Universidade de São Paulo
Escola Superior de Agricultura “Luiz de Queiroz”

Noções de programação estatística usando o software livre R (Versão 1)

Cristian Villegas

Março de 2013
Piracicaba

Resumo

O profissional de hoje em dia precisa conhecer muito bem algum *software* estatístico ou não estatístico para realizar análises de dados. Alguns exemplos de *softwares* estatísticos são SAS, Minitab, SPSS, Stata, Statistica, Splus e alguns exemplos de *softwares* não estatísticos são Matlab, Maple e Mathematica. O problema dos *softwares* mencionados anteriormente é que são pagos e, muitas vezes, a licença comercial tem um alto custo que nem sempre empresas e usuários particulares podem pagar. Por outro lado, não existe nenhum *software* que tenha todas as ferramentas estatísticas para desenvolver as análises mais diversas, como por exemplo, as que envolvem modelos não lineares generalizados, modelos não lineares mistos, modelos com distribuições simétricas ou assimétricas, séries temporais, superficies de resposta, análise multivariada, análise de sobrevivência. Pelas razões citadas precisamos ter uma noção de programação estatística para poder resolver desde os problemas mais simples até os mais complexos. Com o intuito de poder programar o que for preciso para fazer uma boa análise estatística, e usando um *software* gratuito, apresento uma breve introdução ao *software* livre R versão 2.15.2 para usuários de Windows. Para melhor aproveitar o curso é necessário ter noções de estatística descritiva, probabilidades, inferência, regressão, modelos lineares, cálculo e álgebra de matrizes.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Operações básicas

1.1 Operadores aritméticos

A seguir definimos as operações elementares: diferença, soma, produto, divisão e potenciação.

1.1.1 Diferença

```
> 1-2
[1] -1
```

1.1.2 Soma

```
> 2+1
[1] 3
```

1.1.3 Produto

```
> 2*3
[1] 6
```

1.1.4 Divisão

```
> 5/8
[1] 0.625
```

1.1.5 Potenciação

```
> 4^4
[1] 256

> 4^6/7
[1] 585.1429

> 4^(6/7)
[1] 3.281341
```

```
# sqrt(4)=4^(1/2)
> sqrt(4)
[1] 2
```

1.2 Funções Básicas

1.2.1 Função modular

$$y = f(x) = |x| = \begin{cases} x, & \text{se} \quad x \geq 0 \\ -x, & \text{se} \quad x < 0 \end{cases}$$

```
> abs(-23)
[1] 23
```

```
> abs(0)
[1] 0
```

```
> abs(35)
[1] 35
```

1.2.2 Funções trigonométricas

1. Função seno: $f : \mathbb{R} \to \mathbb{R}$, $f(x) = \sin x$.
2. Função cosseno: $f : \mathbb{R} \to \mathbb{R}$, $f(x) = \cos x$.
3. Função tangente: $f : \mathbb{R} - \left\{\frac{\pi}{2} + h\pi, h \in \mathbb{Z}\right\} \to \mathbb{R}$, $f(x) = \tan x$.

```
> sin(Pi) #Erro!
Error: object 'Pi' not found
```

```
> sin(pi)   #Correto!
[1] 1.224646799e-16
```

```
> cos(pi)
[1] -1
```

```
> tan(pi)
[1] -1.224646799e-16
```

```
> sin(0)
[1] 0
```

```
> cos(0)
[1] 1
```

```
> tan(0)
[1] 0
```

```
> sin(pi/2)#sin(90)=1
[1] 1
```

```
> cos(pi/2)#cos(90)=0
[1] 6.123233996e-17
```

```
> tan(pi/2)#cuidado com este resultado! lembrar que tan(90) não está definido
[1] 1.633123935e+16
```

Funções trigonométricas inversas

1. Função Arcoseno: $f : [-1, 1] \to \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, $f(x) = \arcsin x$
2. Função Arco-cosseno: $f : [-1, 1] \to [0, \pi]$, $f(x) = \arccos x$
3. Função Arco-tangente: $f : \mathbb{R} \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, $f(x) = \arctan x$

```
> asin(pi/4)#definido fora do intervalo (-Inf, -1] e [1, Inf)
[1] 0.9033391108
```

```
> acos(pi/4)#definido fora do intervalo (-Inf, -1] e [1, Inf)
[1] 0.667457216
```

```
> atan(pi/4)
[1] 0.66577375
```

Funções trigonométricas hiperbólicas

```
> sinh(pi)#sinh(x)=(exp(x)-exp(-x))/2
[1] 11.54873936
```

```
> cosh(pi)#cosh(x)=(exp(x)+exp(-x))/2
[1] 11.59195328
```

```
> tanh(pi)#tanh(x)=sinh(x)/cosh(x)
[1] 0.9962720762
```

Funções trigonométricas hiperbólicas inversas

```
> asinh(pi)
[1] 1.862295743
```

```
> acosh(pi)
[1] 1.811526272
```

```
> atanh(pi)#Erro!
[1] NaN
Warning message:
In atanh(pi) : NaNs produced
```

1.2.3 Função exponencial

$f : \mathbb{R} \to \mathbb{R}^+$, $f(x) = a^x$ e $1 \neq a > 0$.

```
> exp(0)
[1] 1
```

```
> exp(1)#tentar options(digits=10);exp(1)
[1] 2.718281828
```

```
> exp(-1)
[1] 0.3678794412
```

1.2.4 Função logarítmica

$f : \mathbb{R}^+ \to \mathbb{R}$, $f(x) = \log_a(x)$ e $1 \neq a > 0$.

Função logarítmica de base e

```
> log(e)#Erro! tentar log(exp(1))
Error: object 'e' not found
```

```
> log(exp(1))#Função logarítmica de base e
[1] 1
```

```
> log(x=1,base=exp(1))#Função logarítmica de base e
[1] 0
```

```
> log(100)
[1] 4.605170186
```

Função logarítmica de base 10

```
> log10(100)
[1] 2
```

```
> log(100,base=10)
[1] 2
```

1.3 Comandos básicos

A seguir descrevemos brevemente os comandos `seq()`, `rep()` e `factorial()`.

1.3.1 `seq()`

O comando `seq(from = , to = , by = , length.out = NULL)` gera uma sequência de números, em que

```
from, to: o valor inicial e final da sequência.
      by:  incremento da sequência.
length.out: comprimento desejado  da sequência.
```

```
> seq(from = 1, to = 10, by = 0.5)
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5
 9.0  9.5 10.0
```

```
> seq(from = 1, to = 10, length.out = 10)
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> seq(from = 1, to = 10, length.out = 13)
 [1]  1.00  1.75  2.50  3.25  4.00  4.75  5.50  6.25  7.00  7.75  8.50  9.25 10.00
```

```
> seq(from = 1, to = 10, by = 0.75)
 [1]  1.00  1.75  2.50  3.25  4.00  4.75  5.50  6.25  7.00  7.75  8.50  9.25 10.00
```

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

1.3.2 `rep()`

Repete elementos de vetores e listas. `rep(x, ...)`, em que ... pode ser “times", “length.out"ou “each".

```
> rep(1:4, times=2)
[1] 1 2 3 4 1 2 3 4
```

```
> rep(1:4, each = 2)       # rep(1:4, each = 2) igual que rep(1:4, 2)?
[1] 1 1 2 2 3 3 4 4
```

```
#rep(1:4, c(2,2,2,2)) e rep(1:4, each = 2) são a mesma coisa
> rep(1:4, times=c(2,2,2,2))
[1] 1 1 2 2 3 3 4 4
```

```
> rep(1:4, times=c(2,1,2,1))
[1] 1 1 2 3 3 4
```

```
> rep(1:4, each = 2, length.out = 4)
[1] 1 1 2 2
```

```
> rep(1:4, each = 2, length.out = 10)
 [1] 1 1 2 2 3 3 4 4 1 1
```

```
> rep(1:4, each = 2, times = 3)
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

1.3.3 `rep()` e `seq()` simultaneamente

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

```
> rep(seq(1:5), length.out=7)
[1] 1 2 3 4 5 1 2
```

1.3.4 `factorial()`

Calcula o fatorial de um numero determinado.

```
> factorial(0) #0!=1 por definição
[1] 1
```

```
> factorial(3) #3!=3*2*1=6
[1] 6
```

```
> factorial(7) #7!=7*6*5*4*3*2*1=5040
[1] 5040
```

Capítulo 2

Tipos de Objetos

2.1 Removendo e listando objetos

```
> rm(list=ls(all=TRUE))#Remove todos os objetos

> ls()          # lista todos os objetos
character(0)    # até agora não temos definido nenhum objeto

> objects()     # equivalente a ls()
character(0)
```

2.2 Vetor de caracteres e vetor numérico

Criando duas variáveis: `nomes` e `idades`.

```
> nomes<- c("Pedro", "Maria", "Italo", "Alessandra")
> idades<-c(34,26,22,25)
```

2.2.1 `length()`

Comprimento de um objeto que pode ser uma lista, fator ou algum outro objeto do R.

```
> length(nomes)
[1] 4

> length(idades)
[1] 4

> NOMES
Error: object 'NOMES' not found

> nomes
[1] "Pedro"      "Maria"      "Italo"      "Alessandra"
```

```
> #Qual é a diferença com os comandos anteriores?
> (nomes<- c("Pedro", "Maria", "Italo", "Alessandra"))
[1] "Pedro"      "Maria"      "Italo"      "Alessandra"

> (idades<-c(34,26,22,25))
[1] 34 26 22 25
```

2.2.2 `class()`

A função `class()` define a classe de objeto que estamos trabalhando

```
> class(nomes)
[1] "character"

> class(idades)
[1] "numeric"

> #Testando o comando ls()
> ls()
[1] "idades" "nomes"
```

2.2.3 `is.character()` e `is.numeric()`

```
> #"nomes" é uma variável que possui caracteres (string)?
> is.character(nomes)
[1] TRUE

> #"nomes" é uma variável numérica?
> is.numeric(nomes)
[1] FALSE

> #"idades" é uma variável que possui caracteres (string)?
> is.character(idades)
[1] FALSE

> #"idades" é uma variável numérica?
> is.numeric(idades)
[1] TRUE
```

2.2.4 Concatenando objetos

```
> dados<- c(idades, nomes)

> dados
[1] "34"          "26"          "22"          "25"
"Pedro"       "Maria"       "Italo"       "Alessandra"

> length(dados)
[1] 8

> class(dados)
[1] "character"
```

2.2.5 `is.vector()`

```
> #"dados" é um vetor?
> is.vector(dados)
[1] TRUE

> is.matrix(dados)
[1] FALSE
```

2.3 matrix()

Com o comando `matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,dimnames = NULL)`, criamos uma matriz.

```
> (matriz.dados<- matrix(data=dados))
     [,1]
[1,] "34"
[2,] "26"
[3,] "22"
[4,] "25"
[5,] "Pedro"
[6,] "Maria"
[7,] "Italo"
[8,] "Alessandra"
```

matriz.dados é uma matriz?

```
> is.matrix(matriz.dados)
[1] TRUE

> class(matriz.dados)
[1] "matrix"
```

Definindo uma matriz com duas linhas

```
> matrix(data=dados,nrow=2)
     [,1] [,2] [,3]     [,4]
[1,] "34" "22" "Pedro" "Italo"
[2,] "26" "25" "Maria" "Alessandra"
```

Definindo uma matriz 4×2

```
> matrix(data=dados,nrow=4,ncol=2)
     [,1] [,2]
[1,] "34" "Pedro"
[2,] "26" "Maria"
[3,] "22" "Italo"
[4,] "25" "Alessandra"
```

Definição de uma matriz 4×2 (entrada por colunas)

```
> matrix(data=dados,nrow=4,ncol=2,byrow=FALSE)
     [,1] [,2]
[1,] "34" "Pedro"
[2,] "26" "Maria"
[3,] "22" "Italo"
[4,] "25" "Alessandra"
```

Definição de uma matriz 4×2 (entrada por linhas)

```
> matrix(data=dados,nrow=4,ncol=2,byrow=TRUE)
     [,1]     [,2]
[1,] "34"     "26"
[2,] "22"     "25"
[3,] "Pedro" "Maria"
[4,] "Italo" "Alessandra"
```

Atribuindo nomes a cada linha e a cada coluna de uma matriz 4×2 (entrada por colunas)

```
> matriz.final<- matrix(data=dados, nrow=4, ncol=2, byrow=FALSE,
+  dimnames = list(c("id 1", "id 2","id 3","id 4"),
+ c("Idade", "Nome")))

> matriz.final
     Idade Nome
id 1 "34"  "Pedro"
id 2 "26"  "Maria"
id 3 "22"  "Italo"
id 4 "25"  "Alessandra"
> length(matriz.final)
[1] 8
```

O comando `dim(x)` determina a dimensão de x, sendo x uma `matrix`, `array` ou `data frame`

```
> dim(matriz.final)
[1] 4 2

> dim(nomes)
NULL

> dim(idades)
NULL

> dim(dados)
NULL
```

2.4 `array()`

`array(data = NA, dim = length(data), dimnames = NULL)`

matriz.final é um array?

