



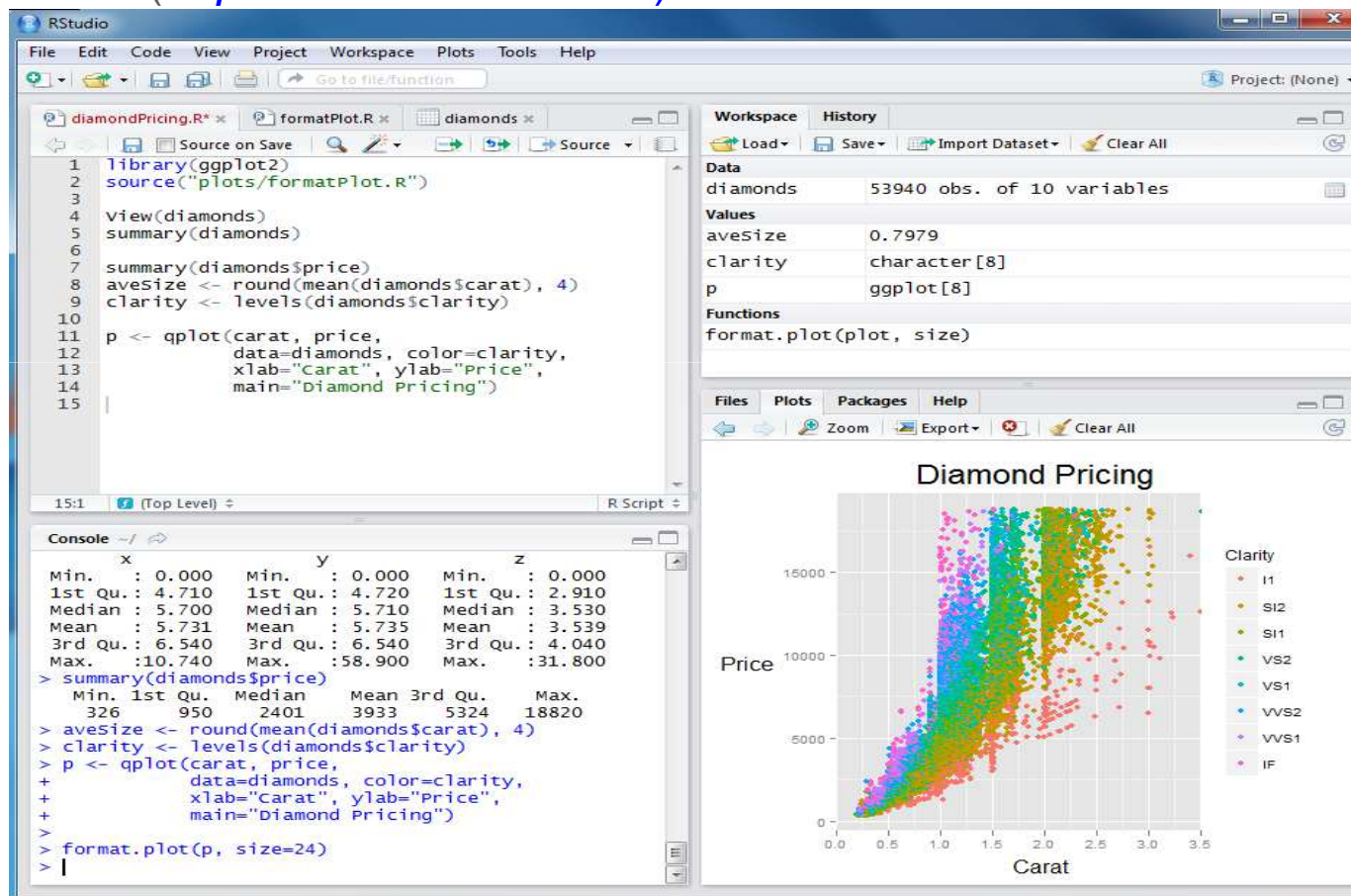
## II. Noções Básicas do R





## II. Noções Básicas do R

- RStudio (<http://www.rstudio.com/>)



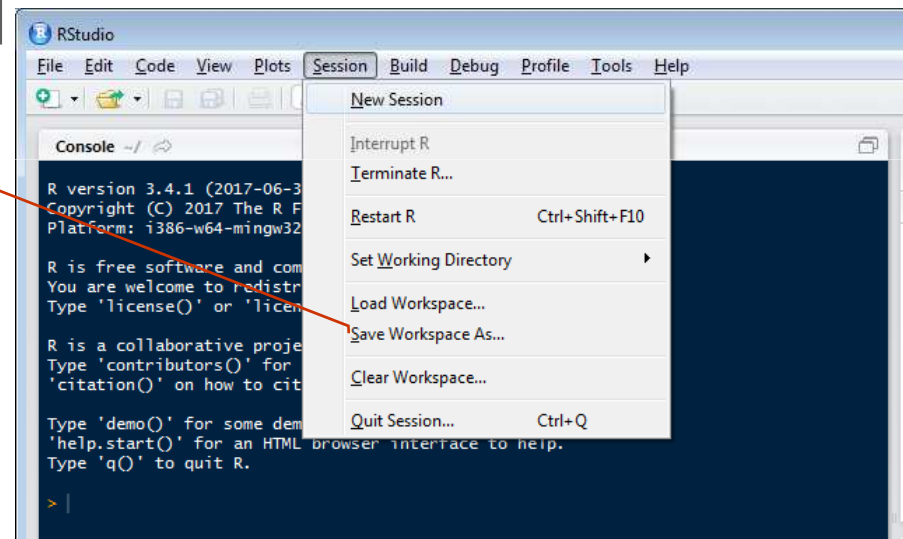


## II. Noções Básicas do R

- ❖ Guardar a sessão de trabalho (*workspace*), `save.image()`

```
> save.image("trabalho.RData")
```

