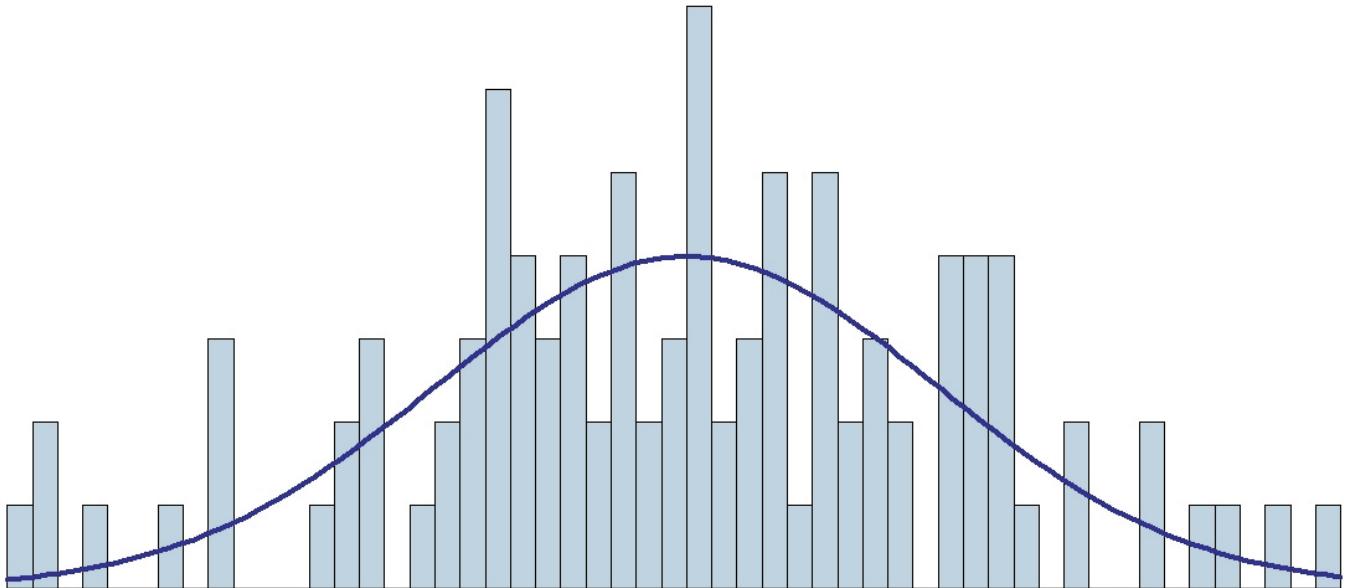




# Minicurso de Estatística Básica: Introdução ao software R



**Ministrantes:**

Bruno Fontana da Silva  
Jean Diniz  
Matias Américo Bortoluzzi

**Nome:** \_\_\_\_\_

<http://www.ufsm.br/pet-ee>

**Programa de Educação Tutorial - Engenharia Elétrica**  
**Universidade Federal de Santa Maria**  
**Santa Maria, Abril de 2009**

# Sumário

---

<b>1 Prefácio</b>	<b>5</b>
1.1 O Projeto R . . . . .	5
1.2 Programa de Educação Tutorial - Engenharia Elétrica (UFSM) . . . . .	6
<b>2 Introdução</b>	<b>7</b>
2.1 Apresentando o R . . . . .	7
2.2 Interface do R . . . . .	7
2.3 Sintaxe do R . . . . .	8
2.4 Tinn-R Editor . . . . .	8
2.5 Tipos de Dados . . . . .	9
2.6 Comandos Básicos . . . . .	10
2.6.1 Utilizando Ajuda . . . . .	11
2.6.2 Atribuição de Valores . . . . .	11
2.6.3 Comandos Auxiliares . . . . .	12
2.6.4 Operações matemáticas simples . . . . .	13
2.6.5 Funções matemáticas simples . . . . .	13
2.6.6 Números complexos . . . . .	14
2.7 Vetores e Matrizes . . . . .	15
2.7.1 Definição . . . . .	15
2.7.2 Declaração de vetores . . . . .	15
2.7.3 Arrays e Matrizes - Definição e Declaração . . . . .	16
2.7.4 Operações e funções com Matrizes . . . . .	17
2.8 Entrada de Arquivos Externos . . . . .	19
2.9 Arquivos provenientes da internet . . . . .	21
<b>3 Gráficos</b>	<b>22</b>
3.1 Introdução . . . . .	22
3.1.1 Comandos Básicos . . . . .	22
3.1.2 Criando Novas Janelas Gráficas e Salvando Gráficos . . . . .	24
3.1.3 Outras Funcionalidades . . . . .	25
3.2 Gráficos de Análise Descritiva . . . . .	27
3.2.1 Histograma . . . . .	27
3.2.2 Barplot . . . . .	29
3.2.3 Boxplot . . . . .	30
3.2.4 Gráfico de Ramo e Folhas . . . . .	32
3.2.5 Gráfico de Pizza . . . . .	33
<b>4 Estatística Descritiva</b>	<b>34</b>
4.1 Introdução . . . . .	34
4.2 Medidas de Posição . . . . .	34

4.2.1	Média Aritmética $\bar{X}$	34
4.2.2	Mediana "Md"	35
4.2.3	Moda "Mo"	36
4.2.4	Quartis (Q)	37
4.2.5	Percentis (P)	38
4.2.6	Decis (D)	39
4.3	Medidas de Dispersão	40
4.3.1	Introdução	40
4.3.2	Amplitude Total (A)	40
4.3.3	Variância ( $\sigma^2$ )	41
4.3.4	Desvio-padrão ( $\sigma$ )	42
4.3.5	Coeficiente de Variação (CV)	43
4.4	Exemplo Aplicado	43
<b>5</b>	<b>Probabilidade</b>	<b>46</b>
5.1	Introdução	46
5.2	Probabilidade - Definição	46
5.3	Axiomas da Probabilidade	49
<b>6</b>	<b>Variáveis Aleatórias</b>	<b>50</b>
6.1	Introdução	50
6.2	Variáveis Aleatórias Discretas	50
6.2.1	Introdução	50
6.2.2	Distribuição Binomial - $X \sim b(n,p)$	50
6.2.3	Distribuição De Poisson - $X \sim P(\lambda)$	53
6.3	Variáveis Aleatórias Contínuas	55
6.3.1	Introdução	55
6.3.2	Distribuição Normal ou Gaussiana - $X \sim N(\mu,\sigma^2)$	55
6.3.3	Distribuição de Weibull - $X \sim W(\delta,\beta)$	58
6.3.4	Distribuições no R	60
<b>7</b>	<b>Inferência Estatística</b>	<b>61</b>
7.1	Introdução	61
7.1.1	Hipóteses Unilaterais e Bilaterais	61
7.2	Testes de Hipótese - Uma amostra	62
7.2.1	Introdução	62
7.2.2	Teste para a Média	62
7.2.3	Teste para a Variância de uma população normal	64
7.2.4	Teste para uma Proporção Binomial	67
7.3	Testes de Hipótese - Duas amostras	68
7.3.1	Introdução	68
7.3.2	Teste para a média	68
7.3.3	Teste para as variâncias de duas populações normais	72
7.3.4	Teste para duas proporções	74
<b>8</b>	<b>Regressão e Correlação Linear Simples</b>	<b>77</b>
8.1	Introdução	77
8.2	Determinando a Equação Linear (Regressão)	78
8.3	Coeficiente de Correlação ( $r$ )	81

8.4	Coeficiente de Determinação ( $r^2$ ) . . . . .	82
8.5	Exemplo Aplicado . . . . .	82
<b>9</b>	<b>Programação em Linguagem R</b>	<b>86</b>
9.1	Introdução . . . . .	86
9.2	Interação com o Usuário . . . . .	86
9.3	Estruturas de Controle da Linguagem R . . . . .	87
9.3.1	InSTRUÇÕES Condicionais . . . . .	87
9.3.2	InSTRUÇÕES Iterativas . . . . .	89
<b>10</b>	<b>Referências Bibliográficas</b>	<b>95</b>

# 1

## Prefácio

---

### 1.1 O Projeto R

R é uma linguagem e ambiente para computação estatística e gráficos. Faz parte da filosofia do Projeto GNU e está disponível como Software Livre sob os termos da "Licença Pública Geral do GNU" da Fundação do Software Livre (*Free Software Foundation's GNU General Public License*) na forma de código fonte. Ele compila e roda sobre uma larga variedade de plataformas UNIX e sistemas similares (incluindo FreeBSB e Linux), Windows e MacOS.

R é uma série integrada de instalações de softwares para manipulação de dados, cálculo e exibição gráfica. Dentre outras coisas, possui:

- uma manipulação de dados eficaz e facilidade de armazenamento;
- uma série de operadores para cálculos com arranjos, especialmente matrizes;
- uma extensa, coerente e integrada coleção de ferramentas intermediárias para análise de dados;
- instalações gráficas para análises de dados e exibição tanto direta no computador quanto para cópia permanente (impressões);
- uma bem desenvolvida, simples e eficaz linguagem de programação (chamada 'S') a qual inclui condições, *loops*, funções recursivas definidas pelo usuário e instalações de entradas e saídas (de fato, a maioria das funções providas no sistema são propriamente escritas na linguagem S).

R é mais um veículo para novos métodos em desenvolvimento de análise de dados. Tem se desenvolvido rapidamente e tem sido estendido por uma extensa coleção de pacotes. Entretanto, muitos programas escritos em R são essencialmente passageiros, escritos para uma pequena parte de análise de dados.

Várias pessoas utilizam o R como um sistema estatístico. Porém, o fato é que o R proporciona um ambiente interior com várias técnicas estatísticas, clássicas e modernas, que foram implementadas dentro do software. Algumas estão compiladas dentro da base do ambiente R, mas várias são disponibilizadas como pacotes. Há em torno de 25 pacotes disponíveis com R (chamados pacotes/*packages* padrões/*standards* e recomendados/*recommended*) e muitos outros disponíveis através da família CRAN em sites da Internet (via <http://CRAN.R-project.org>), dentre outros. Você pode buscar estes pacotes e sua documentação de acordo com a necessidade e avanço na utilização do R.

## 1.2 Programa de Educação Tutorial - Engenharia Elétrica (UFSM)

Esta apostila foi elaborada pelo grupo PET Engenharia Elétrica (PET-EE) da Universidade Federal de Santa Maria (UFSM) com o objetivo de fornecer um guia de introdução ao software R e sua aplicação básica na área de estatística. Esta documentação é de distribuição livre para qualquer pessoa que obter acesso a mesma, podendo ser editada, modificada e redistribuída da forma que o usuário bem entender.

O Programa de Educação Tutorial (PET) foi criado para apoiar atividades acadêmicas que integram ensino, pesquisa e extensão. Formado por grupos tutoriais de aprendizagem, o PET propicia aos alunos participantes, sob a orientação de um tutor, a realização de atividades extracurriculares que complementem a formação acadêmica do estudante e atendam às necessidades do próprio curso de graduação. O estudante e o professor tutor recebem apoio financeiro de acordo com a Política Nacional de Iniciação Científica.

O Programa de Educação Tutorial em Engenharia Elétrica (PET-EE) da UFSM é um grupo que consiste de doze alunos bolsistas, seis não-bolsistas e vários voluntários de diversos semestres do curso, orientados por um professor tutor. O programa busca propiciar aos alunos condições para a realização de atividades extra-curriculares, que favoreçam a sua formação acadêmica tanto para a integração no mercado de trabalho como para o desenvolvimento de estudos em programas de pós-graduação.

São características básicas do PET a formação acadêmica ampla, a interdisciplinaridade, a atuação coletiva, a interação contínua entre os petianos e os corpos docente e discente de graduação e pós-graduação, além do planejamento e execução de um programa diversificado de atividades como: leituras e seminários, grupos de estudo, organização de conferências e palestras, elaboração e desenvolvimento de projetos de pesquisa, estudo de, pelo menos, um idioma estrangeiro, entre outros. Essas características tornam o PET um programa abrangente, pois os petianos se envolvem em atividades de ensino, pesquisa e extensão durante toda a sua permanência no grupo.

# 2

## Introdução

---

### 2.1 Apresentando o R

O R foi criado originalmente por Ross Ihaka e por Robert Gentleman na universidade de Auckland, Nova Zelândia, e foi desenvolvido por um esforço colaborativo de pessoas em vários locais do mundo. O nome R provém em parte das iniciais dos criadores e também de um jogo figurado com a linguagem S (da Bell Laboratories, antiga AT&T).

O R é ao mesmo tempo uma linguagem de programação e um ambiente para computação estatística e gráfica. Trata-se de uma linguagem de programação especializada em computação com dados. Algumas das suas principais características são o seu caráter gratuito e a sua disponibilidade para uma gama bastante variada de sistemas operacionais. Neste documento iremos concentrar a nossa atenção na versão Windows, mas basicamente tudo o que aqui é descrito também se aplica às outras versões, dadas as muito reduzidas diferenças entre as versões para as diversas plataformas. Apesar do seu caráter gratuito o R é uma ferramenta bastante poderosa com boas capacidades ao nível da programação.

O R é também altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas. Um conjunto de pacotes é incluído com a instalação do software, mas muitos outros estão disponíveis na rede de distribuição do R (em inglês CRAN).

### 2.2 Interface do R

Ao iniciar o R abrirá automaticamente o Console que é a janela onde os comandos são digitados. Internamente ao Console, se encontra o prompt, conforme figura abaixo, que é um sinal indicador de que o programa está pronto para receber comando.

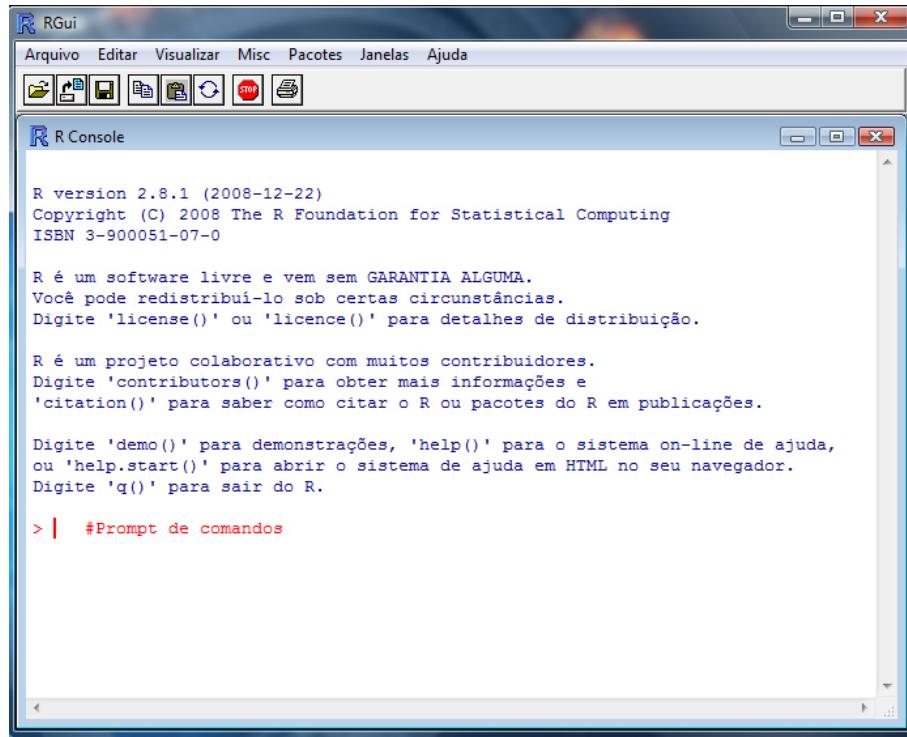


Figura 2.1: Interface do R - Console e Prompt.

## 2.3 Sintaxe do R

Tecnicamente o R é uma linguagem de expressões com regras e sintaxe muito simples. Faz distinção entre maiúsculas e minúsculas, de modo que os caracteres “A” e “a” são entendidos como sendo símbolos diferentes, referindo-se, portanto, a variáveis diferentes.

Os comandos ou ordens elementares consistem em expressões ou atribuições. Se uma ordem ou comando é uma expressão, o seu valor é calculado e visualizado sendo perdido em seguida. Uma atribuição, ao contrário, calcula a expressão e atribui o resultado que não é mostrado automaticamente, apenas é salvo no endereço de alguma variável que está sendo usada no R.

Os comandos são separados por ponto e vírgula (“;”) ou são inseridos em nova linha. Podem agrupar-se dentro de chaves (“{...}”) vários comandos elementares numa expressão mais complexa.

Se ao terminar uma linha, o comando não está sintaticamente completo o R mostra o símbolo “+” que é o comando de continuação do comando inicial.

## 2.4 Tinn-R Editor

O Tinn-R é um editor de texto muito utilizado para a escrita do código R. Com a seguinte definição “*This is not notepad*”, ou seja, *isto não é um bloco de notas*, verifica-se que este editor surge como alternativa ao popular *notepad*.

O Tinn-R possui todos os menus presentes no *notepad* e, além disso, agrupa vários outros recursos extras. O Tinn-R possui a seguinte interface:

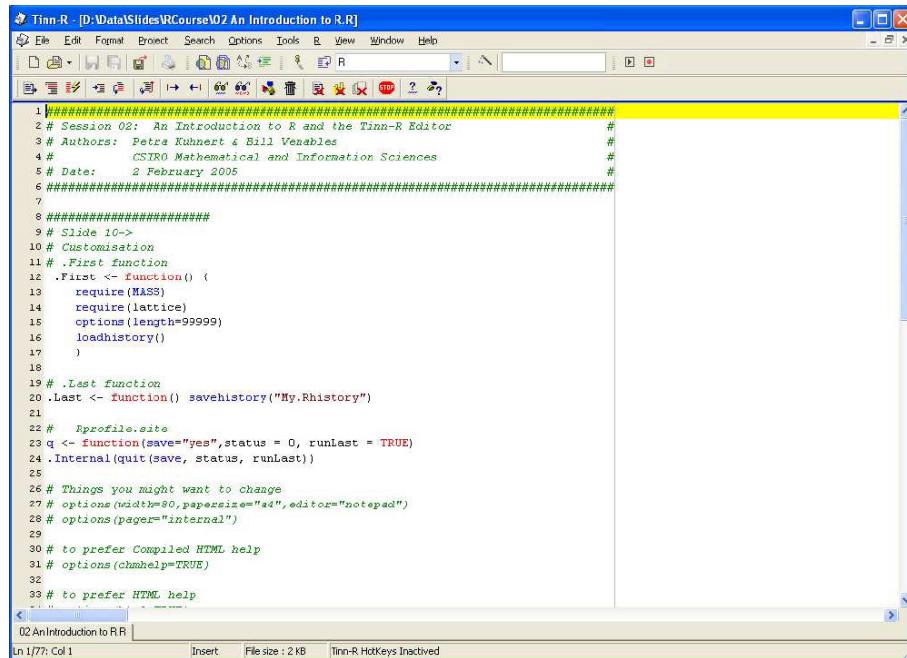


Figura 2.2: Interface do Tinn-R Editor.

A grande vantagem de apresentarmos um editor de texto é haver a possibilidade de correção e execução automática do código, facilitando o trabalho do programador junto ao R. Depois de editado um código no Tinn-R, basta executá-lo na interface do Tinn-R e o programa executará automaticamente os comandos previstos na console do R.

Como alternativa ao Tinn-R na correção de códigos podemos utilizar o bloco de notas que pode ser aberto diretamente no R. Basta utilizar o comando:

```
> edit()
```

Em seguida, deve-se guardar os dados introduzidos no bloco de notas em algum diretório, podendo, quando necessário, ser importado para o Console do R (como veremos a seguir).

Já para editar uma série de dados dentro do R, sem digitá-la novamente, basta atuar com o comando:

```
> variavel = edit(variavel) #carregue a variável para a edição
```

Ou através do editor de dados, pelo comando:

```
> variavel = de(variavel)
```

```
#Exemplo: entre com o vetor {48, 49, 51, 50, 49} e,  
#em seguida, acrescente os valores 60 e 63.  
> x<-c(48, 49, 51, 50, 49)      # entrada do dados  
> x                               # apresentação dos dados  
[1] 48 49 51 50 49  
> x=edit(x)                      # comando de edição dos dados  
> x                               # reapresentação dos dados  
[1] 48 49 51 50 49 60 63
```

## 2.5 Tipos de Dados

Basicamente temos quatro tipos de dados no R: n<sup>umericos</sup>, c, l e n<sup>umeros</sup> c. Cada objeto possui dois atributos: tipo (*mode*) e o tamanho (*length*). Essas informações s<ão> bastante importantes durante a manipulação de dados. Por exemplo, vetores devem possuir obrigatoriamente todos os elementos do mesmo tipo (exceto n<sup>umericos</sup> e complexos, que podem ser agrupados). Veja abaixo alguns exemplos de tipos de dados no R:

```
> #Numérico  
> valor <- 605  
> valor  
[1] 605  
>  
> #Caracteres  
> string <- "Olá, mundo!"  
> string  
[1] "Olá, mundo!"  
>  
> #Lógicos  
> 2 < 6  
[1] TRUE  
>  
> #Números complexos  
> nc <- 2 + 3i  
> nc  
[1] 2+3i
```

```
> mode(valor)
[1] "numeric"
> length(valor)
[1] 1
> mode(string)
[1] "character"
> length(string)
[1] 1
> mode(2<4)
[1] "logical"
> length(2<4)
[1] 1
> mode(nc)
[1] "complex"
> length(nc)
[1] 1
> mode(sin)
[1] "function"
```

## 2.6 Comandos Básicos

### 2.6.1 Utilizando Ajuda

Durante a utilização do software é possível consultar a sintaxe de algum comando ou obter mais informações sobre determinada função. Para isso o R conta com o comando *help*. A sintaxe do comando é a seguinte:

```
> help(comando)      #sintaxe

#Exemplo:
> help(sqrt)
```

Ao executar o exemplo acima, uma interface do menu de ajuda será executada mostrando o tópico da função *sqrt*, que é função matemática para a raiz quadrada. Para realizar uma busca em arquivos de ajuda sobre um tópico desejado, podemos utilizar os seguintes comandos:

```

> help.search("expressão")      #sintaxe
> ??expressão                 #sintaxe

#Exemplos:
> help.search("negative binomial")
> ??weibull

```

Os exemplos acima retornarão janelas de informação indicando os tópicos de ajuda que possuem a expressão procurada.

Observe nos exemplos anteriores o símbolo sustenido (`#`). No console do R, ele anula todos os comandos da linha escritos após a sua inserção. Esse artifício é amplamente utilizado para realizar comentários dentro de um código de programação. Utilizaremos sustenidos nesta apostila para realizar comentários dentro dos códigos e exemplos abordados.