```
> is.array(matriz.final)
[1] TRUE
```

```
> array(data = 1:6, dim = c(3,2))
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
> matrix(data=1:6,nrow=3,ncol=2,byrow=FALSE)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
> (arranjo<-array(data=1:24, dim=c(2,4,3)))
, , 1
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

, , 2
     [,1] [,2] [,3] [,4]
[1,]    9   11   13   15
[2,]   10   12   14   16

, , 3
     [,1] [,2] [,3] [,4]
[1,]   17   19   21   23
[2,]   18   20   22   24
```

```
> arranjo[1,,]
     [,1] [,2] [,3]
[1,]    1    9   17
[2,]    3   11   19
[3,]    5   13   21
[4,]    7   15   23
```

```
> arranjo[,2,]
     [,1] [,2] [,3]
[1,]    3   11   19
[2,]    4   12   20
```

```
> arranjo[,,3]
     [,1] [,2] [,3] [,4]
[1,]   17   19   21   23
[2,]   18   20   22   24
```

```
> arranjo[2,,3]
[1] 18 20 22 24
```

```
> arranjo[,3:4,]
, , 1
     [,1] [,2]
[1,]    5    7
[2,]    6    8

, , 2
     [,1] [,2]
[1,]   13   15
[2,]   14   16

, , 3
     [,1] [,2]
[1,]   21   23
[2,]   22   24
```

2.5 factor()

```
sexo <- c(rep("Femenino",2), rep("Masculino", 3))
sexo <- factor(sexo)
class(sexo)
sexo;levels(sexo)
plot(sexo)

sexo2<-factor(x = c(rep("A", 2), rep("B",3)),
labels = c("Feminino", "Masculino"))
class(sexo);sexo2;levels(sexo2)
plot(sexo2)

sexo3<-factor(x = rep(1:2, times=c(2,3)),
labels = c("Feminino", "Masculino"))
class(sexo3);sexo3;levels(sexo3)
plot(sexo3)
```

```
bebe.brejas <- c(rep("muitas",4), rep("poucas", 1),rep("nada", 3))
bebe.brejas <- ordered(x=bebe.brejas)
class(bebe.brejas)
bebe.brejas
levels(bebe.brejas)
plot(bebe.brejas)
```

2.5.1 cbind() e rbind()

```
> cbind(nomes, idades)
     nomes         idades
[1,] "Pedro"       "34"
[2,] "Maria"       "26"
[3,] "Italo"       "22"
[4,] "Alessandra"  "25"

> rbind(nomes, idades)
       [,1]    [,2]    [,3]    [,4]
nomes  "Pedro" "Maria" "Italo" "Alessandra"
idades "34"    "26"    "22"    "25"
```

2.6 data.frame()

```
> data.frame(nomes, idades)
       nomes idades
1      Pedro     34
2      Maria     26
3      Italo     22
4 Alessandra     25

> data.frame(X=nomes, Y=idades)
           X  Y
1      Pedro 34
2      Maria 26
3      Italo 22
4 Alessandra 25
```

```
> summary(data.frame(nomes, idades))
       nomes       idades
 Alessandra:1   Min.   :22.00
 Italo     :1   1st Qu.:24.25
 Maria     :1   Median :25.50
 Pedro     :1   Mean   :26.75
                3rd Qu.:28.00
                Max.   :34.00
```

2.6.1 names(), colnames() e rownames()

```
> DATA.FRAME<- data.frame(X=nomes, Y=idades)

> DATA.FRAME
           X  Y
1      Pedro 34
2      Maria 26
3      Italo 22
4 Alessandra 25

> names(DATA.FRAME)
[1] "X" "Y"

> colnames(DATA.FRAME)
[1] "X" "Y"

> rownames(DATA.FRAME)
[1] "1" "2" "3" "4"
```

```
> colnames(DATA.FRAME)<- c("XX","YY")

> rownames(DATA.FRAME)<- c("id1","id2","id3","id4")

> names(DATA.FRAME)
[1] "XX" "YY"
```

```
> brinca.dados<- data.frame(X=idades,Y=idades*2,Z=idades+idades*2)

> brinca.dados
   X  Y   Z
1 34 68 102
2 26 52  78
3 22 44  66
4 25 50  75
```

2.6.2 `colSums()` e `colMeans()`

`colSums()` e `colMeans()` funções que calculam soma e média das colunas.

```
> colSums(brinca.dados)
  X   Y   Z
107 214 321

> colMeans(brinca.dados)
    X     Y     Z
26.75 53.50 80.25
```

2.6.3 `rowSums()` e `rowMeans()`

`rowSums()` e `rowMeans()` funções que calculam soma e média das linhas

```
> rowSums(brinca.dados)
[1] 204 156 132 150

> rowMeans(brinca.dados)
[1] 68 52 44 50
```

Seleccionando elementos de um data.frame

```
> brinca.dados
   X  Y   Z
1 34 68 102
2 26 52  78
3 22 44  66
4 25 50  75

> brinca.dados[1,]#selecciono primeira linha
   X  Y   Z
1 34 68 102

> brinca.dados[,2]#selecciono segunda coluna
[1] 68 52 44 50

> brinca.dados$Y
[1] 68 52 44 50

> brinca.dados[,"Y"]
[1] 68 52 44 50
```

Seleção de colunas 1 e 3

```
> brinca.dados[,c(1,3)]
   X   Z
1 34 102
2 26  78
3 22  66
4 25  75
```

```
> brinca.dados[,-2]
   X   Z
1 34 102
2 26  78
3 22  66
4 25  75

> brinca.dados[,c("X","Z")]
   X   Z
1 34 102
2 26  78
3 22  66
4 25  75
```

2.6.4 `sort()` e `order()`

Ordenação de < a >

```
> brinca.dados[,"X"]
[1] 34 26 22 25

> sort(brinca.dados[,"X"],decreasing = FALSE)
[1] 22 25 26 34

> order(brinca.dados[,"X"], decreasing = FALSE)
[1] 3 4 2 1
```

Ordenação de > a <

```
> brinca.dados[,"X"]
[1] 34 26 22 25

> sort(brinca.dados[,"X"],decreasing = TRUE)
[1] 34 26 25 22

> order(brinca.dados[,"X"], decreasing = TRUE)
[1] 1 2 4 3
```

```
> #Listando objetos novamente
> ls()
 [1] "a1"           "a2"           "a3"           "a4"           "brinca.dados"
 [6] "dados"        "DATA.FRAME"   "idades"       "matriz.dados" "matriz.final"
[11] "nomes"

> rm(a1,a2,dados)#remove o objeto "a1", "a2" e "dados"
> ls()#listando objetos
[1] "a3"           "a4"           "brinca.dados" "DATA.FRAME"   "idades"
[6] "matriz.dados" "matriz.final" "nomes"
```

2.7 `list()`

Listas de objetos, estudaremos para criar funções!!

```
> minha.lista<- list(um=7:9, dois=c("Marcelo, Gisele"), tres=factor(letters[4:7]))
```

```
> minha.lista
$um
[1] 7 8 9

$dois
[1] "Marcelo, Gisele"

$tres
[1] d e f g
Levels: d e f g
```

2.8 Opções com data.frame

2.8.1 `subset()`

Subconjuntos de vetores, matrizes e data frames

```
> data(airquality)#?airquality
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

```
> str(airquality)
'data.frame':   153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...

> dim(airquality)
[1] 153   6

> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

```
> summary(airquality[,1:4])
     Ozone           Solar.R           Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
 NA's   : 37.00   NA's   :  7.0
```

```
> class(airquality)
[1] "data.frame"

> subset(x=airquality, subset=Wind > 20, select=c(Ozone, Day))
   Ozone Day
9      8   9
48    37  17
```

2.8.2 apply()

```
> apply(X=airquality, MARGIN=2, FUN=mean, na.rm=F)
    Ozone   Solar.R      Wind      Temp     Month       Day
       NA        NA  9.957516 77.882353  6.993464 15.803922
```

```
> apply(X=airquality, MARGIN=2, FUN=mean, na.rm=T)
     Ozone    Solar.R       Wind       Temp      Month        Day
 42.129310 185.931507   9.957516  77.882353   6.993464  15.803922
```

2.8.3 replace()

```
> replace(x=c(4,7,9), list=c(1,2,3), values=c(5,8,0))
[1] 5 8 0
```

2.8.4 lapply()

```
> data(cars)
> lapply(X=cars,FUN=mean)
$speed
[1] 15.4

$dist
[1] 42.98
```

2.8.5 sapply()

```
> sapply(X=cars,FUN=mean)
speed  dist
15.40 42.98
```

2.8.6 tapply()

```
> head(warpbreaks)
  breaks wool tension
1     26    A       L
2     30    A       L
3     54    A       L
4     25    A       L
5     70    A       L
6     52    A       L
```

```
> str(warpbreaks)
'data.frame':   54 obs. of  3 variables:
 $ breaks : num  26 30 54 25 70 52 51 26 67 18 ...
 $ wool   : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
 $ tension: Factor w/ 3 levels "L","M","H": 1 1 1 1 1 1 1 1 1 2 ...
```

```
> summary(warpbreaks)
     breaks      wool   tension
 Min.   :10.00   A:27   L:18
 1st Qu.:18.25   B:27   M:18
 Median :26.00          H:18
 Mean   :28.15
 3rd Qu.:34.00
 Max.   :70.00
```

```
> tapply(X=warpbreaks$breaks, INDEX=warpbreaks[,-1], FUN=sum)
    tension
wool   L   M   H
   A 401 216 221
   B 254 259 169
```

2.8.7 sweep()

```
> dadosX<- matrix(data=1:10,nrow=5,ncol=2)
> dadosX
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

> colMeans(dadosX)
[1] 3 8
```

```
> dadosY<- sweep(x=dadosX, MARGIN=2, STATS=colMeans(dadosX), FUN="-")
> dadosY
     [,1] [,2]
[1,]   -2   -2
[2,]   -1   -1
[3,]    0    0
[4,]    1    1
[5,]    2    2
```

2.8.8 merge()

```
> autores <- data.frame( sobrenome = I(c("Tukey", "Venables", "Tierney",
"Ripley", "McNeil")),nacionalidade = c("US", "Australia", "US", "UK",
"Australia"), sexo = c("M", "F", "M", "M", "F"))
```

```
> autores
  sobrenome nacionalidade sexo
1     Tukey            US    M
2  Venables     Australia    F
3   Tierney            US    M
4    Ripley            UK    M
5    McNeil     Australia    F
```

```
> livros <- data.frame(
+     nome = I(c("Tukey", "Venables", "Tierney",
+              "Ripley", "Ripley", "McNeil", "R Core")),
+     titulo = c("Exploratory Data Analysis",
+                "Modern Applied Statistics with S",
+                "LISP-STAT",
+                "Spatial Statistics", "Stochastic Simulation",
+                "Interactive Data Analysis",
+                "An Introduction to R") )
> livros
      nome                           titulo
1    Tukey        Exploratory Data Analysis
2 Venables Modern Applied Statistics with S
3  Tierney                       LISP-STAT
4   Ripley               Spatial Statistics
5   Ripley           Stochastic Simulation
6   McNeil        Interactive Data Analysis
7   R Core             An Introduction to R
```

```
(m1 <- merge(x=autores, y=livros, by.x = "sobrenome", by.y = "nome"))
  sobrenome nacionalidade sexo                           titulo
1    McNeil     Australia    F        Interactive Data Analysis
2    Ripley            UK    M               Spatial Statistics
3    Ripley            UK    M           Stochastic Simulation
4   Tierney            US    M                       LISP-STAT
5     Tukey            US    M        Exploratory Data Analysis
6  Venables     Australia    F Modern Applied Statistics with S
```

```
(m2 <- merge(livros, autores, by.x = "nome", by.y = "sobrenome"))
      nome                           titulo nacionalidade sexo
1   McNeil        Interactive Data Analysis     Australia    F
2   Ripley               Spatial Statistics            UK    M
3   Ripley           Stochastic Simulation            UK    M
4  Tierney                       LISP-STAT            US    M
5    Tukey        Exploratory Data Analysis            US    M
6 Venables Modern Applied Statistics with S     Australia    F
```

2.8.9 with()

```
> library(MASS)
> names(anorexia)
[1] "Treat"  "Prewt"  "Postwt"
```

```
> str(anorexia)
'data.frame':   72 obs. of  3 variables:
 $ Treat : Factor w/ 3 levels "CBT","Cont","FT": 2 2 2 2 2 2 2 2 2 2 ...
 $ Prewt : num  80.7 89.4 91.8 74 78.1 88.3 87.3 75.1 80.6 78.4 ...
 $ Postwt: num  80.2 80.1 86.4 86.3 76.1 78.1 75.1 86.7 73.5 84.6 ...
```

```
> summary(anorexia)
  Treat        Prewt           Postwt
 CBT :29   Min.   :70.00   Min.   : 71.30
 Cont:26   1st Qu.:79.60   1st Qu.: 79.33
 FT  :17   Median :82.30   Median : 84.05
           Mean   :82.41   Mean   : 85.17
           3rd Qu.:86.00   3rd Qu.: 91.55
           Max.   :94.90   Max.   :103.60
```

```
> anorexia2<- anorexia[1:10,]

> anorexia2
   Treat Prewt Postwt
1   Cont  80.7   80.2
2   Cont  89.4   80.1
3   Cont  91.8   86.4
4   Cont  74.0   86.3
5   Cont  78.1   76.1
6   Cont  88.3   78.1
7   Cont  87.3   75.1
8   Cont  75.1   86.7
9   Cont  80.6   73.5
10  Cont  78.4   84.6
```

```
> with(data=anorexia2, expr=Postwt - Prewt)
 [1]  -0.5  -9.3  -5.4  12.3  -2.0 -10.2 -12.2  11.6  -7.1   6.2
```

2.8.10 within()

```
> within(data = anorexia2, expr=wtDiferença <- Postwt - Prewt)
   Treat Prewt Postwt wtDiferença
1   Cont  80.7   80.2        -0.5
2   Cont  89.4   80.1        -9.3
3   Cont  91.8   86.4        -5.4
4   Cont  74.0   86.3        12.3
5   Cont  78.1   76.1        -2.0
6   Cont  88.3   78.1       -10.2
7   Cont  87.3   75.1       -12.2
8   Cont  75.1   86.7        11.6
9   Cont  80.6   73.5        -7.1
10  Cont  78.4   84.6         6.2
```

2.8.11 transform()

```
> transform(anorexia2, wtDiferença = Postwt - Prewt)
   Treat Prewt Postwt wtDiferença
1   Cont  80.7   80.2        -0.5
2   Cont  89.4   80.1        -9.3
3   Cont  91.8   86.4        -5.4
4   Cont  74.0   86.3        12.3
5   Cont  78.1   76.1        -2.0
6   Cont  88.3   78.1       -10.2
7   Cont  87.3   75.1       -12.2
8   Cont  75.1   86.7        11.6
9   Cont  80.6   73.5        -7.1
10  Cont  78.4   84.6         6.2
```

Capítulo 3

Obtendo ajuda

3.1 help() ou ?

Obtendo ajuda de um comando conhecido

```
?sqrt
help(sqrt)
```

3.2 help.search()

Como obter ajuda quando não sei o comando?