Opção alternativa



O ficheiro *workspace* é **\*.RData**





## II. Noções Básicas do R

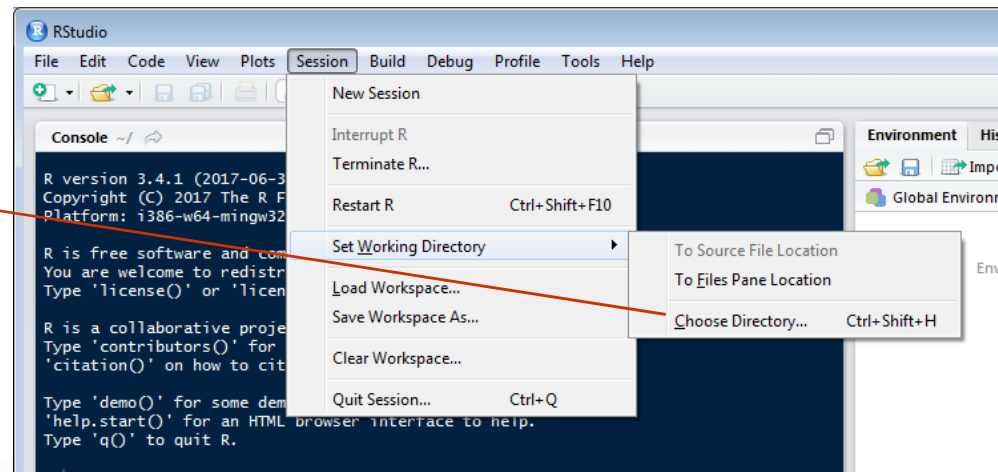
- Os ficheiros de sessão são sempre guardados na diretoria atual de trabalho, que pode ser consultada, `getwd()` :

```
> getwd()  
[1] "C:/Documents and Settings/.../My Documents"
```

- Alterar a diretoria de trabalho, `setwd()`

```
> setwd("C:\\Formacao\\CursoR")  
> setwd("C:/Formacao/CursoR")
```

Opção alternativa



## II. Noções Básicas do R

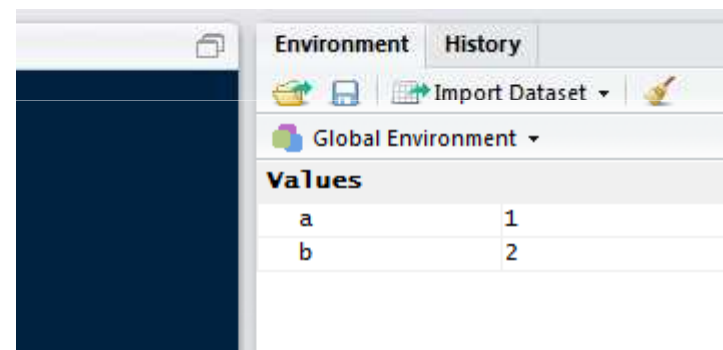
- ❖ Listagem de todos os objetos criados no R, `ls()`, `objects()`

```
> a = 1  
> b = 2  
> ls()  
[1] "a" "b"
```

```
> a = 1  
> b = 2  
> objects()  
[1] "a" "b"
```

- ❖ Objetos que começam por “a”

```
> ls(pattern="^a")  
[1] "a"  
> objects(pattern= "^a")  
[1] "a"
```



The screenshot shows the RStudio Environment pane. The 'Environment' tab is selected, showing the 'Global Environment'. Below the tab, there is a table of values:

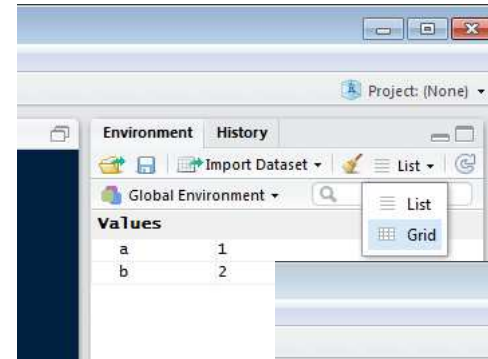
Values	
a	1
b	2



## II. Noções Básicas do R

- ❖ Remover um objeto específico, `rm()`

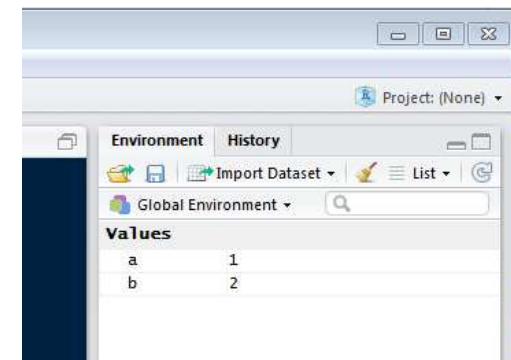
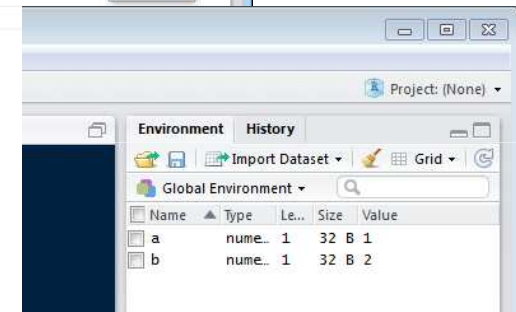
```
> rm(a)
> ls()
[1] "b"
```



- ❖ Remover todos os objetos

```
> rm(list=ls())
> ls()
character(0)
```

```
> rm(list=objects( ))
> ls()
character(0)
```

