Introdução ao R

Probabilidades e Estatística C

2005/2006

1 O que é o R?

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

2 Instalação

1. Pode ser obtido, por exemplo, a partir do endereço:

   http://neacm.fe.up.pt/CRAN/

   seguindo os links

   R-Binaries⟶Windows⟶base

2. Descarregar o ficheiro executável, por exemplo,

   http://neacm.fe.up.pt/CRAN/bin/windows/base/R-2.2.0-win32.exe

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

3 Documentação sobre o R

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.*

4 Documentação sobre o R

Documentação geral

- **R** para principiantes, de *E. Paradis*, em

  http://neacm.fe.up.pt/CRAN/doc/contrib/rdebuts_en.pdf

- FAQ;
- Manuais online sobre tratamento estatístico.

5 Início de sessão no R

Num ambiente de Windows,

1. Criar uma pasta onde quer guardar os seus trabalhos - e.g., C:/PED_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:/PED_R;
3. Duplo clique sobre o ícone do **R**, carregando o programa;
4. Aguardar o prompt ">".

Nota: Aconselha-se o uso de um ficheiro de texto (Wordpad, e.g.) onde se vão escrevendo as instruções a serem dadas no R. Fica-se assim com um registo do que se faz, bastando copiar as instruções para o R.

6 Comandos elementares

- Sair do programa

  $> q(\ )$

- 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 objectos criados

  $> ls(\ )$

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

7 Principais operações aritméticas

Soma	$+$
Diferença	-
Multiplicação	$*$
Divisão	$/$
Potência	$\hat{}$
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 (vectores)	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.

8 Alguns tipos de objectos

- Vector (colecção ordenada de elementos do mesmo tipo);
- Array (generalização multidimensional de vector, 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*(*objecto*) informa ou atribui o tipo de *objecto*.

8.1 Vectores

8.1.1 Criação de vectores e seu manuseamento

- Criamos um vector através da função $c(\ )$.

$$
\begin{array}{ll}
> & x <- c(2,4,9) \\
> & x \\
[1] & \quad 2 \quad 4 \quad 9 \\
> & cores <- c(\text{"}vermelho\text{"},\text{"}verde\text{"},\text{"}verde\text{"},\text{"}preto\text{"}) \\
> & cores[4] \\
[1] & \text{"}preto\text{"}
\end{array}
$$

- Podemos ler um vector de um ficheiro exterior, "dados.txt", usando a função $read.table(\text{"}dados.txt\text{"})$. Ver esta função mais à frente.

- De diversos modos podemos extrair elementos de um vector.

$$
\begin{array}{ll}
> & dados <- c\,(2.4, 2.6, 1.9, 2.0, 2.0, 2.4, 2.8, 2.6, 2.3, 1.8) \\
> & dados \\
[1] & \quad 2.4 \quad 2.6 \quad 1.9 \quad 2.0 \quad 2.0 \quad 2.4 \quad 2.8 \quad 2.6 \quad 2.3 \quad 1.8 \\
> & dados\,[3:6] \\
[1] & \quad 1.9 \quad 2.0 \quad 2.0 \quad 2.4 \\
> & dados\,[dados < 2] \\
[1] & \quad 1.9 \quad 1.8
\end{array}
$$

- Podemos omitir elementos de um vector.

$$
\begin{array}{ll}
> & dados\,[-c\,(3:6)] \\
[1] & \quad 2.8 \quad 2.6 \quad 2.3 \quad 1.8
\end{array}
$$

8.1.2 Criação de vectores com sequências e repetições

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

$> \quad seq1 <- 3:8$ $> \quad seq1$ $[1] \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8$	$> \quad rep1 <- \ rep(20,6)$ $> \quad rep1$ $[1] \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20$
$> \quad seq2 <- \ -2:1$ $> \quad seq2$ $[1] \quad -2 \quad -1 \quad 0 \quad 1$	$> \quad rep2 <- rep(seq(4),2)$ $> \quad rep2$ $[1] \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4$
$> \quad seq3 <- seq(1,3,0.5)$ $> \quad seq3$ $[1] \quad 1 \quad 1.5 \quad 2 \quad 2.5 \quad 3$	$> \quad seq4 <- seq(from=-1,to=7,by=2)$ $> \quad seq4$ $[1] \quad -1 \quad 1 \quad 3 \quad 5 \quad 7$