```
help.search("linear models")#Função procurada "lm"
example("lm")
```

3.3 RSiteSearch()

Obtendo ajuda da web.

```
RSiteSearch("quality control")
```

3.4 demo()

Todos os demos disponíveis dos pacotes instalados

```
demo(package = .packages(all.available = TRUE))
```

Mostra um demo, pausando entre páginas

```
demo(topic=lm.glm, package="stats", ask=TRUE)
```

Mostra um demo, sem pausar entre páginas

```
demo(topic=lm.glm, package="stats", ask=FALSE)
```

Outro exemplo

```
demo(package="graphics", ask=TRUE)  #demos no pacote "graphics"
demo(topic=graphics,package="graphics", ask=TRUE)
```

Capítulo 4

Funções estatísticas básicas

4.1 summary()

Trabalhando dados de soja do pacote nlme

```
> rm(list=ls(all=TRUE))
> library(nlme) #carregando pacote
> data()        #lista todos os dados disponíveis
```

Lista todos os dados de todos os pacotes disponiveis

```
> data(package = .packages(all.available = TRUE))
Warning messages:
1: In data(package = .packages(all.available = TRUE)) :
  datasets have been moved from package 'base' to package 'datasets'
2: In data(package = .packages(all.available = TRUE)) :
  datasets have been moved from package 'stats' to package 'datasets'
```

```
> data(Soybean) # carrega dados "Soybean" do pacote nlme
> ?Soybean      # obtendo descrição dos dados
> Soybean       # mostra o conjunto de dados completo
```

```
> head(Soybean, n=15L)#mostra primeiras 15 linhas
Grouped Data: weight ~ Time | Plot
     Plot Variety Year Time   weight
1  1988F1       F 1988   14  0.10600
2  1988F1       F 1988   21  0.26100
3  1988F1       F 1988   28  0.66600
4  1988F1       F 1988   35  2.11000
5  1988F1       F 1988   42  3.56000
6  1988F1       F 1988   49  6.23000
7  1988F1       F 1988   56  8.71000
8  1988F1       F 1988   63 13.35000
9  1988F1       F 1988   70 16.34170
10 1988F1       F 1988   77 17.75083
11 1988F2       F 1988   14  0.10400
12 1988F2       F 1988   21  0.26900
13 1988F2       F 1988   28  0.77800
14 1988F2       F 1988   35  2.12000
15 1988F2       F 1988   42  2.93000
```

```
> str(as.data.frame(Soybean))#identifica tipo de cada variável
'data.frame':   412 obs. of  5 variables:
 $ Plot   : Ord.factor w/ 48 levels "1988F4"<"1988F2"<..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Variety: Factor w/ 2 levels "F","P": 1 1 1 1 1 1 1 1 1 1 ...
 $ Year   : Factor w/ 3 levels "1988","1989",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Time   : num  14 21 28 35 42 49 56 63 70 77 ...
 $ weight : num  0.106 0.261 0.666 2.11 3.56 ...

> dim(Soybean)
[1] 412   5
```

```
> summary(Soybean)
      Plot      Variety    Year          Time           weight
 1988F4 : 10   F:204   1988:156   Min.   :14.00   Min.   : 0.029
 1988F1 : 10   P:208   1989:128   1st Qu.:27.00   1st Qu.: 0.596
 1988F8 : 10           1990:128   Median :41.00   Median : 3.485
 1988F6 : 10                      Mean   :43.67   Mean   : 6.362
 1988F3 : 10                      3rd Qu.:63.00   3rd Qu.:10.914
 1988P1 : 10                      Max.   :84.00   Max.   :30.272
 (Other):352
```

Resumo da variável weight do banco de dados Soybean

```
> summary(Soybean$weight)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.029   0.596   3.485   6.362  10.910  30.270

> as.matrix(summary(Soybean$weight))
           [,1]
Min.      0.029
1st Qu.   0.596
Median    3.485
Mean      6.362
3rd Qu. 10.910
Max.     30.270
```

4.2 Medidas de resumo

4.2.1 min()

```
> min(Soybean$weight)           #mínimo de "weight"
[1] 0.029
```

4.2.2 quantile()

```
> quantile(Soybean$weight, probs=c(0.25, 0.50, 0.75)) #Q1, Q2 e Q3 de "weight"
      25%       50%       75%
 0.596042  3.485000 10.913573
```

4.2.3 median()

```
> median(Soybean$weight)    #Q2=Mediana de "weight"
[1] 3.485
```

4.2.4 mean()

```
> mean(Soybean$weight)      #média de "weight"
[1] 6.362162
```

4.2.5 max()

```
> max(Soybean$weight)           #máximo de "weight"
[1] 30.2717
```

4.3 Medidas de variabilidade

4.3.1 var() e sd()

```
> sd(Soybean$weight)
[1] 6.914517

> var(Soybean$weight)
[1] 47.81055
```

4.4 Função soma

4.4.1 sum()

```
> sum(Soybean$weight)
[1] 2621.211

> (n<- length(Soybean$weight))
[1] 412
```

4.4.2 cumsum()

```
> Soybean.reduzido<- Soybean[1:10,]
> cumsum(Soybean.reduzido$weight)
 [1]  0.10600  0.36700  1.03300  3.14300  6.70300 12.93300 21.64300 34.99300
 [9] 51.33470 69.08553
```

Usando a função soma para calcular a média

```
> sum(Soybean$weight)/n
[1] 6.362162

> mean(Soybean$weight)
[1] 6.362162
```

```
> (sum(Soybean$weight^2)-n*mean(Soybean$weight)^2)/(n-1)
[1] 47.81055

> var(Soybean$weight)
[1] 47.81055
```

4.5 Função produto

4.5.1 prod()

```
> prod(Soybean$weight)
[1] 1.366752e+144
```

4.5.2 cumprod()

```
> Soybean.reduzido<- Soybean[1:10,]
> cumprod(Soybean.reduzido$weight)
 [1] 1.060000e-01 2.766600e-02 1.842556e-02 3.887792e-02 1.384054e-01 8.622657e-01
 [7] 7.510334e+00 1.002630e+02 1.638467e+03 2.908415e+04
```

4.6 Como somar por colunas?

4.6.1 colSums()

```
> #Primeiro selecionamos 10 linas e todas as colunas para este exemplo
> Soybean.reduzido<- Soybean[1:10,]
```

```
> colSums(Soybean.reduzido[,c("Time", "weight")])
     Time    weight
455.00000  69.08553

> apply(X=Soybean.reduzido[,c("Time", "weight")],MARGIN=2,FUN=sum)
     Time    weight
455.00000  69.08553
```

4.7 Como somar por linhas?

4.7.1 rowSums()

```
> rowSums(Soybean.reduzido[,c("Time", "weight")])
       1        2        3        4        5        6        7        8
14.10600 21.26100 28.66600 37.11000 45.56000 55.23000 64.71000 76.35000
       9       10
86.34170 94.75083
```

```
> apply(X=Soybean.reduzido[,c("Time", "weight")],MARGIN=1,FUN=sum)
       1        2        3        4        5        6        7        8
14.10600 21.26100 28.66600 37.11000 45.56000 55.23000 64.71000 76.35000
       9       10
86.34170 94.75083
```

Capítulo 5

Leitura de dados

5.1 `getwd()` e `setwd()`

```
> rm(list=ls(all=TRUE))

> getwd()                                  #mostra o diretorio atual
[1] "D:/DropBox/Dropbox/2013/minicurso R/Codigo R"

> setwd("D:/DropBox/Dropbox/2013/minicurso R")          #muda o diretorio
> setwd("D:/DropBox/Dropbox/2013/minicurso R/Codigo R") #voltando ao diretorio original
```

5.2 Dados de crescimento de árvores de laranja

O conjunto de dados `Orange` possui 35 linhas e 3 colunas, cada linha representa o crescimento de árvores de laranja avaliado em 3 tempos.

```
Tree um fator indicando a árvore que foi medida.
age um vetor numérico mostrando a idade das árvores (dias desde 31/12/1968)
circumference um vetor numérico indicando o diâmetro à altura do peito (dap).
```

5.2.1 Carregando dados no formato txt

```
> orange.txt<- read.table("Orange.txt")#Erro, por que?
> head(orange.txt)
    V1   V2            V3
1 Tree  age circumference
2    1  118            30
3    1  484            58
4    1  664            87
5    1 1004           115
6    1 1231           120
```

```
> orange.txt<- read.table("Orange.txt", header=TRUE)
> head(orange.txt)
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
```

```
> dim(orange.txt)
[1] 35  3

> class(orange.txt)
[1] "data.frame"
```

```
> summary(orange.txt)
      Tree        age         circumference
 Min.   :1   Min.   : 118.0   Min.   : 30.0
 1st Qu.:2   1st Qu.: 484.0   1st Qu.: 65.5
 Median :3   Median :1004.0   Median :115.0
 Mean   :3   Mean   : 922.1   Mean   :115.9
 3rd Qu.:4   3rd Qu.:1372.0   3rd Qu.:161.5
 Max.   :5   Max.   :1582.0   Max.   :214.0
```

5.2.2 Carregando dados no formato csv

```
> orange.csv<- read.csv("Orange.csv")#Erro, porque?
> head(orange.csv)
  Tree.age.circumference
1              1;118;30
2              1;484;58
3              1;664;87
4             1;1004;115
5             1;1231;120
6             1;1372;142
> dim(orange.csv)
[1] 35  1
```

```
> orange.csv<- read.csv("Orange.csv", sep=";")#Corrigindo Erro

> head(orange.csv)
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
```

```
> dim(orange.csv)
[1] 35  3

> class(orange.csv)
[1] "data.frame"
```

```
> summary(orange.csv)
      Tree        age         circumference
 Min.   :1   Min.   : 118.0   Min.   : 30.0
 1st Qu.:2   1st Qu.: 484.0   1st Qu.: 65.5
 Median :3   Median :1004.0   Median :115.0
 Mean   :3   Mean   : 922.1   Mean   :115.9
 3rd Qu.:4   3rd Qu.:1372.0   3rd Qu.:161.5
 Max.   :5   Max.   :1582.0   Max.   :214.0
```

5.3 Dados perdidos `na.omit()`

5.3.1 Carregando dados no formato "txt" com valores NA

```
> rm(list=ls(all=TRUE))
> orangeNA<- read.table("OrangeNA1.txt", header=TRUE)#Erro, porque?
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,  :
  line 1 did not have 3 elements
```

```
> orangeNA<- read.table("OrangeNA2.txt", header=TRUE)#Corrigindo erro

> head(orangeNA)
  Tree  age circumference
1    1   NA            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231            NA
6    1 1372           142
```

```
> dim(orangeNA)
[1] 35  3

> class(orangeNA)
[1] "data.frame"
```

Resumo para dados com NA

```
> summary(orangeNA)
      Tree        age        circumference
 Min.   :1   Min.   : 118   Min.   : 30.0
 1st Qu.:2   1st Qu.: 484   1st Qu.: 69.0
 Median :3   Median :1004   Median :115.0
 Mean   :3   Mean   : 944   Mean   :118.2
 3rd Qu.:4   3rd Qu.:1372   3rd Qu.:167.0
 Max.   :5   Max.   :1582   Max.   :214.0
             NA's   :   2   NA's   :  2.0
```

Calculando a média com dados perdidos