## 2.6.2 Atribuição de Valores

Como todo tipo de programação (inclusive funcional), é comum que tenhamos que atribuir valores para algumas variáveis antes de utilizá-las (esse processo também é conhecido como inicialização de variáveis). No R podemos fazer uma atribuição de valores de várias formas, conforme os exemplos abaixo:

```

> x <- 10          #x é a variável que recebe o valor 10;
> 0.56 -> x       #x é a variável que recebe o valor 0.56;
> x = -8          #x é a variável que recebe o valor -8;
> assign("x", 2i)   #x é a variável que recebe o imaginário 2i;

```

Na maior parte do tempo utilizaremos os símbolos “`->`” e “`=`” para atribuição de valores. Para mostrar o valor armazenado em uma variável, basta digitar a variável na Console e apertar Enter. Qualquer valor digitado sem atribuição pode ser mostrado na tela. O último valor inserido (em uma atribuição ou não) é sempre armazenado em uma variável especial, denominada `.Last.value`. Esta variável pode ser utilizada para realizar operações, mas é preciso tomar cuidado, pois seu valor está sendo constantemente modificado.

```
#Exemplo:
> x = 5
> x
[1] 5
> .Last.value
[1] 5
> y = 10
> 89
[1] 89
> .Last.value
[1] 89
```

### 2.6.3 Comandos Auxiliares

Abaixo veremos uma tabela com os principais comandos que ajudam a manipular os objetos e a “*workspace*” que estão sendo utilizados durante a execução do programa.

Função	Descrição
<i>ls()</i> ou <i>objects()</i>	lista curta de variáveis definidas
<i>ls.str()</i>	lista detalhada de variáveis definidas
<i>str(x)</i>	ver informações detalhadas de <i>x</i>
<i>ls.str(ab)</i>	ver informações detalhadas sobre todas as variáveis com “ <i>ab</i> ” em seu nome
<i>rm(x)</i>	deletar variável <i>x</i>
<i>rm(x, y)</i>	deletar as variáveis <i>x</i> e <i>y</i>
<i>rm(list = ls())</i>	deletar todas as variáveis (limpar a <i>workspace</i> )
<i>class(x)</i>	ver que tipo de objeto é <i>x</i>
<i>q()</i>	sair do R com a opção de salvar a <i>workspace</i> em um arquivo (“Name.RData”) e o histórico de comandos em outro arquivo (“Name.RHistory”)
<i>ctrl + L</i>	no teclado, pressione “ctrl+L” para limpar a tela da console

### 2.6.4 Operações matemáticas simples

Expressões aritméticas podem ser construídas através dos operadores usuais e das regras de precedência:

1	$\wedge$	Potenciação
2	/	Divisão à direita
2	*	Multiplicação
3	+	Adição
3	-	Subtração

```
#Exemplo:
> x=3; y=5; z=10
> h = 4*sqrt(3*x) + 15/(y-z) - x^2
> h
[1] 0
```

Além disso, o R também possui os operadores relacionais e operadores lógicos:

Símbolo	Descrição
<	Menor
<=	Menor ou igual
>	Maior
>=	Maior ou igual
==	Igual (comparação)
!=	Diferente
&	AND
	OR
!	NOT
TRUE ou 1	Valor booleano verdadeiro (1)
FALSE ou 0	Valor booleano falso (0)

```
#Exemplo:
> x = 3
> 2*(1>(3&(4<=x)*!0))
[1] 2
> x = 5
> 2*(1>(3&(4<=x)*!0))
[1] 0
```

## 2.6.5 Funções matemáticas simples

O R possui várias funções matemáticas já implementadas. Abaixo segue uma tabela listando as principais funções com sua respectiva descrição e posteriormente alguns exemplos. Para mais informações, consulte o *help* do R.

Função	Descrição
$abs(x)$	valor absoluto de $x$
$log(x, b)$	logaritmo de $x$ com base $b$
$log(x)$	logaritmo natural de $x$
$log10(x)$	logaritmo de $x$ com base 10
$exp(x)$	exponencial elevado a $x$
$sin(x)$	seno de $x$
$cos(x)$	cosseno de $x$
$tan(x)$	tangente de $x$
$round(x, digits = n)$	arredonda $x$ com $n$ decimais
$ceiling(x)$	arredondamento de $x$ para o maior valor
$floor(x)$	arredondamento de $x$ para o menor valor
$length(x)$	número de elementos do vetor $x$
$sum(x)$	soma dos elementos do vetor $x$
$prod(x)$	produto dos elementos do vetor $x$
$max(x)$	seleciona o maior elemento do vetor $x$
$min(x)$	seleciona o menor elemento do vetor $x$
$range(x)$	retorna o menor e o maior elemento do vetor $x$

```
#Exemplo:
> x = 30; y = 60
> (sin(x))^2 + (cos(x))^2
[1] 1
> round(tan(2*y),digits =3)
[1] 0.713
> floor(tan(2*y))
[1] 0
> ceiling(tan(2*y))
[1] 1
```

## 2.6.6 Números complexos

Para utilizar números complexos possuímos a variável especial “ $i$ ”, que representa a unidade imaginária  $\sqrt{-1}$ . Porém, o R precisa saber que estamos trabalhando com números complexos. Caso contrário ele pode acabar retornando, em algumas ocasiões, o valor *NaN* (*Not a Number*), que indica uma indefinição matemática. Portanto, mesmo que a parte imaginária seja nula, é necessário evidenciá-la para que o R retorne valores complexos.

```
#Exemplo:
> sqrt(-17)
[1] NaN
Warning message:
In sqrt(-17) : NaNs produzidos
> sqrt(-17+0i)
[1] 0+4.123106i
```

## 2.7 Vetores e Matrizes

### 2.7.1 Definição

Vetores são conjuntos de dados unidimensionais. Sua principal utilidade é poder armazenar diversos dados em forma de lista e aplicar funções e operações sobre todos os dados pertencentes a determinado vetor com apenas poucos comandos. Trabalharemos apenas com vetores numéricos.

### 2.7.2 Declaração de vetores

Podemos tratar as atribuições de valores vistas anteriormente como vetores unidimensionais e unitários, ou seja, que só contém um elemento. Para declarar mais de um elemento dentro de um vetor, utilizaremos a seguinte sintaxe:

```
#sintaxe:

$$x = c(a_1, a_2, a_3, \dots, a_{n-1}, a_n)$$


#Exemplos:
> vec <- c(1, 4, 10.5, 54.48, 9, 10)
> vec
[1] 1.00 4.00 10.50 54.48 9.00 10.00
> vec2 <- (1:10)
> vec2
[1] 1 2 3 4 5 6 7 8 9 10
> vec3 <- c((1:3),(3:1))
> vec3
[1] 1 2 3 3 2 1
> vec4 <- c(0, vec3, 0)
> vec4
[1] 0 1 2 3 3 2 1 0
```

Essa sintaxe também serve para vetores de caracteres. Porém, devemos considerar que caracteres devem ser declarados entre aspas.

Outra forma de declarar vetores é a seguinte:

```
#vetor de "a" até "z"
seq(from= a, to= z)

#vetor de "a" até "z" com passo "n"
seq(from= a, to= z, by= n )

#vetor de "a" até "z" com "n" elementos
seq(from= a, to= z, length.out= n)

#Exemplos:
> seq(from=1, to=5)
[1] 1 2 3 4 5
> seq(from=1, to=5, by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(from=0, to=10, length.out= 4)
[1] 0.000000 3.333333 6.666667 10.000000
```

### 2.7.3 Arrays e Matrizes - Definição e Declaração

Podemos definir *arrays* como um conjunto de elementos de dados, geralmente do mesmo tamanho e tipo de dados. Elementos individuais são acessados por sua posição no array. A posição é dada por um índice, também chamado de subscrição. O índice geralmente utiliza uma sequência de números naturais. Arrays podem ser de qualquer tipo, porém neste capítulo abordaremos apenas arrays numéricos, devido a sua grande importância para declaração de matrizes. Existem arrays unidimensionais e multidimensionais. Arrays numéricos unidimensionais nada mais são do que vetores, como já vimos. Já arrays numéricos multidimensionais podem ser usados para representação de matrizes. Vejamos abaixo a sintaxe para declaração de um array:

```
#sintaxe:
x <- array( dados , dim= vetor_dimensão )
#Exemplo:
> x <- array(c(1:10), dim = c(2,5))
> x
[,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Como pode ser visto, o preenchimento de uma array multidimensional (leia-se *matriz* a partir de agora) é realizado dispondo os elementos de entrada ordenadamente coluna por coluna. Existe uma outra função para declaração de matrizes que permite alterar a ordem de disposição dos elementos:

```
#sintaxe:
x <- matrix(data = dados, nrow = m, ncol = n, byrow = Q)

onde "m" é o número de linhas, "n" é o número de colunas e
se Q = 1 #ativa disposição por linhas
se Q = 0 #mantém disposição por colunas

#Exemplo:
> A <- matrix(c(1:10),2,5,1)
> A
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

Para selecionar um elemento de uma matriz utilizamos a indexação por colchetes na variável que representa a matriz com os índices separados por vírgula:

```
#Exemplos:
> A[2,4]
[1] 9
> A[2,4] - x[1,5]
[1] 0
> A[2,]
[1] 6 7 8 9 10
> A[,2:4]
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    7    8    9
> A[,]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

## 2.7.4 Operações e funções com Matrizes

Vamos definir duas matrizes ( $A$  e  $B$ ) e um vetor  $v$ . Abaixo segue uma tabela com as principais operações e funções realizadas entre matrizes. Posteriormente, alguns exemplos.

Função	Descrição
$A * B$	produto elemento a elemento de $A$ e $B$
$A \% * \% B$	produto matricial de $A$ por $B$
$B = aperm(A)$	matriz transposta: $B = A^t$
$B = t(A)$	matriz transposta: $B = A^t$
$B = solve(A)$	matriz inversa: $B = A^{-1}$
$x = solve(A, b)$	resolve o sistema linear $Ax = b$
$det(A)$	retorna o determinante de $A$
$diag(v)$	retorna uma matriz diagonal onde o vetor $v$ é a diagonal
$diag(A)$	retorna um vetor que é a diagonal da matriz $A$
$diag(n)$	sendo $n$ um inteiro, retorna uma matriz identidade de ordem $n$
$eigen(A)$	retorna os autovalores e autovetores de $A$
$eigen(A)\$values$	retorna os autovalores de $A$
$eigen(A)\$vectors$	retorna os autovetores de $A$

```
#Exemplos:
> B = t(A)
> B
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> b= array(c(0,1,5),dim=c(3,1));
> C= matrix(c(c(1,1,0),c(0,1,4),c(0:2)),3,3,1);
> y = solve(C,b)
> y
     [,1]
[1,]   -9
[2,]    9
[3,]   -2
> Cinv = solve(C)
> Cinv%*%b
     [,1]
[1,]   -9
[2,]    9
[3,]   -2
```

## 2.8 Entrada de Arquivos Externos

Se os dados estiverem salvos em arquivos, sob forma de planilhas, tabelas, etc., deve-se fazer com que o R leia estes arquivos, transformando-os em um objeto. Para que o R reconheça o conjunto de dados do arquivo é necessário que as colunas sejam separadas. Caso isso não ocorra o R não conseguirá separar as colunas e emitirá uma mensagem de erro. Um modo fácil de resolver este problema é salvar a planilha de dados com o formato (**.csv**) que utiliza vírgula (,) como elemento separador das colunas.

Porém, antes de iniciar a entrada de dados no R deve-se alterar a pasta de trabalho padrão em que o arquivo de dados **.csv** será salvo. Para isso basta ir em *Arquivo/Mudar dir...* e alterar o diretório em que será salvo o arquivo. Ao abrir a página de alteração do diretório, escolha o diretório em que será salvo o arquivo.

Depois de salvar o arquivo no diretório especificado, carregue o arquivo no console do R. Utilizaremos, nesta apostila, a pasta de trabalho como sendo **C:\Rdados**.

Outra maneira de alterar o diretório é utilizar o seguinte comando, especificando, como argumento, o diretório requerido:

```
> setwd('C:\\Rdados')      #diretório C:\\Rdados
```

Conferindo o diretório atualizado através do comando:

```
> getwd()
```

De posse da pasta de trabalho e do arquivo no formato **.csv** na pasta **Rdados**, procederemos com o seguinte comando:

```
> dir()
```

Com este comando o R irá verificar se há algum arquivo na pasta de trabalho. Como previamente havíamos salvo um arquivo **.csv**, sabemos que o R irá encontrar este arquivo no diretório especificado anteriormente.

Em seguida, devemos dar o comando para que o R carregue o arquivo **.csv** no console de trabalho. Para isso digite o seguinte comando:

```
> carregar <- read.table("arquivo.csv",header=T,sep=",",dec=".")
```

Onde:

- **carregar**: é o objeto no qual os dados lidos serão reconhecidos pelo R;
- **< -**: sinal que atribui os dados lidos ao objeto **carregar**;
- **read.table**: função que lê o arquivo do tipo **.csv**.

O parâmetro “header” nos permite indicar se o arquivo de dados (**data.frame**) tem ou não o nome nas colunas (título) na primeira linha de dados. O parâmetro “sep” permite

indicar o tipo de separador dos dados presentes no arquivo. Finalmente o parâmetro “dec” permite indicar o caractere usado como separador de casas decimais dos números reais.

*Observação:* existem outras sintaxes para carregar dados no console do R (verifique isso utilizando o comando “*help(read.table)*”), porém os argumentos permanecem idênticos aos apresentados anteriormente.

Caso o arquivo tenha título, podemos verificar o nome destes títulos através do comando:

```
> names()      #no argumento vai sempre o nome do objeto desejado
```

Podemos ver a dimensão do arquivo carregado por meio do seguinte comando:

```
> dim()
```

Isto porque o R, agora, considera o arquivo carregado como uma matriz. Desta forma, podemos localizar linhas, colunas e elementos desta matriz. Para isso, utilize os comandos abaixo:

```
> carregar[1,1]      #localiza o elemento a(1,1) da matriz
> carregar[1:5,]     #localiza as primeiras cinco linhas da matriz
```

Vamos agora desenvolver um exemplo com um arquivo **.txt**:

```
> dir() #verifica a presença de arquivos no diretório de trabalho
[1] "arquivoteste.txt"  #localiza arquivo .txt
> carregar <- read.table('arquivoteste.txt',header=T,dec='.')
> #carrega o arquivo no objeto "carregar"
> carregar      #ilustra o arquivo
   mpg engine horse weight accel year origin cylinder
 1 18     307    130   3504  12.0    70      1       8
 2 15     350    165   3693  11.5    70      1       8
 3 18     318    150   3436  11.0    70      1       8
 4 16     304    150   3433  12.0    70      1       8
 5 17     302    140   3449  10.5    70      1       8
 6 15     429    198   4341  10.0    70      1       8
 7 14     454    220   4354   9.0    70      1       8
 8 14     440    215   4312   8.5    70      1       8
```

Podemos ainda carregar um arquivo de qualquer diretório sem precisar informar este diretório no comando. Para isso, basta utilizar a sintaxe abaixo:

```
> carregar <- read.table("C:/Rdados/arquivoteste.txt",
+ header = TRUE, dec=".")
```

## 2.9 Arquivos provenientes da internet

O R permite acessar um banco de dados disponível na web. Esta tarefa é importante, pois facilita o acesso aos dados provenientes da internet, uma vez que os dados não necessitam ser copiados para algum diretório e, posteriormente, carregados para o R.

```
> read.table("endereço") #Sintaxe  
  
#Exemplo:  
> read.table("http://www.leg.ufpr.br/~paulojus/dados/gam01.txt")
```

Caso você queira fazer um download de dados provenientes da internet, sem utilizar seu navegador, utilize no R a função *download.file()*.

```
> #> download.file('endereço','ficheiro de destino') #sintaxe  
>  
> # Exemplo  
> download.file('http://www.leg.ufpr.br/~paulojus/dados/gam01.txt',  
+ 'C:\\R teste\\dados.txt')
```

Observe que o primeiro argumento indica o URL, e o segundo o nome do diretório do computador onde os dados serão guardados. Note que para indicar um caminho do ficheiro se deve separar o nome das pastas por dois caracteres “\\”.

# 3

## Gráficos

### 3.1 Introdução

As capacidades gráficas são uma componente muito importante e extremamente versátil do ambiente R. O R consegue plotar desde gráficos bidimensionais simples até graficos tridimensionais mais complexos por meio de comandos simples. Dá-se muita ênfase no R aos gráficos estatísticos, tais como histogramas, curvas de distribuições, gráfico de barras dentre outros.

#### 3.1.1 Comandos Básicos

O comando básico para a criação gráfica é o **plot()**. A função “*plot(dados)*” gera um gráfico simples, atribuindo pontos em coordenadas cartesianas. Confira o exemplo abaixo:

```
#Exemplo:  
> a <- 1:20  
> b <- a^2  
> plot(a,b)
```

Para tornar o gráfico acima contínuo, deve-se acrescentar o argumento *type='l'* na função *plot()*.

```
> plot(a,b, type="l")
```

Além deste argumento, existem inúmeros outros argumentos para a configuração do gráfico que podem ser acessados com o comando *help(plot)*

Através dos comandos *lines()* e *points()* é possível adicionar, após dado um comando de *plot()*, linhas e pontos, respectivamente, a um gráfico já existente. Veja o exemplo abaixo:

```
#Exemplo:  
> a <- 1:20  
> b <- a^2  
> plot(a,b)  
> lines(rev(a),b) #adição de linhas  
> points(a, 400-b) #adição de pontos
```

O R permite que sejam feitas mudanças na representação dos indicadores gráficos (pontos) através do parâmetro “*pch*=” nos comandos *plot()* e *points()*. Veja o exemplo abaixo:

```
#Exemplo:  
> a <- 1:20; b <- a^2  
> plot(a,b,pch=2)  
> points(a,400-b, pch=5)  
> points(a,200-b, pch=10)  
> windows()  
> plot(0:20,0:20,pch=0:20)
```

Ainda, é possível realizar mudanças nas características das linhas. Para isso, basta utilizar os comandos “*lwd*=” e “*lty*=” que modificam, respectivamente, a largura e o estilo da linha. Veja o exemplo seguinte:

```
#Exemplo:  
> a <- 1:20; b <- a^2  
> plot(a,b,type="l")  
> lines(a,2*b,lwd=4)  
> lines(a,0.5*b,lty=2)  
> lines(a,3*b,lty=3)  
> lines(a,4*b,lty=2,lwd=4)
```

Para alterar a dimensão dos intervalos, pode-se primeiro plotar um gráfico em branco, ajustando os limites da abscissa e da ordenada e depois gerar o gráfico desejado. Observe no exemplo como proceder:

```
#Exemplo:  
> plot(c(-pi,pi),c(-1,1), type="n") #gerando um gráfico em branco  
> x<-seq(-pi,pi,0.1)  
> a <- sin(x)  
> b <- sin(x-2/3*pi)  
> c <- sin(x+2/3*pi)  
> lines(x,a,col=2,lwd=1)  
> lines(x,b,col=3,lwd=2)  
> lines(x,c,col=4,lwd=3)
```

Pode-se ainda acrescentar o nome dos eixos através dos parâmetros “*xlab*=” e “*ylab*=” no comando *plot()*. O título do texto pode ser adicionado com o parâmetro “*main*=” no comando *plot()* ou através do comando *title("título", "subtítulo")*.

A legenda do gráfico pode ser acrescida através do comando *text()* que possui como argumentos as coordenadas do ponto em que se quer colocar a legenda e o texto desejado. Observe o exemplo:

```
#Exemplo:
> plot(c(-pi,pi),c(-1,3),xlab="Período", ylab="Fases", type="n")
> title("Representação das tensões trifásicas","Fases ABC")
> lines(x,a,col=2,lwd=1)
> lines(x,b,col=3,lwd=2)
> lines(x,c,col=4,lwd=3)
> text(0,2,"Observe a defasagem de 120° entre as fases")
```

Outra utilidade do comando `text()` é acrescentar textos às coordenadas cartesianas  $x$  e  $y$ . Observe a sintaxe seguida do exemplo:

```
#Sintaxe:
text(x,y,"etiquetas")
#Exemplo:
> a <- seq(from=0, to=20, by=2); b <- a^2
> plot(a,b,type="n") #plota um gráfico vazio
> text(a,b,"R")      #aplica a etiqueta no sítio dos pontos
```

### 3.1.2 Criando Novas Janelas Gráficas e Salvando Gráficos

Ao executarmos sucessivos comandos `plot()` os gráficos gerados são sobrepostos na mesma janela gráfica chamada de device ACTIVE. Para evitar este problema, podemos proceder de duas formas, conforme a conveniência:

- Os gráficos podem ser salvos imediatamente ao serem gerados. Existem vários formatos em que o R pode salvar imagens gráficas. Alguns deles são: JPEG, BMP, PDF, TIFF, PNG. Faremos abaixo um exemplo utilizando o formato JPEG, mas tenha em mente que a sintaxe para qualquer formato segue idêntica.

```
#Exemplo
jpeg(file='figure.jpeg') #figure é o nome do arquivo imagem
plot(rnorm(10))      #gráfico que estou salvando
dev.off()            #fecha a janela gráfica automaticamente
```

- Por vezes é necessário gerar várias janelas gráficas (devices). Para isso basta utilizar o comando `windows()` ou `X11` entre os sucessivos `plot()`. Confira o exemplo abaixo:

```
#Exemplo:
plot(rnorm(10)) #plotando o primeiro gráfico
windows()        #criação de uma nova janela gráfica
plot(rnorm(20)) #plotando o segundo gráfico
```

Observe que o segundo gráfico vai surgir numa outra janela, o que permite ao utilizador ver os gráficos simultaneamente. Observe, também, que posteriores gráficos que venham a ser plotados serão sobrepostos ao gráfico do último device aberto (leia na parte superior da janela “*Device(ACTIVE)*”)

### 3.1.3 Outras Funcionalidades

Uma funcionalidade bastante útil do R consiste na utilização de identificadores gráficos quando se deseja identificar um ponto ou um conjunto de pontos em um gráfico. Para tanto, existem dois identificadores que podem ser utilizados:

- **locator()**: permite que o utilizador selecione regiões do gráfico utilizando o botão esquerdo do mouse até que se tenha um número n de pontos selecionados ou até pressionar o botão direito do mouse. Cada clique que é dado com o botão esquerdo do mouse o R retorna na console as coordenadas do clique. Veja a sintaxe e o exemplo:

```
#Sintaxe:
locator(n) #localiza n pontos
#Exemplo:
> x=1:20
> y=sqrt(x)
> plot(x,y)
>
> text(locator(1),"mas ba tchê")
> #onde for dado o clique será escrita a mensagem
>
> #ou de outra forma:
> plot(x,y)
> locator(2)
> #localiza dois pontos e dá suas coordenadas na console
>
> x
[1] 12.557587 3.424694
y
[1] 2.427596 3.819199
```

- **identify()**: comando semelhante ao locator(), porém apresenta a capacidade de identificar pontos particulares de um gráfico e não apenas sua posição.

Vejamos um exemplo: representar as coordenadas de oito diferentes cidades, nomeá-las e identificá-las graficamente.

