code>	Função <code>as.type()</code>
Array	<code>is.array()</code>	<code>as.array()</code>
Dataframe	<code>is.data.frame()</code>	<code>as.data.frame()</code>
Factor	<code>is.factor()</code>	<code>as.factor()</code>
List	<code>is.list()</code>	<code>as.list()</code>
Matrix	<code>is.matrix()</code>	<code>as.matrix()</code>
Vector	<code>is.vector()</code>	<code>as.vector()</code>
Mode	Função <code>is.mode()</code>	Função <code>as.mode()</code>
Character	<code>is.character()</code>	<code>as.character()</code>
Complex	<code>is.complex()</code>	<code>as.complex()</code>
Logical	<code>is.logical()</code>	<code>as.logical()</code>
Numeric	<code>is.numeric()</code>	<code>as.numeric()</code>

```
> is.matrix(m)
[1] TRUE

> as.vector(m)
[1] 5 6 2 8
```

```
> is.factor(f)
[1] TRUE

> as.numeric(f)
[1] 2 2 2 1 2 1
```

- As funções são também objetos. Existe um conjunto muito alargado de funções *built-in*.
- Algumas funções matemáticas: `sum(x)`, `sqrt(x)`, `log(x)`, `log(x,n)`, `exp(x)`, `choose(n,x)`, `rank(x)`, `factorial(x)`, `floor(x)`, `ceiling(x)`, `round(x, digits)`, `abs(x)`, `cos(x)`, `sin(x)`, `tan(x)`, `acos(x)`, `acosh(x)`, `gamma(x)`

```
> floor(3.5)
[1] 3
> ceiling(3.5)
[1] 4
```

- Algumas estatísticas: `max(x)`, `min(x)`, `mean(x)`, `median(x)`, `range(x)`, `var(x)`, `cor(x,y)`, `quantile(x)`, `cumsum(x)`, `cumprod(x)`, `cummax(x)`, `cummin(x)`

```
> y<-c(3.4,6.9,9.4,5.1,3.6)
> cummax(y)
[1] 3.4 6.9 9.4 9.4 9.4
```

- Existe uma família de funções, particularmente útil, para lidar com estruturas multidimensionais tipo matriz e dataframe

```
> apropos("apply")  
[1] ".rs.applyTransform" "apply"           "dendraply"  
[4] "eapply"             "kernapply"       "lapply"  
[7] "mapply"             "rapply"          "sapply"  
[10] "tapply"            "vapply"
```

- Um exemplo simples:

```
> apply(m,1,mean)  
[1] 3.5 7.0  
  
> apply(m,1,sd)  
[1] 2.121320 1.414214  
  
> apply(m,2,sum)  
[1] 11 10
```

## Outras funções genéricas

- Para aceder a objetos dentro de outros objetos: `with()`
- Para aceder a objetos dentro de outros objetos, e fazer modificações: `within()`
- Para ver os primeiros e últimos elementos de um objecto: `head()` e `tail()`

```
> #example use of with and within
> data(cars)
> head(cars)

  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10

> with(cars, mean(speed))
[1] 15.4

> temp<-within(cars, assign("ratio", speed/dist))
> head(temp, 4)

  speed dist    ratio
1     4    2 2.0000000
2     4   10 0.4000000
3     7    4 1.7500000
4     7   22 0.3181818

> rm(temp)
```



Na implementação base do R existem disponíveis inúmeras distribuições, por exemplo:

Distribuição	Nome no R	Argumentos
Beta	beta	shape1, shape2
Binomial	binom	size, prob
Binomial negativa	nbinom	size, prob
Cauchy	cauchy	location, scale
Exponencial	exp	rate
F-Snedecor	f	df1, df2
Gama	gamma	shape, scale
Geométrica	geom	prob
Hipergeométrica	hyper	m, n, k
Log-normal	lnorm	meanlog, sdlog
Logística	logis	location, scale
Normal	norm	mean, sd
Poisson	pois	lambda
Qui-quadrado	chisq	df
t-Student	t	df
Uniforme	unif	min, max
Weibull	weibull	shape, scale

Existem quatro funções-tipo aplicáveis às distribuições:

- **Função densidade/massa de probabilidade,  $f(x)$ :** `dnome(x, ...)`

Fornece o valor da função —  $f(a)$  — para um dado valor  $x = a$ , i.e., a probabilidade no ponto:  $P(X = a)$ .