```
> mean(orangeNA$circumference)# Erro, porque?
[1] NA

> mean(orangeNA$circumference,na.rm = FALSE)# Corrigindo Erro?
[1] NA

> mean(orangeNA$circumference,na.rm = TRUE)# Definitivo
[1] 118.2424
```

Calculando min, max e quantis com dados perdidos

```
> min(orangeNA$circumference,na.rm = TRUE)
[1] 30

> quantile(orangeNA$circumference,probs=c(0.25,0.50,0.75),na.rm = TRUE)
25% 50% 75%
 69 115 167

> median(orangeNA$circumference,na.rm = TRUE)
[1] 115

> max(orangeNA$circumference,na.rm = TRUE)
[1] 214
```

Calculando var e sd com dados perdidos

```
> var(orangeNA$circumference,na.rm = TRUE)
[1] 3290.502

> sd(orangeNA$circumference,na.rm = TRUE)
[1] 57.3629
```

Usando comando "na.omit" para trabalhar com dados perdidos

```
> head(orangeNA,n=10L)
   Tree  age circumference
1     1   NA            30
2     1  484            58
3     1  664            87
4     1 1004           115
5     1 1231            NA
6     1 1372           142
7     1 1582           145
8     2  118            NA
9     2  484            69
10    2  664           111
```

```
> dim(orangeNA)
[1] 35  3
```

```
> head(na.omit(orangeNA),n=10L)
   Tree  age circumference
2     1  484            58
3     1  664            87
4     1 1004           115
6     1 1372           142
7     1 1582           145
9     2  484            69
10    2  664           111
12    2 1231           172
13    2 1372           203
14    2 1582           203
```

```
> dim(na.omit(orangeNA))
[1] 31  3
```

Compare os comandos a seguir com os comandos anteriores

```
> summary(orangeNA)
      Tree        age         circumference
 Min.   :1   Min.   : 118   Min.   : 30.0
 1st Qu.:2   1st Qu.: 484   1st Qu.: 69.0
 Median :3   Median :1004   Median :115.0
 Mean   :3   Mean   : 944   Mean   :118.2
 3rd Qu.:4   3rd Qu.:1372   3rd Qu.:167.0
 Max.   :5   Max.   :1582   Max.   :214.0
             NA's   :   2   NA's   :  2.0
```

```
> summary(na.omit(orangeNA))
      Tree            age          circumference
 Min.   :1.000   Min.   : 118.0   Min.   : 30.0
 1st Qu.:2.000   1st Qu.: 574.0   1st Qu.: 72.0
 Median :3.000   Median :1004.0   Median :115.0
 Mean   :3.194   Mean   : 961.4   Mean   :119.9
 3rd Qu.:4.000   3rd Qu.:1372.0   3rd Qu.:169.5
 Max.   :5.000   Max.   :1582.0   Max.   :214.0
```

5.4 Operadores lógicos

```
> rm(list=ls(all=TRUE))
> idade<- c(34,23,56,43)
> sexo <- c("F", "F", "F", "M")
```

```
> idade[idade<30]
[1] 23
```

```
> idade[idade<23 && idade>25]
numeric(0)

> idade[idade<23 & idade>25]
numeric(0)
```

```
> idade[idade<23 || idade>25]
[1] 34 23 56 43

> idade[idade<23 |idade>25]
[1] 34 56 43
```

```
> idade<34
[1] FALSE  TRUE FALSE FALSE
```

```
> idade>34
[1] FALSE FALSE  TRUE  TRUE
```

```
> idade<=34
[1]  TRUE  TRUE FALSE FALSE
```

```
> idade>=34
[1]  TRUE FALSE  TRUE  TRUE
```

```
> idade==34
[1]  TRUE FALSE FALSE FALSE
```

```
> any(idade<34)#Alguma idade é menor do que 34?
[1] TRUE
```

```
> all(idade<34)#Todos as idades são menores do que 34?
[1] FALSE
```

Capítulo 6

Tipos de gráficos 2D

6.1 plot()

```
> rm(list=ls(all=TRUE))
> ?cars          #objendo informações do banco de dados
> data(cars)     #carrega dados
```

```
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

```
> str(cars)
'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...

> dim(cars)
[1] 50  2
```

```
> summary(cars$speed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    12.0    15.0    15.4    19.0    25.0
```

```
> attach(what=cars) #Anexar objetos ao "Caminho de pesquisa"

> summary(speed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    12.0    15.0    15.4    19.0    25.0

> detach(cars)  #Retire objetos do "Caminho de pesquisa"
```

```
> summary(speed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    12.0    15.0    15.4    19.0    25.0
```

```
> find(what="glm")
[1] "package:stats"

> find(what="lm")
[1] "package:stats"

> find(what="cor")
[1] "package:stats"
```

`lty` Tipo de linha

`lwd` Largura da linha

`X11()` Abre nova janela para um gráfico

`graphics.off()` Fecha todos os gráficos abertos

```
> X11()
> plot(x=speed, y=dist)
> savePlot(filename="plot_1", type="eps")

> X11()
> plot(x=speed, y=dist, type="l")
> savePlot(filename="plot_2", type="eps")
```

```
> X11()
> plot(x=speed, y=dist, type="l", col="blue")
> savePlot(filename="plot_3", type="eps")

> X11()
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo")
> savePlot(filename="plot_4", type="eps")
```

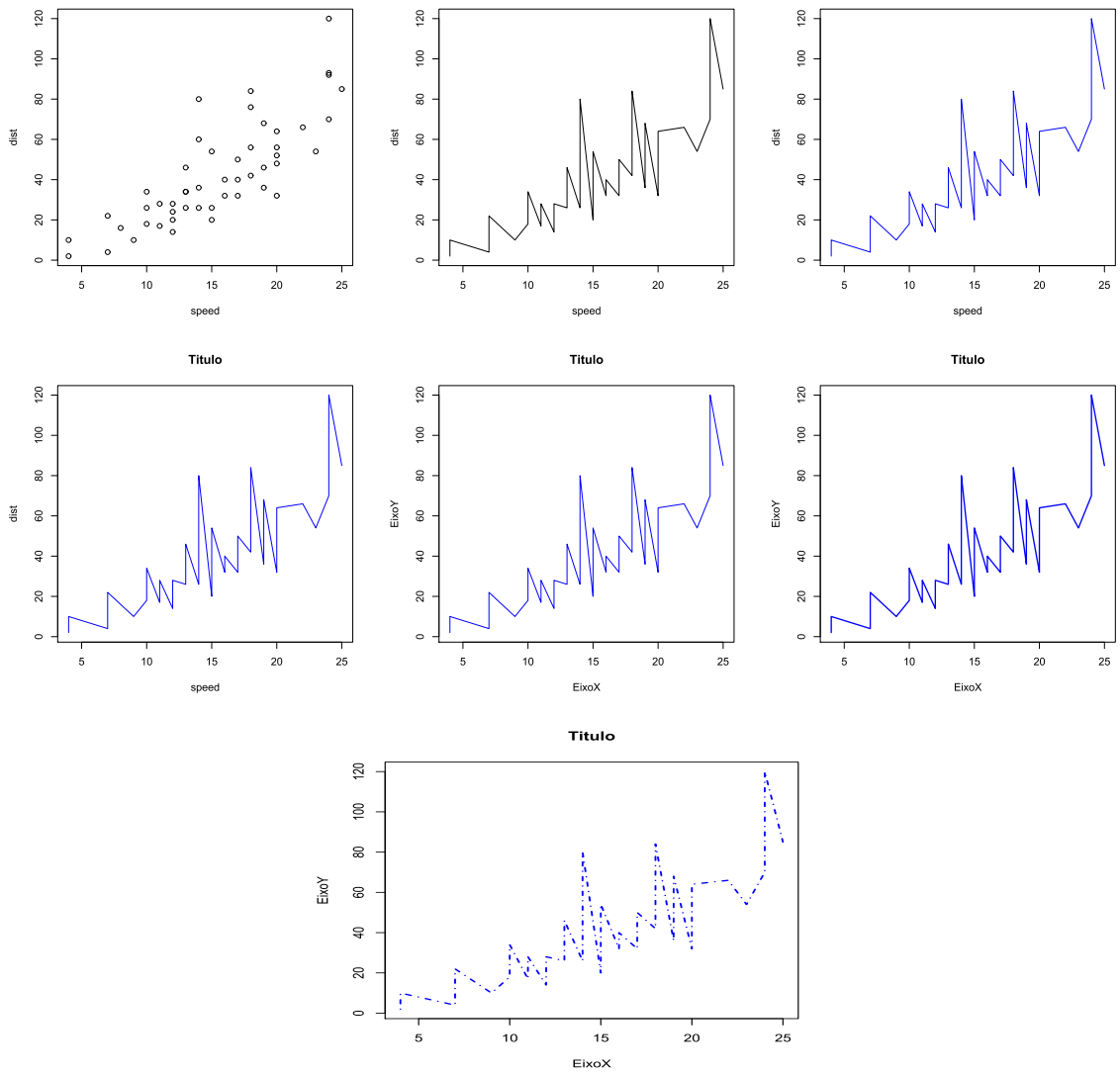

Figura 6.1: Opções do comando `plot()`.

```
> X11()
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY")
> savePlot(filename="plot_5", type="eps")

> X11()
> plot(x=speed, y=dist,type="l",col="blue",main="Titulo",xlab="EixoX",ylab="EixoY",lwd=2)
> savePlot(filename="plot_6", type="eps")

> X11()
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=2,lty=4)
> savePlot(filename="plot_7", type="eps")
```

6.1.1 `par()`

Como ter quatro gráficos na mesma janela?

```
> X11()
> par(mfrow=c(2,2))
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=1,lty=1)
> plot(x=speed, y=dist, type="l", col="green",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=2,lty=2)
> plot(x=speed, y=dist, type="l", col="red",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=3,lty=3)
> plot(x=speed, y=dist, type="l", col="black",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=4,lty=4)
> savePlot(filename="quatro_figuras_juntas", type="eps")
```

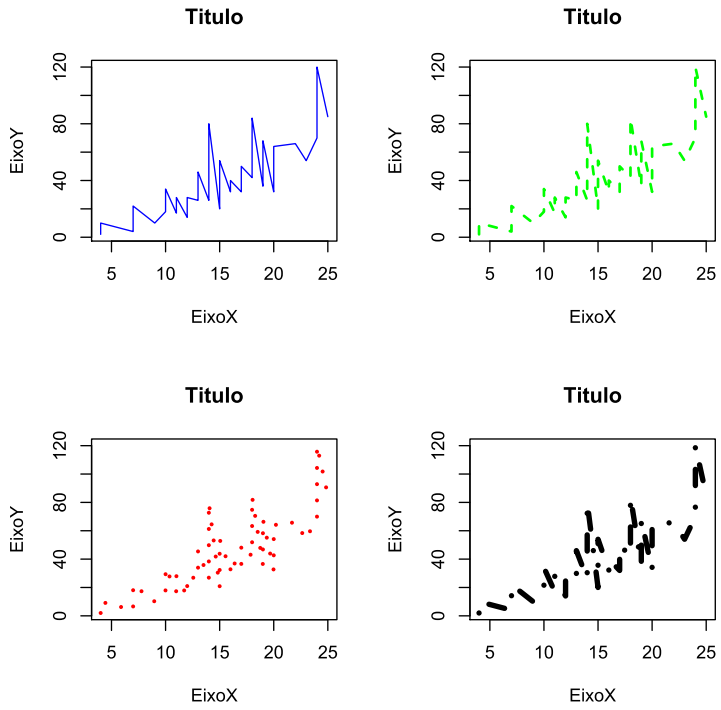

Figura 6.2: Quatro figuras na mesma janela.

Usando comando `par()`

```
> X11()
> par(ask="TRUE")
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=1,lty=1)
> savePlot(filename="fig1", type="eps")
```

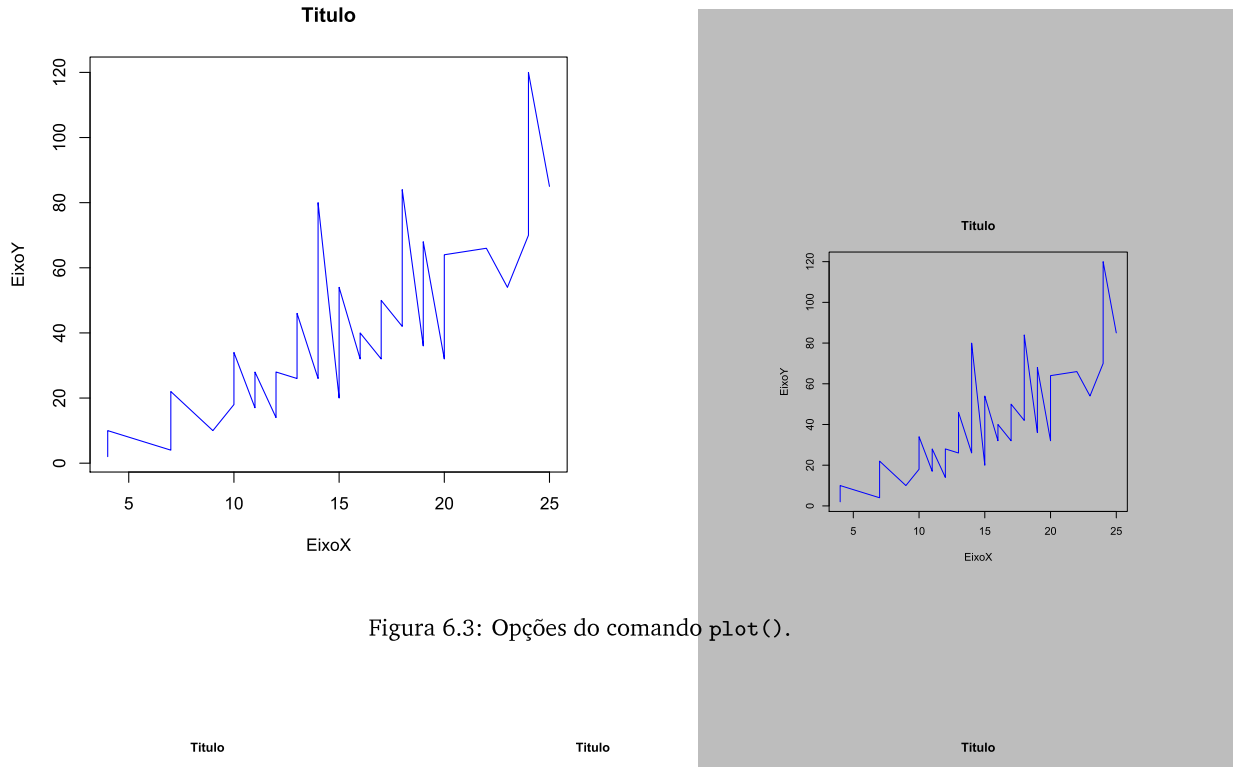

Figura 6.3: Opções do comando `plot()`.