```
#Exemplo:
> x <- c(2,3,4,5,6,7,8,9) #Representação das coordenadas
  "x" das cidades
> y <- c(15,26,45,8,74,11,61,32) #Coordenadas y das cidades
> #Descrevendo o nome das cidades:
> nomes <- paste("cidade", LETTERS[1:8], sep= " ")
> cidades <- data.frame(x,y,row.names=nomes) #Juntando os dados
> cidades #visualizando o objeto cidades
      x   y
cidade A 2 15
cidade B 3 26
cidade C 4 45
cidade D 5 8
cidade E 6 74
cidade F 7 11
cidade G 8 61
cidade H 9 32
> #Visualizando graficamente os pontos que representam as cidades:
> plot(cidades)
> #representa-se as coordenadas gráficas dos pontos, o vetor que
> #será descrito e o número de pontos a serem identificados:
> identify(x,y,nomes,n=4)
[1] 2 3 6 7
```

Depois de adicionado o comando `identify` e definidos seus parâmetros, deve-se clicar nos pontos que se deseja identificar.

O R permite acrescentar gráficos múltiplos basta através dos comandos **`par(mfrow=c(x,y))`** e **`par(mfcol=c(x,y))`** que apresentam comportamentos idênticos. No vetor `c(x,y)`, `x` define o número de divisões horizontais (linhas) e `y` o número de divisões verticais (colunas). Os parâmetros dos gráficos podem ser encontrados no help através do comando `?par`. Proceda com o exemplo abaixo:

```
#Exemplo:
> par(mfrow=c(1,2))
> x<-1:10
> y<- c(2,5,9,6,7,8,4,1,3,10)
> x;y
[1]  1  2  3  4  5  6  7  8  9 10
[1]  2  5  9  6  7  8  4  1  3 10
> plot(x,y)
> plot (x,y, xlab="Eixo X", ylab="Eixo Y",
+ main="Personalizando um gráfico", xlim=c(0,10), ylim=c(0,10),
+ col="red", pch=22, bg="blue", tcl=0.4, las=1, cex=1.5, bty="l")
```

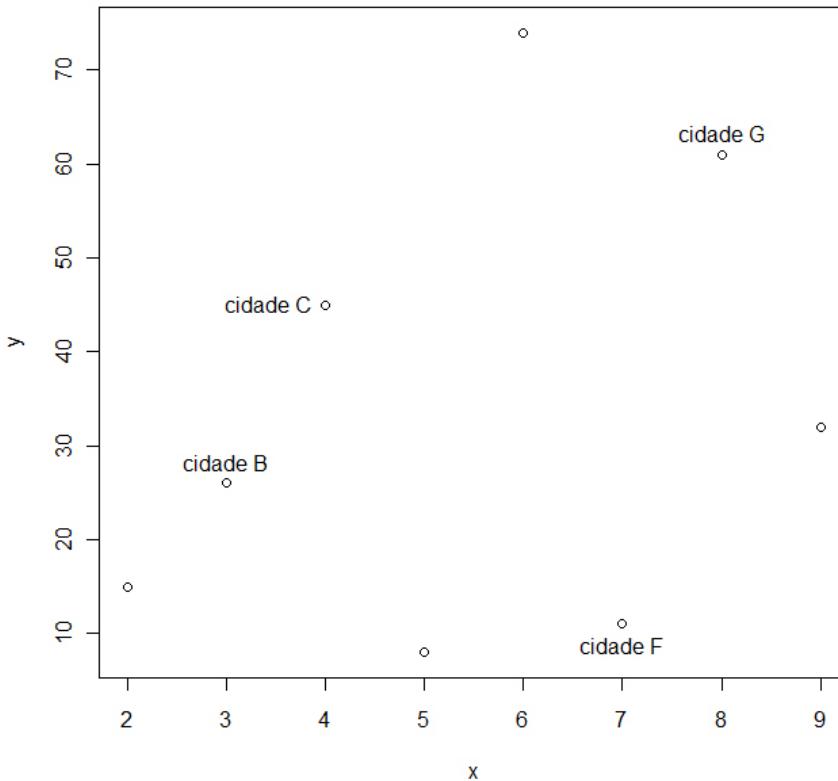


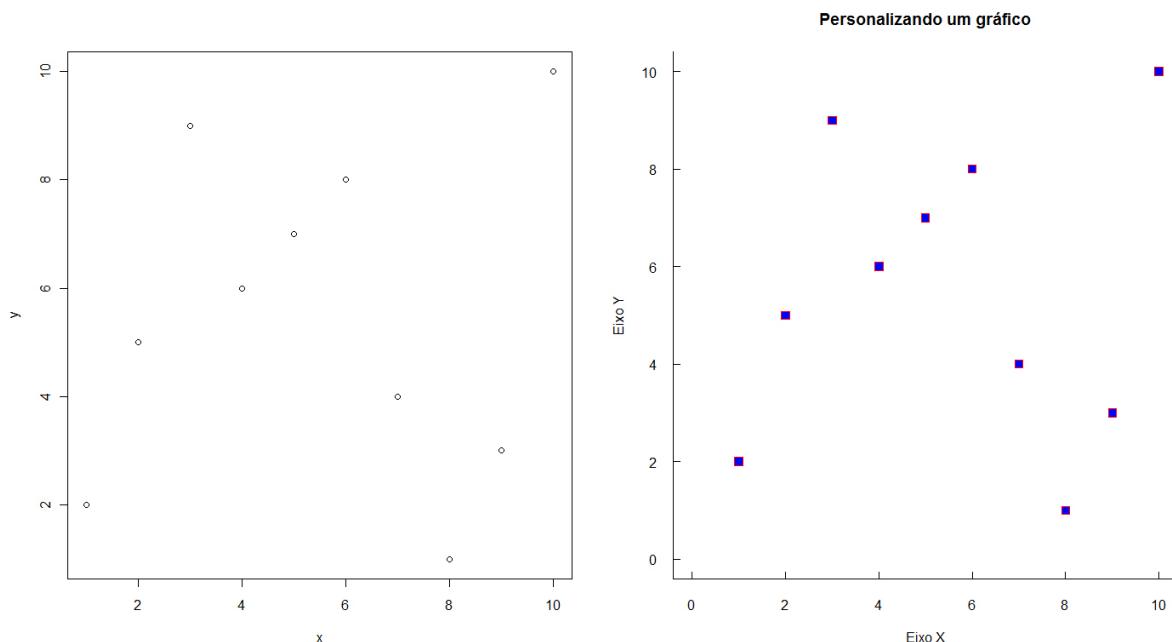
Figura 3.1: Gráfico dos dados de coordenadas de cidades identificadas

## 3.2 Gráficos de Análise Descritiva

Apresentaremos agora três gráficos fundamentais na análise descritiva dos dados: histograma, gráfico de barras e gráfico de caixas. São reconhecidos no R pelos nomes `hist`, `barplot` e `boxplot`.

### 3.2.1 Histograma

Um histograma divide uma série de dados em diferentes classes igualmente espaçadas e mostra a frequência de valores em cada classe. Em um gráfico, o histograma mostra diferentes barras, com bases iguais e amplitudes relativas às frequências dos dados em cada classe. O eixo das ordenadas, portanto, mostra a frequência relativa de cada classe e o eixo das abscissas os valores e intervalos das classes. Abaixo é apresentada a sintaxe do comando e um exemplo ilustrativo:

Figura 3.2: Múltiplos gráficos usando “`mfrow()`”

```
#Sintaxe:  
> hist(dados,nclass=k,) #k é o número de classes do histograma  
  
#Exemplo:  
> rest <- c(96,96,102,102,102,104,104,108,  
+ 126,126,128,128,140,156,160,160,164,170,  
+ 115,121,118,142,145,145,149,112,152,144,  
+ 122,121,133,134,109,108,107,148,162,96)  
> par(mfrow=c(1,2))  
> hist(rest,nclass=12)  
> hist(rest,nclass=6)
```

Do exemplo anterior, teremos os seguintes histogramas, respectivamente:

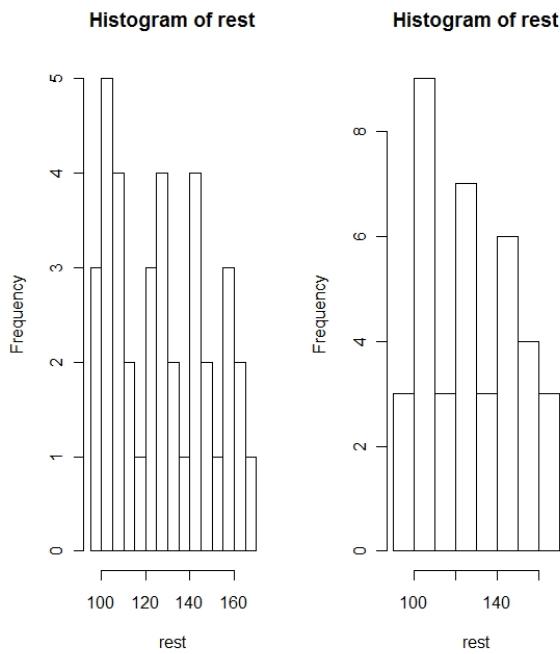


Figura 3.3: Histograma das resistências com 12 classes

### 3.2.2 Barplot

A função `barplot()` produz gráfico de barras, onde cada barra representa a medida de cada elemento de um vetor, ou seja, as barras são proporcionais com a “dimensão” do elemento. A sintaxe geral da função está abaixo:

```
#Sintaxe:  
> barplot(x, col=" ", legend.text=" ", xlab=" ", ylab=" ")  
  
x - é o vetor ou arquivo de dados;  
col=" " - define-se a cor de exibição do gráfico de barras;  
legend.text=" " - legenda do gráfico (o que representa a altura dos gráficos);  
xlab=" " e ylab=" " - nome das grandezas expressas nos eixos x e y, respectivamente.
```

Faremos um exemplo simples utilizando um vetor qualquer e após utilizaremos o dataset “euro” que descreve as taxas de conversão entre as diversas moedas e o euro nos países da união européia. Observe os gráficos e veja os argumentos utilizados na descrição da função `barplot()`.

```
#Exemplos:  
> x <- c(1,2,3,4,5,6,7)  
> barplot(x)  
> barplot(euro,xlab="Euro conversions",col="red",  
+ legend.text="Valor da taxa")
```

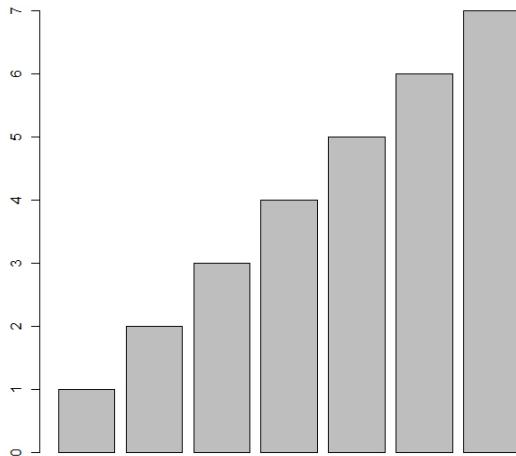


Figura 3.4: Gráfico de Barras de x

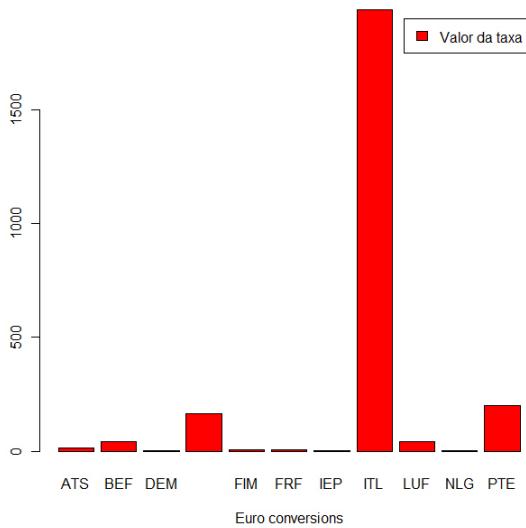


Figura 3.5: Gráfico de barras do dataset “euro”

### 3.2.3 Boxplot

O boxplot é um gráfico que possibilita representar a distribuição de um conjunto de dados com base em alguns de seus parâmetros descritivos (mediana e os quartis). Ele permite avaliar a simetria dos dados e a sua dispersão. É especialmente recomendado para a comparação de dois ou mais conjuntos de dados correspondentes às categorias de uma variável qualitativa.

Veja o gráfico abaixo:

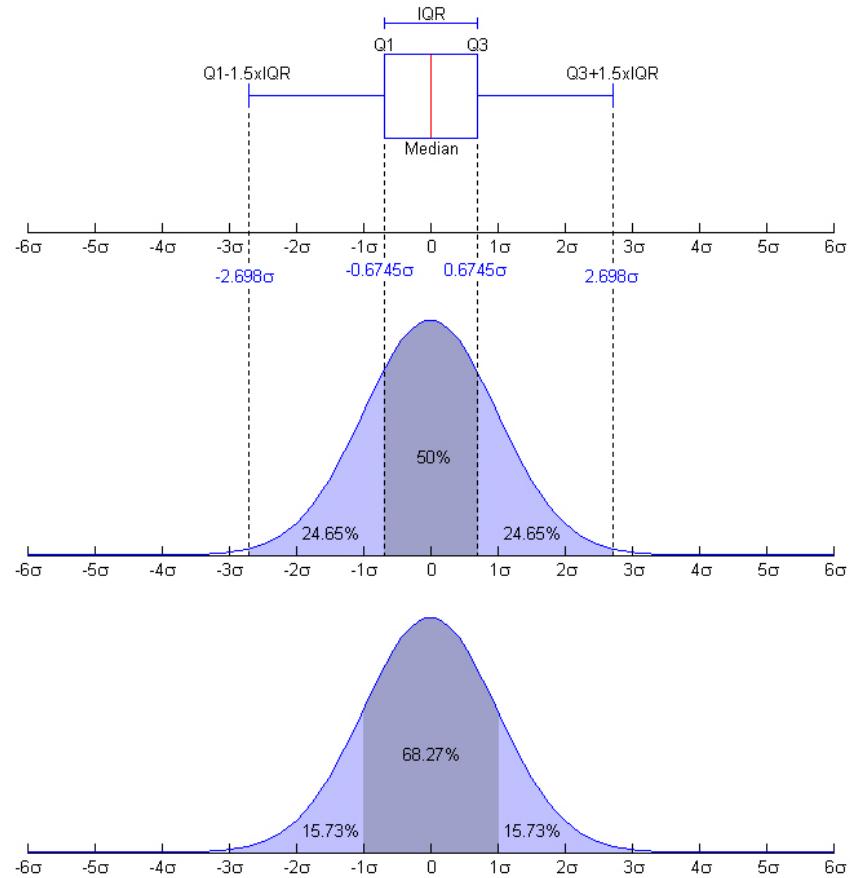


Figura 3.6: Interpretação do gráfico de caixas (boxplot)

Com base neste gráfico, podemos identificar em um boxplot os seguintes parâmetros:

- A linha central marca a mediana do conjunto de dados;
- A parte inferior da caixa é delimitada pelo primeiro quartil ( $Q_1$ ) e a parte superior pelo terceiro quartil ( $Q_3$ );
- Podemos, com isso, verificar também o intervalo interquartil dado pela diferença entre o primeiro e o terceiro quartil ( $IQR = Q_3 - Q_1$ );
- As hastes inferiores e superiores se estendem, respectivamente, do quartil inferior até o menor valor não inferior a  $Q_1 - 1.5 * IQR$  e do quartil superior até o maior valor não superior a  $Q_3 + 1.5 * IQR$ ;
- Os valores inferiores a  $Q_1 - 1.5 * IQR$  e superiores a  $Q_3 + 1.5 * IQR$  são representados individualmente no gráfico sendo estes valores caracterizados como outliers, ou seja, que estão fora do intervalo  $Q_1 - 1.5 * IQR < \text{valor} < Q_3 + 1.5 * IQR$ ;
- As quantidades  $Q_1 - 1.5 * IQR$  e  $Q_3 + 1.5 * IQR$  delimitam as cercas inferior e superior, respectivamente, e constituem limites para além dos quais, como visto, os dados passam a ser considerados outliers.

O comando utilizado no R é o *boxplot()*. Este comando possui vários argumentos. Utilize o comando *help(boxplot)* para maiores informações.

```
#Exemplo:  
> x = c(5,5,5,13,7,11,11,9,8,9)  
> y = c(11,8,4,5,9,5,10,5,4,10)  
> boxplot(x,y) #para plotar no mesmo gráfico (comparação)  
> boxplot(x); boxplot(y) #para plotar em gráficos diferentes
```

Apresentamos no exemplo seguinte um conjunto de dados (dataset) presente no banco de dados do R criado por contribuintes do mundo inteiro [veja em *help(datasets)*]. O dataset utilizado no exemplo é chamado “InsectSprays” e apresenta a contagem de insetos em unidades experimentais agrícolas tratados com diferentes inseticidas.

```
#Exemplo:  
> boxplot(count~spray,data=InsectSprays,xlab="Tipo de Spray",  
+ ylab="Contagem de Insetos",main="InsectSprays data",  
+ col="yellow")
```

### 3.2.4 Gráfico de Ramo e Folhas

O gráfico de ramo e folhas é uma representação gráfica dos números que permite organizar os dados de forma a chamar a atenção para algumas características do conjunto de dados. São elas: forma da distribuição (simetria/assimetria), dispersão dos dados e

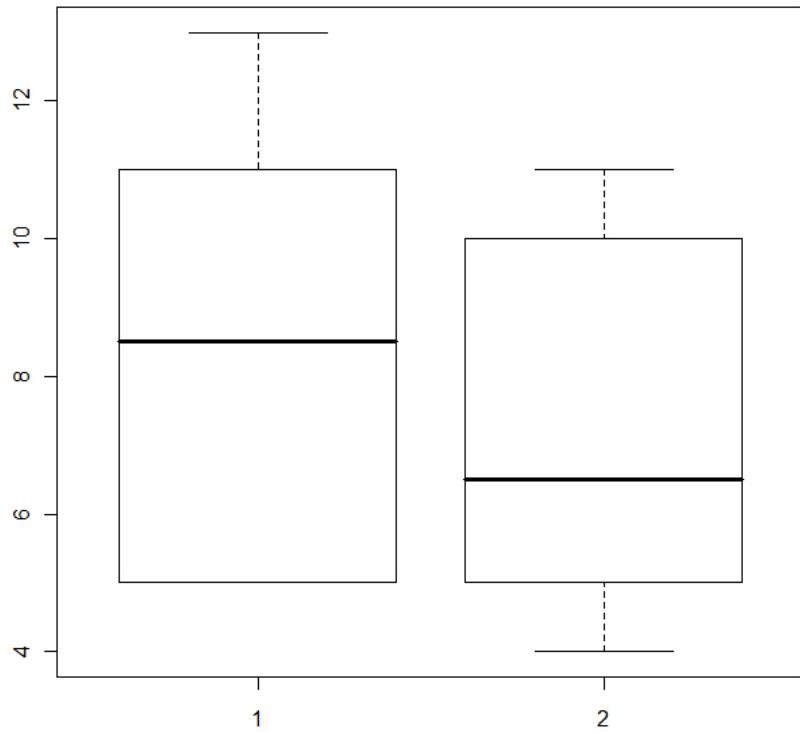


Figura 3.7: Boxplot comparativo dos vetores x e y

existência de outliers.

O gráfico de ramo e folhas possui muita semelhança com o histograma, porém possui a vantagem de exibir o formato da distribuição sem que haja perda de informação. A desvantagem do ramo-e-folhas está no fato de ser um gráfico que deve ser utilizado com conjuntos dados de pequena dimensão.

Um gráfico de ramo e folhas é construído dispondo os dados em duas colunas: uma para os números inteiros (ramos) situada à esquerda, e outra à direita composta pelos números situados depois do ponto decimal dos dados (folhas). As colunas subsequentes à segunda coluna representam as diversas aparições de um mesmo ramo na série de dados em ordem crescente. As linhas apresentam os números dispostos em ordem crescente.

Para a construção de um gráfico ramo e folhas no R basta utilizar o comando **stem()**. Veja o exemplo abaixo:

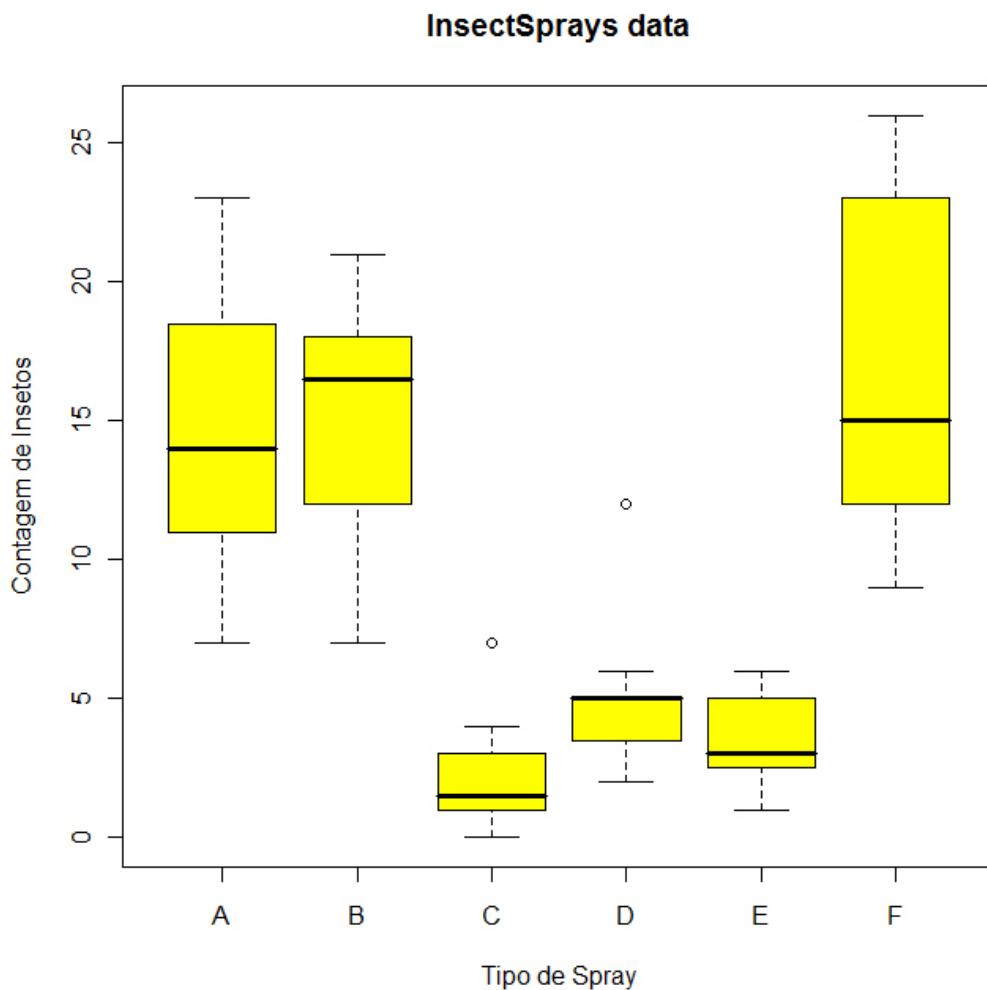


Figura 3.8: Boxplot do dataset “InsectSprays”