*Exemplo:*  $X \sim N(8.5, 2.3) \Rightarrow P(X = 10) = ?$

```
> dnorm(9, 8.5, 2.3)
```

```
[1] 0.1694026
```

- **Função de distribuição,  $F(x)$ :** `pnome(q, ...)`

Fornece o valor da função —  $F(x)$  — para um dado valor  $x$ , i.e., a probabilidade acumulada no ponto  $x$ :  $P(X \leq x)$ .

*Exemplo:*  $X \sim N(8.5, 2.3) \Rightarrow P(X \leq 10) = ?$

```
> pnorm(10, mean=8.5, sd=2.3)
```

```
[1] 0.7428555
```

- **Função quantil (inversa da função de distribuição),  $F^{-1}(x)$ :** `qnome(p, ...)`

Fornece o valor —  $x$  — dado o valor de  $F(x)$ , i.e., conhecida a probabilidade acumulada no ponto:  $P(X \leq x)$  indica qual o valor de  $x$ .

*Exemplo:*  $X \sim N(8.5, 2.3) \Rightarrow P(X \leq ?) = 0.743$

```
> qnorm(0.743, 8.5, 2.3)
[1] 10.00103
```

- **Geração de variáveis pseudoaleatórias:** `rnome(n, ...)`

```
> rn<-rnorm(1000)
```

- Estruturas condicionais: **if** e **ifelse**

```
if(condição) expr1 else expr2
```

```
> notas<-c(12,5.3,15,7.0,17)
> if(notas[1]<9.5) print("r") else print("a")
[1] "a"
```

```
ifelse(condição,expr1,expr2)
```

```
> ifelse(notas<9.5,"reprovado","aprovado")
[1] "aprovado" "reprovado" "aprovado" "reprovado" "aprovado"
```

- Estruturas de repetição: **for**, **while** e **repeat** . Instrução **break**.

```
for (nome in expr1) expr2
```

```
> for(x in c(4,9,16,25)) print(sqrt(x))
[1] 2
[1] 3
[1] 4
[1] 5
```

`while (condição) expr`

```
> a <- 0; b <- 1
> while (b < 4) {
  print(b)
  temp <- a + b
  a <- b
  b <- temp
}

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

`repeat expr`

`break`

```
> x<-1
> repeat{print(x)
  x = x+1
  if(x == 4) {break}}

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

- As funções criadas pelo utilizador são definidas por um conjunto de instruções e aplicadas sobre um conjunto de *argumentos*.
- Sintaxe de uma função: `> nome <- function(arg1,arg2,...) {expressão}`
- Exemplo de uma função muito simples e bem conhecida  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

```
> minhamedia0<-function(x){sum(x)/length(x)}  
> y  
[1] 3.4 6.9 9.4 5.1 3.6  
> minhamedia0(y)  
[1] 5.68
```

```
> minhamedia1<-function(x){  
  soma<-sum(x)  
  n<-length(x)  
  m<-soma/n  
  return(m)  
}
```

- Desafio - função densidade de probabilidade Gaussiana:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad -\infty < x < +\infty, \quad -\infty < \mu < +\infty, \quad 0 < \sigma < +\infty$$

```
> gaussianal<-function(x,m,s){  
  (1/(sqrt(2*pi)*s))*exp(-0.5*((x-m)/s)^2)  
}  
> gaussianal(0,0,1)  
[1] 0.3989423
```

```
> gaussian2<-function(x,m,s){  
  z<-(x-m)/s  
  c<-(1/(sqrt(2*pi)*s))  
  p<- -0.5*z^2  
  g<-c*exp(p)  
  return(g)  
}
```

```
> gaussian2(0,0,1)  
[1] 0.3989423  
> dnorm(0,0,1)  
[1] 0.3989423
```

## Bases de dados internas:

- Para visualizar ou aceder às bases de dados disponíveis nos packages em uso, utiliza-se a função `data()`

```
> data(package="datasets") #Mostra as bases de dados existentes  
> data(Puromycin,package="datasets") #Acede à base de dados  
> head(Puromycin)
```