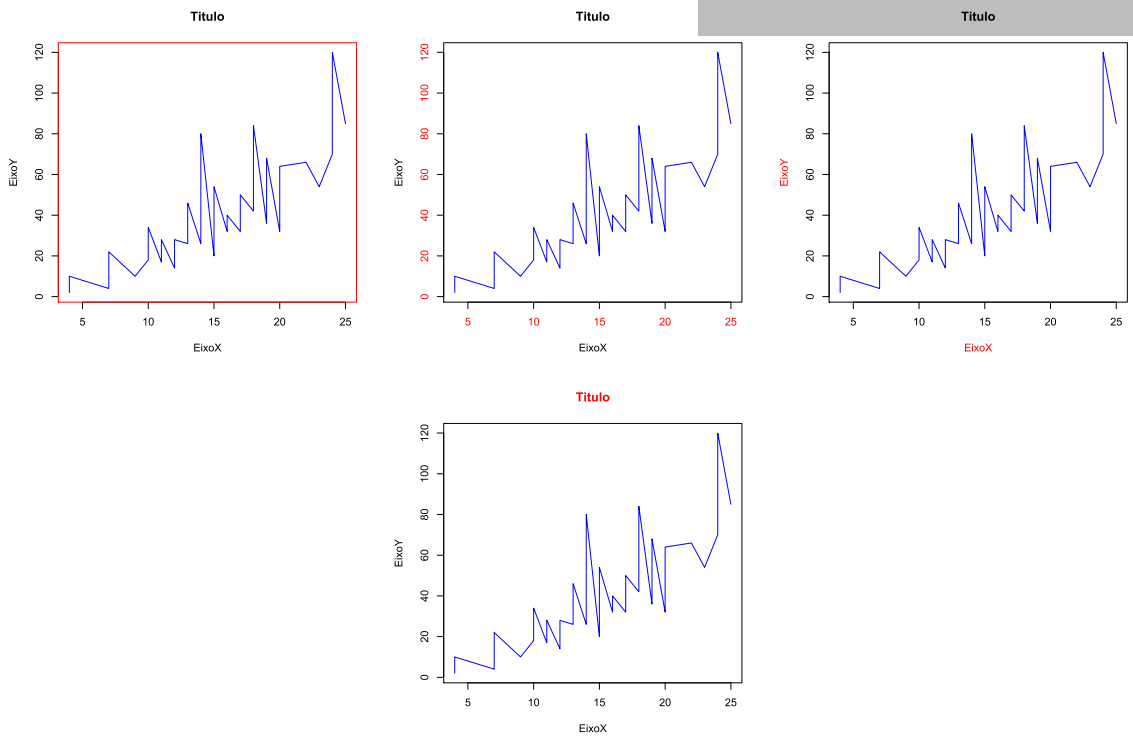

Figura 6.4: Opções do comando `par()`.

```
> X11()
> par(bg="gray")
> plot(x=speed,y=dist,type="l",col="blue",main="Titulo",xlab="EixoX",ylab="EixoY",
lwd=1,lty=1)
> savePlot(filename="fig2", type="eps")
```

```
> X11()
> par(col="red")
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=1,lty=1)
> savePlot(filename="fig3", type="eps")

> X11()
> par(col.axis="red")
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=1,lty=1)
> savePlot(filename="fig4", type="eps")
```

```
> X11()
> par(col.lab="red")
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=1,lty=1)
> savePlot(filename="fig5", type="eps")

> X11()
> par(col.main="red")
> plot(x=speed, y=dist, type="l", col="blue",main="Titulo",xlab="EixoX", ylab="EixoY",
lwd=1,lty=1)
> savePlot(filename="fig6", type="eps")
```

6.1.2 lines(), points()

```
> X11()
> par(mfrow=c(2,2))
> plot(x=speed, y=dist, type="l", col="blue",lwd=1,lty=1)
> points(x=speed-1, y=dist+10, col="red", lwd=2)
> plot(x=speed, y=dist, type="l", col="blue",lwd=1,lty=1)
> points(x=speed-1, y=dist+10, type="p", col="red", lwd=2,pch=4)
> plot(x=speed, y=dist, type="l", col="blue",lwd=1,lty=1)
> lines(x=speed-2, y=dist+3, col="red", lwd=2)
> plot(x=speed, y=dist, type="l", col="blue",lwd=1,lty=1)
> ablines(x=speed-2, y=dist+3, col="red", lwd=2)
> savePlot(filename="comando_lines", type="eps")
```

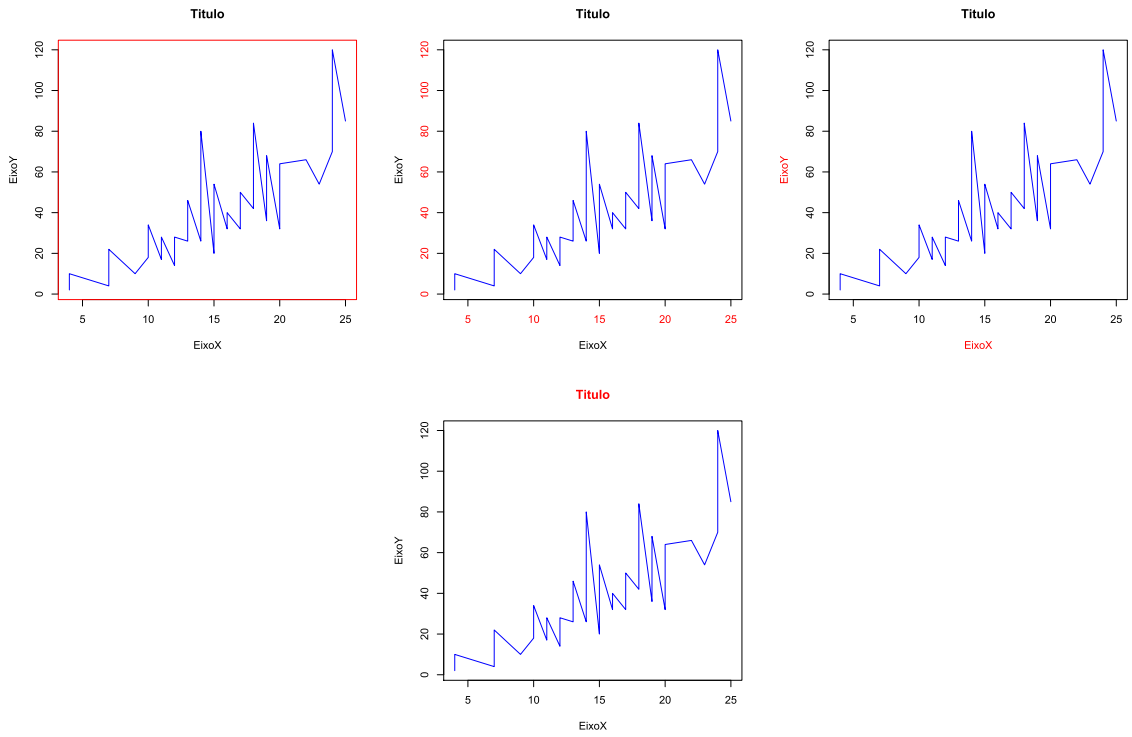

Figura 6.5: Opções do comando `par()`?.

6.1.3 `abline()`

```
> ajuste<-lm(formula=dist~speed)
> (intercepto  <- coef(ajuste)[1])
> (coef.angular<- coef(ajuste)[2])
```

```
> X11()
> par(mfrow=c(2,2))
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=2)
> abline(a=intercepto, b=coef.angular, col="red", lwd=2,lty=4)
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=3)
> abline(h=60, col="black", lwd=2,lty=4)
```

```
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5)
> abline(v=15, col="green", lwd=2,lty=4)
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5)
> abline(v=15, col="green", lwd=2,lty=4)
> savePlot(filename="comando_abline", type="eps")
```

6.1.4 `legend()`, `axis()`, `locator()`, `identify()`

```
> par(mfrow=c(2,2))
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5)
> identify(x=speed, y=dist, labels=c("min", "max"),n=2)
```

```
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5)
> legend( locator(n=1), c("Série temporal"), lty=5, lwd=1, col="blue")

> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5)
> legend(x=5, y = 120, legend=c("Série Temporal"), col="blue",lty=5, lwd=1)

> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5)
> legend(x="topleft",legend=c("Série Temporal"), col="blue",lty=5, lwd=1)
> savePlot(filename="comando_locator", type="eps")
```

```
> par(mfrow=c(1,2))
> plot(x=speed, y=dist, type="l", col="blue",lwd=1, lty=5, axes=F)
> axis(side=1,at=seq(0,25,5))
> axis(side=2,at=seq(0,120,20))

> plot(x=speed, y=dist, type="l", col="red",lwd=2, lty=2, axes=F)
> axis(side=1)
> axis(side=2)
> savePlot(filename="comando_axis", type="eps")
```

6.2 barplot()

```
> idade<- c(20,17,18,25,18)
> sexo<- c("f","m","f","f","m")
> nome<- c("maria", "pedro", "taciana", "renata","edson")
> altura<- c(178, 190, 152,169,187)
> (dados<- data.frame(idade,sexo,nome,altura))
> attach(dados)
```

6.3 hist()

```
> hist(x=idade,main="Hist. de idade",xlab = "idade",ylab="Freq. abs.")
> savePlot(filename="histograma", type="eps")
```

6.4 barplot()

```
> barplot(table(sexo,idade),col=c("gray","white"),legend.text = c("Femenino", "Masculino"),
> args.legend = list(x = "topright"))
> savePlot(filename="barplot", type="eps")
```

6.5 dotchart()

```
> dotchart(idade,labels=nome,xlim=c(16,26),xlab="idade")
> savePlot(filename="dotchart", type="eps")
```

6.6 pie()

```
> cervejas <- c(0.25, 0.50, 0.10,0.15)
> names(cervejas) <- c("Skol", "Bohemia","Antartica", "Itaipava")
> pie(cervejas,col=c("blue","red", "green","white"),
main="Preferências de marcas de cervejas")
> savePlot(filename="pie", type="eps")
```

6.7 boxplot()

```
> boxplot(formula=idade~sexo, names=c("Femenino", "Masculino"),ylab="Idade",xlab="Sexo")
> savePlot(filename="boxplot", type="eps")
```

6.8 matplot()

```
> (id<- sort(rep(1:3,3)))
> (semana<- sort(rep(1:3,3)))
> (medicao<-c(23,24,27,  31,22,25, 12,15,17 ))
> (data.matplot<- data.frame(id,semana,medicao))
> matplot(1:3,matrix(data.matplot[,3],3,3),type="b",xlab="Semanas", ylab="Medição",axes=F,
> col=1:3,lty=2:4,lwd=2,ylim=c(10,35),pch=1:3)
> title("Gráfico de Perfis")
> axis(1,at=1:3,labels=c("Semana 1", "Semana 2", "Semana 3"))
> axis(2,lty=2)
> legend(locator(1),legend=c("Ind. 1","Ind. 2","Ind. 3"),col=1:3,lty=2:4,lwd=2,pch=1:3)
> savePlot(filename="matplot", type="eps")
```

6.9 curve()

```
> par(mfrow=c(2,2))
> curve(x^2, from=-4, to=4, type="l",col="red",xlab="x", ylab="f(x)")
> curve((1/sqrt(2*pi))*exp(-0.5*(x^2)), from=-4, to=4, type="l",col="blue",xlab="x",
ylab="f(x)")
> curve(dnorm(x,mean=0,sd=1), from=-4, to=4, type="l",col="black",xlab="x", ylab="f(x)")
> curve(abs(x), from=-4, to=4, type="l",col="green",xlab="x", ylab="f(x)")
> savePlot(filename="curve_plot", type="eps")
```