```
#Exemplo:
> rf<-c(5.50,5.61,4.88,5.07,5.26,5.55,5.36,5.29,5.58,5.65,5.57,5.53,5.62,5.29,5.4
> stem(rf)
The decimal point is 1 digit(s) to the left of the |
48 | 8
49 |
50 | 7
51 | 0
52 | 6799
53 | 04469
54 | 2467
55 | 03578
56 | 12358
57 | 59
58 | 5
```

Observe que no exemplo acima as folhas contêm o último dígito decimal e os ramos se apresentam em sequencia. Se os números tiverem muitos algarismos significativos serão arredondados para a casa decimal que mais se aproximar do ramo.

### 3.2.5 Gráfico de Pizza

Gráficos de pizza exibem dados como proporção de um todo o que permite fazer comparações entre grupos. Este tipo de gráfico não apresenta nenhum eixo. Quando um dado é solto em um gráfico de pizza, o gráfico calcula a porcentagem de cada valor em relação a toda pizza. Veja a sintaxe e o exemplo abaixo:

```
#Sintaxe:  
pie(dados,opções)  
  
#Exemplo:  
> a<-c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)  
> names(a)<-c("a","b","c","d","e","f")  
> pie(a,col = c("red","blue","green","gray", "brown", "black"))
```

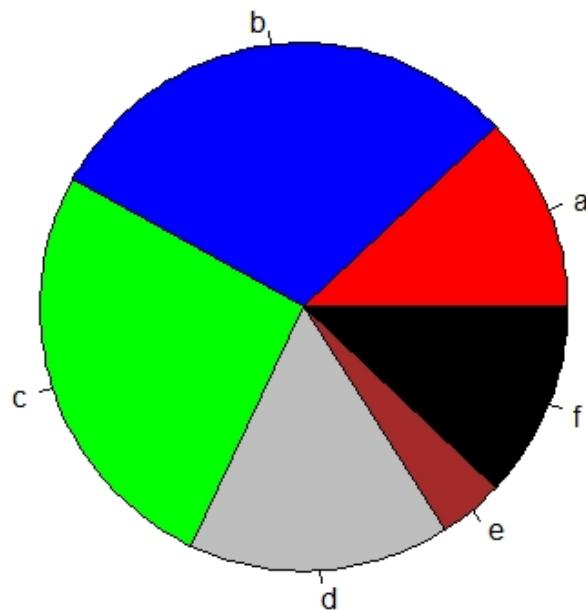


Figura 3.9: Gráfico de Pizza

# 4

## Estatística Descritiva

### 4.1 Introdução

Apresentaremos neste capítulo alguns comandos básicos do R na manipulação da estatística descritiva. A estatística descritiva ocupa-se da organização, apresentação e sintetização dos dados. Desenvolveremos abaixo os componentes da estatística descritiva, bem como seus comandos. Ao final desta seção abordaremos um exemplo ilustrativo, demonstrando alguns dos comandos mencionados.

### 4.2 Medidas de Posição

São as estatísticas que representam uma série de dados orientando-nos quanto à posição da distribuição em relação ao eixo horizontal (eixo "x") do gráfico da curva de freqüência. As medidas de posição mais importantes são as **medidas de tendência central**, no qual se verifica uma tendência dos dados observados a se agruparem em torno dos valores centrais. Passaremos, então, a apresentar as medidas de tendência central mais utilizadas:

#### 4.2.1 Média Aritmética $\bar{X}$

A média aritmética é igual ao quociente entre a soma dos valores do conjunto e o número total dos valores.

$$\bar{X} = \sum_{i=1}^n \frac{X_i}{n} \quad (4.1)$$

onde  $X_i$  são os dados amostrais e  $n$  o número de valores amostrais. O comando para calcularmos a média aritmética segue abaixo, juntamente com um exemplo:

```
#sintaxe:  
          > mean(dados)  
  
#Exemplo:  
> x <- c(10, 14, 13, 15, 16, 18, 12)  
> mean(x)  
[1] 14
```

## 4.2.2 Mediana “Md”

A mediana de um conjunto de valores, dispostos segundo uma ordem (crescente ou decrescente) é o valor situado de tal forma no conjunto que o separa em dois subconjuntos de mesmo número de elementos. Existe um método prático para o cálculo da mediana. Vamos a ele:

- **Se a série dada tiver número ímpar de termos** - O valor mediano será o termo de ordem dado pela fórmula:

$$Md = \frac{n + 1}{2} \quad (4.2)$$

**Exemplo:** Calcule a mediana da série  $\{1, 3, 0, 0, 2, 4, 1, 2, 5\}$

1. Ordenar a série:  $\{0, 0, 1, 1, 2, 2, 3, 4, 5\}$ ;
2.  $n = 9$  elementos.
3. Pela fórmula  $(n + 1) / 2$  é dado por  $(9+1) / 2 = 5$ ;
4. Logo, o quinto elemento da série ordenada será a mediana. Este elemento é o número 2.

- **Se a série dada tiver número par de termos** - O valor mediano será o termo de ordem dado pela fórmula:

$$Md = \frac{\left[\left(\frac{n}{2}\right) + \left(\frac{n}{2} + 1\right)\right]}{2} \quad (4.3)$$

onde  $(\frac{n}{2})$  e  $(\frac{n}{2} + 1)$  são termos de ordem e devem ser substituídos pelo seu valor correspondente.

**Exemplo:** Calcule a mediana da série  $\{1, 3, 0, 0, 2, 4, 1, 3, 5, 6\}$ .

1. Ordenar a série  $\{0, 0, 1, 1, 2, 3, 3, 4, 5, 6\}$ ;
2.  $n = 10$  elementos.
3. Pela fórmula  $[(10/2) + (10/2 + 1)]/2$  resultará na realidade  $(5^{\circ} \text{ termo} + 6^{\circ} \text{ termo})/2$ . Estes termos são 2 e 3, respectivamente.
4. Logo a mediana será  $(2+3)/2$ , ou seja,  $Md = 2,5$ .

Algumas observações:

- Quando o número de elementos da série estatística for ímpar, haverá coincidência da mediana com um dos elementos da série.
- Quando o número de elementos da série estatística for par, nunca haverá coincidência da mediana com um dos elementos da série. A mediana será sempre a média aritmética dos dois elementos centrais da série.

O comando para o cálculo da mediana é o seguinte:

```
#sintaxe:  
  
          > median(dados)  
  
#Exemplo: Tomando os exemplos acima para simples  
conferência, obteremos:  
> k <- c(1,3,0,0,2,4,1,2,5)  
> median(k)  
[1] 2  
> g <- c(1,3,0,0,2,4,1,3,5,6)  
> median(g)  
[1] 2.5
```

### 4.2.3 Moda “Mo”

É o valor que ocorre com maior frequência em uma série de valores. A moda é facilmente reconhecida, basta, de acordo com definição, procurar o valor que mais se repete.

*Observação:* há séries em que não existe valor modal, isto é, série nas quais nenhum valor apareça mais vezes que os outros. Nestes casos dizemos que a série é amodal. Porém, em outros casos, pode haver dois ou mais valores de concentração. Dizemos, então, que a série tem dois valores (bimodal) ou mais.

Existem duas formas que podemos utilizar para encontrarmos a moda de uma série de dados. São elas:

- **table()**: este comando ordena em ordem crescente os dados e indica o número de vezes em que o elemento se repete na série de dados apresentada. É utilizado para encontrar a moda em pequenas amostras.
- **subset()**: em oposição ao item anterior esta função é utilizada quando o tamanho da amostra é grande.

O comando para a obtenção da moda é dado abaixo:

```
#sintaxe:  
  
          > subset(table(), table() == max(table()))
```

Este comando retorna a moda bem como o número de ocorrências do elemento em questão.

Tomemos como exemplo a série {7, 8, 9, 10, 10, 10, 11, 12}:

```
#Exemplo:
> y <- c(7,8,9,10,10,10,11,12)
> table(y)
y
 7 8 9 10 11 12
 1 1 1 3 1 1
> subset(table(y),table(y)==max(table(y)))
10
 3
```

Observe que ambos os comandos indicam qual é o valor da série de dados que mais se repete. No exemplo, este valor é o dado 10, com três ocorrências.

#### 4.2.4 Quartis (Q)

Denominamos quartis os valores de uma série que a dividem em quatro partes iguais. Precisamos, portanto, de três quartis ( $Q_1$ ,  $Q_2$  e  $Q_3$ ) para dividir a série em quatro partes iguais. Veja a figura abaixo:

*Observação:* o quartil  $Q_2$  sempre será igual à mediana da série.

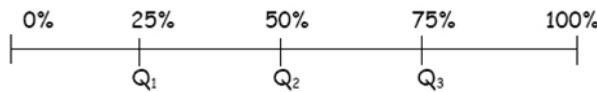


Figura 4.1: Quartis de uma série de dados

**Exemplo:** Calcule os quartis da série: {5, 2, 6, 9, 10, 13, 15}

1. Inicialmente se deve ordenar em ordem crescente os valores. Isto resulta: {2, 5, 6, 9, 10, 13, 15}.
2. O valor que divide a série acima em duas partes iguais é o elemento 9, logo a mediana e o quartil 2 ( $Q_2$ ) é 9.
3. Temos agora {2, 5, 6, 9} e {9, 10, 13, 15} como sendo os dois grupos contendo 50% das informações sobre os dados da série. Para o cálculo do primeiro e do terceiro quartis, basta calcular as medianas dos dois grupos resultantes.
4. Logo em {2, 5, 6, 9} a mediana é 5.5, ou seja, o quartil  $Q_1$  é 5.5 e em {9, 10, 13, 15} a mediana é 11.5, ou seja, o quartil  $Q_3$  é 11.5.

Podemos encontrar os quartis através do comando:

```
#sintaxe:
> summary(dados)
```

Este comando calcula e apresenta o resultado de outros comandos. Desta forma, a função `summary` é capaz de resumir vários tipos de objetos em uma única função. Dentre esses objetos encontram-se o primeiro e o terceiro quartil, sendo que o segundo quartil é dado indiretamente através da mediana.

Calculando o exemplo acima apresentado através do R:

```
#Exemplo:  
> z <- c(5,2,6,9,10,13,15)  
> summary(z)  
Min. 1st Qu. Median Mean 3rd Qu. Max.  
2.000 5.500 9.000 8.571 11.500 15.000
```

Observe que o comando `summary` calcula outras medidas além dos quartis.

#### 4.2.5 Percentis (P)

São as medidas que dividem a amostra em 100 partes iguais. Veja a ilustração:

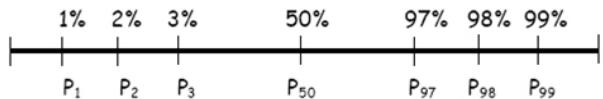


Figura 4.2: Percentis de uma série de dados

Por padrão o R calcula os quantis (partes em que a série de dados é dividida)  $q_0$ ,  $q_{25}$ ,  $q_{50}$ ,  $q_{75}$ ,  $q_{100}$ , os quais são obtidos através do comando:

```
#sintaxe:  
> quantile(dados)
```

Por exemplo, para o vetor 48, 49, 51, 50, 49 podemos calcular os quantis citados da seguinte forma:

```
#Exemplo:  
> q <- c(48,49,51,50,49)  
> quantile(q)  
0% 25% 50% 75% 100%  
48 49 49 50 51
```

A especificação dos percentis pode ser feita pelo comando:

```
#sintaxe:
```

```
> quantile(dados, c(valores dos percentis))
```

Para o exemplo anterior:

```
#Exemplo:
```

```
> percentis = seq(.01,.99,.01)
> quantile(q, percentis)
   1%    2%    3%    4%    5%    6%    7%    8%
48.04 48.08 48.12 48.16 48.20 48.24 48.28 48.32
   9%   10%   11%   12%   13%   14%   15%   16%
48.36 48.40 48.44 48.48 48.52 48.56 48.60 48.64
   ...
  97%   98%   99%
50.88 50.92 50.96
```

#### 4.2.6 Decis (D)

A definição dos decis obedece ao mesmo princípio dos quartis e, portanto, dividem a série de dados em dez partes iguais. Observe:

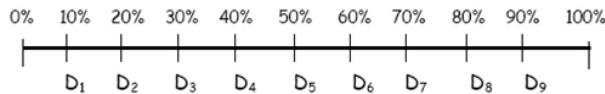


Figura 4.3: Decis de uma série de dados

Indicamos os decis por  $D_1, D_2, \dots, D_9$ . Deste modo, precisamos de nove decis para dividir uma série em dez partes iguais. De especial interesse é o quinto decil que divide o conjunto em duas partes iguais. Assim sendo, o quinto decil é igual ao segundo quartil e, também, igual a mediana da série de dados.

O comando para calcularmos os decis é, também, o `quantile()`. Isso porque basta indicarmos quais os percentuais queremos que este comando calcule. Assim, para os decis:

```
#sintaxe:
```

```
> quantile(dados, seq(0.10, 0.9, 0.1))
```

No exemplo que desenvolvemos para os percentis, temos:

```
#Exemplo:
> d <- c(48,49,51,50,49)
> quantile(d,seq(0.10,0.9,0.1))
 10% 20% 30% 40% 50% 60% 70% 80% 90%
48.4 48.8 49.0 49.0 49.0 49.4 49.8 50.2 50.6
```

*Observação:* como regra geral podemos utilizar o comando "*quantile()*" para os quartis, decís e para os percentis. Basta, para isso, utilizar um vetor no segundo argumento do comando com os valores percentuais desejados.

## 4.3 Medidas de Dispersão

### 4.3.1 Introdução

Além das medidas de posição que estudamos, há outras que, consideradas individualmente, não são medidas de posição, mas apresentam a característica de avaliar a dispersão dos dados em torno dos valores centrais. A motivação para o estudo das medidas de dispersão está ilustrada no seguinte exemplo:

Considere as séries de dados a seguir:

- a) 20, 20, 20, 20, 20
- b) 15, 10, 20, 25, 30

Observamos que para ambas as séries a média é igual a 20. Nota-se, entretanto, que os valores da série na letra "a" se concentram totalmente na média, enquanto os valores da série "b" se dispersam em torno do mesmo valor. Ou seja, a série "a" não apresenta dispersão e os valores da série "b" estão dispersos em torno média 20.

As medidas de dispersão são a amplitude total, a variância, o desvio-padrão e o coeficiente de variação. Vejamos a seguir descrição e exemplos para cada medida de dispersão.

### 4.3.2 Amplitude Total (A)

É a diferença entre o maior e menor dos valores da série. Ou seja:

$$A = X_{max} - X_{min} \quad (4.4)$$

A utilização da amplitude total como medida de dispersão é muito limitada, pois é uma medida que depende apenas dos valores extremos, não sendo afetada pela variabilidade interna dos valores da série.

Em uma série de dados podemos encontrar os valores máximos e mínimos através dos seguintes comandos:

```
#Sintaxe:
> max(dados)
> min(dados)
```

Outra forma de obter o maior e menor valor da série de dados é utilizar o comando:

```
#Sintaxe:  
> range(dados)
```

Já para o cálculo da amplitude total da série deve-se fazer a diferença entre os valores apontados por um dos comandos anteriores da seguinte forma:

```
#Sintaxe:  
> max(dados) - min(dados)
```

Nesse mesmo contexto, podemos encontrar a quantidade total de elementos que a série de dados possui. Basta utilizar a sintaxe a seguir:

```
#Sintaxe:  
> length(dados)  
  
#Exemplo: Dada a série de dados {20,23,23,28,33,37,37,37,40,44}:  
> a <- c(20,23,23,28,33,37,37,37,40,44)  
> max(a)  
[1] 44  
> min(a)  
[1] 20  
> range(a)  
[1] 20 44  
> Amplitude = max(a)-min(a)  
> Amplitude  
[1] 24  
> length(a) #número de elementos da série de dados  
[1] 10
```

### 4.3.3 Variância ( $\sigma^2$ )

A variância é a medida de dispersão mais empregada geralmente, pois leva em consideração a totalidade dos valores da variável em estudo. Baseia-se nos desvios em torno da média aritmética, sendo um indicador de variabilidade.

Considerando nosso propósito de medir o grau de variabilidade dos valores em torno da média, nada mais interessante do que estudarmos o comportamento dos desvios de cada valor individual da série em relação à média. Desta forma, o desvio individual é dado por:  $d_i = (x_i - \bar{x})$ , onde  $x_i$  representa cada um dos i-ésimos valores da amostra e  $\bar{x}$  é a média da amostra.

Entretanto, por uma das propriedades da média tem-se que  $\sum_{i=1}^n (x_i - \bar{x}) = 0$ . Temos então que solucionar o seguinte problema: queremos calcular a média dos desvios, porém sua soma pode ser nula.

Como solução a esse problema a variância considera o quadrado de cada desvio

$(x_i - \bar{x})^2$ , evitando com isso que o somatório seja nulo. Assim, a variância é dada por:

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2 F_i}{n - 1} \quad (4.5)$$

onde  $F_i$  é o número de ocorrências de  $x_i$ .

O comando para o cálculo da variância é o seguinte:

```
#Sintaxe:  
> var(dados)  
  
#Exemplo: Para o vetor {10,11,9,10,10,9,11} obtenha a variância.  
> v <- c(10,11,9,10,10,9,11)  
> var(v)  
[1] 0.6666667
```

#### 4.3.4 Desvio-padrão ( $\sigma$ )

Seguindo a mesma linha de raciocínio usado para o cálculo da variância, necessitamos, agora, aproximar a medida de dispersão da variável original. Para isso, calculamos o desvio padrão, que é a raiz quadrada da variância. Assim:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2 F_i}{n - 1}} \quad (4.6)$$

Podemos representar o desvio padrão por uma distribuição normal, conforme gráfico abaixo:

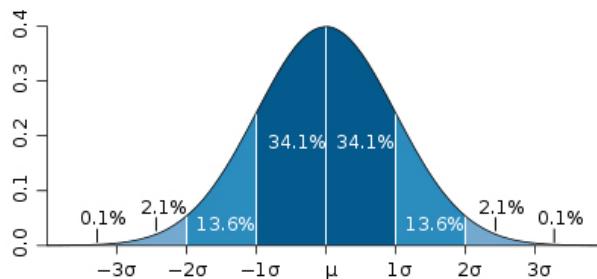


Figura 4.4: Gráfico da distribuição normal em função do desvio-padrão

- 68,26% das ocorrências se concentrarão na área do gráfico demarcada por um desvio padrão à direita e um desvio padrão à esquerda da linha média;
- 95,44% das ocorrências estão a dois desvios padrão, para a direita e a esquerda da média e, finalmente;
- 99,72% das ocorrências ocorrem a três desvios padrão ao redor da média aritmética.

Esta particularidade torna as distribuições normais previsíveis, ou seja, se pudermos levantar seu desvio padrão poderemos fazer previsões sobre os eventos representados dentro das probabilidades definidas.

O comando para o cálculo do desvio padrão segue abaixo:

```
#Sintaxe:  
> sd(dados)  
  
#Exemplo: Para o exemplo anterior, do cálculo da variância.  
> v <-c(10,11,9,10,10,9,11)  
> sd(v)  
[1] 0.8164966  
> sqrt(var(v))  
[1] 0.8164966
```

### 4.3.5 Coeficiente de Variação (CV)

Trata-se de uma medida relativa de dispersão, útil para a comparação em termos relativos do grau de concentração em torno da média de séries distintas. É dado por:

$$CV = \frac{\sigma}{\bar{x}} \cdot 100\% \quad (4.7)$$

A importância de se estudar o coeficiente de variação se dá, pois o desvio-padrão é relativo à média. E como duas distribuições podem ter médias diferentes, o desvio destas distribuições não é comparável. Logo, o coeficiente de variação é muito utilizado para comparação entre amostras.

O R calcula o coeficiente de variação conforme mostrado abaixo:

```
#Sintaxe:  
> 100*sd(dados)/mean(dados) #dado em porcentagem  
  
#Exemplo: Para o exemplo anterior, do cálculo do desvio-padrão.  
> v <-c(10,11,9,10,10,9,11)  
> CV = 100*sd(v)/mean(v)  
> CV  
[1] 8.164966 #em torno de 8%
```

## 4.4 Exemplo Aplicado

O seguinte exercício foi adaptado a partir de [5].

**Exemplo:** Um artigo no Journal of Structural Engineering (Vol. 115, 1989) descreve um experimento para testar a resistência resultante em tubos circulares com calotas soldadas nas extremidades. Os primeiros resultados (em kN) são: 96; 96; 102; 102; 102; 104; 104; 108; 126; 126; 128; 128; 140; 156; 160; 160; 164 e 170. Pede-se:

- a)** Calcule a média da amostra e dê uma interpretação prática para ela.
- b)** Calcule os percentis 9%, 25,5% e 69,67%.
- c)** Calcule o segundo quartil ou mediana.
- d)** Calcule a amplitude da amostra.
- e)** Calcule a variância e o desvio padrão da amostra.
- f)** Qual a fonte de maior variabilidade deste experimento.

Para resolvemos este problema no R basta carregarmos o vetor com as medidas de resistência das calotas soldadas. Veja abaixo:

```
> rest <- c(96,96,102,102,102,104,104,108,
+ 126,126,128,128,140,156,160,160,164,170)
> rest
[1] 96 96 102 102 102 104 104 108 126 126
[11] 128 128 140 156 160 160 164 170
```

- a)** Para calcular a média basta fazer:

```
> mean(rest)
[1] 126.2222
```

Com este valor podemos concluir que a resistência da solda das calotas circulares se concentra, na maioria dos testes, em torno do valor médio. Isto é, se pegarmos aleatoriamente uma calota soldada é de se esperar que a resistência da solda se concentre em torno (e próximo) da média.

- b)** Obtemos pelo seguinte comando:

```
> quantile(rest,c(.09,0.255,.6967))
  9%    25.5%   69.67%
99.1800 102.6700 138.1268
```

**c)** Como sabemos, o segundo quartil coincide com a mediana da amostra e calculamos da seguinte forma:

```
> summary(rest)
  Min. 1st Qu. Median     Mean 3rd Qu.     Max.
  96.0    102.5   126.0    126.2    152.0    170.0
> #ou
> median(rest)
[1] 126
```

- d)** A amplitude pode ser obtida de duas formas diferentes. Vamos a elas:

```
> range(rest)
[1] 96 170
> A = 170-96
> #ou
> A = max(rest)- min(rest)
> A
[1] 74
```

e) Obteremos com os comandos apresentados abaixo:

```
> var(rest)
[1] 683.2418
> sd(rest)
[1] 26.13889
> sqrt(var(rest))
[1] 26.13889
```

f) Como a estatística se preocupa com a variabilidade dos dados amostrais, devemos apontar suas causas. Neste exemplo, podemos apontar como possíveis causas de variabilidade os erros de medição da resistência da solda, soldagem feita por soldadores diferentes (caso não seja automatizado), etc. Enfim, devemos reduzir a variabilidade para termos garantias de qualidade e, num cenário ideal, eliminá-la.

# 5

# Probabilidade

## 5.1 Introdução

Um experimento pode apresentar diferentes resultados quando conduzido mais de uma vez. Desta forma, mesmo tomando-se todo o cuidado para sua realização, existem variáveis como variações na temperatura, quantidade de impurezas na composição química de determinado material, alterações nos medidores, dentre outras, que não podem ser controladas, mas influenciam nas conclusões a respeito de tal experimento.

Estas variáveis podem provocar variações muito pequenas nos resultados e serem desprezadas, ou sua influência pode ser relativamente grande e as conclusões não serem óbvias. Por este motivo, as distribuições de probabilidade buscam modelar e analisar os resultados experimentais em que as variáveis não controladas possam provocar alterações significativas nos resultados.

Alguns conceitos importantes devem estar consolidados para que seja compreendido o estudo da probabilidade. A seguir são apresentados tais conceitos:

- **Experimento Aleatório:** experimento que pode apresentar diferentes resultados mesmo quando conduzido sob as mesmas condições.
- **Espaço amostral:** conjunto de todos os possíveis resultados de um experimento.
- **Eventos:** subconjunto do espaço amostral de um experimento aleatório.

## 5.2 Probabilidade - Definição

A probabilidade é um número atribuído a cada membro de uma coleção de eventos a partir de um experimento aleatório. Ela é normalmente quantificada de maneira a representar o grau de crença que determinado evento possa ocorrer. Por exemplo, pode-se afirmar que a probabilidade de chover no fim de semana é de 40%.

Quando se quantifica determinado acontecimento, atribui-se um valor entre 0 e 1 ou em porcentagem. Quando a probabilidade é igual a zero, o evento não ocorrerá. Já quando a probabilidade é igual a um, ele ocorrerá com certeza.

Pode-se definir a probabilidade de um experimento da seguinte forma: Se um espaço amostral consistir em  $N$  resultados possíveis que sejam igualmente prováveis, as possibilidades de ocorrência de cada resultado é de  $1/N$ .

**Exemplo 1:** Em um lote de 100 diodos, 30% satisfazem os requerimentos mínimos de potência de um consumidor específico. Se um diodo for selecionado ao acaso (cada diodo tem a mesma chance de ser selecionado) qual será a probabilidade de que as exigências do consumidor sejam satisfeitas?

**Solução:** Denotando “E” como evento em que o diodo selecionado satisfaça às exigências do consumidor, temos que E é o subconjunto de 30 diodos que está contido no espaço amostral de 100 diodos do lote ( $N=100$ ).

A probabilidade de cada diodo ser selecionado é determinada da seguinte maneira:

$$P(N) = \frac{1}{100} = 0,01$$

Onde N são os possíveis resultados e P(N) a probabilidade individual de seleção de cada diodo.

Como E tem 30 possíveis resultados e cada resultado tem probabilidade de 0,01, então a probabilidade de ocorrência do evento E, ou a probabilidade de que sejam satisfeitas as exigências do consumidor, será:

$$P(E) = 30 \times 0.01 = 0,3 = 30\%$$

No R a probabilidade é calculada da seguinte forma:

```
#Sintaxe:  
# Exemplo 1: Lote de Diodos  
> n = 100      # número de possíveis resultados  
> p = 1/n      # probabilidade de ocorrência de um resultado  
> p  
[1] 0.01  
> E = 30*p     # probabilidade do evento E  
> E  
[1] 0.3
```

**Exemplo 2:** Uma inspeção visual de um produtor local de pastilhas provenientes de um processo de fabricação de semicondutores resultou na seguinte tabela:

Número de contaminantes	Proporção de Partículas
0	0,4
1	0,2
2	0,15
3	0,10
4	0,05
5 ou mais	0,10

Se desse processo uma pastilha for selecionada ao acaso e o local for inspecionado, pergunta-se:

- a) Qual será a probabilidade de que ele não contenha partículas contaminantes?
- b) Qual será a probabilidade de uma pastilha conter três ou mais partículas do sítio inspecionado?
- c) Qual a probabilidade de uma partícula conter 0 ou mais de três partículas no sítio inspecionado?

**Solução:**

- a) A probabilidade requerida depende somente do número de partículas contaminantes. Pode-se considerar que o espaço amostral apresentado na tabela represente as

seis categorias que resumem o número de partículas contaminantes em uma pastilha. Assim, o evento em que não há nenhuma partícula no local inspecionado é denotado pelo evento  $E_a$ :

$$P(E_a) = P(0) = 0,4 \quad (5.1)$$

```
#Sintaxe:  
# Exemplo 2.a):  
> p_0 = 0.4 # probabilidade de 0 "partículas" por pastilha  
> p_a = p_0 # probabilidade do item a é igual a probabilidade 0  
> p_a # mostrar probabilidade do item a  
[1] 0.4
```

**b)** Fazendo  $E_b$  denotar o evento em que a pastilha contém três ou mais partículas no sítio inspecionado, verificamos que  $E_b$  será a soma das três últimas categorias (3, 4, 5 ou mais):

$$P(E_b) = P(3, 4, 5, > 5) = 0,1 + 0,05 + 0,1 = 0,25$$

```
#Sintaxe:  
# Exemplo 2.b):  
> p_3 = 0.1 # probabilidade de 3 partículas por pastilha  
> p_4 = 0.05 # probabilidade de 4 partículas por pastilha  
> p_5 = 0.1 # probabilidade de 5 ou mais partículas por pastilha  
> p_c = p_3 + p_4 + p_5 # probabilidade do item b é  
> # a soma das três probabilidades  
> p_c # mostrar probabilidade do item b  
[1] 0.25
```

**c)** Fazendo  $E_c$  denotar o evento em que a pastilha contém 0 ou mais que três partículas no sítio inspecionado,  $E_c$  será a soma dos resultados (0, 4, 5 ou mais):

$$P(E_c) = P(0, 4, \geq 5) = 0,4 + 0,05 + 0,1 = 0,55$$

```
#Sintaxe:  
# Exemplo 2.c):  
> p_0 = 0.4 # probabilidade de 0 partículas por pastilha  
> p_4 = 0.05 # probabilidade de 4 partículas por pastilha  
> p_5 = 0.1 # probabilidade de 5 ou mais partículas por pastilha  
> p_c = p_0 + p_4 + p_5 # probabilidade do item c é >  
> # a soma das três probabilidades  
> p_c # mostrar probabilidade do item c  
[1] 0.55
```

## 5.3 Axiomas da Probabilidade

Os axiomas da probabilidade são regras que garantem que as probabilidades de um determinado evento sejam coerentes com o nosso entendimento intuitivo. Eles possibilitam que as probabilidades de alguns eventos sejam calculadas a partir do conhecimento das probabilidades de outros eventos. Assim:

Considerando  $S$  o espaço amostral e  $E$  como qualquer evento do experimento aleatório, tem-se:

- 1)  $P(S) = 1$
- 2)  $0 \leq P(E) \leq 1$
- 3) Para dois eventos  $E_1$  e  $E_2$  com  $E_1 \cap E_2 = \emptyset$ ,  $P(E_1 \cup E_2) = P(E_1) + P(E_2)$

# 6

## Variáveis Aleatórias

---

### 6.1 Introdução

A utilização dos recursos da estatística descritiva requer que o espaço amostral não-numérico de um determinado experimento aleatório seja transformado em um espaço amostral numérico. Em outras palavras, é necessário representar o resultado de um experimento aleatório através de um número.

Tal transformação é feita por uma função que confere um número real a cada resultado do experimento aleatório. A esta função dá-se o nome de variável aleatória.

Em geral, as variáveis aleatórias são representadas pela letra maiúscula X, e elas podem ser de dois tipos: discretas ou contínuas.

### 6.2 Variáveis Aleatórias Discretas

#### 6.2.1 Introdução

Em alguns experimentos, como por exemplo, o número de bits transmitidos que são recebidos com erro, tem-se uma medida limitada a números inteiros. Ou ainda, quando se deseja registrar que 0,0042 de 1000 bits foram recebidos com erro, tem-se uma medida fracionária mais ainda assim limitada na linha dos números reais. A variável que representa uma medida com uma faixa finita de valores ou infinita contável é denominada **variável aleatória discreta** ou **VAD**.

Como exemplo de variáveis aleatórias discretas, tem-se a quantidade de partes defeituosas em n testadas, número de bits transmitidos com erro, ou seja, tudo que estiver relacionado com a contagem de determinados elementos.

#### 6.2.2 Distribuição Binomial - $X \sim b(n,p)$

Um experimento binomial diz respeito a um experimento aleatório que consiste em repetidas tentativas que apresentam apenas dois resultados possíveis (tentativas de Bernoulli) e possui as seguintes características:

- As tentativas são independentes, ou seja, o resultado de uma não altera o resultado da outra;
- Cada repetição do experimento admite apenas dois resultados: sucesso ou fracasso;
- A probabilidade de sucesso ( $p$ ), em cada tentativa, é constante.

A variável aleatória **X** denota o número de tentativas que resultaram em sucesso e possui uma distribuição binomial com parâmetros **p** e **n = 1,2,3,...**

A função de probabilidade de  $\mathbf{X}$  é:

$$P(X = x) = f(x) = \binom{n}{x} p^x (1-p)^{n-x}, x = 0, 1, 2, \dots, n. \quad (6.1)$$

onde

$$\binom{n}{x} = \frac{n!}{x!(n-x)!} \quad (6.2)$$

$n$  - número de tentativas;  $p$  - probabilidade de sucesso;  $x$  - número de sucessos;

A esperança (média) e a variância são dadas por:

$$\mu = E(X) = np \quad (6.3)$$

$$\sigma^2 = V(x) = np(1-p) \quad (6.4)$$

A distribuição binomial é obtida através da expansão binomial, fórmula que pode ser aplicada em vários exemplos:

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} \quad (6.5)$$

Considerando  $a = p$  e  $b = 1-p$ , observa-se que a soma das probabilidades para uma variável aleatória discreta é igual a um.

O nome binomial é devido ao fato de o resultado apresentar duas possibilidades.

A distribuição binomial é uma particularidade da distribuição multinomial.

**Exemplo 1:** Uma amostra de ar tem 10% de chance de conter certa molécula rara. Considere que as amostras sejam independentes com relação à presença da molécula rara. Encontre a probabilidade de que nas próximas 18 amostras, exatamente 2 contenham a molécula rara.

**Solução:**  $x$  é o número de amostras de ar que contêm a molécula rara nas próximas 18 amostras analisadas.

$$p = 10\% = 0,1;$$

$$n = 18;$$

$$x = 2;$$

```
#Exemplo 1:
> p <- 0.1 #probabilidade
> n <- 18 #número de amostras
> x <- 2 #número de sucessos em 18 amostras
> dbinom(x, n, p)
[1] 0.2835121
```

O resultado representa que a probabilidade de que exatamente 2 moléculas raras sejam encontradas nas próximas 18 amostras analisadas.

**Exemplo 2:** A probabilidade de uma peça artesanal ser feita com perfeição por um artesão é de 50%. Considerando que o artesão produz, de maneira independente, 6 peças por dia, pede-se:

a) Obter a distribuição de probabilidades, ou seja, as probabilidades associadas aos

possíveis valores da variável aleatória discreta  $x$ , em que  $x = \text{número de peças perfeitas produzidas pelo artesão num único dia}$ .

*Observação:*  $x = \{0,1,2,3,4,5,6\}$ ,  $n = 6$ ,  $p = 0.5$ .

**b)** Plotar o gráfico com os valores da probabilidade calculada.

```
#Exemplo 2:  
> x <- 0:6  
> n <- 6  
> p <- 0.5  
> bino <- dbinom(x, n, p)  
> bino  
[1] 0.015625 0.09375 0.234375 0.312500 0.234375 0.093750 0.015625  
> plot(x,bino,type="h",xlab="Nº de peças com perfeição",  
+ ylab="Probabilidade",main="Distribuição binomial")
```

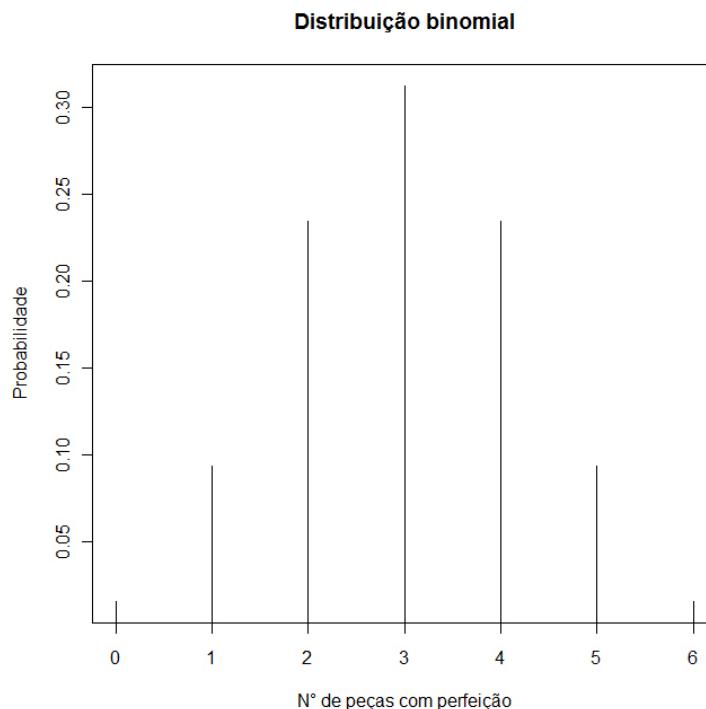


Figura 6.1: Gráfico do Exemplo 2.b: Distribuição binomial de probabilidades

### 6.2.3 Distribuição De Poisson - $X \sim P(\lambda)$

A distribuição de Poisson permite evidenciar a probabilidade de experimentos em que o número de amostras pode aumentar no tempo e a probabilidade de sucesso diminuir, mantendo a esperança  $E(X)$  constante.

Um experimento no qual dado um intervalo de números reais, sejam realizadas contagens através do intervalo. Se este intervalo puder ser dividido subintervalos com comprimentos tão pequenos tais que:

- A probabilidade de mais de uma contagem em um subintervalo seja zero;
- A probabilidade de uma contagem em um subintervalo seja a mesma para todos os subintervalos e proporcional ao comprimento do intervalo;
- A contagem de cada subintervalo seja independente dos outros subintervalos;

Tal experimento é chamado *processo de Poisson*.

Assim a variável aleatória  $X$ , que denota o número de contagens no intervalo, possui uma distribuição de Poisson com parâmetro  $\lambda$  e é representada através da função:

$$f(x) = \frac{e^{-\lambda} \lambda^x}{x!}, x = 0, 1, 2, \dots, n. \quad (6.6)$$

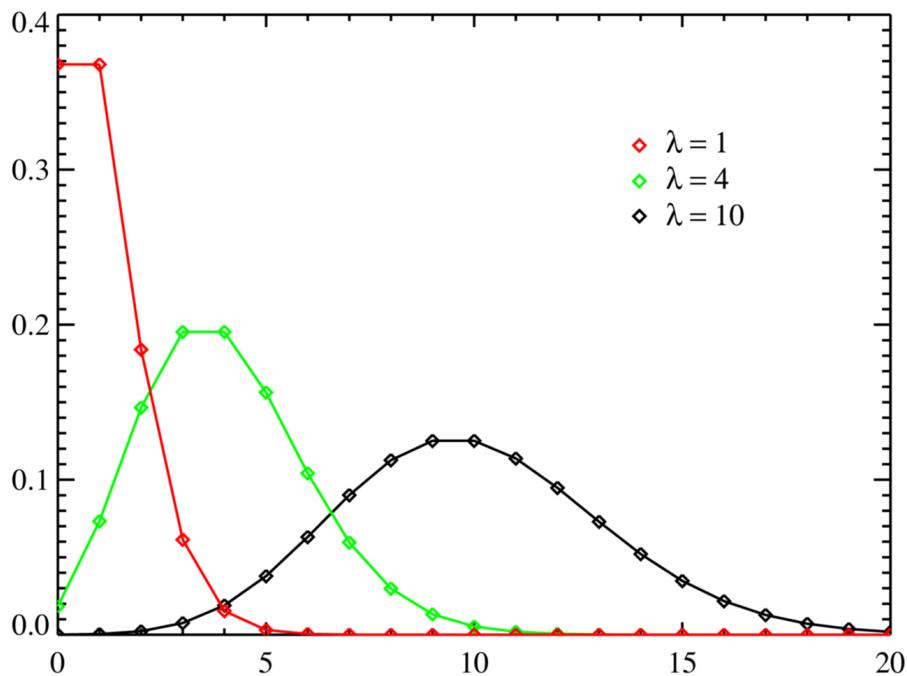
A média e a variância de uma distribuição de Poisson são representadas por:

$$\mu = E(X) = \lambda = pn \quad (6.7)$$

$$\sigma^2 = V(x) = \lambda \quad (6.8)$$

Exemplos da utilização da distribuição de Poisson são quaisquer experimentos que incluem intervalo de tempo, superfície ou volume: partículas de contaminação na fabricação de semi-condutores, interrupções de energia, entre outras.

Abaixo, observa-se o gráfico da distribuição de Poisson.

Figura 6.2: Distribuição de Poisson para diferentes parâmetros  $\lambda$ 

A distribuição binomial pode ser aproximada por uma Poisson, com  $\lambda = n.p$  quando o tamanho da amostra é grande ( $n \rightarrow \infty$ ) e a probabilidade é pequena ( $p \rightarrow 0$ ). Em outras palavras, quando  $n > 30$  e  $p < 0,05$ .

**Exemplo 3:** Em um fio delgado de cobre, o número de falhas no fio segue a distribuição de Poisson, com uma média de 2,3 falhas por milímetro.

- a) Determine a probabilidade de existir exatamente 2 falhas em um milímetro de fio.
- b) Sabendo que o número máximo de erros no teste de qualidade é de 10 erros/mm, verifique as probabilidades de que ocorram de 0 a 10 falhas no fio. Plote o gráfico da distribuição.

**Solução:**

- a) X representa o número de falhas em 1 milímetro de fio:  $E(X) = \mu = \lambda = 2,3$  falhas/mm.