8.1.3 Operações sobre vectores

Exercício

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

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

Exercício 1.2 das folhas da cadeira

Considere o vector dos dados do exercício 1.2. 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*

Exercício 1.2

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

> summary(dados)

> L< −max(dados)-min(dados) # Amplitude dos dados

> k< −ceiling(1+log(20)/log(2)) # Calcula o menor inteiro não inferior a 1+log(20)/log(2)

> l < −L/k # Amplitude de cada classe

> for(i in 1:7) {

+ quebra[i]< −min(dados)+(i-1)*l } # Cálculo dos extremos de cada classe.

> quebra

Exercício 1.2

```
> Fi<- -rep(0,k)
> for(i in 1:k){
+ Fi[i]<- -sum(dados<=quebra[i+1])}   # Frequências absolutas acumuladas
> Fi
> fi<- -rep(0,k)
> fi[1]<-Fi[1]
> for(i in 2:k){
+ fi[i]<- -Fi[i]-Fi[i-1]}   # Frequências absolutas
> fi
> fi_ast<- -fi/Fi[k]   # Frequências relativas
```

Exercício 1.2

> Fi_ast< −Fi/Fi[k] # Frequências relativas acumuladas

> cl_inf< −quebra[-(k+1)] # Extremos inferiores das classes

> cl_sup< −quebra[-1] # Extremos superiores das classes

> sepa< −rep(-",k) # Vector de separadores

> write(t(cbind(cl_inf,sepa,cl_sup,fi,Fi,fi_ast,Fi_ast)),"ex12.txt",ncol=7) # Escreve para o ficheiro ex12.txt os resultados

>

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

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

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

Exercício 1.2

> 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 caracteristica *counts* do *hist(dados)*

Exercício 1.2

$> postscript(file = "histograma.ps", horizontal = FALSE)$

$> hist(dados, c(170, 255, 340, 425, 510, 595, 680), xlab = "n^{o}palavras",$
$ylab = "Frequencia", main =)$

$> fi < -hist(dados, c(170, 255, 340, 425, 510, 595, 680), xlab = "n^{o}palavras",$
$ylab = "Frequencia", main =)\$counts$

$> lines(c(212.5, 297.5, 382.5, 467.5, 552.5, 637.5), fi, col = 2)$

$> dev.off()$

$> postscript(file = "caixadebigodes.ps", horizontal = FALSE)$

$> boxplot(dados, range = 0, horizontal = TRUE)$

$> dev.off()$

8.2 Matrizes

8.2.1 Criação de matrizes e seu manuseamento

- Podemos criar uma matriz através da função *matrix*.

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

  $> M <- t(M)$

- 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

- 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

8.3 Data frames

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.

8.3.1 Criação de data frames e seu manuseamento

$> x1 < -1 : 10$

$> x2 < -11 : 20$

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

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

$> d1$

$> attributes(d1)$

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: 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. "$c : /Dados/dados.txt$"

$> dados$

$> dim(dados)$

- Podemos atribuir nomes às colunas (por defeito as linhas estão identificadas pelos seus números):
  $> names(dados) <- c("Nome1","Nome2", \ldots)$ # Tantos nomes como colunas

- 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.

- Adicionemos uma nova coluna (variável), *Nome.x*, ao data frame (Nome.x tem de ser um vector 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:

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

  $> attach(dados)$

8.4 Listas

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

Criação de listas

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

$> uma.lista$

$> uma.lista\$uma.palavra$

$> uma.lista[1]$

$> attributes(uma.lista)$

$> mode(uma.lista)$

8.5 Funções

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") }
}
}
```

8.6 Regressão Linear Simples

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

$> x <- c(20, 40, 61, 85, 68)$

$> y <- c(1, 1.5, 3, 4.2, 3.3)$

$> lm(y \sim x)$

$> names(lm(y \sim x))$

$> lm(y \sim x)\$coefficients$ #Dá os parâmetros de regressão estimados

$> lm(y \sim x)\$residuals$ #Dá os resíduos da regressão

$> lm(y \sim x)\$fitted$ #Dá os valores estimados de regressão, $\hat{Y}_i$