6.10 coplot()

```
> set.seed(344)
> peso<- runif(22,30,40)
> altura<- runif(22,155,190)
> sexo<- rep(c("F","M"),11)
> coplot(formula=peso~altura|sexo,lwd=2,col="blue",pch=4, xlab="altura",ylab="peso")
> savePlot(filename="coplot", type="eps")
```

6.11 image()

```
> rm(list=ls(all=TRUE))

> require(grDevices) # para colores

> x <- 10*(1:nrow(volcano))
> y <- 10*(1:ncol(volcano))

> image(x, y, volcano, col = rainbow(100), axes = FALSE)
> savePlot(filename="image_comando", type="eps")
```

6.12 contour()

```
> contour(x, y, volcano, levels = seq(90, 200, by = 5),
> add = TRUE, col = "peru")
> axis(1, at = seq(100, 800, by = 100))
> axis(2, at = seq(100, 600, by = 100))
> box()
> title(main = "Mapa de Piracicaba", font.main = 4)
> savePlot(filename="contour_comando", type="eps")
```

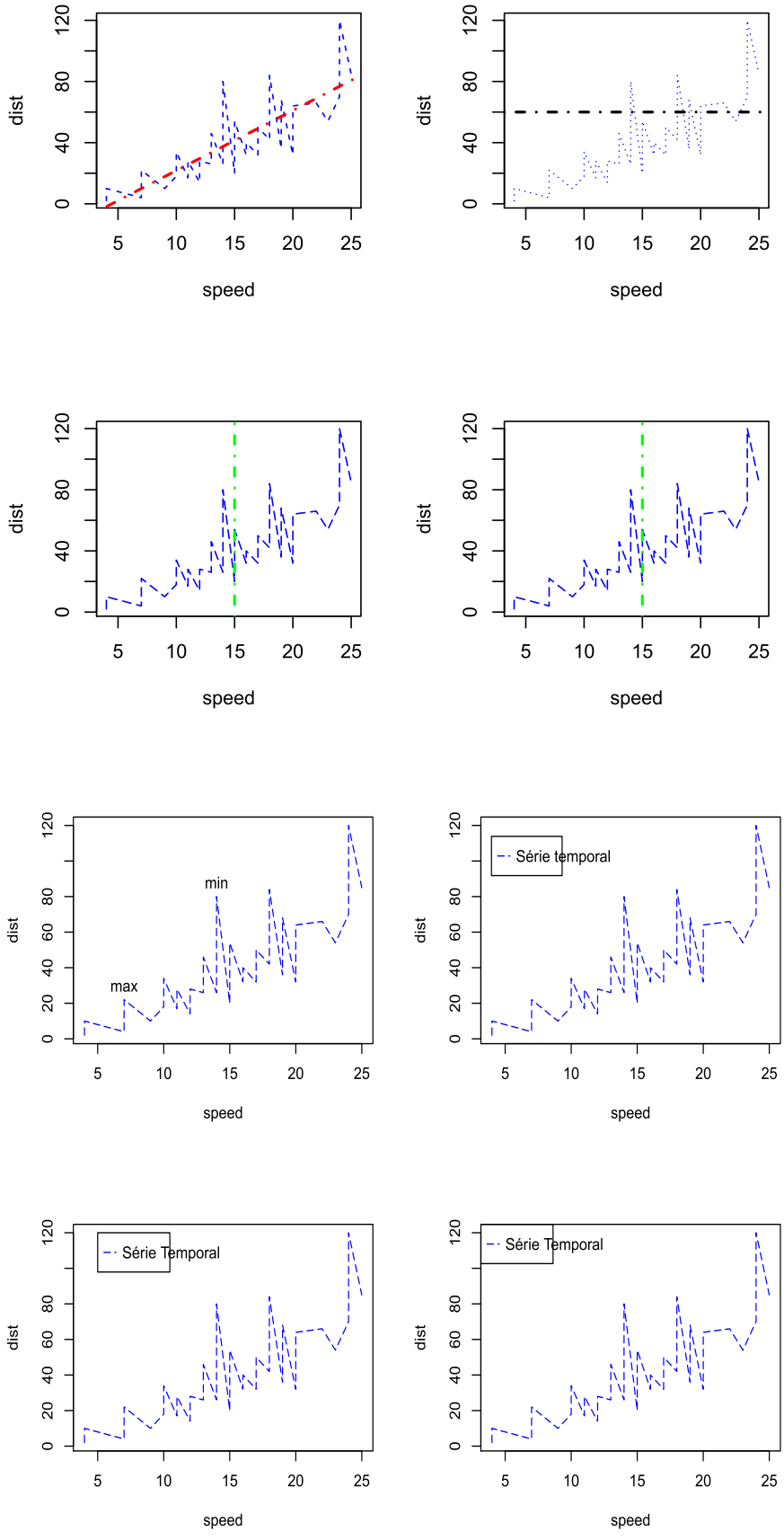

Figura 6.6: abline() e lines().

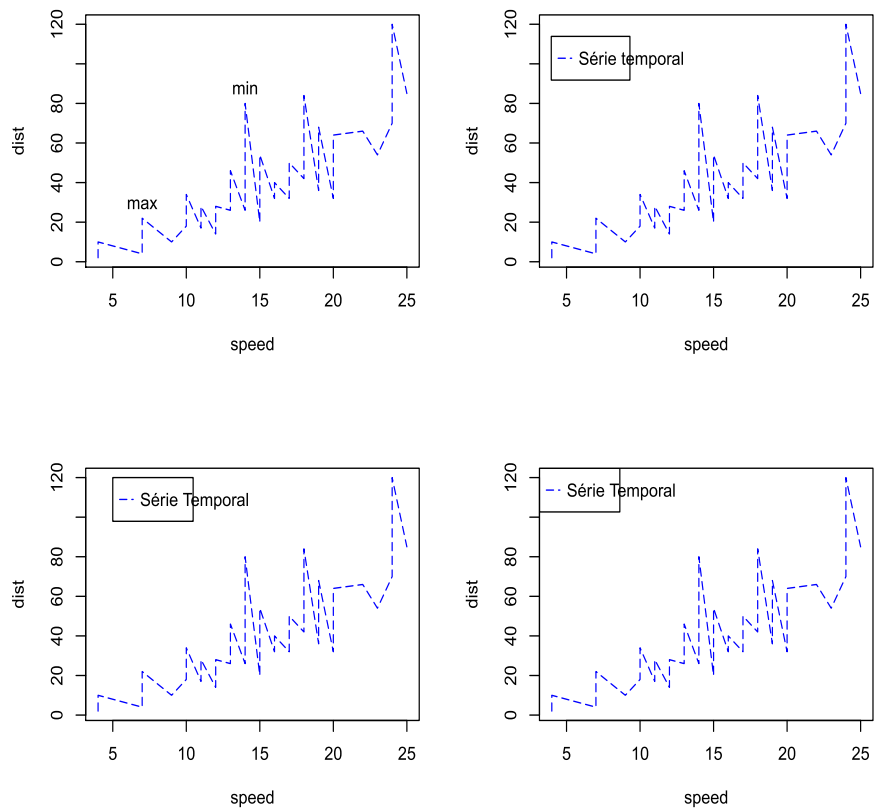

Figura 6.7: locator().

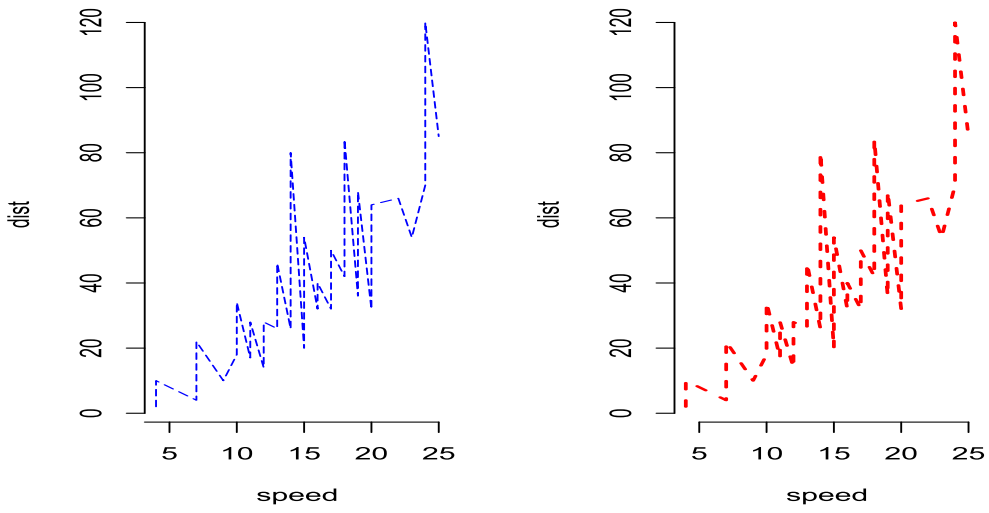

Figura 6.8: axis().

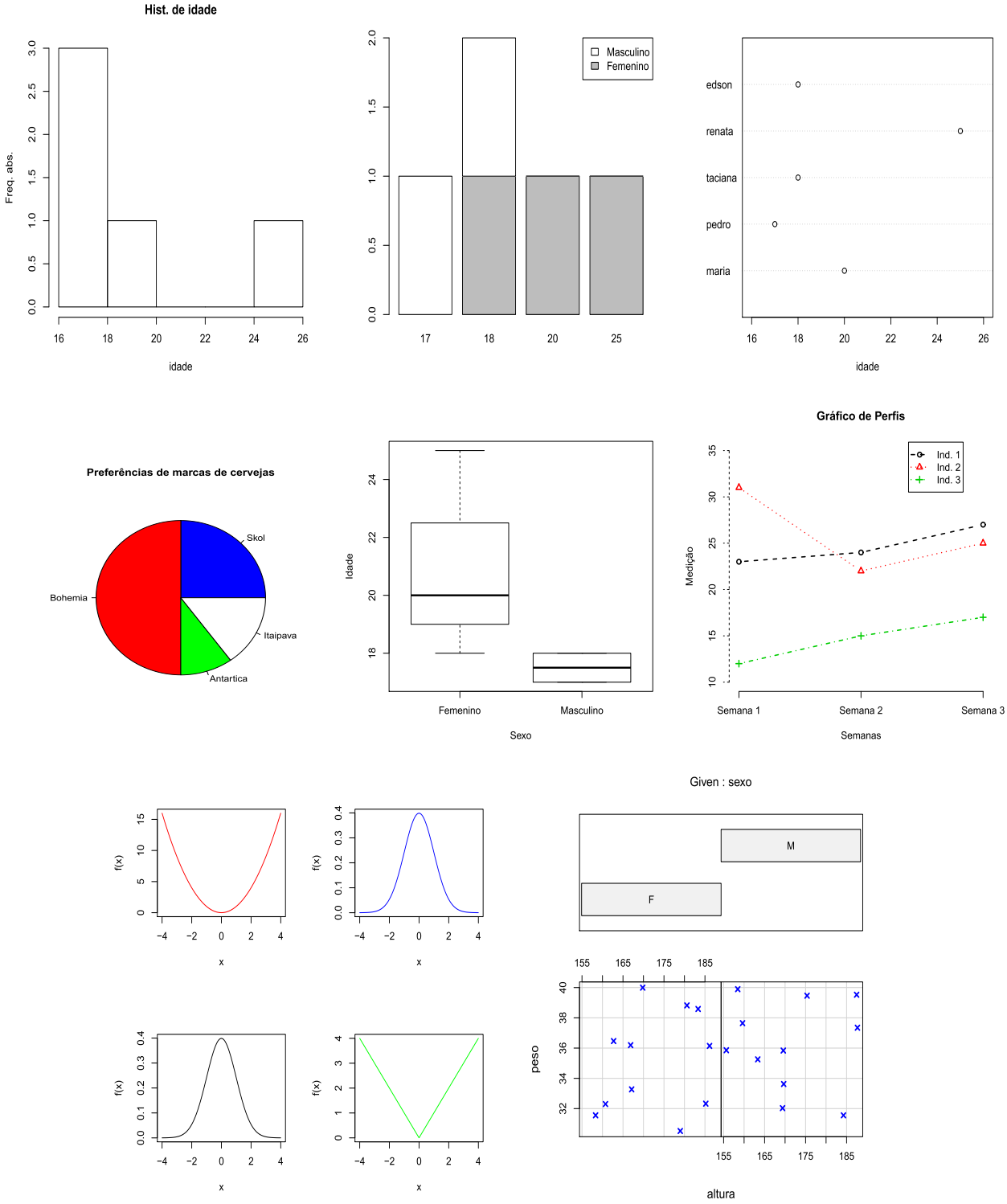

Figura 6.9: Opções de gráficos com R.