```
#Exemplo 3a:
> x<-2
> lambda<-2.3
> #distribuição de Poisson com parâmetros x e lambda:
> dpois(x,lambda)
[1] 0.2651846
```

b) Fazendo x variar de 0 a 10 erros/mm, temos:

```
#Exemplo 3b:
> x <- 0:10
> poisson <- dpois(x, lambda)
> plot(x,poisson, xlab= "Nº de erros por milímetro",
+ ylab="Probabilidade de Poisson",main="Distribuição de Poisson")
> lines(x,poisson)
```

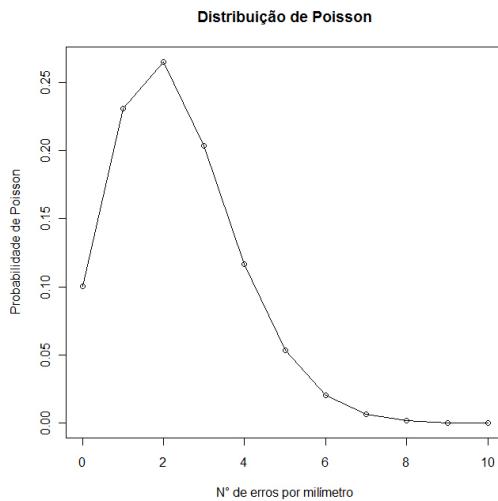


Figura 6.3: Gráfico do Exemplo 3.b: Distribuição Poisson de probabilidades

## 6.3 Variáveis Aleatórias Contínuas

### 6.3.1 Introdução

A corrente elétrica em um fio condutor ou o comprimento de uma peça são exemplos de experimentos aleatórios que apresentam a medida de interesse como um número real. Desta forma se pode ter uma precisão arbitrária da medida. A variável aleatória que representa uma medida, com um intervalo finito ou infinito de números reais para sua faixa denomina-se **variável aleatória contínua** ou **VAC**.

Como exemplos de variáveis aleatórias contínuas têm-se: a corrente elétrica, comprimento, pressão, temperatura, tempo, tensão, peso.

### 6.3.2 Distribuição Normal ou Gaussiana - $X \sim N(\mu, \sigma^2)$

É um dos mais importantes modelos de probabilidade para VACs. Aplicado em inúmeros fenômenos e muito utilizado no desenvolvimento teórico e na área da inferência estatística.

A distribuição gaussiana serve como aproximação para o cálculo de outras distribuições quando o número de observações aumenta. O teorema central do limite explica esta propriedade, ele diz que: “Toda a soma de variáveis aleatórias independentes de média finita e variância limitada é aproximadamente Normal, desde que o número de termos da soma seja suficientemente grande” [5]. Assim fica determinado que quando as réplicas de um experimento aleatório se aproximarem de um resultado médio, o experimento tenderá a ter uma distribuição normal na medida em que o número de réplicas se torne grande.

As variáveis aleatórias com diferentes médias e variâncias podem ser modeladas pelas funções densidade de probabilidade normal.  $E(X) = \mu$  determina o centro do gráfico em forma de sino, e  $V(X) = \sigma^2$  determina a largura da distribuição, como pode-se observar:

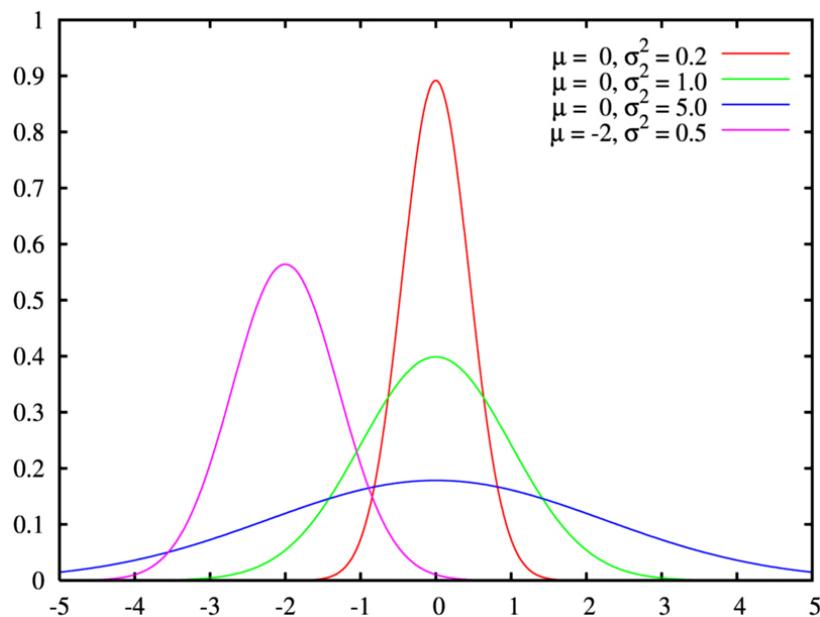


Figura 6.4: Distribuição Normal com diferentes parâmetros

A representação da função densidade de probabilidade é dada por:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (6.9)$$

Em que:

$$-\infty < x < \infty \quad (6.10)$$

$$\sigma > 0 \quad (6.11)$$

$$E(X) = \mu \quad (6.12)$$

$$V(X) = \sigma^2 \quad (6.13)$$

**Exemplo 4:** Um pesquisador coletou os dados da estatura de jovens em idade de alistamento militar. Sabe-se que a estatura de uma população segue a distribuição normal, com média 170 cm e variância 36 cm<sup>2</sup> (desvio padrão de 6 cm<sup>2</sup>).

- a) Qual a probabilidade de se encontrar um jovem com mais de 1,79 m de altura?
- b) Qual a altura em que a probabilidade de encontrarmos valores menores que ela seja de 80%?
- c) Represente graficamente a curva da distribuição normal para este problema e identifique as respostas dos itens a e b.

### Solução:

```
#Exemplo 4:
> #Item a)
> 1-pnorm(179,170,6) #pnorm(x,média,desvio padrão)
[1] 0.0668072
> #Item b)
> qnorm(0.8, 170,6)
[1] 175.0497
> #Item c)
> curve(dnorm(x,170,6),170-3*6,170+3*6,xlab="Alturas (cm)",
+ ylab="Probabilidade de se encontrar a altura x",
+ main="Distribuição Normal")
> lines(c(179,179),c(0,0.022),col="red")
> lines(c(175.0497,175.0497),c(0,0.0465),col="blue")
```

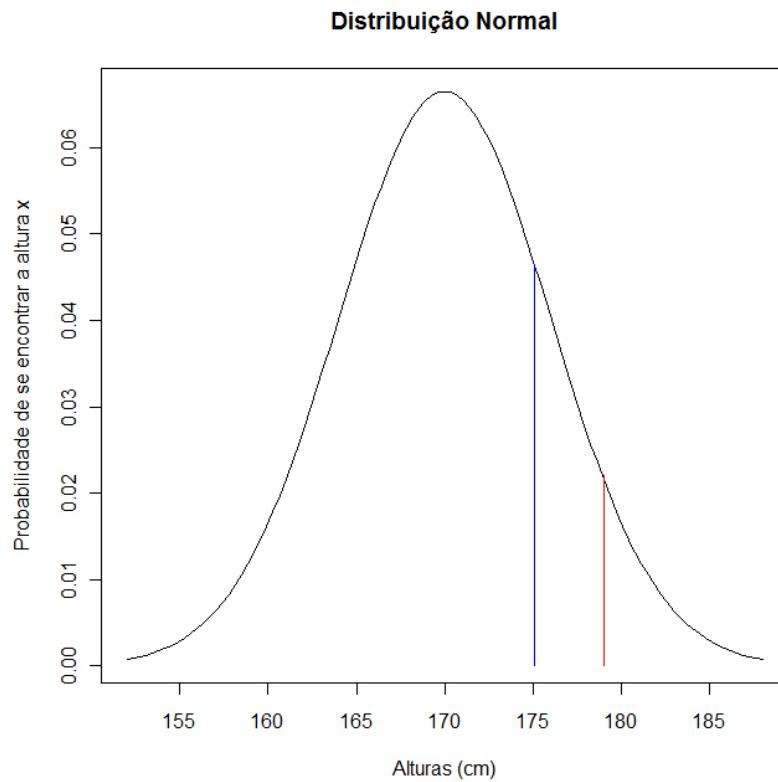


Figura 6.5: Distribuição normal do Exemplo 4.c

### 6.3.3 Distribuição de Weibull - $X \sim W(\delta, \beta)$

A distribuição Weibull, geralmente é usada para modelar o tempo até uma falha de sistemas físicos (diversos tipos). Os parâmetros da distribuição fornecem grande flexibilidade para modelar sistemas em que o número de falhas aumenta com o tempo, diminui com o tempo, ou permanece constante com o tempo.

Sua função densidade de probabilidade é representada por:

$$f(x) = \frac{\beta}{\delta} \left( \frac{x}{\delta} \right)^{\beta-1} e^{-\left( \frac{x}{\delta} \right)^\beta} \quad (6.14)$$

onde a média e a variância de X são, respectivamente:

$$\mu = \delta \Gamma \left( 1 + \frac{1}{\beta} \right) \quad (6.15)$$

$$\sigma^2 = \delta^2 \Gamma \left( 1 + \frac{2}{\beta} \right) - \delta^2 \left[ \Gamma \left( 1 + \frac{1}{\beta} \right) \right]^2 \quad (6.16)$$

**Exemplo 5:** O tempo de falha (em horas) de um mancal em um eixo mecânico é satisfatoriamente modelado como uma variável aleatória de Weibull com  $\beta = \frac{1}{2}$  e  $\delta = 5000$ .

- a) Determine a probabilidade de um mancal durar no mínimo 6000 h.
- b) Verificar a probabilidade de um mancal ter problemas nas primeiras 6000 h. Plotar

o resultado.

**Solução:**

```
#Exemplo 5:  
> 1-pweibull(6000,0.5,5000)  
[1] 0.3343907  
> x <- 0:6000  
> curve(dweibull(x,0.5,5000),0,7000) #dweibull(x, beta,delta)
```

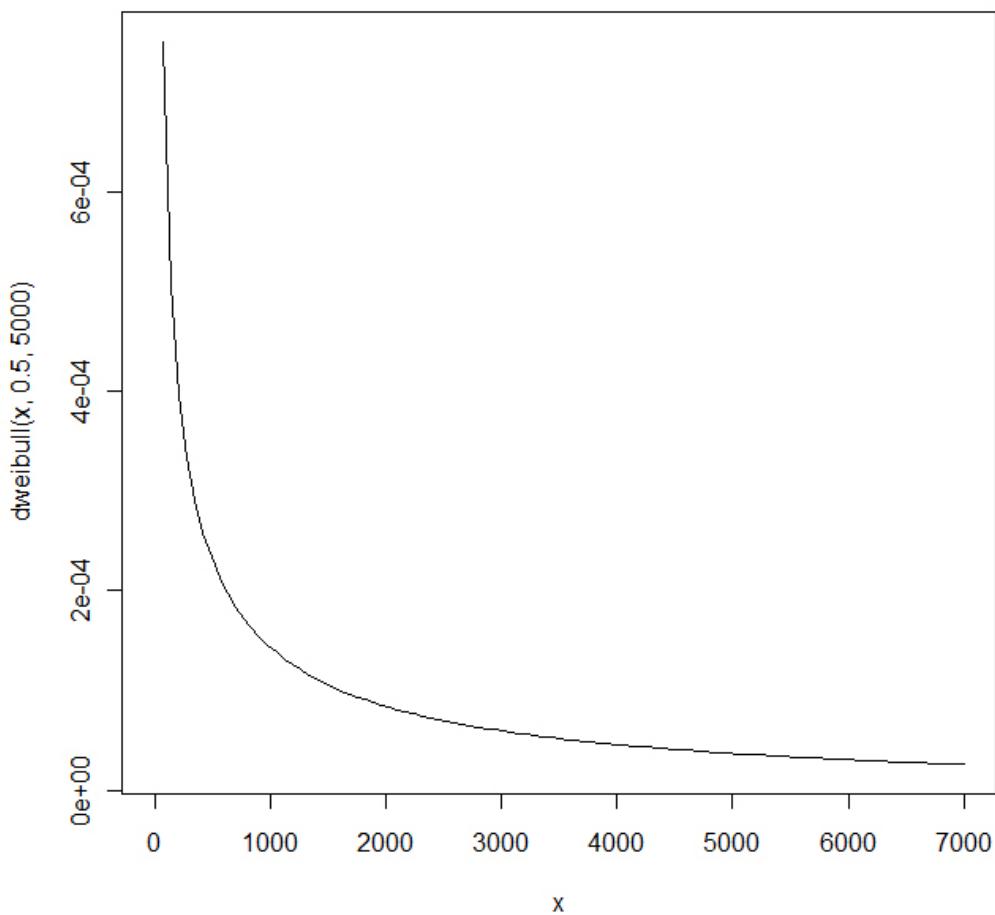


Figura 6.6: Gráfico do Exemplo 5: Distribuição Weibull

### 6.3.4 Distribuições no R

Abaixo é possível conferir algumas distribuições disponíveis no R.

Distribuição/função	Função
<b>beta</b>	<code>dbeta(n, shape1, shape2)</code>
<b>binomial</b>	<code>dbinom(n, size, prob)</code>
<b>binomial negativa</b>	<code>dnbinom(n, size, prob)</code>
<b>Cauchy</b>	<code>dcauchy(n, location=0, scale=1)</code>
<b>estatística de Wilcoxon's</b>	<code>dwilcox(nn, m, n, n), dsignrank(nn,n)</code>
<b>exponencial</b>	<code>dexp(n, rate=1)</code>
<b>Fischer-Snedecor(F)</b>	<code>df(n, df1, df2)</code>
<b>gamma</b>	<code>dgamma(n, shape, scale=1)</code>
<b>Gauss(normal)</b>	<code>dnorm(n, mean=0, sd=1)</code>
<b>geométrica</b>	<code>dgeom(n, prob)</code>
<b>hipergeométrica</b>	<code>dhyper(nn, m, n, k)</code>
<b>logística</b>	<code>dlogis(n, location=0, scale=1)</code>
<b>log-normal</b>	<code>dlnorm(n, meanlog=0, sdlog=1)</code>
<b>Poisson</b>	<code>dpois(n, lambda)</code>
<b>qui-quadrado</b>	<code>dchisq(n, df)</code>
<b>"Student" (t)</b>	<code>dt(n, df)</code>
<b>uniforme</b>	<code>dunif(n, min=0, max=1)</code>
<b>Weibull</b>	<code>dweibull(n, shape, scale=1)</code>

Figura 6.7: Funções de Distribuições disponíveis no R

## 7

# Inferência Estatística

---

## 7.1 Introdução

Muitos problemas, especialmente na área de engenharia, requerem que decidamos entre aceitar ou rejeitar uma hipótese acerca de algum parâmetro. A afirmação é chamada de hipótese e o procedimento de tomada de decisão sobre a hipótese é chamado de teste de hipóteses.

Definição: Uma hipótese estatística é uma afirmação sobre os parâmetros de uma ou mais populações.

É importante destacar que hipóteses são sempre afirmações sobre a população ou distribuição sob estudo, não afirmações sobre a amostra.

Em um teste de hipótese trabalhamos sempre com duas afirmações: a hipótese nula ( $H_0$ ) e a hipótese alternativa ( $H_1$ ). O valor do parâmetro especificado na hipótese nula é geralmente determinado por uma de três formas:

- Ele é resultado de experiência passada, conhecimento do processo ou de testes e experimentos prévios.
- É determinado a partir de alguma teoria ou do modelo relativo ao processo sob estudo.
- É resultado de considerações externas, tais como projeto ou especificações de engenharia ou obrigações contratuais.

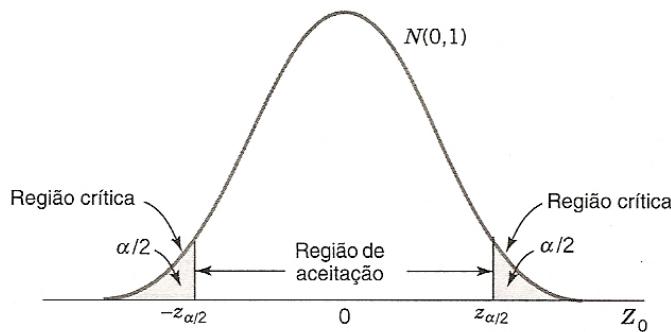
Trataremos aqui a hipótese nula de forma que ela seja sempre estabelecida especificando um valor exato do parâmetro (igualdade). Já a hipótese alternativa permitirá ao parâmetro assumir vários valores, dependendo do tipo de hipótese em estudo.

### 7.1.1 Hipóteses Unilaterais e Bilaterais

Um teste de qualquer hipótese, tal como

$$\begin{aligned} H_0 &: \mu = \mu_0 \\ H_1 &: \mu \neq \mu_0 \end{aligned}$$

é chamado de teste **bilateral**, porque é importante detectar diferenças em relação ao valor da média  $\mu_0$  usado na hipótese, que esteja em ambos os lados de  $\mu_0$ . Em tal caso, a região critica é dividida em duas partes com (geralmente) igual probabilidade colocada em cada extremidade da distribuição da estatística do teste.



A distribuição de  $Z_0$  quando  $H_0: \mu = \mu_0$  for verdadeira, com região critica para  $H_1: \mu \neq \mu_0$ .

Figura 7.1: Gráfico de um teste bilateral para a média de distribuição normal

Muitos problemas de teste de hipótese envolvem, naturalmente, uma hipótese alternativa **unilateral**, tal como:

$$H_0: \mu = \mu_0$$

$$H_1: \mu < \mu_0$$

ou

$$H_0: \mu = \mu_0$$

$$H_1: \mu > \mu_0$$

Se a hipótese alternativa for  $H_1: \mu > \mu_0$ , a região crítica deve estar na extremidade superior da distribuição da estatística de teste, enquanto se a hipótese alternativa for  $H_1: \mu < \mu_0$ , a região critica deve estar na extremidade inferior da distribuição. Esses testes são chamados **unilaterais**.

## 7.2 Testes de Hipótese - Uma amostra

### 7.2.1 Introdução

Esta seção trata de inferências acerca dos parâmetros média, variância e proporção de uma população simples.

### 7.2.2 Teste para a Média

Podemos realizar o teste de hipótese para a média de uma população normal em duas situações:

#### Variância conhecida

Neste caso utiliza-se a seguinte estatística de teste:

$$Z_0 = \frac{\bar{X} - \mu_0}{\frac{\sigma}{\sqrt{n}}} \tag{7.1}$$

e o seguinte intervalo de confiança:

$$\bar{x} - z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{x} + z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \tag{7.2}$$

## Variância desconhecida

Neste caso utiliza-se a seguinte estatística de teste:

$$T_0 = \frac{\bar{X} - \mu_0}{\frac{s}{\sqrt{n}}} \quad (7.3)$$

e o seguinte intervalo de confiança:

$$\bar{x} - t_{\frac{\alpha}{2}, n-1} \frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\frac{\alpha}{2}, n-1} \frac{s}{\sqrt{n}} \quad (7.4)$$

Nesta apostila trabalharemos apenas com o teste “t” (para variância desconhecida) em ambos os casos. Isso se justifica pelo fato de que o teste “z” é uma particularidade do teste “t” para um número razoável de amostras ( $n > 30$ ) e a utilização de “t” em casos de variância conhecida não é um problema, pelo contrário, cobre o teste para os casos de erro do tipo I.

A função de teste de hipótese para uma média é chamada “*t.test()*”. Posteriormente veremos que a mesma função servirá para comparação entre duas médias. Veja abaixo a sintaxe da função:

#sintaxe:

*t.test(x, alternative = "hipótese", mu = media, conf.level = β)*

onde:

*hipótese* - pode assumir os seguintes valores:

- greater (teste unilateral de  $H_0$  maior que a média)
- less (teste unilateral de  $H_0$  menor que a média)
- two.sided (teste bilateral de  $H_0$  igual à média)

*media* - valor da média a ser testada

*β* - confiança do teste ( $1 - \alpha_{(significância)}$ )

Para entender como utilizar a função, faremos um exemplo retirado de Montgomery [5].

**Exemplo:** Um artigo periódico *Materials Engineering* (1989, Vol. II, No. 4, pp. 275-281) descreve os resultados de teste de tensão quanto à adesão em 22 corpos de prova de liga U-700. A carga no ponto de falha do corpo de prova é dada a seguir (em MPa):

19,8	18,5	17,6	16,7	15,8
15,4	14,1	13,6	11,9	11,4
11,4	8,8	7,5	15,4	15,4
19,5	14,9	12,7	11,9	11,4
10,1	7,9			

O objetivo é verificar se os dados sugerem que a carga média na falha excede 10 MPa. Considerar que a carga na falha tem uma distribuição normal e utilizar  $\alpha = 0.05$ .

Primeiramente, devemos associar os dados da tabela a um vetor. Feito isso, devemos realizar um teste unilateral a fim de verificar se a média na falha é maior que a 10 MPa. A seguir podemos verificar a resolução utilizando a função “*t.test()*” no R:

```
#Exemplo:
> x<-c(19.8, 18.5, 17.6, 16.7, 15.8, 15.4, 14.1, 13.6,
+ 11.9, 11.4, 11.4, 8.8, 7.5, 15.4, 15.4, 19.5,
+ 14.9, 12.7, 11.9, 11.4, 10.1, 7.9)
> x
[1] 19.8 18.5 17.6 16.7 15.8 15.4 14.1 13.6 11.9 11.4 11.4
[12] 8.8 7.5 15.4 15.4 19.5 14.9 12.7 11.9 11.4 10.1 7.9
> t.test(x,alternative="greater",mu=10,conf.level=0.95)

One Sample t-test

data: x
t = 4.9017, df = 21, p-value = 3.781e-05
alternative hypothesis: true mean is greater than 10
95 percent confidence interval:
12.40996      Inf
sample estimates:
mean of x
13.71364
```

A função retorna a hipótese alternativa, o intervalo de confiança, a média amostral, o valor de “t” calculado (valor crítico), os graus de liberdade da distribuição e o p-value.

P-value é a probabilidade de obter um resultado pelo menos muito próximo do que o valor que foi testado, assumindo que a hipótese nula é verdadeira. O fato de o P-value ser baseado nessa premissa é crucial para sua correta interpretação. Um P-valor próximo de 0 indica que a hipótese nula é falsa. Quando próximo de 1, indica que não há evidências suficientes para rejeitar a hipótese nula.

Para interpretar os testes de hipótese no R, temos que comparar o P-value com a significância testada:

$p\text{-value} > \alpha \longrightarrow \text{"Aceita"} H_0$ $p\text{-value} < \alpha \longrightarrow \text{Rejeita } H_0$
--------------------------------------------------------------------------------------------------------------------------------

Em nosso caso, percebemos que *p-value* é muito menor que  $\alpha$ . Logo, através das evidências amostrais, rejeita-se a hipótese nula ( $H_0 : \mu = 10$  MPa), evidenciando que a carga média é maior que 10 MPa ( $H_1 : \mu > 10$  MPa).

### 7.2.3 Teste para a Variância de uma população normal

Algumas vezes são necessários testes de hipótese e intervalos de confiança para a variância de uma população. O teste que estudaremos é válido quando a população for modelada por uma distribuição normal.

A estatística de teste é dada pela seguinte fórmula:

$$X_0^2 = \frac{(n - 1)S^2}{\sigma_0^2} \quad (7.5)$$

com os seguintes intervalos de confiança:

- Intervalo superior

$$\sigma^2 \leq \frac{(n - 1)s^2}{\chi_{1-\alpha,n-1}^2} \quad (7.6)$$

- Intervalo inferior

$$\frac{(n - 1)s^2}{\chi_{\alpha,n-1}^2} \leq \sigma^2 \quad (7.7)$$

Devemos entender que a estatística desse teste segue uma distribuição qui-quadrado com  $n-1$  graus de liberdade, abreviada por  $\chi_{n-1}$ . O R não possui nenhuma função pronta para calcular esse teste para a variância. Portanto, utilizaremos a estatística de teste e o conceito de P-valor para criar uma função simples deste caso. Para tanto, utilizaremos um exemplo de [5].

**Exemplo:** Uma máquina automática de enchimento é usada para encher garrafas com detergente líquido. Uma amostra aleatoria de 20 garrafas resulta em uma variância da amostra do volume de enchimento de  $s^2 = 0,0153$  (onça fluída)<sup>2</sup>. Se a variância do volume exceder 0,01 (onça fluída)<sup>2</sup>, existirá uma proporção inaceitável de garrafas cujo enchimento não foi completo e cujo enchimento foi em demasia. Há evidências nos dados da amostra sugerindo que o fabricante tenha um problema com garrafas cheias com falta e excesso de detergente? Use  $\alpha = 0,05$  e considere que o volume de enchimento tenha uma distribuição normal.

Para resolver este problema, devemos calcular a estatística de teste e verificar o p-valor dessa estatística. Caso o P-valor seja menor que o alfa desejado, devemos rejeitar a hipótese nula, “aceitando”  $H_1 : \sigma^2 > 0,01$ . Vejamos como resolver isso no R:

```
#Exemplo:
> nam = 20 #n = tamanho da amostra
> varS = 0.0153 #varS = variância amostral
> sig0 = 0.01 #sig0 = valor de variância testada
> alfa = 0.05 #alfa = significância do teste
> #estatística de teste
> quicalc = (nam-1)*varS/sig0
> quicalc
[1] 29.07
> #pvalor do teste
> pvalor=pchisq(q=quicalc,df=nam-1, lower.tail = F)
> pvalor
[1] 0.064892
```

Podemos observar no exemplo resolvido que se utilizou a função de probabilidade da distribuição qui-quadrado para obter o *p-value*. O argumento “lower.tail=F” indica

que queremos um teste do tipo  $H_1 : \sigma^2 > \sigma_0^2$ . A análise aqui é a mesma: como  $p-value > \alpha$ , não existem evidências amostrais para rejeitar a hipótese nula. Portanto, não há evodências suficientes para concluirmos que o volume do enchimento excede 0,01 (onça fluída)<sup>2</sup>.

Vamos generalizar este exercício criando uma função simples para teste de uma variância. Veja abaixo a função e o resultado para o mesmo exemplo anterior:

```
onevartest <- function(varS, nam, sig0, alfa, tail)
{
  #nam = tamanho da amostra
  #varS = variância amostral
  #sig0 = valor de variância testada
  #alfa = significância do teste

  quicalc = (nam-1)*varS/sig0; #estatística de teste

  #Testes unilaterais apenas

  if(tail==0) #intervalo superior
  {
    #pvalor do teste
    pvalor=pchisq(q=quicalc,df=nam-1, lower.tail = F)
    Intervalo=((nam-1)*varS)/(qchisq(p=alfa,df=nam-1, lower.tail=F))
    print("P-Valor e Valor crítico do intervalo superior")
  }

  else #intervalo inferior
  {
    #pvalor do teste
    pvalor=pchisq(q=quicalc,df=nam-1, lower.tail = T);
    Intervalo=((nam-1)*varS)/(qchisq(p=alfa,df=nam-1, lower.tail=T))
    print("P-Valor e Valor crítico do intervalo inferior")
  }

  return(list(c("Pvalor =", pvalor),c("Intervalo = ", Intervalo)))
}

#Exemplo:
> onevartest(0.0153,20,0.01,0.05,0)
[1] "P-Valor e Valor crítico do intervalo superior"
[[1]]
[1] "Pvalor ="           "0.0648920049393868"

[[2]]
[1] "Intervalo = "       "0.00964386145047913"
```

## 7.2.4 Teste para uma Proporção Binomial

Testes de proporção estão relacionados com variáveis aleatórias que possuem parâmetros de distribuições binomiais. Frequentemente é necessário testar hipóteses e construir intervalos de confiança para proporções de uma população. A metodologia para os testes de proporções é a mesma utilizada anteriormente. O teste de hipótese para uma proporção binomial tem a seguinte estatística de teste:

$$Z_0 = \frac{X - np_0}{\sqrt{np_0(1 - p_0)}} \quad (7.8)$$

e o seguinte intervalo de confiança:

$$\hat{p} - z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}} \leq p \leq \hat{p} + z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}} \quad (7.9)$$

A função do teste de hipótese para uma proporção é chamada “*prop.test()*”. Veja abaixo a sintaxe da função:

#sintaxe:

*prop.test(x, p = prop, alternative = "hipótese", conf.level = β)*

onde:

*x* - é uma matriz contendo o número de sucessos e falhas, respectivamente.

*hipótese* - pode assumir os seguintes valores:

- greater (teste unilateral de  $H_0$  maior que a proporção  $p$ )
- less (teste unilateral de  $H_0$  menor que a proporção  $p$ )
- two.sided (teste bilateral de  $H_0$  igual proporção  $p$ )

*prop* - valor da proporção a ser testada

*β* - confiança do teste ( $1 - \alpha_{(significância)}$ )

**Exemplo:** Queremos provar, através de dados amostrais, que a fração defeituosa num processo de fabricação de semicondutores não excede 6% do total produzido. Para tanto, retirou-se uma amostra aleatória de 200 semicondutores e se encontrou 4 defeituosos.

Para resolver esse problema, vamos gerar uma matriz (de uma linha e duas colunas) que contenha o total de sucessos (4) e falhas (196), respectivamente. Como estamos observando as falhas do processo, nosso sucesso serão os semicondutores defeituosos. Após, escolhemos o tipo de teste (neste caso, unilateral;  $H_1 : p < 0.06$ ) e o grau de confiança do teste. A solução do exemplo está a seguir:

```
#Exemplo:  
> x = matrix(c(4,196),1,2)  
> x  
[,1] [,2]  
[1,]    4   196  
> prop.test(x,p=0.06,alternative="less",conf.level=0.97)  
  
 1-sample proportions test with continuity correction  
  
data: x, null probability 0.06  
X-squared = 4.9867, df = 1, p-value = 0.01277  
alternative hypothesis: true p is less than 0.06  
97 percent confidence interval:  
 0.00000000 0.05203808  
sample estimates:  
 p  
0.02
```

A interpretação do teste é dada pela mesma forma explicada em [7.2.2](#), com o P-value. Como temos um P-value menor que nossa significância (3%), rejeitamos a hipótese nula, evidenciando que a proporção de peças defeituosas do processo é menor que 6%.

## 7.3 Testes de Hipótese - Duas amostras

### 7.3.1 Introdução

Esta seção trata de inferências acerca dos parâmetros média, variância e proporção de duas populações independentes.

### 7.3.2 Teste para a média

Podemos realizar testes para a diferença entre duas médias em dois casos: com variâncias conhecidas ou variâncias desconhecidas. Ainda para variâncias desconhecidas, podemos afirmar que elas são iguais ou diferentes. Veja nas figuras abaixo (retiradas de [\[5\]](#)) a definição para cada situação e também as definições dos intervalos de confiança:

<b>Teste de Hipóteses para <math>\mu_1 - \mu_2</math> com Variâncias Conhecidas</b>	
Hipótese nula:	$H_0: \mu_1 - \mu_2 = \Delta_0$
Estatística de teste:	$Z_0 = \frac{\bar{X}_1 - \bar{X}_2 - \Delta_0}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$
Hipóteses Alternativas	Critério de Rejeição
$H_1: \mu_1 - \mu_2 \neq \Delta_0$	$z_0 > z_{\alpha/2}$ ou $z_0 < -z_{\alpha/2}$
$H_1: \mu_1 - \mu_2 > \Delta_0$	$z_0 > z_\alpha$
$H_1: \mu_1 - \mu_2 < \Delta_0$	$z_0 < -z_\alpha$

Figura 7.2: Teste para a diferença de duas médias com variâncias conhecidas

<b>Definição: Intervalo de Confiança para a Diferença entre Duas Médias com Variâncias Conhecidas</b>	
Se $\bar{x}_1 - \bar{x}_2$ forem as médias de duas amostras aleatórias independentes de tamanhos $n_1$ e $n_2$ , provenientes de populações com variâncias conhecidas $\sigma_1^2$ e $\sigma_2^2$ , respectivamente, então um intervalo de confiança de $100(1 - \alpha)\%$ para $\mu_1 - \mu_2$ é	
	$\bar{x}_1 - \bar{x}_2 - z_{\alpha/2} \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \leq \mu_1 - \mu_2 \leq \bar{x}_1 - \bar{x}_2 + z_{\alpha/2} \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$
	sendo $z_{\alpha/2}$ o ponto percentual superior $\alpha/2$ da distribuição normal.

Figura 7.3: Intervalo de confiança para a diferença de duas médias com variâncias conhecidas

<u>O Teste <i>t</i> Combinado para Duas Amostras<sup>1</sup></u>	
Hipótese nula:	$H_0: \mu_1 - \mu_2 = \Delta_0$
Estatística de teste:	$T_0 = \frac{\bar{X}_1 - \bar{X}_2 - \Delta_0}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$
<u>Hipótese Alternativa</u>	<u>Critério de Rejeição</u>
$H_1: \mu_1 - \mu_2 \neq \Delta_0$	$t_0 > t_{\alpha/2, n_1 + n_2 - 2}$ OU $t_0 < -t_{\alpha/2, n_1 + n_2 - 2}$
$H_1: \mu_1 - \mu_2 > \Delta_0$	$t_0 > t_{\alpha, n_1 + n_2 - 2}$
$H_1: \mu_1 - \mu_2 < \Delta_0$	$t_0 < -t_{\alpha, n_1 + n_2 - 2}$

Figura 7.4: Teste para a diferença de duas médias com variâncias desconhecidas

<u>Definição: Intervalo de Confiança para a Diferença nas Médias de Duas Distribuições Normais, com Variâncias Desconhecidas, porém Iguais</u>
Se $\bar{x}_1$ , $\bar{x}_2$ , $s_1^2$ e $s_2^2$ forem as médias e as variâncias de duas amostras aleatórias de tamanhos $n_1$ e $n_2$ , respectivamente, provenientes de duas populações normais independentes, com variâncias desconhecidas, porém iguais, então um intervalo de confiança de $100(1 - \alpha)\%$ para a diferença nas médias $\mu_1 - \mu_2$ é
$\bar{x}_1 - \bar{x}_2 - t_{\alpha/2, n_1 + n_2 - 2} S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$
$\leq \mu_1 - \mu_2 \leq \bar{x}_1 - \bar{x}_2 + t_{\alpha/2, n_1 + n_2 - 2} S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$
em que
$S_p = \sqrt{[(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2]/(n_1 + n_2 - 2)}$
é a estimativa combinada do desvio-padrão comum da população comum e $t_{\alpha/2, n_1 + n_2 - 2}$ é o ponto percentual superior $\alpha/2$ da distribuição <i>t</i> com $n_1 + n_2 - 2$ graus de liberdade.

Figura 7.5: Intervalo de confiança para a diferença de duas médias com variâncias desconhecidas, porém iguais

Conforme explicado na Seção 7.2.2, trabalharemos apenas com o teste “*t*” (para variância desconhecida) em ambos os casos.

A sintaxe para realizar um teste para duas amostras (no caso da comparação de duas médias) é bastante semelhante ao teste para uma amostra. A única diferença é um segundo vetor numérico de dados que será comparado com o primeiro. Veja abaixo:

**Definição: Intervalo de Confiança para a Diferença nas Médias de Duas Distribuições Normais com Variâncias Desconhecidas e Desiguais**

Se  $\bar{x}_1$ ,  $\bar{x}_2$ ,  $s_1^2$  e  $s_2^2$  forem as médias e as variâncias de duas amostras aleatórias de tamanhos  $n_1$  e  $n_2$ , respectivamente, provenientes de duas populações normais independentes, com variâncias desconhecidas e desiguais, então um intervalo de confiança de  $100(1 - \alpha)\%$  para a diferença nas médias  $\mu_1 - \mu_2$  é

$$\bar{x}_1 - \bar{x}_2 - t_{\alpha/2, v} \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \leq \mu_1 - \mu_2 \leq \bar{x}_1 - \bar{x}_2 + t_{\alpha/2, v} \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

em que  $v$  é dado pela Eq. 9.18 e  $t_{\alpha/2, v}$  é o ponto percentual superior  $\alpha/2$  da distribuição  $t$ , com  $v$  graus de liberdade.

Figura 7.6: Intervalo de confiança para a diferença de duas médias com variâncias desconhecidas e diferentes

#sintaxe:

`t.test(x, y, alternative = "hipótese", mu = difmedias, conf.level = β)`

onde:

*hipótese* - pode assumir os seguintes valores:

- greater (teste unilateral de  $H_0 : \mu_x > \mu_y$ )
- less (teste unilateral de  $H_0 : \mu_x < \mu_y$ )
- two.sided (teste bilateral de  $H_0 : \mu_x = \mu_y$ )

*difmedias* - testar ( $H_1$ ) se  $\mu_x - \mu_y$  é maior, menor ou diferente do valor “difmedias”  
 $\beta$  - confiança do teste ( $1 - \alpha(significancia)$ )

Vamos estudar o teste a partir de um exemplo retirado de [5].

**Exemplo:** Dois catalisadores estão sendo analisados para determinar como eles afetam o rendimento médio de um processo químico. Especificamente, o catalisador 1 está correntemente em uso, mas o catalisador 2 é aceitável. Uma vez que o catalisador 2 é mais barato, ele deve ser adotado, desde que ele não mude o rendimento do processo. Um teste é feito em uma planta piloto, resultando nos dados mostrados na tabela abaixo. Há alguma diferença entre os rendimentos médios? Use  $\alpha = 0.05$  e considere variâncias iguais.

Número da Observação	Catalisador 1	Catalisador 2
1	91,50	89,19
2	94,18	90,95
3	92,18	90,46
4	95,39	93,21
5	91,79	97,19
6	89,07	97,04
7	94,72	91,07
8	89,21	92,75

O objetivo é verificar se há diferença entre as médias de rendimento do catalisador 1 e do catalisador 2.

Primeiramente, devemos associar os dados da tabela a dois vetores ( $x$  e  $y$ , respectivamente). Feito isso, devemos realizar um teste bilateral a fim de verificar se  $\mu_x \leq \mu_y$ . A seguir podemos verificar a resolução utilizando a função “`t.test()`” no R:

```
#Exemplo:
> x <- c(91.50, 94.18, 92.18, 95.39,
+ 91.79, 89.07, 94.72, 89.21)
> y <- c(89.19, 90.95, 90.46, 93.21,
+ 97.19, 97.04, 91.07, 92.75)
> x
[1] 91.50 94.18 92.18 95.39 91.79 89.07 94.72
[8] 89.21
> y
[1] 89.19 90.95 90.46 93.21 97.19 97.04 91.07
[8] 92.75
> t.test(x,y,alternative="two.sided",mu=0)

Welch Two Sample t-test

data: x and y
t = -0.3536, df = 13.353, p-value = 0.7292
alternative hypothesis: true difference in means isn't equal to 0
95 percent confidence interval:
-3.387118 2.432118
sample estimates:
mean of x mean of y
92.2550 92.7325
```

Conforme o resultado do teste, percebemos que  $p\text{-value}$  é maior que  $\alpha$ . Portanto, não há evidências amostrais suficientes para rejeitar a hipótese nula  $H_0 : \mu_x = \mu_y$ . Concluímos então que não há diferença entre os rendimentos dos catalisadores.

### 7.3.3 Teste para as variâncias de duas populações normais

Esta seção apresentará um teste para as variâncias de duas populações normais. A estatística deste teste é dada pela razão F (conhecida como teste F):

$$F_0 = \frac{S_1^2}{S_2^2} \quad (7.10)$$

e possui o seguinte intervalo de cofiança:

$$\frac{s_1^2}{s_2^2} f_{1-\alpha/2, n_2-1, n_1-1} \leq \frac{\sigma_1^2}{\sigma_2^2} \leq \frac{s_1^2}{s_2^2} f_{\alpha/2, n_2-1, n_1-1} \quad (7.11)$$

Para realizar este teste, contaremos com a função “`var.test()`”, cuja sintaxe está descrita abaixo:

#sintaxe:

`var.test(x, y, ratio =  $\frac{s_x^2}{s_y^2}$ , alternative = "hipótese", conf.level =  $\beta$ )`

onde:

$x, y$  - são vetores numéricos das populações estudadas

$\frac{s_x^2}{s_y^2}$  - a razão a ser testada (default: 1)

*hipótese* - pode assumir os seguintes valores:

- greater (teste unilateral de  $H_0 : \frac{s_x^2}{s_y^2} > ratio$ )
- less (teste unilateral de  $H_0 : \frac{s_x^2}{s_y^2} < ratio$ )
- two.sided (teste bilateral de  $H_0 : \frac{s_x^2}{s_y^2} = ratio$ )

$\beta$  - confiança do teste ( $1 - \alpha(significância)$ )

Resolveremos um exemplo de [5] para aplicar esta função.

**Exemplo:** Uma companhia fabrica propulsores para uso em motores de turbinas de avião. Uma das operações envolve esmerilhar o acabamento de uma superfície particular para um componente de liga de titânio. Dois processos diferentes para esmerilhar podem ser usados, podendo produzir peças com iguais rugosidades médias na superfície. Uma amostra aleatória de  $n_1 = 11$  peças, proveniente do primeiro processo, resulta em um desvio-padrão de  $s_1 = 5,1$  micropolegadas. Uma amostra aleatória de  $n_2 = 16$  peças, proveniente do segundo processo, resulta em um desvio-padrão de  $s_2 = 4,7$  micropolegadas. Verifique se a razão entre as duas variâncias  $\frac{\sigma_1^2}{\sigma_2^2}$  é diferente de 1 com um nível de confiança de 90%. Considere que os dois processos sejam diferentes e a rugosidade na superfície seja normalmente distribuída.

Podemos encontrar a solução para esse problema facilmente. Basta criarmos dois vetores com distribuições normais aleatórias, especificando o tamanho da amostra e o desvio-padrão de cada população. Após isso, basta aplicarmos a função “`var.test()`”. Veja a solução a seguir:

```
#Exemplo:
> x <- rnorm(11, sd=5.1)
> y <- rnorm(16, sd=4.7)
> x
[1] -1.0894190 2.6676766 4.5793885 1.2510806 7.3023286
[6] -3.6150530 -3.1051404 -6.7414156 0.1396896 1.3428483
[11] -1.5461805
> y
[1] -3.1878255 5.7939292 -3.4140953 -5.9682770 5.8940996
[6] 0.1104748 -0.5214162 0.5847299 1.8692529 3.7037109
[11] -5.5122767 -1.4200343 4.6508673 -3.3598818 0.3133970
[16] 4.3929997
> var.test(x,y,ratio=1,alternative="t",conf.level=0.9)

      F test to compare two variances

data: x and y
F = 1.032, num df = 10, denom df = 15, p-value = 0.9264
alternative hypothesis: true ratio of variances is not equal to 1
90 percent confidence interval:
0.4056976 2.9360014
sample estimates:
ratio of variances
1.031980
```

Como é possível observar, o *p-value* é maior que nossa significância  $\alpha$ . Logo, não há evidências amostrais suficientes para rejeitarmos  $H_0$  e afirmar que as variâncias são diferentes, o que nos leva a aceitar a hipótese de que a razão entre as variâncias é 1.

### 7.3.4 Teste para duas proporções

Esta seção abordará testes para duas proporções. É comum quando se comparam resultados binomiais de duas populações (casos clássicos como pesquisas de opinião entre homens e mulheres, estudos epidemiológicos, falhas apresentadas em dois processos de fabricação para um mesmo produto, etc.). Abaixo é dada a definição para a estatística desse teste:

$$Z = \frac{\hat{P}_1 - \hat{P}_2 - (p_1 - p_2)}{\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}} \quad (7.12)$$

que tem o seguinte intervalo de confiança:

$$\hat{p}_1 - \hat{p}_2 - z_{\alpha/2} \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}} \leq \hat{p}_1 - \hat{p}_2 \leq \hat{p}_1 - \hat{p}_2 + z_{\alpha/2} \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}} \quad (7.13)$$

Semelhante a Seção 7.3.2, usaremos o “*prop.test()*”. Mas desta vez, ao invés de utilizar uma matriz, usaremos dois vetores: um vetor linha de dois elementos para os

sucessos das populações 1 e 2 e outro vetor de mesma dimensão cujos elementos serão justamente as populações 1 e 2. Veja abaixo a sintaxe:

```
#sintaxe:
```

```
prop.test(X = c(x1, x2), Y = c(n1, n2), alternative = "hipótese", conf.level = β)
```

onde:

$X$  - vetor com os sucessos  $x_1$  (da população 1) e  $x_2$  (da população 2)

$Y$  - vetor com os tamanhos  $n_1$  (da população 1) e  $n_2$  (da população 2)

*hipótese* - pode assumir os seguintes valores:

- greater (teste unilateral de  $H_0 : \hat{p}_1 > \hat{p}_2$ )
- less (teste unilateral de  $H_0 : \hat{p}_1 < \hat{p}_2$ )
- two.sided (teste bilateral de  $H_0 : \hat{p}_1 = \hat{p}_2$ )

$\beta$  - confiança do teste ( $1 - \alpha_{(significância)}$ )

Para entender o uso da função aplicada a duas amostras, resolveremos um exemplo retirado de [5].

**Exemplo:** O experimento, em 1954, da vacina Salk contra a poliomielite focou a sua eficiência de compate à paralisia. Notou-se que, sem um grupo de controle de crianças, não haveria uma base para avaliar a eficácia da vacina Salk. Então a vacina foi administrada a um grupo e um placebo (visualmente idêntico à vacina, porém sem efeito algum) foi administrado a um segundo grupo. Por razões éticas e por se suspeitar de que o conhecimento da administração da vacina afetaria os diagnósticos subsequentes, o experimento foi conduzido de uma maneira tal que a identidade das crianças não fosse revelada. Ou seja, nem os indivíduos e nem os administradores sabiam quem havia recebido a vacina e quem havia recebido o placebo. Os dados reais para esse experimento são apresentados a seguir:

Grupo do Placebo	Grupo da Vacina
$n = 201299$	$n = 200745$
Casos observados de pólio: 110	Casos observados de pólio: 33

(a) Use o procedimento de teste de hipótese para determinar se a proporção de crianças nos dois grupos que contraiu paralisia é estatisticamente diferente. Use  $\alpha = 0.05$ .

(b) Repita o item (a) com  $\alpha = 0.01$ .

Para resolver este problema no R, primeiramente devemos criar os vetores X (sucessos  $x_1$  e  $x_2$ ; no caso, casos de pólio) e Y (tamanho das populações). Após isso, simplesmente aplicamos o teste conforme a sintaxe. Veja a resolução abaixo:

```
#Exemplo:  
> X = c(110,33)  
> Y = c(201299, 200745)  
> X  
[1] 110 33  
> Y  
[1] 201299 200745  
> #Item (a)  
> prop.test(X,Y,alternative="t")  
  
 2-sample test for equality of proportions  
    with continuity correction  
  
data: X out of Y  
X-squared = 40.1968, df = 1, p-value = 2.296e-10  
alternative hypothesis: two.sided  
95 percent confidence interval:  
 0.0002606084 0.0005035179  
sample estimates:  
 prop 1      prop 2  
0.0005464508 0.0001643877  
  
> #Item (b)  
> prop.test(X,Y,alternative="t",conf.level=0.99)  
  
