Introdução ao R

Estatística Numérica Computacional

Isabel Natário

2018/2019

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Matemática

- Um *software* estatístico de distribuição gratuita;
- Permite a análise estatística de dados, através de:
  - Utilização das bibliotecas que possui;
  - Programas feitos pelo utilizador.

1. Pode ser obtido em:

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

   selecionando um *CRAN mirror*, escolhendo a versão associada ao sistema operativo adequado (Windows, e.g.) e seguindo as instruções dadas;

2. Descarregar o ficheiro executável (e.g.) R-3.5.1-win.exe;

3. Executar esse ficheiro, o que permitirá a instalação do sistema base e dos *packages* (pacotes, bibliotecas) recomendados.

Os manuais sobre o **R**, incluídos em todas as instalações, são:

- *An introduction to R* (leitura obrigatória);
- *Writing R extensions*;
- *R data import/export*;
- *The R language definition*;
- *R installation and administration*.

- Rstudio: ambiente de desenvolvimento para usar o R de forma mais eficiente, programa de distribuição gratuita
- Pode ser obtido em: http://www.rstudio.com/
- Descarrega-se o Rstudio, versão *desktop*, por exemplo
- Escolher um ficheiro executável que seja adequado ao nosso sistema operativo e corre-lo

Num ambiente de Windows,

1. Criar uma pasta onde quer guardar os seus trabalhos - e.g., C:/ENC_R;
2. Com o botão direito do rato carregue no atalho do programa R e escolha "propriedades"; No espaço reservado ao "start in" altere o caminho para o local onde criou a sua pasta - e.g., C:/ENC_R;
3. Duplo clique sobre o ícone do **R**, carregando o programa;
4. Aguardar o prompt ">".

Notas: Aconselha-se a criação e uso de um ficheiro *script* do R onde se vão escrevendo as instruções a serem dadas. Fica-se assim com um registo do que se faz, bastando depois fazer CTRL+R para a instrução correr no R. Adicionalmente, aconselha-se a instalação do editor de texto Tinn-r, para melhor facilitar o trabalho da edição das instruções do referido ficheiro.

Num ambiente de Windows,

1. Criar uma pasta onde quer guardar os seus trabalhos - e.g., C:/ENC_R;
2. Carregar o programa Rstudio;
3. Aguardar o prompt ">".
4. No separador "Session" selecione a opção "Set Working Directory", aí selecione "Choose Directory" e finalmente escolha a diretoria que criou anteriormente

Notas: Aconselha-se a criação e uso de um ficheiro *script* do R onde se vão escrevendo as instruções a serem dadas. Fica-se assim com um registo do que se faz, bastando depois fazer CTRL+R para a instrução correr no R ou carregar na tecla de "Run".

- Expressões Por exemplo, queremos calcular 6+48

  $> 6 + 48$

- Atribuições Atribuímos o valor a um escalar através da sintaxe

  $>$ *escalar* $<-$ *expressao*

  Por exemplo,

  $> x <- 6 + 48$

- Listar todos os objetos criados

  $>$ *ls*( )

Nota: O **R** faz a distinção entre maiúsculas e minúsculas. Por exemplo, X e x são objetos diferentes.

Soma	$+$
Diferença	-
Multiplicação	*
Divisão	$/$
Potência	$\widehat{}$
Raiz quadrada de x	$sqrt\,(x)$
Módulo	%%
Logaritmos	log, $log10$, $log2$, $logb\,(x, base)$
Exponencial	exp

Trignométricas	*sin*, *cos*, *tan*
Arredondamento de x com n casas decimais	*round* (x, n)
Outras (vetores)	*max*, *min*, *range*, *mean*, *sum*, *var*, *sd*, *prod*, *sort*, *order*, etc.

Nota: Sempre que tiver dúvida sobre uma qualquer *função* pode pedir ajuda no R através do comando help(*função*). A função apropos(*conceito*) informa-o sobre todas as funções que o R tem que envolvam *conceito*. A função demo( ) mostra-lhe alguns exemplos.

Observação: Para utilizar o R num ambiente de janelas em vez da linha de comandos utilize *package* **Rcmdr**.