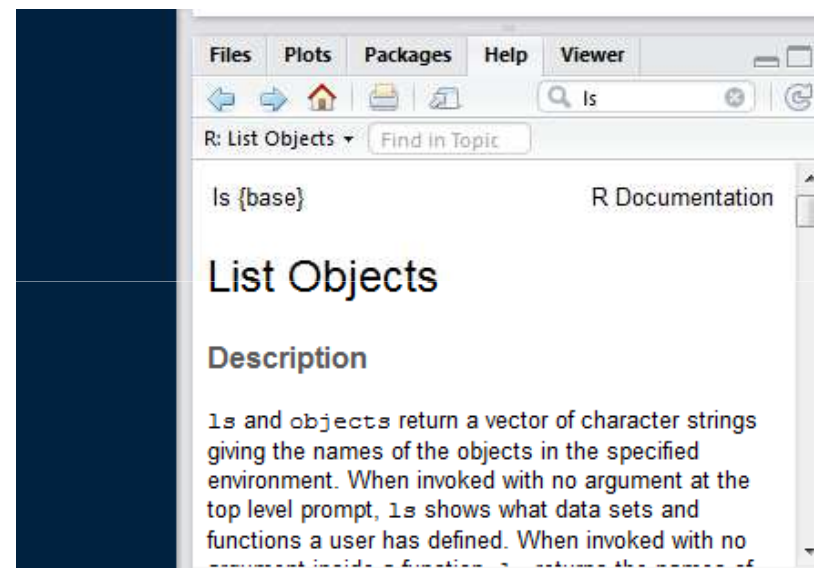


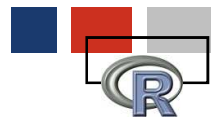
## II. Noções Básicas do R

- Opções de ajuda sobre comandos do R, `help()`, `?`, `example()`, F1

```
> help(rm)
> ?rm
```

```
> example(rm)
> tmp <- 1:4
> ## work with tmp and cleanup
> rm(tmp)
```



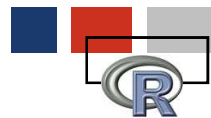


## II. Noções Básicas do R

- O R, como a maioria das linguagens que nasce em ambientes UNIX, é **case sensitive**, pelo que por as letras 'a' e 'A' podem corresponder a diferentes variáveis.
- O R **ignora espaços**, isto é, '8+3' e '8+ 3' dá origem exatamente ao mesmo resultado.
- Os **comandos** devem ser **separados** por ';' ou por uma **nova linha**.
- Podemos **agrupar comandos**, para serem executados em simultâneo, se estiverem entre chavetas '{ }'.
- O '#' é utilizado para **comentários**.
- Quando um **comando não** está **completo**, o R coloca o sinal de '+' na linha seguinte, permitindo que este seja completado.







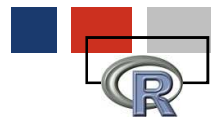
## II. Noções Básicas do R

- O R é uma **linguagem baseada em objetos**, ou seja, tudo o que usamos está guardado na memória do computador sob a forma de um objeto.
- Todos os objetos têm um nome associado. Este nome não pode começar por números ou nem conter ^, !, \$, @, +, -, /, \* e é case-sensitive.
- Um objeto pode armazenar diferentes conteúdos: números, texto, vetores, matrizes, expressões, chamadas a funções, etc.
- Para armazenamento num objeto usamos o operador de atribuição, '<-', '=' ou '->', e para visualizar o conteúdo do objeto basta digitar o nome do objeto.

```
> numero <- 3  
> numero  
[1] 3
```

```
> texto <- "teste"  
> texto  
[1] "teste"
```





## II. Noções Básicas do R

'+'	Soma
'-'	Subtração
'*'	Multiplicação
'%*%'	Multiplicação de matrizes
'/'	Divisão
'^'	Exponenciação
'%/%'	Divisão Inteira
'%%'	Resto da divisão inteira
'.'	Criação de sequências

```
> 8+3
[1] 11

> 8-3
[1] 5

> 8*3
[1] 24

> 8/3
[1] 2.666667

> 8^3
[1] 512

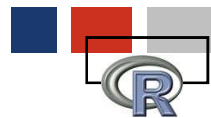
> 8%/%3
[1] 2

> 8%%3
[1] 2

> 8:3
[1] 8 7 6 5 4 3

> 3:8
[1] 3 4 5 6 7 8
```

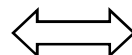




## II. Noções Básicas do R

- O **vetor** é considerado a estrutura de dados mais simples e consiste numa coleção organizada de elementos.
- A atribuição é feita a partir da função `c()`, cujos argumentos correspondem aos próprios elementos do vetor.

```
> x <- c(3.5,1.4,5,2.6,7,4.8)
> x
[1] 3.5 1.4 5.0 2.6 7.0 4.8
```



```
> c(3.5,1.4,5,2.6,7,4.8) -> x
> x
[1] 3.5 1.4 5.0 2.6 7.0 4.8
```

- A atribuição pode ser feita também por intermédio da função `assign( )` que é particularmente útil nas atribuições automáticas.

```
> assign("x",c(3.5,1.4,5,2.6,7,4.8))
> x
[1] 3.5 1.4 5.0 2.6 7.0 4.8
```





## II. Noções Básicas do R

- Os vetores podem ser usados para formação de novos vetores, seja por associação ou mesmo pelo uso de operações aritméticas.

```
> y <- c(x,1.5,x)
> y
[1] 3.5 1.4 5.0 2.6 7.0 4.8 1.5 3.5 1.4 5.0 2.6 7.0 4.8
```

- No caso das operações aritméticas, com vetores, a operação é efetuada elemento a elemento.

```
> x
[1] 3.5 1.4 5.0 2.6 7.0 4.8
> y <- 2*x + 5
> y
[1] 12.0 7.8 15.0 10.2 19.0 14.6
```

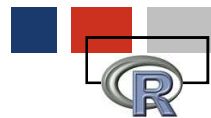


## II. Noções Básicas do R

- Nas operações aritméticas com vetores são também utilizados os habituais operadores: `+`, `-`, `*`, `/` e `^`.
- As funções aritméticas mais usuais são também aplicáveis a vetores: `exp()`, `log()`, `log10()`, `sin()`, `cos()`, `tan()`, `abs()`, `sqrt()`...
- Algumas das funções aritméticas correspondem a funções de **agregação**, pelo que quando aplicadas a vetores dão origem a um valor único: `min()`, `max()`, `length()`, `sum()`, `prod()`, `mean()`, `var()`...
- Outras funções especiais: `range()`, `cumsum()`, `cumprod()`...

```
> x
[1] 3.5 1.4 5.0 2.6 7.0 4.8
> log(x)
[1] 1.2527630 0.3364722 1.6094379 0.9555114 1.9459101 1.5686159
> max(x)
[1] 7
> range(x)
[1] 1.4 7.0
```