## Bases de dados externas:

- Para facilitar a leitura, os ficheiros de dados devem estar organizados de acordo com algumas regras básicas:
  - 1 Colunas - variáveis; Linhas - casos (observações);
  - 2 A primeira linha deve conter o nome das variáveis (não pode começar por um número);
  - 3 Dados omissos devem ser codificados com NA ou deixar célula em branco;
- Hoje em dia já existem packages específicos para ler as bases de dados externas mais usuais (e.g., Excel, SPSS,...). Em particular,
  - 1 `heaven` lê SPSS, Stata, e ficheiros SAS;
  - 2 `readxl` lê ficheiros Excel (.xlsx ou .xls).



- A leitura de bases de dados externas produz "tibbles"
- *Tibbles* são data frames "simplificadas" - por exemplo, não converte strings em fatores, admite nome com sintaxe "desadequada",...

```
> library(readxl)
> dados<-read_excel("imc.xlsx",sheet = 1,col_names = T)
> dados
```

```
# A tibble: 540 x 7
  escola idade sexo    imc pabdom panca mgorda
  <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl>
1      2      7 F    15.8    59    67    14.9
2      2      6 F    13.8    50    60    11.1
3      2      7 M    16.3    62    69    16.5
4      2      6 M    17.1    62    70     21
5      2      7 F    20.2    70    79    30.2
6      2      7 F    13.3    50    61    11.4
7      2      7 M    14.6    56    62    14.9
8      2      6 M    15.2    53    65    16.4
9      2      7 F    18.8    65    67    26.7
10     2      6 F    15.6    53    67    14.3
# ... with 530 more rows
```

- Por questões de familiarização com as estruturas optaremos por converter as tibbles em data frames
- Existem funções exclusivas para tibbles (potencialmente simplificadoras de alguns processos) que deixamos para o aluno explorar

```
> dados<-as.data.frame(dados)
```

```
> dados[1:6,]
```

	escola	idade	sexo	imc	pabdom	panca	mgorda
1	2	7	F	15.78	59	67	14.9
2	2	6	F	13.84	50	60	11.1
3	2	7	M	16.27	62	69	16.5
4	2	6	M	17.10	62	70	21.0
5	2	7	F	20.18	70	79	30.2
6	2	7	F	13.26	50	61	11.4

```
> dados[1:6, 4]
```

```
[1] 15.78 13.84 16.27 17.10 20.18 13.26
```

```
> dados[1:6, "panca"]
```

```
[1] 67 60 69 70 79 61
```

```
> mean(dados$imc)
```

```
[1] 17.2073
```

- A dimensão da base de dados é dada pelas funções `dim()` ou `nrow()` e `ncol()`

```
> dim(dados)
[1] 540  7
```

- Para modificar uma variável lida como numérica para factor usa-se a função `factor()`

```
> dados$escola <- factor(dados$escola, labels=c("EscA","EscB","EscC","EscD"))
```

- A função `str()` resume a base de dados

```
> str(dados)

'data.frame':      540 obs. of  7 variables:
 $ escola: Factor w/ 4 levels "EscA","EscB",...: 2 2 2 2 2 2 2 2 2 2 ...
 $ idade : num  7 6 7 6 7 7 7 6 7 6 ...
 $ sexo  : chr  "F" "F" "M" "M" ...
 $ imc   : num  15.8 13.8 16.3 17.1 20.2 ...
 $ pabdom: num  59 50 62 62 70 50 56 53 65 53 ...
 $ panca : num  67 60 69 70 79 61 62 65 67 67 ...
 $ mgorda: num  14.9 11.1 16.5 21 30.2 11.4 14.9 16.4 26.7 14.3 ...
```

- Para seleccionar um subconjunto de dados, tem-se a função `subset()`

```
> EscolaA<-subset(dados,escola=="EscA")
```

- A selecção de dados também é possível indexando as bases de dados a determinadas condições

```
> dados[dados$escola=="EscA",c(3,4)]
```

```
> dados[dados$imc>25 & dados$sexo=="M",]
```

	escola	idade	sexo	imc	pabdom	panca	mgorda
529	EscC	9	M	25.52	84	93	28.2
540	EscC	9	M	25.08	79	85	29.8