- Vetor (coleção ordenada de elementos do mesmo tipo);
- Array (generalização multidimensional de vetor, com elementos do mesmo tipo);
- Data frame (como o array, mas com colunas de diferentes tipos);
- Factor (tipo de vector para dados categóricos);
- Lista

Nota: A função *mode*(*objeto*) informa ou atribui o tipo de *objeto*.

Criação de vetores e seu manuseamento

- Criamos um vetor através da função *c*( )

```
> x <- c(2, 4, 9)
> x
[1] 2 4 9
> cores <- c("vermelho", "verde", "verde", "preto")
> cores[4]
[1] "preto"
```

- Podemos ler um vetor de um ficheiro exterior, "dados.txt", usando a função *read.table*("*dados.txt*"). Ver esta função mais à frente
- De diversos modos podemos extrair elementos de um vetor

```
> dados <- c(2.4, 2.6, 1.9, 2.0, 2.0, 2.4, 2.8, 2.6, 2.3, 1.8)
> dados
[1] 2.4 2.6 1.9 2.0 2.0 2.4 2.8 2.6 2.3 1.8
> dados[3 : 6]
[1] 1.9 2.0 2.0 2.4
> dados[dados < 2]
[1] 1.9 1.8
```

Criação de vetores com sequências e repetições

- Podemos omitir elementos de um vetor

```
> dados[-c(3:6)]
[1] 2.8 2.6 2.3 1.8
```

- As funções *seq* e *rep* são úteis na criação de vetores

```
> seq1 <- 3:8
> seq1
[1] 3 4 5 6 7 8
```

```
> seq2 <- -2:1
> seq2
[1] -2 -1 0 1
```

```
> seq3 <- seq(1, 3, 0.5)
> seq3
[1] 1 1.5 2 2.5 3
```

```
> rep1 <- rep(20, 6)
> rep1
[1] 20 20 20 20 20 20
```

```
> rep2 <- rep(seq(4), 2)
> rep2
[1] 1 2 3 4 1 2 3 4
```

```
> seq4 <- seq(from = -1, to = 7, by = 2)
> seq4
[1] -1 1 3 5 7
```

Operações sobre vetores

Exercício

Considere os vetores $x = (1, 2, 3, 4)$, $y = (0, 1, 2, 3)$ e $z = (3, 4)$.
Determine

$$x + y \quad y - z \quad y/x \quad \sqrt{y} \quad e^y \quad \ln x - \cos y$$

Outro Exercício

Considere o vetor dos dados
$(205, 377, 292, 300, 179, 240, 300, 190, 680, 250,$
$180, 170, 211, 266, 303, 350, 375, 288, 360, 225)$. Ordene-o, de modo não decrescente (função *sort*) e determine as ordens dos seus elementos (função *order*).
Faça uso das funções seguintes para o ajudar a responder ao exercício:
summary, *mean*, *var*, *median*, *hist*, *boxplot*

Operações sobre vetores

Outro Exercício

```
> dados<- c(205, 377, 292, 300, 179, 240, 300, 190, 680, 250, 180, 170, 211, 266, 303, 350, 375, 288, 360, 225)

> summary(dados)

> hist(dados)  # Faz o histograma dos dados, usando a regra de Sturges para escolher o nº de classes.

> quebra <- c(min(dados),(max(dados)-min(dados))/2,max(dados))  # Novos pontos de corte

> hist(dados,breaks=quebra)  # Para forçar os extremos das classes

> dev.off() # Fecha o dispositivo gráfico.
```

Operações sobre vetores

Outro Exercício

> par(mfrow=c(i,j)) # Para dividir a área do dispositivo gráfico em i linhas e j colunas - podendo-se então desenhar $i \times j$ gráficos na mesma folha

```
> par(mfrow=c(1,2))
> hist(dados)
> hist(dados, breaks=quebra)
> dev.off()
```

> names(hist(dados)) # Porque *hist(dados)* é na verdade uma *data frame* (a ver à frente), tem a si associada uma série de características cuja função *names* permite ver

> hist(dados)$counts # Acede-se desta forma à característica *counts* do *hist(dados)*