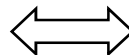




## II. Noções Básicas do R

- Para vetores de texto, podemos recorrer à mesma função `c()` desde que os elementos estejam “entre plicas” ou “entre aspas”.

```
> trim <- c("jan", "fev ", "mar")  
> trim  
[1] "jan" "fev" "mar"
```



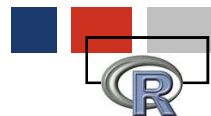
```
> trim <- c('jan','fev','mar')  
> trim  
[1] "jan" "fev" "mar"
```

### ❖ Concatenação de texto, `paste()`, `paste0()`

```
> paste(trim,2016)  
[1] "jan 2016" "fev 2016" "mar 2016"  
> paste(trim,"2016", sep="")  
[1] "jan2016" "fev2016" "mar2016"
```

```
> paste0(trim,2016)  
[1] "jan2016" "fev2016" "mar2016"
```





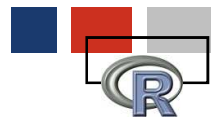
## II. Noções Básicas do R

- Os **vetores lógicos** são gerados por condições e podem assumir os valores *TRUE*, *FALSE* e *NA* (*not available*); nas condições podem ser usados diferentes operadores lógicos.

<code>==</code>	Igual	<code>is.na()</code>
<code>&lt;</code>	Menor	<code>is.nan()</code>
<code>&lt;=</code>	Menor ou igual	<code>is.character()</code>
<code>&gt;</code>	Maior	<code>is.factor()</code>
<code>&gt;=</code>	Maior ou igual	
<code>&amp;</code>	Intersecção (e)	
<code> </code>	Reunião (ou)	
<code>!</code>	Negação	

```
> x <- c (3.5,1.4,5,2.6,7,4.8)
> x
[1] 3.5 1.4 5.0 2.6 7.0 4.8
> x > 3
[1] TRUE FALSE TRUE FALSE TRUE TRUE
> sum(x > 3)
[1] 4
```





## II. Noções Básicas do R

- Nem sempre são conhecidos todos os elementos de um vetor e nesses casos o valor considerado, por defeito, é o *NA*.

❖ Identifica os NA's, `is.na()`

```
> a <- c(1,2.5,4,NA)
> a
[1] 1.0 2.5 4.0 NA
> a+2
[1] 3.0 4.5 6.0 NA
> is.na(a)
[1] FALSE FALSE FALSE TRUE
```

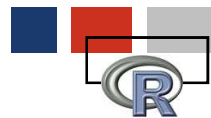
- Em determinadas situações os cálculos numéricos também dão origem a um outro tipo de “*missing values*”, que são os *NaN* (*Not a Number*).

❖ Identifica os NAN's, `is.nan()`

```
> b <- c(0/0, NA)
> b
[1] NaN NA
> is.na(b)
[1] TRUE TRUE
> is.nan(b)
[1] TRUE FALSE
```







## II. Noções Básicas do R

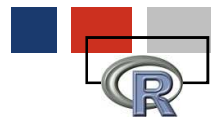
- ❖ Gera uma **sequência** numérica (por defeito, com incremento 1 ou -1)  
`[from]:[to]`

```
> 1:6  
[1] 1 2 3 4 5 6  
> 6:1  
[1] 6 5 4 3 2 1
```

- ❖ Cria uma **sequência**, `seq()`

```
> seq(from=0,to=1,by=0.25)  
[1] 0.00 0.25 0.50 0.75 1.00  
> seq(from=0,to=1,length.out=5)  
[1] 0.00 0.25 0.50 0.75 1.00  
> paste("Quartil", seq(from=0,to=1,by=0.25))  
[1] "Quartil 0"      "Quartil 0.25" "Quartil 0.5"  "Quartil 0.75" "Quartil 1"
```





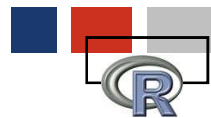
## II. Noções Básicas do R

- ❖ Cria uma **sequência** por repetição de dígitos ou texto, `rep()`

```
> rep(1, times=3)
[1] 1 1 1
> rep(1:3, times=3)
[1] 1 2 3 1 2 3 1 2 3
> rep(1:3, each=3)
[1] 1 1 1 2 2 2 3 3 3
> rep(1:3, 1:3)
[1] 1 2 2 3 3 3
```

```
> trim
[1] "jan" "fev" "mar"
> rep(trim, times=2)
[1] "jan" "fev" "mar" "jan" "fev" "mar"
> rep(trim, each=2)
[1] "jan" "jan" "fev" "fev" "mar" "mar"
> rep(trim, c(1,2,2))
[1] "jan" "fev" "fev" "mar" "mar"
```





## II. Noções Básicas do R

- Para além dos **vetores** existem outros tipos de objetos:
  - ❖ **Factor** – Formas compactas de armazenamento de dados categóricos;
  - ❖ **Lists** – São em geral vetores que podem ter muitos elementos e que não têm de ser todos do mesmo tipo;
  - ❖ **Matrix** – Generalização multi-dimensional de vetores (todos os vetores devem ser do mesmo tipo);
  - ❖ **Data frame** – Têm uma estrutura do tipo matriz mas em que as colunas podem ter diferentes formatos;
  - ❖ **Funções** – São objetos do R que podem ser guardados no ambiente de trabalho e que permitem a reutilização de procedimentos.





## II. Noções Básicas do R

- Por vezes um vetor por ser usado para especificar uma classificação discreta de variável e nesses casos é aconselhado usar um fator.