- A selecção aleatória de uma subamostra faz-se usando a função `sample()`

```
> n.amostra<-4
```

```
> dados[sample(1:nrow(dados),n.amostra),]
```

	escola	idade	sexo	imc	pabdom	panca	mgorda
127	EscB	8	F	15.32	60	71	17.2
435	EscC	8	M	16.90	63	74	13.4
517	EscC	9	F	18.36	69	77	23.2
528	EscC	9	F	19.56	69	77	26.7

- A ordenação de uma base de dados, em função dos valores de uma das variáveis pode fazer-se usando a função `order()` (ordenação crescente) ou utilizando as funções `rev()` e `order()` (ordenação decrescente)

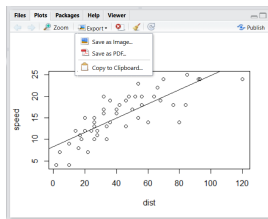
```
> dados[order(dados$idade),]  
> dados[rev(order(dados$idade)),]
```

- Para ordenar em função dos valores de duas variáveis

```
> dados[1:5,]  
  escola idade sexo   imc pabdom panca mgorda  
1  EscB     7    F 15.78    59   67   14.9  
2  EscB     6    F 13.84    50   60   11.1  
3  EscB     7    M 16.27    62   69   16.5  
4  EscB     6    M 17.10    62   70   21.0  
5  EscB     7    F 20.18    70   79   30.2  
  
> dados_ord<-dados[order(dados$escola,dados$idade),]  
> dados_ord[1:5,]
```

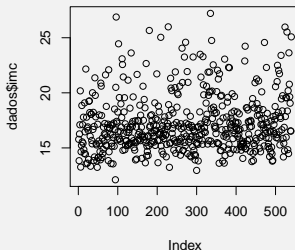
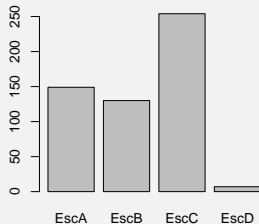
```
  escola idade sexo   imc pabdom panca mgorda  
160  EscA     6    F 16.88    58   63   21.5  
163  EscA     6    M 16.00    57   64   17.8  
164  EscA     6    M 20.60    70   74   25.7  
165  EscA     6    M 17.51    60   63   19.8  
166  EscA     6    M 15.75    53   58   19.2
```

- Existem três sistemas gráficos:
  - 1 **base** (package graphics)
  - 2 **lattice** (packages lattice)
  - 3 **ggplot2** (packages ggplot2)
- As funções disponíveis para desenhar gráficos agrupam-se em três classes:
  - 1 **Funções High-level**: geram gráficos completos
  - 2 **Funções Low-level**: adicionam componentes a gráficos já existentes
  - 3 **Funções interativas**: permitem extrair informação por interação com gráficos já desenhados
- Sempre que um gráfico é criado é possível copiar ou salvar o gráfico em diferentes formatos (por exemplo, .pdf, .png, .jpg, .eps, .ps,...).



- A função `plot()` é uma função genérica que gera um gráfico cujo tipo depende da classe dos seus argumentos

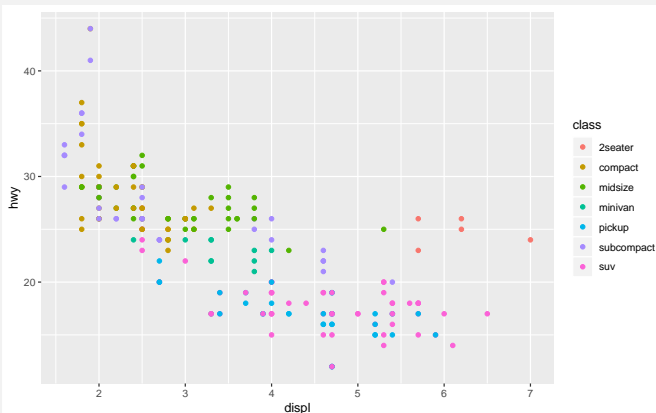
```
> plot(dados$escola)
> plot(dados$imc)
```



- Outras funções comuns para gráficos incluem: `boxplot()`, `barplot()`, `hist()`, `pie()`, `qqnorm()`, `qqplot()`, `curve()`, ... Dica: `apropos("plot")` mostra muitas outras.