Operações sobre vetores

Outro Exercício

```
> jpeg(file="histograma.jpeg")
> hist(dados, c( 170,255,340,425,510,595,680), xlab="n°
palavras", ylab="Frequencia", main=" ")
> fi <-  hist(dados,c(170,255,340,425,510,595,680), xlab="n°
palavras", ylab="Frequencia", main=" ")$counts
> lines(c(212.5,297.5,382.5,467.5,552.5,637.5), fi, col=2)
> dev.off()


> jpeg(file="caixadebigodes.jpeg")
> boxplot(dados, range=0, horizontal=TRUE)
> dev.off()
```

Criação de matrizes e seu manuseamento

- Podemos criar uma matriz através da função *matrix*

$> M <- matrix(seq(1,9,1), ncol = 3) \quad \#$ Ou

$M <- matrix(seq(1,9,1), nrow = 3)$

$> M$

	[,1]	[,2]	[,3]
[,1]	1	4	7
[,2]	2	5	8
[,3]	3	6	9

Repare que os elementos de $seq(1,9,1) = (1,2,3,4,5,6,7,8,9)$ são colocados na matriz por linha. Se quisermos que seja por coluna, basta transpor a matriz:

$> M <- t(M)$

Criação de matrizes e seu manuseamento

- Podemos também criar matrizes através das funções *cbind* e *rbind*.

  $> M1 <- cbind(c(1,2,3), c(4,5,6,), c(7,8,9,))$

  $> M2 <- rbind(c(1,4,7), c(2,5,8,), c(3,6,9,))$

- De diversos modos podemos extrair dados de uma matriz.

  $> M1[2,3]$

  [1] 8

  $> M1[2,]$

  [1] 2 5 8

  $> M1[1:2,3]$

  [1] 7 8

Criação de matrizes e seu manuseamento

- Podemos omitir elementos de uma matriz

```
> M1[-1,]
     [,1] [,2] [,3]
[,1]    2    5    8
[,2]    3    6    9
> M1[-1,-1]
     [,1] [,2]
[,1]    5    8
[,2]    6    9
```

Construção de data frames e seu manuseamento

Um data frame é uma base de dados. Pode ser vista como uma matriz, com colunas de modos e atributos eventualmente diferentes. Cada coluna contém a informação sobre uma variável. Pode-se dispor na forma matricial, sendo as suas linhas e colunas acedidas pelas usuais convenções de índices.

Exemplo - construção de um data frame:

$> x1 <- 1:10$

$> x2 <- 11:20$

$> x3 <- letters[1:10]$

$> d1 <- data.frame(x1, x2, x3)$

$> d1$

$> attributes(d1)$

Construção de data frames e seu manuseamento

Os data frames são uma boa forma de introduzir dados que se encontram num ficheiro exterior, digamos "dados.txt", para dentro do R, através da função read.table( ):

$> dados <- read.table("dados.txt", header = TRUE)$ # A opção *header=TRUE* usa-se quando as colunas dos dados no ficheiro se encontram encabeçadas pelos nomes das quantidades que representam

Nota 1: O ficheiro "dados.txt"tem de estar na nossa pasta de trabalho. Se não estiver, temos de indicar o caminho da sua localização, e.g. "*c : /Dados/dados.txt*"

Nota 2: Usando o R-studio os dados podem ser importados de forma mais automática usando a opção "*Import Dataset*" na divisória do 1º quadrante do programa

$> dados$

$> dim(dados)$

- Podemos atribuir nomes às colunas (por defeito as linhas estão identificadas pelos seus números):

$> names(dados) <- c("Nome1","Nome2",...)$ # Tantos nomes como colunas

Construção de data frames e seu manuseamento

- Consultemos a 1ª linha e a 2ª coluna do data frame *dados* (supondo que as tem):

  $> dados[1, \ ]$

  $> dados[\ , "Nome2"]$

- Alternativamente, se quisermos trabalhar apenas com a 2º coluna:

  $> dados\$Nome2$

- Como é muito importante trabalharmos com as colunas de um data frame, a função *attach* permite que se faça referência às colunas de um modo mais cómodo:

  $attach(dados)$

- Para acedermos à 2º coluna basta agora chamar:

  $> Nome2$

Nota: A função *detach* liberta a atribuição anterior. O efeito da função attach deixa de se fazer sentir quando saímos do programa R.