- ❖ Cria um fator, `factor()`

```
> regioao <- c("norte","centro","algarve","lvt","centro","norte","alentejo","lvt","centro")
> regioao <- as.factor(regiao)
> str(regiao)
Factor w/ 5 levels "alentejo","algarve",...: 5 3 2 4 3 5 1 4 3
```

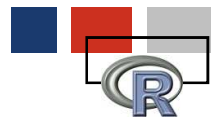
- ❖ Mostra ou alterar os níveis de um fator, `levels()`

```
> levels(regiao)
[1] "alentejo" "algarve" "centro" "lvt" "norte"
> levels(regiao)<-c("Alentejo","Algarve", "Centro","LVT","Norte","RAA","RAM" )
> regioao
[1] Norte Centro Algarve LVT Centro Norte Alentejo LVT Centro
Levels: Alentejo Algarve Centro LVT Norte RAA RAM
```

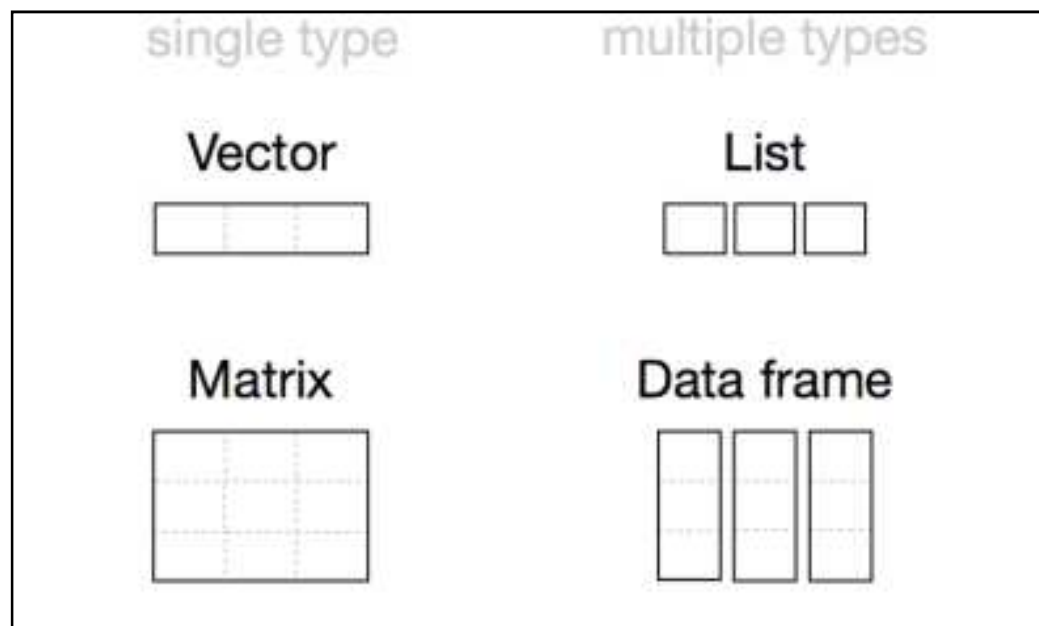
- ❖ Mostra uma tabela de frequências para um fator, `table()`

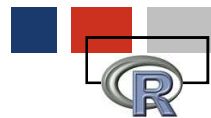
```
> table(regiao)
regiao
Alentejo Algarve Centro LVT Norte RAA RAM
      1      1      3      2      2      0      0
```





## II. Noções Básicas do R





## II. Noções Básicas do R

- O **data frame** é muito semelhante a uma matriz mas, em geral, as colunas têm nomes e podem conter dados de diferentes tipos. Pode ser visto como uma tabela de uma base de dados, em que cada linha corresponde a um registo e cada coluna corresponde a uma variável.

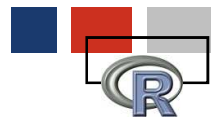
❖ Cria um *data frame*, `data.frame()`

```
> temp <- data.frame(Dias=c("terca","quarta","quinta","sexta"), Dia_Temp=c(14,15,20,12))
> temp
Dias Dia_Temp
1  terca      14
2 quarta     15
3 quinta     20
4  sexta     12

> names(temp) OU colnames(temp)
[1] "Dias"      "Dia_Temp"

> rownames(temp)
[1] "1" "2" "3" "4"
```





## II. Noções Básicas do R

### ❖ Explorar um *data frame* (indexação)

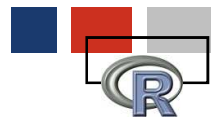
```
> temp
  Dias Dia_Temp
1  terca      14
2  quarta     15
3  quinta     20
4  sexta      12
> temp[3,2]
[1] 20
> temp$Dias[4]
[1] sexta
> temp[2:3,"Dia_Temp"]
[1] 15 20
> temp$Dia_Temp[2:3]
[1] 15 20
> temp[temp$Dia_Temp<15,1]
[1] terca sexta

> temp[temp$Dia_Temp<15,]
  Dias Dia_Temp
1  terca      14
4  sexta      12
```

**df[** *linhas* **,** *colunas* **]**

**df\$coluna[** *linhas* **]**





## II. Noções Básicas do R

Como atualizar a temperatura de quinta para 19 graus?

```
> temp
  Dias Dia_Temp
1  terca      14
2  quarta     15
3  quinta     20
4  sexta      12
```

```
> temp[3,2]<-19
```

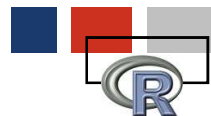
```
> temp[temp$Dias=="quinta", "Dia_Temp"]<-19
```

```
> temp$Dia_Temp[temp$Dias=="quinta"]<-19
```

```
> temp$Dia_Temp[3]<-19
```







## II. Noções Básicas do R

- Existem funções que se utilizam especificamente para ordenação de vetores ou fatores, quer seja por ordem crescente ou decrescente.
  - ❖ Ordena os índices do objeto segundo um critério, `order()`

```
> order(temp$Dia_Temp)
[1] 4 1 2 3

> temp[order(temp$Dia_Temp),]
  Dias Dia_Temp
4 sexta      12
1  terça      14
2  quarta     15
3  quinta     20

> temp[order(-temp$Dia_Temp),]
  Dias Dia_Temp
3  quinta     20
2  quarta     15
1  terça      14
4  sexta      12

> temp<- temp [order(temp$Dia_Temp),]
```

Dias	Dia_Temp
1  terça	14
2  quarta	15
3  quinta	20
4  sexta	12





## II. Noções Básicas do R

- As funções são também objetos que ficam armazenados em R e que permitem guardar procedimentos que podem ser reutilizados.
- Estrutura básica de uma função:

❖ nome atribuído à função

❖ expressão que cria uma nova função

```
> nome.funcao <- function( argumento 1,. . . , argumento n)
```

❖ argumentos necessários à função

comando 1

❖ entre as chavetas "{}" são listados os comandos da função

. . .

comando n

❖ retorna o resultado

```
return(resultado)
```





## II. Noções Básicas do R

- Exemplo:

```
> teste <- function(a,b) {print(a+b)}
```

```
> teste(2,3)
```

```
[1] 5
```

```
teste <- function(a,b) {
```

```
    print(a+b)
```

```
    d <- (b-a)
```

```
    return(d)
```

```
}
```

```
> resultado<-teste(2,3)
```

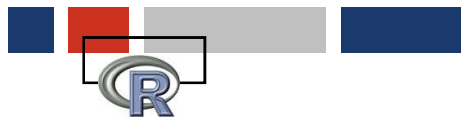
```
[1] 5
```

```
> resultado
```

```
[1] 1
```

❖ A função `return()` permite especificar um ou mais resultados (em lista) da função

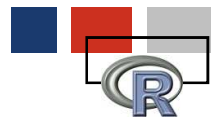




# Importação de dados

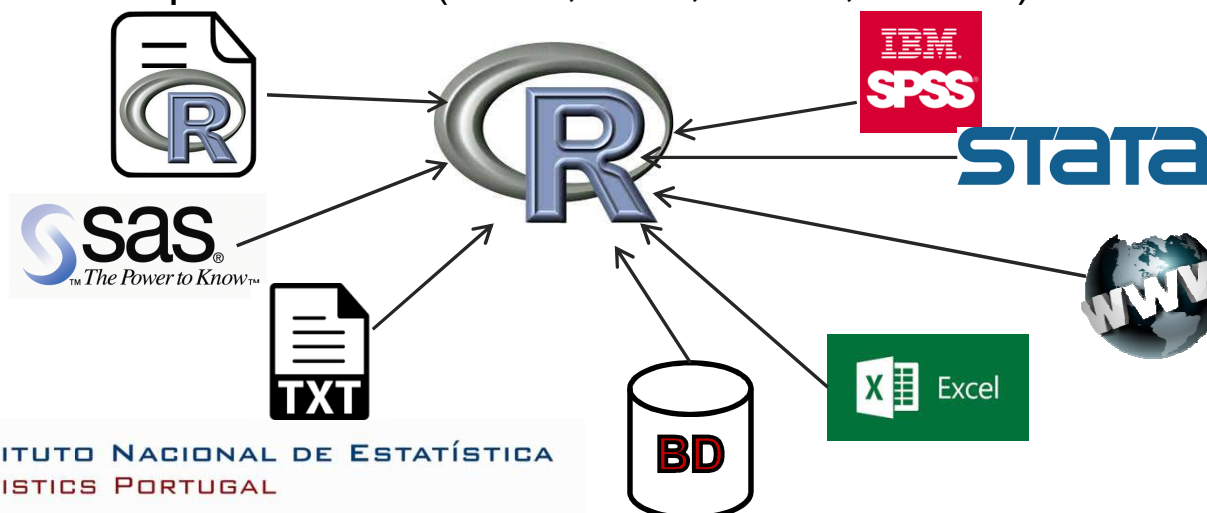


INSTITUTO NACIONAL DE ESTATÍSTICA  
STATISTICS PORTUGAL

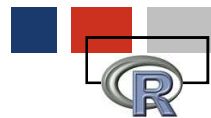


## II. Noções Básicas do R Importação

- O R dispõe de um conjunto de funções que permitem a importação ou exportação de informação para o ambiente de trabalho.
  - Importar funcionalidades adicionais incluídas em *packages* ou ficheiros texto.
  - Importar ou exportar dados provenientes de ficheiros externos das diferentes plataformas (Excel, SAS, SPSS, CSV...)



INSTITUTO NACIONAL DE ESTATÍSTICA  
STATISTICS PORTUGAL



## II. Noções Básicas do R

### Importação

- Para importar procedimentos em R ou dados para o ambiente de trabalho o R dispõe de algumas soluções, como por exemplo o comando, `source()`

```
> source("C:/Documents and Settings/.../teste.R")  
> teste(5,8)  
[1] 13  
> d  
[1] 3
```

A screenshot of a Notepad window titled "teste.R - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains the following R code:

```
teste <- function(a,b)  
{  
  print(a+b)  
  d <- (b-a)  
}
```

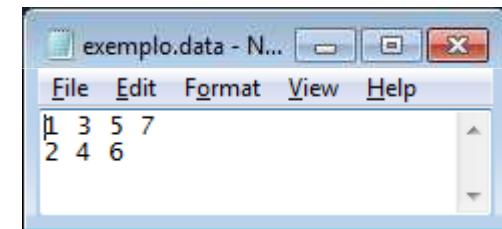




## II. Noções Básicas do R Importação

- Para atribuir os dados de um ficheiro a uma lista ou vetor pode ser utilizado o comando `scan()`.
- ❖ Carregar um vetor ou lista com dados provenientes de um ficheiro de texto.

```
> scan("exemplo.data")
Read 7 items
[1] 1 3 5 7 2 4 6
> scan("exemplo.data", skip = 1)
Read 3 items
[1] 2 4 6
```



- ❖ Carregar um vetor ou lista com dados introduzidos diretamente no teclado

```
> x <- scan()
1: 4
2: 5
3: 8
4: 1
5:
Read 4 items
> x
[1] 4 5 8 1
```