- O ggplot2 é um sistema de criação de gráficos baseado numa construção por camadas, sendo a primeira dada pela função `ggplot()`

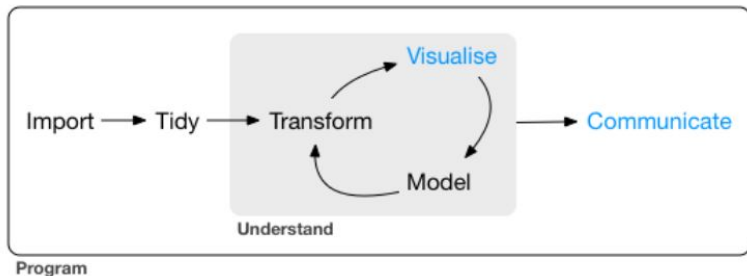
```
> library(ggplot2)
> ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```





## R Markdown

- Uma parte importante da ciência dos dados é a **comunicação** dos métodos e resultados



<https://r4ds.had.co.nz/communicate-intro.html>

Fonte:

- O R Markdown é uma ferramenta que permite integrar texto, código e resultados, possuindo diversos tipos de outputs possíveis, e.g., HTML, pdf, word, slideshows,...
- Os documentos .Rmd permitem reproduzir por completo a análise e resultados obtidos (*reproducible research*)

O R Markdown permite integrar a utilização dos seguintes recursos:

- R — linguagem de programação (análise de dados) através do RStudio — ambiente de desenvolvimento integrado (IDE - *integrated development environment*). Em particular, serão necessários os packages `rmarkdown` e `knitr`:

```
> install.packages(rmarkdown)
> install.packages(knitr)
```

- TeX/LaTeX — Linguagem/sistema tipográfico, adequado para produzir documentos que integram conteúdos de matemática, com grande qualidade tipográfica, através dos compiladores MiKTeX/MacTeX

Como se processa?

- O R Markdown envia o ficheiro `.Rmd` para o `knitr` que interpreta e executa o código, produzindo um documento markdown (`.md`) que inclui o **código** e os **resultados da sua execução**.
- Este ficheiro é processado pelo `pandoc` (conversor de formatos) criando o output final no formato escolhido



Fonte: <https://r4ds.had.co.nz/r-markdown.html>

Um ficheiro .Rmd inclui:

- Cabeçalho — Estrutura que inclui os metadados do documento. Inicializado e finalizado por duas linhas de três traços:

```
---  
title: "Exemplo"  
author: "RB"  
date: "3 de setembro de 2019"  
output: html_document  
---
```

- *Chunks* — Pedacos de código R. Código inicializado e finalizado por 3 *backticks* (acento agudo):

código R:|