Construção de data frames e seu manuseamento

- Adicionemos uma nova coluna (variável), *Nome.x*, ao data frame (Nome.x tem de ser um vetor chamado Nome.x com um número de elementos igual ao número de linhas do data frame):

  $> dados <- data.frame(dados, Nome.x)$

- Ou, mais facilmente:

  $> dados\$Nome.x <- Nome.x$

  $> attach(dados)$

Exemplo de construção do *data frame*, empresas.

Em determinado país estão registadas na câmara do comércio apenas 40 empresas de importação de flores. Neste registo constam a identificação das firmas pelo seu nome, o número de sócios, o capital social (em milhares de euros), o volume médio mensal de negócios do ano transato (em milhares de euros) e o número de empregados. Estas informações encontram-se, em colunas e pela ordem atrás indicada, no ficheiro empresas.txt. Copie este ficheiro para a sua pasta.

(Dados inventados. Qualquer semelhança com a realidade é pura coincidência.)

- Comecemos por ler os dados do ficheiro:

  $> empresas <- read.table("empresas.txt")$

  $> empresas$

  $> attributes(empresas)$

  $> dim(empresas)$

- Atribuímos nomes às colunas (por defeito as linhas estão identificadas pelos números de 1 a 40):

  $> names(empresas) <- c("Nome","N.Socios","C.Social","VMM","N.Empregados")$

Data frame empresas

- Consultemos a 1ª linha e a 3ª coluna:

  $> empresas[1, \ ]$

  $> empresas[\ , "C.Social"]$

- Alternativamente, se quisermos trabalhar apenas com a 3º coluna:

  $> empresas\$C.Social$

- Como é muito importante trabalharmos com as colunas de um data frame, a função *attach* permite que se faça referência às colunas de um modo mais cómodo:

  $attach(empresas)$

- Para acedermos a 3º coluna basta agora chamar:

  $> C.Social$

Nota: A função *detach* liberta a atribuição anterior

Data frame empresas

- Adicionemos uma nova coluna (variável) ao data frame. Suponhamos que temos informação adicional sobre a antiguidade de cada firma e sobre se a empresa se classifica de pequena (1), média (2) ou grande (3):

  $> Antig <- c(10, 3, 9, 1, 1, 4, 5, 4, 4, 6, 10, 2, 8, 3, 2, 6, 7, 6, 3, 6, 3, 6, 10, 5, 1, 9, 4, 5, 7, 7, 9, 5, 1, 10, 5, 3, 8, 2, 5, 8)$

  $> Tamanho <- c(1, 2, 2, 1, 2, 1, 2, 2, 3, 2, 1, 1, 1, 2, 1, 2, 2, 1, 3, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1)$

  $> empresas <- data.frame(empresas, Antig, Tamanho)$

- Ou, mais facilmente:

  $> empresas\$Antig <- Antig$

  $> empresas\$Tamanho <- Tamanho$

  $> attach(empresas)$

  Nota: Num data frame, os vetores de caracteres são imediatamente interpretados como *fatores*.

Data frame empresas

- Apresentam-se agora algumas funções estatísticas bem como alguns comandos para a construção de gráficos estatísticos.

Breve informação estatística

```
> summary(empresas)

> summary(empresas$Antig)   # Ou simplesmente
> summary(Antig)
```

✓ Média e desvio padrão de *C.Social*:

```
> mean(empresas$C.Social)   # Ou simplesmente
> mean(C.Social)

> sd(C.Social)
```

✓ Média e desvio padrão de *VMM* por *N.Empregados*:

```
> tapply(VMM, N.Empregados, mean)

> tapply(VMM, N.Empregados, sd)
```

Data frame empresas

Alguns gráficos

✓ Gráfico de barras do N.Empregados

$> barplot(N.Empregados)$

✓ Gráfico de barras do conjunto de médias e de desvios padrão de VMM por N.Empregados