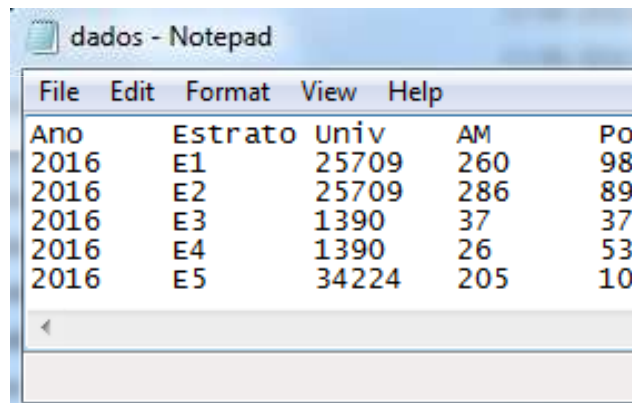




## II. Noções Básicas do R

### Importação

- Para carregar ficheiros de dados *TXT* ou *CSV* em formato de tabela existem funções mais específicas (dependendo do tipo de ficheiro).
  - ❖ Ler dados de ficheiro *TXT* para uma tabela, `read.table()`, `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()`

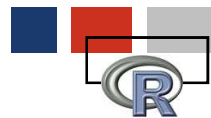


Ano	Estrato	Univ	AM	Po
2016	E1	25709	260	98
2016	E2	25709	286	89
2016	E3	1390	37	37
2016	E4	1390	26	53
2016	E5	34224	205	10

```
> read.table(file="C:/.../dados.txt", sep="\t", dec=",", header = TRUE)
      Ano Estrato Univ  AM Ponderador
1 2016      E1 25709 260  98.88077
2 2016      E2 25709 286  89.89161
3 2016      E3  1390  37  37.56757
4 2016      E4  1390  26  53.46154
5 2016      E5 34224 205 106.94634
> dados <-
read.table(file="C:/.../dados.txt", sep="\t", dec=",", header = TRUE)
> dim(dados)
[1] 5 5
```







## II. Noções Básicas do R

### Importação

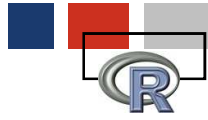
- A importação e exportação de dados em formato *Excel*, *SAS*, *SPSS*,... pode ser feita com recurso a funcionalidades adicionais de outros Packages
- ❖ Importar dados do *Excel*, recorrendo *package* “*rio*”, `import()`

```
> library(rio)
> dados <- import(file="dados.xlsx")
> dados
  Ano Estrato Univ AM Ponderador
1 2016 E 1 25709 260 98.88077
2 2016 E 2 25709 286 89.89161
3 2016 E 3 1390 37 37.56757
...
```

- ❖ Outro exemplo de importação de um ficheiro *SPSS* contendo variáveis com *value labels*. Neste caso será considerado um fator em R.

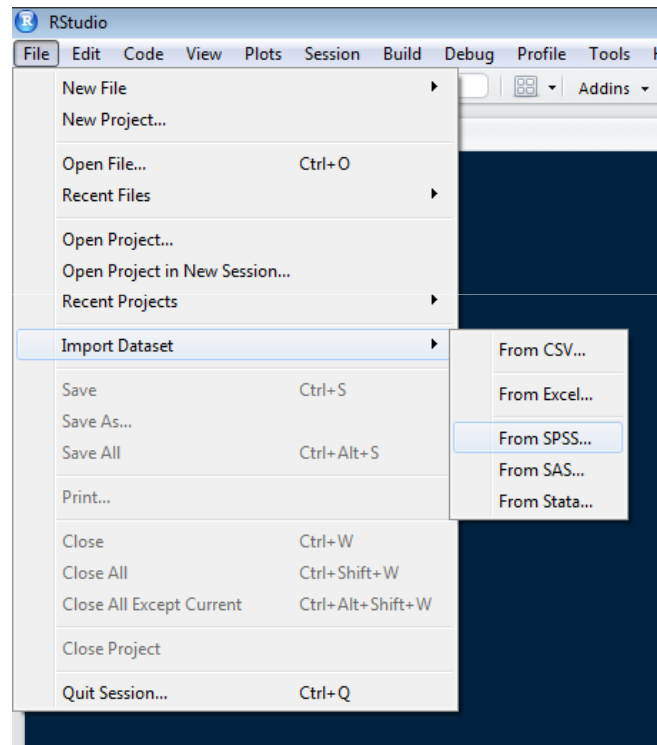
```
> library(rio)
> dados <- factorize(import(file="dados.sav"))
```

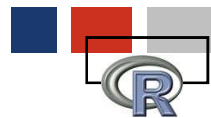




## II. Noções Básicas do R Importação

- Outra possibilidade de importação de dados é recorrendo aos menus do Rstudio.





## II. Noções Básicas do R

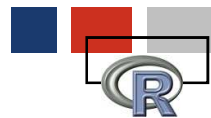
### Importação rodbc

- Muitas vezes a quantidade de informação justifica a necessidade de importar ou exportar informação em base de dados. O R dispõe de um conjunto de *packages* para ligação a sistemas de organização de bases de dados.

Package	Sistema BD
RPgSQL	PostgreSQL
RMySQL	MySQL
RmSQL	MiniSQL
RSQLite	SQLite
ROracle	Oracle

- O *package* **RODBC** é mais geral permitindo o acesso a diferentes tipos de ficheiros ou bases de dados, desde que seja criada uma ligação ODBC (*Open Data Base Connectivity*).





## II. Noções Básicas do R

### Importação rodbc

- Instalação do package RODBC

```
> install.packages("RODBC")  
> library("RODBC")
```

- Criação de uma ligação à base de dados.

```
> conexao<- odbcConnectExcel2007("c:\\User\\Ficheiro.xlsx"  
> conexao<- odbcConnectAccess2007 ("c:\\User\\Database.accdb"  
> conexao<- odbcConnect(dsn, uid = "pedro.sousa", pwd = "segredo")
```

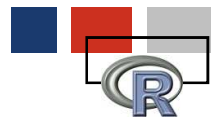
- A seleção de informação pode ser feita por comandos SQL tais como `select`, `from`, `where` , etc.

```
> dados <- sqlQuery (conexao, "select * from tabela")
```

- Selecionar todos os conteúdos de uma tabela `tbl`

```
> dados <- tbl(conexao, in_schema("SIGUA", "V_UNIV_ACT_01"))
```





## II. Noções Básicas do R

### Importação rodbc

- A criação de uma nova tabela é feita a partir do comando `sqlSave()`.

```
> sqlSave(conexao, dados, "TabelaDados", rownames = FALSE)
```