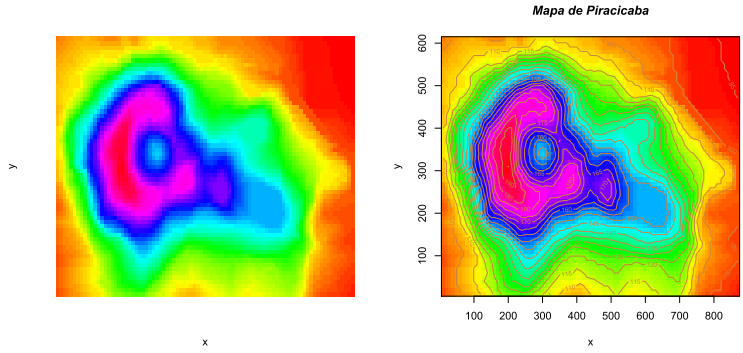

Figura 6.10: `image()` e `contour()`.

Capítulo 7

Tabelas

7.1 table()

```
>rm(list=ls(all=TRUE))

>idade<- c(23,34,34,18,19)
>sexo<- c("f","m","m","f","f")
>dados<- data.frame(idade,sexo)
>attach(dados)
```

```
>table(sexo,idade)

    idade
sexo 18 19 23 34
   f  1  1  1  0
   m  0  0  0  2
```

```
> set.seed(23)
> age<- runif(100)*50

> table(cut(age,c(0,10,20,30,40,50)))

 (0,10] (10,20] (20,30] (30,40] (40,50]
     17      17      20      34      12
```

7.2 prop.table()

```
> prop.table(x=table(sexo,idade),margin=1)
> prop.table(x=table(sexo,idade),margin=2)
```

7.3 margin.table()

```
> margin.table(x=table(sexo,idade),margin=1)
> margin.table(x=table(sexo,idade),margin=2)
```

Capítulo 8

Criando funções

8.1 Estrutura de uma função

```
Nome<- function(Args,...) {
comandos
return(resultado)
}
```

8.2 Criando uma função para calcular a média

8.2.1 Versão 1

```
> rm(list=ls(all=TRUE))
> media.v1 <- function(x){
+ n<- length(x);media<- sum(x)/n
+ resposta<- list(X=x,N=n,SOMA=sum(x),MEDIA=media)
+ return(resposta)
+ }
```

```
> media.v1(x=c(3,4,5))
$X
[1] 3 4 5

$N
[1] 3

$SOMA
[1] 12

$MEDIA
[1] 4
```

```
> media.v1(x=c(3,4,NA))
$X
[1]  3  4 NA

$N
[1] 3

$SOMA
[1] NA

$MEDIA
[1] NA
```

```
> media.v1(x=c(3,4,"dd"))
Error in sum(x) : invalid 'type' (character) of argument
```

8.2.2 Versão 2

```
> media.v2 <- function(x){
+ if (!is.numeric(x)) stop("x deve ser numerico!")
+ n<- length(x); media<- sum(x)/n
+ resposta<- list(X=x,N=n,SOMA=sum(x),MEDIA=media)
+ return(resposta)
+ }
```

```
> media.v2(x=c(3,4,5))
$X
[1] 3 4 5

$N
[1] 3

$SOMA
[1] 12

$MEDIA
[1] 4
```

```
> media.v2(x=c(3,4,NA))
$X
[1]  3  4 NA
$N
[1] 3
$SOMA
[1] NA
$MEDIA
[1] NA
```

```
> media.v2(x=c(3,4,"dd"))
Error in media.v2(x = c(3, 4, "dd")) : x deve ser numerico!
```

8.2.3 Versão 3

```
> media.v3 <- function(x){
+
+ if (!is.numeric(x)) stop("x deve ser numerico!")
+ if (any(is.na(x)))  x<-na.omit(x)
+
+ n<- length(x)
+ media<- sum(x)/n
+ resposta<- list(X=x,N=n,SOMA=sum(x),MEDIA=media)
+ return(resposta)
+ }
```

```
> media.v3(x=c(3,4,5))
$X
[1] 3 4 5

$N
[1] 3

$SOMA
[1] 12

$MEDIA
[1] 4
```

```
> media.v3(x=c(3,4,NA))
$X
[1] 3 4
attr(,"na.action")
[1] 3
attr(,"class")
[1] "omit"

$N
[1] 2

$SOMA
[1] 7

$MEDIA
[1] 3.5
```

```
> media.v3(x=c(3,4,"dd"))
Error in media.v3(x = c(3, 4, "dd")) : x deve ser numerico!
```

```
> funcao.varias<- function(x){
+
+ respostas<- list(raiz=sqrt(x),
+ media=mean(x),
+ variancia=var(x),
+ clase.objeto=class(x))
+
+ return(respostas)
+ }
```

Aprendendo coisas de funcao.varias()

```
> aux<- funcao.varias(x=c(4,5,6))
> names(aux)
[1] "raiz"          "media"         "variancia"     "clase.objeto"
> aux
$raiz
[1] 2.000000 2.236068 2.449490

$media
[1] 5

$variancia
[1] 1

$clase.objeto
[1] "numeric"
```

```
> aux$raiz
[1] 2.000000 2.236068 2.449490
> aux[1]
$raiz
[1] 2.000000 2.236068 2.449490

> aux[["raiz"]]
[1] 2.000000 2.236068 2.449490

> aux$raiz[2]
[1] 2.236068

> aux[[1]][2]
[1] 2.236068

> aux[["raiz"]][2]
[1] 2.236068

> aux$clase.objeto
[1] "numeric"
```

8.3 Uso de loops

8.3.1 for()

```
> for (i in 1:3){
+ cat("Oi\n")
+ }
Oi
Oi
Oi
```

8.3.2 while()

```
> funcao.while<- function(n){
+ while(n<3){
+ cat("Oi\n")
+ n<-n+1
+ }
+ }
> funcao.while(n=10)

> funcao.while(n=0)
Oi
Oi
Oi
```

8.3.3 if() e repeat()

```
> funcao.repeat<- function(n){
+ repeat{
+ if (n>=3) break
+ cat("Oi\n")
+ n<- n+1
+ }
+ }
```

```
> funcao.repeat(n=0)
Oi
Oi
Oi
```

```
> funcao.repeat(n=1)
Oi
Oi
```

```
> funcao.repeat(n=2)
Oi
```

```
> funcao.repeat(n=3)
```

8.3.4 if() e ifelse()

```
> meu.aleatorio<- function(n,distribuicao,shape){
+ if(distribuicao=="gamma") rgamma(n,shape) else
+ if(distribuicao=="normal") rnorm(n) else
+ if(distribuicao=="exponencial") rexp(n)
+ else stop("Não conheço essa distribuição")
+ }
```

```
> meu.aleatorio(n=10,distribuicao="gamlss", shape=2)
Error in meu.aleatorio(n = 10, distribuicao = "gamlss", shape = 2) :
  Não conheço essa distribuição
```

```
> meu.aleatorio(n=10,distribuicao="gamma", shape=2)
 [1] 1.6891046 0.5923445 4.2200586 2.5762123 0.1132899 3.0921128 1.5507007
 [8] 3.8247920 0.4851958 3.0634147
```

```
> ifelse(test=cars$speed<18, yes=sqrt(cars$speed), no=NA)
 [1] 2.000000 2.000000 2.645751 2.645751 2.828427 3.000000 3.162278 3.162278
 [9] 3.162278 3.316625 3.316625 3.464102 3.464102 3.464102 3.464102 3.605551
[17] 3.605551 3.605551 3.605551 3.741657 3.741657 3.741657 3.741657 3.872983
[25] 3.872983 3.872983 4.000000 4.000000 4.123106 4.123106 4.123106       NA
[33]       NA       NA       NA       NA       NA       NA       NA       NA
[41]       NA       NA       NA       NA       NA       NA       NA       NA
[49]       NA       NA
```

8.3.5 swith()

```
> funcao.switch<- function(x,opcao){
+
+ switch(opcao,
+ raiz=sqrt(x),
+ media=mean(x),
+ variancia=var(x),
+ stop("Deve escolher raiz, media ou variancia"))
+ }
```

```
> funcao.switch(x=c(4,5,6),opcao="raiz")
[1] 2.000000 2.236068 2.449490
```

```
> funcao.switch(x=c(4,5,6),opcao="media")
[1] 5
```

```
> funcao.switch(x=c(4,5,6),opcao="variancia")
[1] 1
```

```
> funcao.switch(x=c(4,5,6),opcao="mediana")
Error in funcao.switch(x = c(4, 5, 6), opcao = "mediana") :
  Deve escolher raiz, media ou variancia
```

8.3.6 menu()

```
> funcao.switch.menu<- function(x){
+
+ switch(EXPR=menu( choices = c("Raiz", "Media", "Variancia") ) ,
+             sqrt(x), mean(x),var(x))
+ }
```

```
> funcao.switch.menu(x=c(4,5,6))

1: Raiz
2: Media
3: Variancia

Selection: 1
[1] 2.000000 2.236068 2.449490
> funcao.switch.menu(x=c(4,5,6))

1: Raiz
2: Media
3: Variancia

Selection: 2
[1] 5
> funcao.switch.menu(x=c(4,5,6))

1: Raiz
2: Media
3: Variancia

Selection: 3
[1] 1
```

8.3.7 readline()

```
> funcao.readline<- function(x){
+ cat("Digite sua opção (raiz, media ou variancia)\n")
+ opcao<- readline()
+ switch(opcao,
+ raiz=sqrt(x),
+ media=mean(x),
+ variancia=var(x),
+ stop("Deve escolher raiz, media ou variancia"))
+ }
```

```
> funcao.readline(x=c(4,5,6))
Digite sua opção (raiz, media ou variancia)
media
[1] 5
```

```
> funcao.readline(x=c(4,5,6))
Digite sua opção (raiz, media ou variancia)
raiz
[1] 2.000000 2.236068 2.449490
```

```
> funcao.readline(x=c(4,5,6))
Digite sua opção (raiz, media ou variancia)
variancia
[1] 1
```

Função binária

```
> minha.binaria.numerica<- function(x){
+
+ n<- length(x)
+
+ for(i in 1:n){
+ if(x[i]>0)
+ x[i]<- 1
+
+ else if(x[i]<=0)
+ x[i]<- 0
+ }
+ return(x)
+ }
> minha.binaria.numerica(x=c(1,2,4,0,5,0,4,0))
[1] 1 1 1 0 1 0 1 0
```

Função binária caracteres

```
> minha.binaria.carateres<- function(x){
+
+ n<- length(x)
+
+ for(i in 1:n){
+ if(x[i]>0)
+ x[i]<- "F"
+
+ else if(x[i]<=0)
+ x[i]<- "M"
+ }
+ return(x)
+ }
> minha.binaria.carateres(x=c(1,2,4,0,5,0,4,0))
[1] "F" "F" "F" "M" "F" "M" "F" "M"
```

Função binária caracteres (opção)

```
> minha.binaria.carateres.opcao<- function(x){
+ ifelse(test= x > 0, yes="F",ifelse(test=x<=0,yes="M",no=0))
+ }
> minha.binaria.carateres.opcao(x=c(3,6,7,0,9,0))
[1] "F" "F" "F" "M" "F" "M"
```

8.3.8 repeat() e break()

Newton Raphson usando repeat()

```
> #Exemplo distribuição Weibull(teta, lamda=2)
> #Fonte: Dobson Capítulo 4
> #================================================
> NR<- function(dados,num.iter,teta.ini){
+ y<- dados
+ n    <- length(y)
+  teta.velho<- teta.ini
+ U.velho   <- -(2*n)/teta.velho+ (2*sum(y^2))/teta.velho^3
+ J.velho<- (2*n)/teta.velho^2-(6*sum(y^2))/teta.velho^4
+ j<- 1
+
+ repeat{
+ teta.novo<- teta.velho - U.velho/J.velho
+ U.novo   <- -(2*n)/(teta.novo) + (2*sum(y^2))/teta.novo^3
+ J.novo<- (2*n)/teta.novo^2-(6*sum(y^2))/teta.novo^4
+
+ if(j>=num.iter)break
+ teta.velho<- teta.novo
+ U.velho<- U.novo
+ J.velho<- J.novo
+ cat("Iteração=",j, "teta=", teta.novo,"\n")
+ j<- j+1
+ }
+ }
```

```
> tempos<- c(1051, 4921, 7886, 10861, 13520,
+ 1337, 5445, 8108, 11026, 13670,1389, 5620, 8546, 11214, 14110,
+ 1921, 5817, 8666, 11362, 14496,1942, 5905, 8831, 11604, 15395,
+ 2322, 5956, 9106, 11608, 16179,3629, 6068, 9711, 11745, 17092,
+ 4006, 6121, 9806, 11762, 17568,4012, 6473, 10205, 11895, 17568,
+ 4063, 7501, 10396, 12044)
```

```
> NR(dados=tempos,num.iter=10, teta.ini=mean(tempos))
Iteração= 1 teta= 9633.777
Iteração= 2 teta= 9875.898
Iteração= 3 teta= 9892.11
Iteração= 4 teta= 9892.177
Iteração= 5 teta= 9892.177
Iteração= 6 teta= 9892.177
Iteração= 7 teta= 9892.177
Iteração= 8 teta= 9892.177
Iteração= 9 teta= 9892.177
```