$> barplot(tapply(VMM, N.Empregados, mean))$

$> barplot(tapply(VMM, N.Empregados, sd))$

✓ Juntar os gráficos anteriores no mesmo écran

$> par(mfrow = c(1, 2))$

$> barplot(tapply(VMM, N.Empregados, mean))$

$> barplot(tapply(VMM, N.Empregados, sd))$

Data frame empresas

- ✓ Podemos adicionar legendas

  $> barplot(tapply(VMM, N.Empregados, mean), main =$
  "Medias de VMM por N° empregado", $xlab =$
  "N° empregado", $ylim =$ "Medias")

  $> barplot(tapply(VMM, N.Empregados, sd), main =$
  "Desvios padrao de VMM por N° empregado", $xlab =$
  "N° empregado", $ylim =$ "Desvios padrao")

- ✓ Fechar o dispositivo gráfico

  $> dev.off( )$

- ✓ Contagem e representação gráfica do número de sócios

  $> table(N.Socios)$

  $> pie(table(N.Socios), main =$ "N° de socios")

  $> barplot(table(N.Socios), main =$ "N° de socios")

Data frame empresas

- ✓ Caixas de bigodes

$> par(mfrow = c(1, 2))$

$> boxplot(VMM)$

$> boxplot(VMM \sim N.Empregados)$

$> dev.off()$

$> boxplot(N.Socios, N.Empregados, col = c("red", "blue"), names = c("N° Socios", "N° Empregados"))$

- ✓ Histogramas

$> hist(VMM, main = "Volume médio mensal de negocios", xlab = "VMM", ylab = "Frequencia absoluta")$

$> par(mfrow = c(1, 3))$

$> tapply(VMM, Tamanho, hist)$

Data frame empresas

- ✓ Gráficos de dispersão

$> plot(VMM, N.Empregados)$

$> plot(VMM, N.Empregados, col = "red")$

$> plot(empresas)$

$>$
$plot(data.frame(N.Socios, VMM, C.Social, N.Empregados), col = "blue")$

Data frame empresas

Output

✓ Escrever informação para fora

$> write(t(cbind(as.character(Nome)," ", N.Socios," ", VMM," ", C.Social," ", N.Empregados)), "RESempresas.txt", ncol = 9)$ # Escreve para o ficheiro RESempresas.txt os nomes das empresas, o seu nº de sócios, o seu capital social e o número de empregados

✓ Gravar gráficos em ficheiro .ps ou .eps

$> postscript(file = "Plotempresas.ps", horizontal = FALSE)$
$> plot(data.frame(N.Socios, VMM, C.Social, N.Empregados), col = "blue")$
$> dev.off()$

✓ Gravar gráficos em ficheiro .pdf

$> pdf(file = "Plotempresas.pdf", horizontal = FALSE)$
$> plot(data.frame(N.Socios, VMM, C.Social, N.Empregados), col = "blue")$
$> dev.off()$

Uma lista é um vetor generalizado em que cada uma das suas componentes pode ser de um tipo e de uma dimensão distinta.

Criação de listas

$> uma.lista <- list(um.vetor = 1:10, uma.palavra = "Ola", uma.matriz = matrix(1:15, ncol = 3), outra.lista = list(a = "flor", b = rep(2, 8)))$

$> uma.lista$

$> uma.lista\$uma.palavra$

$> uma.lista[1]$

$> attributes(uma.lista)$

$> mode(uma.lista)$

Devem-se construir funções quando se pretendem executar as mesmas operações repetidas vezes.

Exemplo de construção da função desconto:

```
> desconto <- function(preco, percentagem){
list(novo.preco = (1 - percentagem/100) * preco, desconto =
(percentagem/100) * preco)
}

> desconto(1000, 20)

> desconto <- function(preco, percentagem = 60){        #Por defeito
usa-se 60%
list(novo.preco = (1 - percentagem/100) * preco, desconto =
(percentagem/100) * preco)
}
```