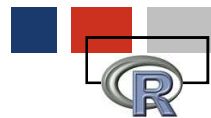
- Para apagar uma dada tabela em Access recorre-se ao comando `sqlDrop()`.

```
> sqlDrop(conexao, "TabelaDados")
```

- No final do acesso aos dados ou antes de estabelecermos uma nova ligação é conveniente fechar as conexões em aberto, `odbcClose()`, `odbcCloseAll()`

```
> odbcClose(conexao)
```





## II. Noções Básicas do R

### Importação ROracle

- Instalação do package ROracle

```
> install.packages("ROracle")  
> library("ROracle")
```

**NOTA:**

- Tem que ser instalado o software *OracleClient*
- É necessário adicionar duas variáveis de ambiente:  
OCI\_INC -> C:\oracle\11.2.0\x64\oci\include  
OCI\_LIB64 -> C:\oracle\11.2.0\x64\BIN

- Criação de uma ligação à base de dados.

```
> drv <- dbDriver("Oracle")  
> connect.string<-"(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=pt-lnx32.ine.pt)  
  (PORT=1521))(CONNECT_DATA=(SID=DW02)))"  
  
> Conexao <- dbConnect(drv,username="pedro.sousa",password="segredo",dbname=connect.string)
```



## II. Noções Básicas do R

### Importação ROracle

- Selecionar todos os conteúdos de uma tabela

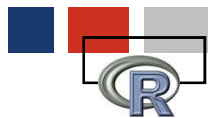
```
> dbReadTable(conexao,"tabela")
```

- A seleção de informação com comandos SQL tais como `select`, `from`, `where`, `update`, etc.

```
> dados <- dbGetQuery(conexao, "select * from tabela")
```

- Fechar as conexões ao servidor Oracle

```
> dbDisconnect(conexao)
```



## II. Noções Básicas do R

### Importação

### Bigdata ROracle

- Trabalhar grandes quantidades de informação. Criar uma ligação a uma BD/tabela sem carregar localmente todo o seu conteúdo.

```
> library(dplyr)
> library(dbplyr)
> AT_04 <- tbl(conexao, in_schema("scheme", "AT_04_2019"))
> AT_04
# Source:   table<AT_04_2019> [?? x 6]
# Database: OraConnection
```

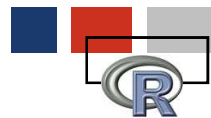
# 82178850

	NIF_EMITENTE	NIF_ADQUIRENTE	NIF_ADQUIRENTE_ENC	PAIS_ADQUIRENTE	ANO_MES_FACTURA~	VALOR_TRIBUTAVEL
	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
1	#####	NA	10100b5b7f525a176d11a1eca323e92d~	PORTUGAL	2019/04	19.7
2	#####	NA	0088e1100e6803247a743141fe1fe023~	PORTUGAL	2019/04	31.8
3	#####	NA	46855bdac89ad3dd9d2fa04edfa7a417~	PORTUGAL	2019/04	5.28
4	#####	NA	461019da3fd7259f0b6e20bdb27726cc~	PORTUGAL	2019/04	61.4
5	#####	NA	13bff3b6fd6e9e2ce9f6f3ddfb7fbcde~	PORTUGAL	2019/04	34.3
6	#####	NA	ad58c069dd8248e9a22275ededa8ee4a~	PORTUGAL	2019/04	19.6
7	#####	NA	0cac2c782ae193e7dc87379d4f70b527~	PORTUGAL	2019/04	4.24
8	#####	NA	00a1487c209ea79ef8e0f633960ee9e8~	PORTUGAL	2019/04	157.
9	#####	NA	6df79ab97e0b982a41e2fb9e3146a2f2~	PORTUGAL	2019/04	74.5
10	#####	NA	a3580d9669cbe49492a0347e18ae750a~	PORTUGAL	2019/04	8.13

# ... with more rows







## II. Noções Básicas do R

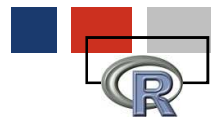
### Importação

Bigdata ROracle

- Fazer operações comuns em grandes bases de

```
> AT_04 %>% filter(PAIS_ADQUIRENTE=="PORTUGAL") %>%  
  group_by(ANO_MES_FACTURACAO) %>%  
  summarise(total = mean(VALOR_TRIBUTAVEL, na.rm=T))  
  
# Source:   lazy query [?? x 2]  
# Database: OraConnection  
  ANO_MES_FACTURACAO  media  
  <chr>              <dbl>  
1 2019/04             323299.  
2 2019/05             333500.  
  
AT_local <- AT_04 %>% select(PAIS_ADQUIRENTE, VALOR_TRIBUTAVEL %>% collect())
```





## II. Noções Básicas do R

### Importação

### WWW

- Obter dados diretamente de uma API (json)

```
> library(jsonlite)
> url <- "https://www.ine.pt/ine/json_indicador/pindica.jsp?op=2&varcd=0001071&lang=PT"
data1 <- fromJSON(url)
```

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

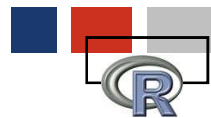
- Obter dados via *webscrapping* (url)

```
> library(rvest)
> url <- "http://ldap.ine.pt/query\_sem\_authent.php"
> pagina <- read_html(url)
> funcionarios<-as.data.frame(pagina %>% html_nodes("table") %>% .[[1]] %>% html_table(fill = T)) %>%
  select(X2,X3,X4) %>%
  rename (Nome=X2, UO=X3,Local=X4) %>%
  slice(18:n())
```

```
> funcionarios
```

	Nome	UO	Local
1	Adélia Veronica da Silva	DRGD :: DRGD/IE :: IE/NLRE	Évora
2	Adelina Maria Saraiva Rodrigues Andrade	DCN :: DCN/ICP	Lisboa-Dep. I
3	Adérito Jesus Alves	DI :: DI/NPAU	Porto
4	Agnelo da Silva Moreira	DRGD :: DRGD/DE :: DE/NDE1	Porto





## II. Noções Básicas do R

**Exercícios1.pdf**