Capítulo 9

Conhecimentos sobre pacotes do R

9.1 Instalando um novo pacote

```
1 install.packages(bs, dependencies="T")
```

9.2 Carregando um pacote instalado

```
1 library(bs)
```

9.3 Atualizando pacotes

```
1 update.packages("bs")
```

9.4 Estudando o código fonte

http://cran.r-project.org/web/packages/gbs

Capítulo 10

Considerações Finais

10.1 Interface com outros softwares

1. C
2. Fortran
3. Python
4. Excel
5. SAS?
6. Matlab?
7. Stata
8. Winbugs

10.2 Tópicos Futuros

1. Rstudio
2. R commander. install.package(Rcmdr, dep="T")
3. Tinn-R.
4. Pacote Lattice.
5. Curso de R para não estatisticos.
6. Curso de R para estatisticos.
7. Criação de um novo pacote.
8. Action!. http://portalaction.com.br/content/sobre-o-action

10.3 Sites de interesse

10.3.1 Português

1. http://leg.est.ufpr.br/doku.php/software:rbr
2. http://leg.est.ufpr.br/doku.php/ridiculas

10.3.2 Espanhol

1. http://www.luiscayuela.blogspot.com.br/
2. http://curso-r-uah2010.wikispaces.com/

10.3.3 Inglês

1. http://www.r-project.org/
2. http://r-forge.r-project.org/
3. http://cran.r-project.org/web/views/
4. http://www.r-bloggers.com/
5. http://www.statmethods.net/
6. https://sites.google.com/site/marcosfs2006/material_r
7. http://www.r-tutor.com/

10.4 Leituras adicionais

Aos interessados em ler uma reportagem publicada, na internet, pelo jornal americano The New York Times o dia 7 de Janeiro de 2009, accesar o site a seguir http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=1

10.5 Referências

Existem muitos livros e materiais na internet para aprender R. Os livros usados para este minicurso são citados a seguir

1. Braun e Murdoch (2007)
2. Chambers (2008)
3. Dalgaard (2008)
4. Gentleman (2009)
5. Spector (2008)
6. Shahbaba (2012)
7. Statistical Sciences, Inc. (1993)

Apêndice A

Álgebra de matrizes

A.1 Operações com matrizes

```
> rm(list=ls(all=TRUE))

> data(stackloss)

> head(stackloss)
  Air.Flow Water.Temp Acid.Conc. stack.loss
1       80         27         89         42
2       80         27         88         37
3       75         25         90         37
4       62         24         87         28
5       62         22         87         18
6       62         23         87         18
```

```
> X<- as.matrix(data.frame(intercepto=rep(1,21),stackloss[,1:3]),21,4)

> dim(X)
[1] 21  4

> Y<- as.matrix(stackloss[,4])

> dim(Y)
[1] 21  1
```

```
> class(X)
[1] "matrix"

> class(Y)
[1] "matrix"
```

```
> A<-X[1:10,2]

> B<-Y[1:10]
```

```
> (MATRIZ<- data.frame(A=A, B=B, soma=A+B, diferenca=A-B, produto=A*B))
    A  B soma diferenca produto
1  80 42  122        38    3360
2  80 37  117        43    2960
3  75 37  112        38    2775
4  62 28   90        34    1736
5  62 18   80        44    1116
6  62 18   80        44    1116
7  62 19   81        43    1178
8  62 20   82        42    1240
9  58 15   73        43     870
10 58 14   72        44     812

> A%*%B#sum(A*B)
      [,1]
[1,] 17163

> t(A)%*%B
      [,1]
[1,] 17163

> A%*%t(B)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,] 3360 2960 2960 2240 1440 1440 1520 1600 1200  1120
 [2,] 3360 2960 2960 2240 1440 1440 1520 1600 1200  1120
 [3,] 3150 2775 2775 2100 1350 1350 1425 1500 1125  1050
 [4,] 2604 2294 2294 1736 1116 1116 1178 1240  930   868
 [5,] 2604 2294 2294 1736 1116 1116 1178 1240  930   868
 [6,] 2604 2294 2294 1736 1116 1116 1178 1240  930   868
 [7,] 2604 2294 2294 1736 1116 1116 1178 1240  930   868
 [8,] 2604 2294 2294 1736 1116 1116 1178 1240  930   868
 [9,] 2436 2146 2146 1624 1044 1044 1102 1160  870   812
[10,] 2436 2146 2146 1624 1044 1044 1102 1160  870   812
```

```
> (beta<- solve(t(X)%*%X)%*%t(X)%*%Y)
                  [,1]
intercepto -39.9196744
Air.Flow     0.7156402
Water.Temp   1.2952861
Acid.Conc.  -0.1521225

> as.matrix(lm(Y~X-1)$coefficients)
                   [,1]
Xintercepto -39.9196744
XAir.Flow     0.7156402
XWater.Temp   1.2952861
XAcid.Conc.  -0.1521225
```

A.2 diag()

```
> diag(t(X)%*%X)
intercepto   Air.Flow Water.Temp Acid.Conc.
        21      78365       9545     156924
```

```
> diag(diag(t(X)%*%X))
     [,1]  [,2] [,3]   [,4]
[1,]   21     0    0      0
[2,]    0 78365    0      0
[3,]    0     0 9545      0
[4,]    0     0    0 156924
```

```
> (COV.X<- cov(X[,-1]))
           Air.Flow Water.Temp Acid.Conc.
Air.Flow   84.05714  22.657143  24.571429
Water.Temp 22.65714   9.990476   6.621429
Acid.Conc. 24.57143   6.621429  28.714286

> (COR.X<- cor(X[,-1]))
            Air.Flow Water.Temp Acid.Conc.
Air.Flow   1.0000000  0.7818523  0.5001429
Water.Temp 0.7818523  1.0000000  0.3909395
Acid.Conc. 0.5001429  0.3909395  1.0000000
```

A.3 det()

```
> det(COV.X)
[1] 7028.451

> apply(X[,-1], 2,mean)
  Air.Flow Water.Temp Acid.Conc.
  60.42857   21.09524   86.28571

> apply(t(X[,-1]), 1,mean)
  Air.Flow Water.Temp Acid.Conc.
  60.42857   21.09524   86.28571
```

A.4 cov() e cor()

```
> COV.X==t(COV.X) #COV.X é uma matriz simétrica?
           Air.Flow Water.Temp Acid.Conc.
Air.Flow       TRUE       TRUE       TRUE
Water.Temp     TRUE       TRUE       TRUE
Acid.Conc.     TRUE       TRUE       TRUE

> eigen(x=COV.X, symmetric=TRUE)
$values
[1] 99.57609 19.58114  3.60468

$vectors
          [,1]       [,2]        [,3]
[1,] 0.9061876 -0.3206905  0.27564787
[2,] 0.2541644 -0.1079241 -0.96112062
[3,] 0.3379713  0.9410154 -0.01629139
```

```
> COR.X==t(COR.X) #COR.X é uma matriz simétrica?
           Air.Flow Water.Temp Acid.Conc.
Air.Flow       TRUE       TRUE       TRUE
Water.Temp     TRUE       TRUE       TRUE
Acid.Conc.     TRUE       TRUE       TRUE

> eigen(x=COR.X, symmetric=TRUE)
$values
[1] 2.1331735 0.6597053 0.2071212

$vectors
          [,1]       [,2]       [,3]
[1,] 0.6313564 -0.2356545  0.7388207
[2,] 0.6035915 -0.4488348 -0.6589572
[3,] 0.4868947  0.8619828 -0.1411357
```

A.5 kronecker()

A.5.1 Produto kronecker de um escalar por uma matriz

```
> kronecker(X=1/2, Y=A)
 [1] 40.0 40.0 37.5 31.0 31.0 31.0 31.0 31.0 29.0 29.0
```

A.5.2 Produto kronecker de uma matriz 2×2 por uma 5 × 3

```
> (I2<- diag(x = 1, nrow=2, ncol=2)#matriz identidade 2x2
     [,1] [,2]
[1,]    1    0
[2,]    0    1

> (XX<- X[1:5,c("Air.Flow", "Water.Temp", "Acid.Conc.")])
     Air.Flow Water.Temp Acid.Conc.
[1,]       80         27         89
[2,]       80         27         88
[3,]       75         25         90
[4,]       62         24         87
[5,]       62         22         87

> kronecker(X=I2, Y=XX)
      [,1] [,2] [,3] [,4] [,5] [,6]
 [1,]   80   27   89    0    0    0
 [2,]   80   27   88    0    0    0
 [3,]   75   25   90    0    0    0
 [4,]   62   24   87    0    0    0
 [5,]   62   22   87    0    0    0
 [6,]    0    0    0   80   27   89
 [7,]    0    0    0   80   27   88
 [8,]    0    0    0   75   25   90
 [9,]    0    0    0   62   24   87
[10,]    0    0    0   62   22   87
```

A.6 svd()

Descomposição em valores singulares de uma matriz quadrada

```
> svd(t(X)%*%X)
$d
[1] 2.438978e+05 8.833823e+02 7.373855e+01 7.432206e-02

$u
             [,1]         [,2]        [,3]         [,4]
[1,] -0.009250513  0.007776382 -0.00443160 -0.999917155
[2,] -0.564869033 -0.780013494  0.26925413 -0.002033749
[3,] -0.196787568 -0.189555193 -0.96193671  0.004609633
[4,] -0.801318948  0.596311417  0.04647944  0.011844760

$v
             [,1]         [,2]        [,3]         [,4]
[1,] -0.009250513  0.007776382 -0.00443160 -0.999917155
[2,] -0.564869033 -0.780013494  0.26925413 -0.002033749
[3,] -0.196787568 -0.189555193 -0.96193671  0.004609633
[4,] -0.801318948  0.596311417  0.04647944  0.011844760
```

Apêndice B

Pequeno resumo dos capítulos anteriores

Tabela B.1: Operadores no R

Operador	Uso
$	seleciona componente
[[[	subindices, elementos
:	operador de sequência
%%	modulo
%/%	divisão inteira
%*%	multiplicação de matrices
< > <= >= == !=	comparação
!	não

Tabela B.2: Operadores lógicos e de comparação

Operador	Explicação	Operador	Explicação
==	Igual a	!=	não é igual a
>	Maior do que	<	Menor do que
>=	Maior igual do que	<=	Menor igual do que
&	Vetorizado e	|	Vetorizado ou
&&	control e	||	control ou
!	não		

Tabela B.3: Funções para testar e converter objetos

Tipo	Testando	Converte a	Tipo	Testando	Converte a
array	is.array	as.array	character	is.character	as.character
complexo	is.complexo	as.complexo	data.frame	is.data.frame	as.data.frame
double	is.double	as.double	factor	is.factor	as.factor
integer	is.integer	as.integer	list	is.list	as.list
logical	is.logical	as.logical	matrix	is.matrix	as.matrix
NA	is.NA	-	null	is.null	as.null
numeric	is.numeric	as.numeric	ts	is.ts	as.ts
vector	is.vector	as.vector			

Tabela B.4: Comandos

ls() or objects()	Lista objetos no espaço de trabalho
rm(object)	Deleta objetos
search()	Caminho de pesquisa

Tabela B.5: Atribuição

$<-$	Atribui valores a uma variável
$->$	Atribui no lado direito
$<<-$	Atribuição Global (funções)

Tabela B.6: Operadores aritméticos

+	Soma
-	Subtração
*	Multiplicação
/	Divisão
$\wedge$	Potenciação

Tabela B.7: Geração de vetores

numeric(25)	25 vezes zero
character(25)	25 vezes " "
logical(25)	25 vezes FALSE
seq(-4,4,0.1)	sequência: -4.0 -3.9 3.8 . . . 3.9 4.0
1:10	igual do que seq(1,10,1)
c(5,7,9,13,1:5)	Concatenação: 5 7 9 13 1 2 3 4 5
rep(1,10)	1 1 1 1 1 1 1 1 1 1
gl(3,2,12)	Fator com 3 níveis, repete cada nivel em blocos de 2, até o tamanho 12 (i.e., 1 1 2 2 3 3 1 1 2 2 3 3)

Tabela B.8: Indices

x[1]	Primeiro elemento
x[1:5]	Subvetor que contem os 5 primeiros elementos
x[c(2,3,5,7,11)]	Seleciono elementos 2, 3, 5, 7 e 11
x[y<=30]	Seleção por uma expressão lógica
x[sex=="male"]	Seleção por variável fator
i <- c(2,3,5,7,11);x[i]	Seleção por variável numérica
l <- (y<=30); x[l]	Seleção por variável lógica

Tabela B.9: Matrizes e data frames

m[4,]	Quarta linha
m[,3]	Terceira coluna
dfr[dfr$var<=30,]	Selecionando elementos de data frame
subset(dfr,var<=30)	Mesma coisa, geralmente mais simples

Referências

Braun, W. J. e Murdoch, D. J. (2007). A First Course in Statistical Programming with R. *Cambridge University Press*.

Chambers, J. M. (2008). Software for data analysis programming with R. *New York: Springer*.

Dalgaard, P. (2008). Introductory Statistics with R. *New York : Springer. Segunda edição*.

Gentleman, R. (2009). R Programming for Bioinformatics. *Chapman & Hall/CRC Computer Science & Data Analysis*.

Shahbaba, B. (2012). Biostatistics with R: An Introduction to Statistics Through Biological Data. *Springer, New York*.

Spector, P. (2008). Data Manipulation with R. *Springer, New York*.

Statistical Sciences, Inc. (1993). S-Plus Programmer's Manual, version 3.1. *Seattle: Statistical Sciences, Inc*.