Função para calcular intervalos de confiança para a média populacional:

```
ICmedia<- function(dados,normal=TRUE,sigma=0,niv.conf=95){
    xbar<- mean(dados)
    n<- dim(as.array(dados))
    if(!sigma) { #Se o sigma é desconhecido
        sigma<- sd(dados)
        if (normal && n < 30){ #Se a populacao é normal e n < 30
            t<- qt(1-(1-niv.conf/100)/2,n-1)
            return(list(IC=c(xbar-t*sigma/sqrt(n),xbar+t*sigma/sqrt(n)))) }
        if (n >= 30) { #Se n ≥ 30, com pop. normal ou não
            z <- qnorm(1-(1-niv.conf/100)/2,0,1)
            return(list(IC=c(xbar-z*sigma/sqrt(n),xbar+z*sigma/sqrt(n)))) }
        else return("Nao se pode determinar IC") }
    else{ #Sigma conhecido
        if (normal) { #Pop. normal
            z<- qnorm(1-(1-niv.conf/100)/2,0,1)
            return(list(IC=c(xbar-z*sigma/sqrt(n),xbar+z*sigma/sqrt(n)))) }
        else {  #Pop. não normal
            if (n >= 30) { #n ≥ 30
                z<- qnorm(1-(1-niv.conf/100)/2,0,1)
                return(list(IC=c(xbar-z*sigma/sqrt(n),xbar+z*sigma/sqrt(n))))}
            else return("Nao se pode determinar IC") }}}
```

Função para calcular intervalos de confiança para a média populacional - 2ª versão:

```
ICmedia1<- function(xbar,dados,normal=TRUE,sigma=0,niv.conf=95,n,s=0) {
    if(sigma==0 && s==0) return("Não se pode determinar o IC")
    if(!sigma) { #Se o sigma é desconhecido
        sigma<- sd(dados)
        if (normal && n < 30){ #Se a populacao é normal e n < 30
            t<- qt(1-(1-niv.conf/100)/2,n-1)
            return(list(IC=c(xbar-t*sigma/sqrt(n),xbar+t*sigma/sqrt(n)))) }
        if (n >= 30) { #Se n ≥ 30, com pop. normal ou não
            z <- qnorm(1-(1-niv.conf/100)/2,0,1)
            return(list(IC=c(xbar-z*sigma/sqrt(n),xbar+z*sigma/sqrt(n)))) }
        else return("Nao se pode determinar IC") }
    else{ #Sigma conhecido
        if (normal) { #Pop. normal
            z<- qnorm(1-(1-niv.conf/100)/2,0,1)
            return(list(IC=c(xbar-z*sigma/sqrt(n),xbar+z*sigma/sqrt(n)))) }
         else { #Pop. não normal
            if (n >= 30) { #n ≥ 30
                z<- qnorm(1-(1-niv.conf/100)/2,0,1)
                return(list(IC=c(xbar-z*sigma/sqrt(n),xbar+z*sigma/sqrt(n)))) }
            else return("Nao se pode determinar IC") } } }
```

Nota: Consulte também os ficheiros "ICmedia.r" e "Ficheiro de ajuda para ICmedia.doc" para mais informações.

Testes de hipóteses:

Uma das muitas funções já programadas do R é a função **t.test**, para efetuar testes de hipótese para a média de uma população.
Mais concretamente, **testamos** $H_0 : \mu = \mu_0$ contra uma hipótese alternativa que pode ser bilateral, unilateral direita ou esquerda.
Assume-se a normalidade da população inerente e que o desvio padrão da mesma é desconhecido:

$> t.test(x, alternative = c("two.sided","less","greater"), mu = 0, conf.level = 0.95)$

Notas:

- ✓ x é a amostra;
- ✓ Por defeito a hipótese alternativa é bilateral e o $\mu_0 = 0$.
- ✓ Esta função devolve ainda um intervalo de confiança para a média, com um nível de confiança a 95% por defeito.

Observação: No entanto todos os testes são programáveis no R!

A regressão linear simples faz-se no R através da função lm( ):

```
> x <- c(20, 40, 61, 85, 68)
> y <- c(1, 1.5, 3, 4.2, 3.3)
> reg1 <- lm(y ~ x)
> names(reg1)
> reg1$coefficients   # Dá os parâmetros de regressão estimados
> reg1$residuals      # Dá os resíduos da regressão
> reg1$fitted         # Dá os valores estimados de regressão, Ŷ_i
> plot(x, y)          # Faz gráfico dos pontos amostrais
> abline(reg1)        # Junta ao gráfico anterior a reta de regressão ajustada
> plot(x, reg1$residuals)   # Faz gráfico dos resíduos
```