```
```{r cars}  
summary(cars)  
```
```

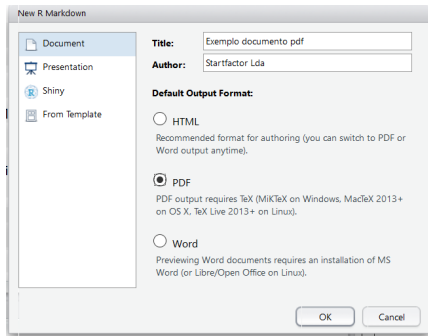
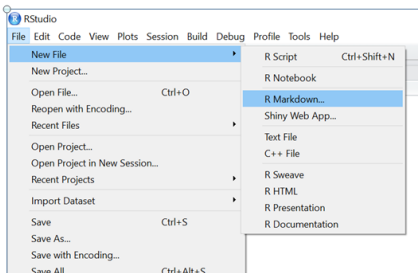
- Texto com formatações básicas (e.g. itálico, bold,...) e pedacos de código inline inicializado e finalizado por 1 *backtick*:

Média igual a ``r round(mean(cars$speed),1)``.

# Criar um ficheiro .Rmd

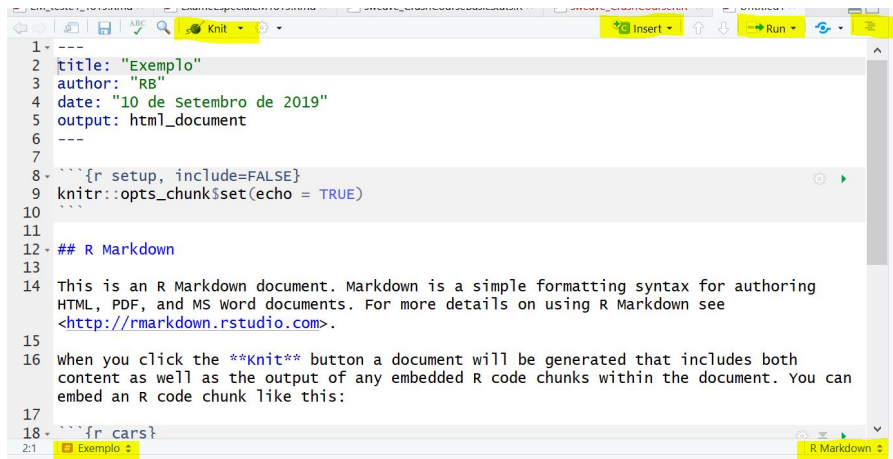
Um ficheiro .Rmd inclui:

- Aceder a *File > New file > R Markdown...*

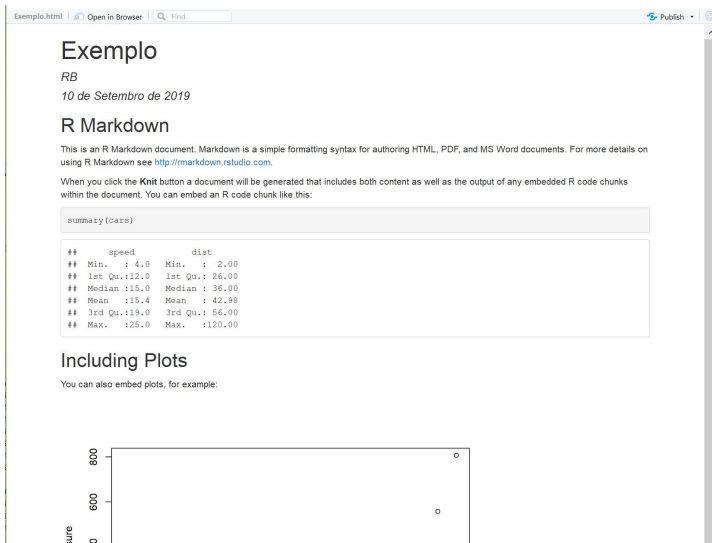


- O R apresenta um *template* para usar como ponto de partida
- Clicar no botão *Knit* para produzir o documento no formato escolhido





```
1 ---
2 title: "Exemplo"
3 author: "RB"
4 date: "10 de Setembro de 2019"
5 output: html_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring
15 HTML, PDF, and MS Word documents. For more details on using R Markdown see
16 <http://rmarkdown.rstudio.com>.
17
18 When you click the Knit button a document will be generated that includes both
19 content as well as the output of any embedded R code chunks within the document. You can
20 embed an R code chunk like this:
21
22 ```{r cars}
```



- Website: <https://rmarkdown.rstudio.com/>
- Online Guide: <https://bookdown.org/yihui/rmarkdown/>
- No menu Help do RStudio estão incluídos dois recursos úteis (sobretudo em fases de aprendizagem iniciais):
  - ❶ R Markdown Cheat Sheet: *Help > Cheatsheets > R Markdown Cheat Sheet*
  - ❷ R Markdown Reference Guide: *Help > Cheatsheets > R Markdown Reference Guide*
- Os dois documentos estão disponíveis <http://rstudio.com/cheatsheets>
- Outros sites:
  - ❶ Rob J Hyndman's personal website: <https://robjhyndman.com> (a very comprehensive academic website).
  - ❷ Amber Thomas's personal website: <https://amber.rbind.io> (a rich project portfolio).
  - ❸ Emi Tanaka's personal website: <https://emitanaka.github.io> (in particular, check out the beautiful showcase page).
  - ❹ "Live Free or Dichotomize" by Nick Strayer and Lucy D'Agostino McGowan: <http://livefreeordichotomize.com> (the layout is elegant, and the posts are useful and practical).