 2-sample test for equality of proportions  
    with continuity correction  
  
data: X out of Y  
X-squared = 40.1968, df = 1, p-value = 2.296e-10  
alternative hypothesis: two.sided  
99 percent confidence interval:  
 0.0002240077 0.0005401186  
sample estimates:  
 prop 1      prop 2  
0.0005464508 0.0001643877
```

Como é possível observar nos dois itens, o *p-value* é muito menor que  $\alpha$ . Portanto, existem evidências amostrais afirmando que devemos rejeitar a hipótese nula, o que nos leva a crer que a hipótese alternativa (proporções diferentes) é verdadeira.

## 8

# Regressão e Correlação Linear Simples

---

## 8.1 Introdução

A regressão e a correlação são técnicas utilizadas para estimar uma relação que possa existir na população, enquanto as técnicas anteriormente estudadas (medidas de tendência central e de dispersão) servem para estimar um único parâmetro populacional.

A análise de correlação e regressão compreende a análise de dados amostrais para saber como duas variáveis estão relacionadas uma com a outra numa população.

A correlação mede a força ou o grau de relacionamento entre duas variáveis. Já a regressão dá a equação que descreve o relacionamento em termos matemáticos, pressupondo alguma relação de causa e efeito entre as variáveis. Por exemplo: a idade e a altura de cada indivíduo.

A regressão linear simples constitui uma tentativa de estabelecer uma equação matemática linear (reta) com apenas uma variável dependente que descreva o relacionamento entre duas variáveis.

A equação linear (reta de regressão) apresenta como principais características:

- O coeficiente angular da reta é dado pela tangente da reta e se denomina “a”;
- A cota da reta em determinado ponto é o coeficiente linear denominado “b” que é o valor de  $y$  quando  $x$  for igual a zero.

Possui a seguinte fórmula:

$$y = ax + b + \varepsilon \quad (8.1)$$

onde:

- $x$  é a variável explicativa, independente ou preditora;
- $y$  é a variável explicada, dependente ou resposta;
- $\varepsilon$  é chamado de erro que corresponde ao desvio entre o valor real e o aproximado (pela reta) de  $y$ . Isso porque sempre teremos observações amostrais que não são pontos da reta.

*Observação:* nem todas as situações são bem aproximadas por uma equação linear. Quando os dados não podem ser aproximados por um modelo linear, as alternativas são procurar um modelo não-linear conveniente, ou transformar os dados para a forma linear. Nesta apostila trataremos apenas da correlação linear.

Para podermos verificar se a correlação é linear devemos fazer uma análise gráfica

do fenômeno em estudo. Para isso, devemos traçar um esboço do gráfico de dispersão. Este gráfico é construído por um conjunto de pontos formados pelos pares ( $x, y$ ) correlacionados entre si.

## 8.2 Determinando a Equação Linear (Regressão)

Devemos estabelecer um critério para a obtenção da equação de primeiro grau (obtenção dos coeficientes “a” e “b”). Utilizaremos o método dos mínimos quadrados de modo que possamos encontrar os valores de “a” e “b” da reta de regressão que minimizem a soma dos quadrados dos desvios individuais. Os desvios individuais correspondem à diferença entre a medida real (do fenômeno estudado) e a medida aproximada pela reta de regressão. O somatório dos quadrados dos desvios é mostrado abaixo:

$$\sum_{i=1}^n d_i^2 = \sum_{i=1}^n (y_i - y_c)^2 \quad (8.2)$$

onde:

- $y_i$  é o valor observado de  $y$ ;
- $y_c$  é o valor calculado de  $y$  na função linear.

Vale enfatizar que o método de ajuste dos mínimos quadrados é preferível porque obtém as melhores estimativas. Por meio deste método, podemos deduzir a reta que melhor explica a relação entre a variável independente e a variável dependente.

Os coeficientes da reta são calculados pelas fórmulas abaixo:

$$a = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \quad (8.3)$$

$$b = \frac{\sum_{i=1}^n y_i - a \sum_{i=1}^n x_i}{n} \quad (8.4)$$

A equação linear pode ser obtida no R por meio da função `lm()` que serve para calcular a regressão linear simples. Podemos obter a regressão linear simples por meio do seguinte comando:

```
#Sintaxe
> lm(y~x,data)    #lm = "linear model"
                    #lê-se "y~x" como sendo y depende de x
```

Onde o argumento “data” recebe o conjunto de dados como uma estrutura chamada `data.frame` a serem analisados no formato ( $x,y$ ). Podemos eliminar este argumento desde que seja dada “entrada”, no R, dos vetores “ $x$ ” e “ $y$ ” antes de utilizarmos a função `lm()`.

Tomaremos como exemplo, para todo este capítulo, os vetores  $x$  e  $y$  carregados abaixo:

```
#Exemplo:
> x <- c(201,225,305,380,560,600,685,735)
> y <- c(17,20,21,23,25,24,27,27)
> dados = data.frame(x,y) #criando um data.frame
> is.data.frame(dados)    #verifica se dados é um data.frame
[1] TRUE
> regressão=lm(y~x,data=dados) #ou apenas "regressão=lm(y~x)"
> regressão

Call:
lm(formula = y ~ x, data = dados)

Coefficients:
(Intercept)          x      #reta de regressão
15.65995        0.01591
```

Para verificarmos os resultados da função `lm()`, inclusive os coeficientes da regressão linear, devemos utilizar o comando:

```
#Sintaxe:
> summary(regressão)
```

Assim como a maioria das funções do R, armazenamos os resultados retornados pela função `lm()` em um objeto. O valor retornado por `lm()` é uma lista. Siga os comandos:

```
#Exemplo:
> is.list(regressão) #verifica se "regressão" é uma lista
[1] TRUE
> names(regressão)
[1] "coefficients"   "residuals"       "effects"        "rank"
[5] "fitted.values"  "assign"          "qr"             "df.residual"
[9] "xlevels"         "call"            "terms"          "model"
```

Neste último comando teremos todos os importantes resultados do modelo ajustado. Os comandos mais importantes listados são os seguintes:

- **`regressão$fitted.values`** ou **`predict()`**: calcula os valores preditos da variável resposta para cada elemento da amostra (faz uma previsão);
- **`regressão$residuals`**: calcula o erro ou os resíduos (valor observado - valor predito) para cada ponto da amostra;
- **`regressão$coefficients`**: obtém uma estimativa dos coeficientes da regressão.

Podemos traçar, como já sabemos, o gráfico de dispersão através do comando `plot(x,y)`. No mesmo gráfico é possível traçarmos a reta da regressão que melhor ajusta

os dados, usando o seguinte comando:

```
> abline(regressão)
```

A função `abline()` carrega consigo um vetor com dois valores (o intercepto e a inclinação da reta) e adiciona uma reta ao gráfico com estes valores.

Para o exemplo do início deste capítulo, temos:

```
#Exemplo:  
> z = plot(x,y)  
> grid(z) #aplicando grid ao gráfico  
> abline(regressão)
```

O resultado do exemplo anterior é apresentado no gráfico abaixo:

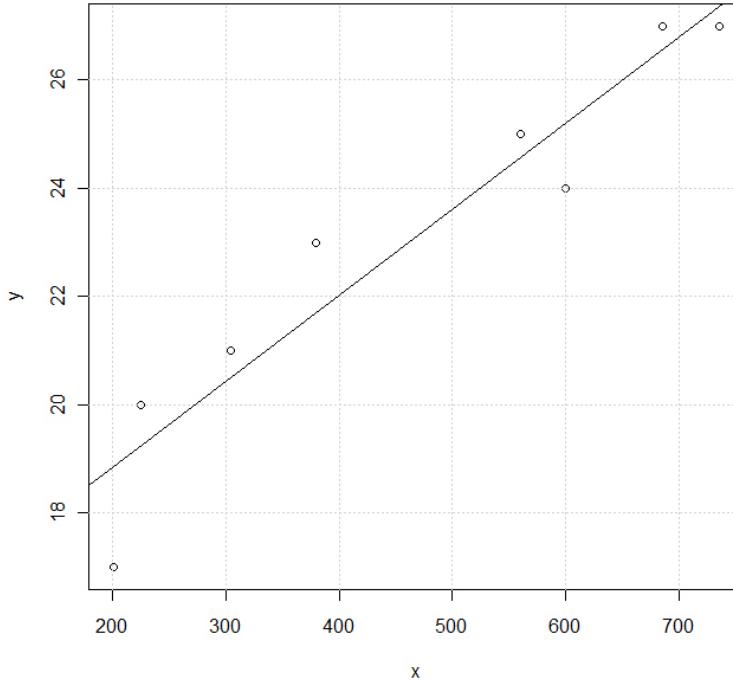


Figura 8.1: Gráfico de dispersão com a reta de regressão.

Outros componentes do ajuste são úteis para propósito de diagnóstico. Por exemplo, pode-se examinar o gráfico de valores ajustados contra resíduos:

```
#Exemplo:  
> plot(regressão$fitted.values,regressão$residuals)
```

Este comando pode nos mostrar os desvios dos dados em relação ao modelo linear (estimado).

Além disso, podemos fazer um histograma dos resíduos para verificar a presença de dados discrepantes (outliers). O comando está abaixo:

```
#Exemplo:  
> hist(regressão$residuals)
```

Podemos identificar diferentes classes que são atribuídas a diferentes objetos do R. Este mecanismo facilita a automação de diversas tarefas no R. Por exemplo, podemos descobrir qual a classe atribuída a um objeto que armazena resultados da função lm() digitando:

```
#Exemplo:  
> class(regressão)  
[1] "lm"
```

O fato que "regressão" é um objeto da classe lm implica que os gráficos diagnósticos podem ser visto simplesmente digitando:

```
#Exemplo:  
> plot(regressão)  
[1] "lm"
```

Este comando do R produz automaticamente uma série de gráficos para o ajuste de um modelo linear. Pressionando sucessivos ENTER's obteremos uma sequência de gráficos relacionados ao ajuste do modelo linear.

## 8.3 Coeficiente de Correlação ( $r$ )

Através do gráfico de dispersão pode-se indicar se a correlação linear é positiva, negativa ou a inexistência de correlação. Além disso, podemos identificar, por meio do coeficiente de correlação ( $r$ ), o tipo de correlação, conforme tabela abaixo:

$ r $	Tipo
0	Nula
0 a 0,3	Fraca
0,3 a 0,7	Regular
0,7 a 0,99	Forte
1	Perfeita

O coeficiente de correlação mede o grau de associação entre as variáveis dependente e independente ( $x$  e  $y$ ). Seu valor varia de -1 (correlação negativa) a 1 (correlação positiva). Existem as seguintes interpretações para o coeficiente de correlação:

- **Correlação nula:**  $y$  não cresce, indicando que a correlação não é linear (constante);

- **Correlação fraca:** indica que  $y$  cresce mais lentamente que  $x$ ;
- **Correlação forte:** indica que  $y$  cresce mais rapidamente que  $x$ ;
- **Correlação perfeita:** indica que  $y$  cresce na mesma velocidade que  $x$ .

Por exemplo, para  $r = +0,953$ , temos uma correlação positiva forte, ou seja, a variável dependente  $y$  cresce quase na mesma proporção que a variável independente  $x$ .

Podemos encontrar o coeficiente de correlação utilizando o comando `cor(x,y)`:

```
#Exemplo:  
> cor(x,y)  
[1] 0.9529494
```

## 8.4 Coeficiente de Determinação ( $r^2$ )

Explica o grau de ajuste do modelo, ou o percentual de variação de  $y$  que é explicada pela variabilidade de  $x$ . Seu valor varia de 0 a 1.

Por exemplo se for  $r^2 = 0,908$  entendemos que, aproximadamente, 91% da variação da variável dependente é explicada pela variável independente. Os outros 9% possuem causas aleatórias desconhecidas (independentes de  $x$ ).

A fórmula para a obtenção do coeficiente de determinação é dada abaixo:

$$r^2 = \frac{(n \sum_{i=1}^n xy - \sum_{i=1}^n x \sum_{i=1}^n y)^2}{[n \sum_{i=1}^n x^2 - (\sum_{i=1}^n x)^2][n \sum_{i=1}^n y^2 - (\sum_{i=1}^n y)^2]} \quad (8.5)$$

*Observação:* se tomarmos a raiz quadrada da equação acima obteremos o coeficiente de correlação (seção anterior).

De acordo com a observação anterior, podemos encontrar o coeficiente de determinação, no R, apenas elevando ao quadrado o coeficiente de determinação:

```
#Exemplo:  
> cor(x,y)^2  
[1] 0.9081126
```

## 8.5 Exemplo Aplicado

O exercício a seguir foi adaptado de [5].

Para uma amostra de oito operadores de máquina, foram coletados o número de horas de treinamento ( $x$ ) e o tempo necessário para completar o trabalho ( $y$ ). Os dados coletados encontram-se na tabela abaixo:

<b>x</b>	5,2	5,1	4,9	4,6	4,7	4,8	4,6	4,9
<b>y</b>	13	15	18	20	19	17	21	16

Pede-se:

- a) Faça o gráfico de dispersão para esses dados.
- b) Determine o modelo de regressão linear simples entre as variáveis “x” e “y”.
- c) Em seguida, trace, no gráfico anterior, a reta de regressão.
- d) Calcule e interterepe os coeficientes de correlação ( $r$ ) e determinação ( $r^2$ ).

### Solução

Inicialmente devemos entrar com os vetores “x” e “y”, da seguinte forma:

```
> y <- c(5.2,5.1,4.9,4.6,4.7,4.8,4.6,4.9)
> x <- c(13,15,18,20,19,17,21,16)
> y #apresentação do vetor y
[1] 5.2 5.1 4.9 4.6 4.7 4.8 4.6 4.9
> x #apresentação do vetor x
[1] 13 15 18 20 19 17 21 16
```

- a) Através do comando apresentado abaixo, obteremos o gráfico da dispersão.

```
> plot(x,y)
```

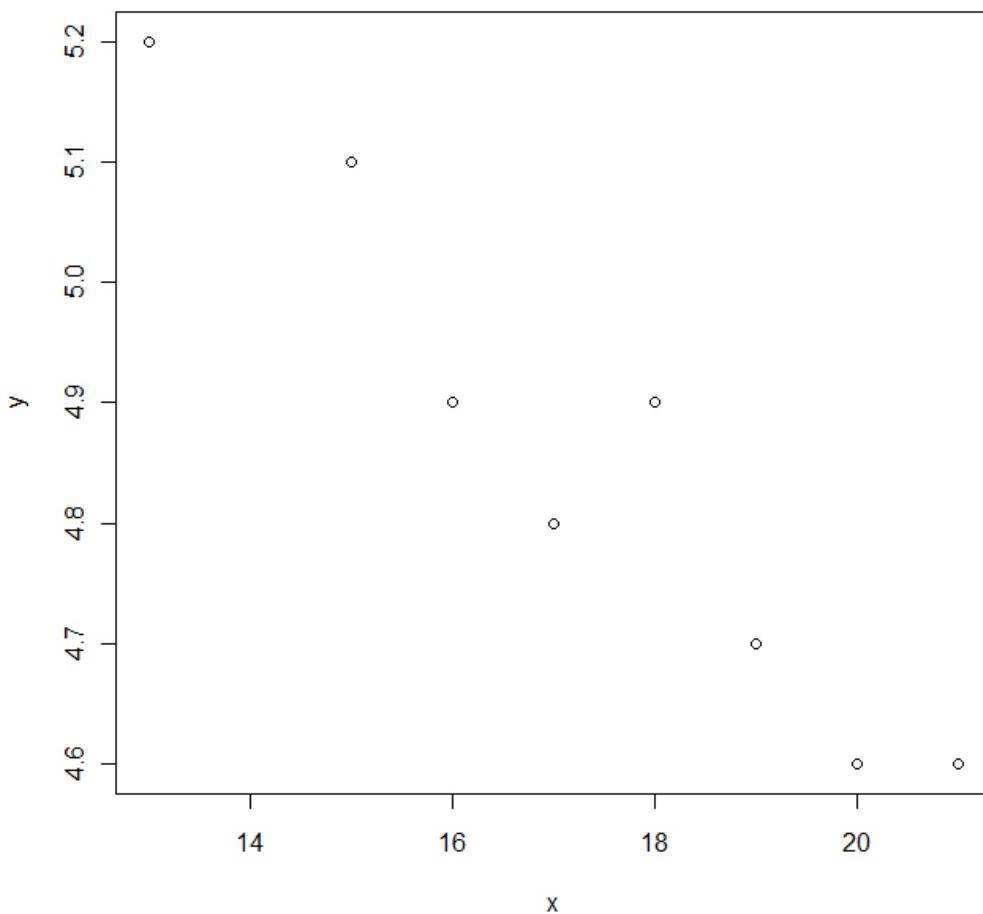


Figura 8.2: Dispersão dos dados

**b)** Um dado importante a ser observado no gráfico acima é que uma reta pode ser utilizada para aproximarmos o modelo apresentado. O modo como encontramos a equação no R é o seguinte:

```
> mod_reg = lm(y~x)
> mod_reg      #resultados do modelo de regressão

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)           x
              6.2261      -0.0792
> abline(mod_reg)
```

c) Através dos resultados fornecidos pelo R, concluímos que a equação da reta que melhor aproxima os pontos é  $y = -0,0792x + 6,2261$ . Além disso, foi traçada a reta de regressão juntamente com o gráfico de dispersão através do comando `abline(mod_reg)`.

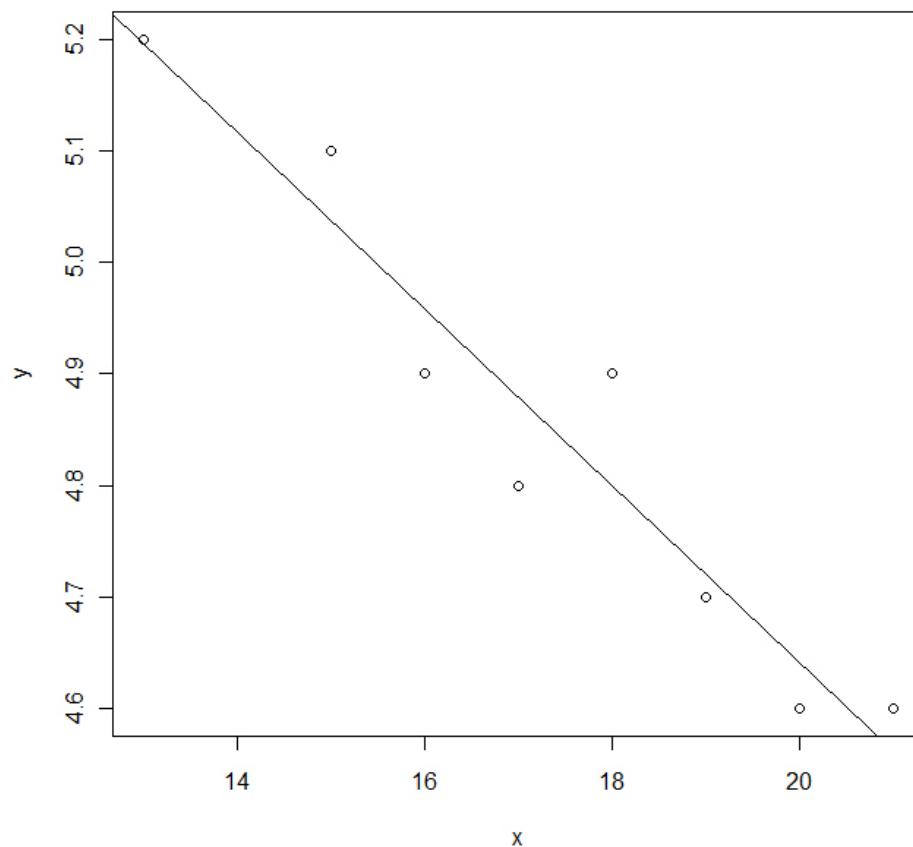


Figura 8.3: Reta de regressão

d) Obteremos o coeficiente de correlação linear por meio do comando:

```
> cor(x,y)  
[1] -0.9592155
```

Podemos interpretar o sinal negativo como sendo causado pelo decrescimento do valor de  $y$  ao longo de  $x$ . Por outro lado, se tomarmos o módulo de  $r$  teremos uma correlação forte, o que indica que o tempo necessário para completar o trabalho decresce quase que na mesma proporção com que aumenta o número de horas do treinamento. Em outras palavras,  $y$  decresce quase na mesma velocidade com que  $x$  cresce.

Podemos, ainda, calcular e interpretar o coeficiente de determinação:

```
> cor(y,x)^2  
[1] 0.9200944
```

Deste resultado, inferimos que aproximadamente 92% da diminuição do tempo de execução do trabalho está relacionada com o aumento do número de horas do treinamento. Em outras palavras, 92% das variações em  $y$  são explicadas por variações em  $x$ . Os outros 8% possuem outras causas.

## 9

# Programação em Linguagem R

## 9.1 Introdução

O R possui uma ferramenta de programação que permite a codificação de pequenos programas. Um programa nada mais é do que a codificação de um algoritmo (ou simplesmente a resolução de um problema) numa linguagem de programação, neste caso a linguagem R. A linguagem R é composta por um conjunto de instruções que são escritas através de um conjunto de códigos (símbolos e palavras). Este conjunto de códigos possui regras de estruturação lógica e sintática própria.

Anotemos, ainda, que devemos gerar um arquivo de programação que conterá o código fonte em qualquer editor de texto. Utilize preferencialmente o Tinn-R ou o bloco de notas. Após o término do código fonte copie o código no Console do R e veja os resultados da programação.

## 9.2 Interação com o Usuário

O R possui diversas funções que têm como objetivo ler dados introduzidos pelo utilizador no teclado, para gerir uma interação com o utilizador.

A função **print()**, por exemplo, pode ser usada para escrever o conteúdo de qualquer objeto. Porém, para objetos muito grandes é mais conveniente utilizar a função **str()**.

A função **cat()** também pode ser usada para escrever objetos ou constantes. Funciona com um número qualquer de argumentos, sendo que sua função é transformar os seus argumentos em "strings" e depois escrevê-los no Console. Veja o exemplo abaixo:

```
#Exemplo:  
> ano=2009  
> cat("Estamos no ano ",ano,"\\n\\tComo passa rápido!\\n")  
Estamos no ano 2009  
      Como passa rápido!
```

Observe que o R interpreta os caracteres entre aspas como sendo uma mensagem introduzida pelo programador. Já os caracteres "\t" e "\n" representam, respectivamente, os comandos de tabulação e de nova linha.

O comando para leitura de dados oriundos do usuário é o **scan()**. Neste comando o R fará a leitura de uma quantidade finita (até que sejam dados dois ENTER's seguidos) de elementos introduzidos pelo usuário. Caso queiramos especificar o número de elementos que o R fará a leitura basta acrescentar como argumento o número de elementos que queremos acrescentar. Veja o exemplo abaixo:

```
#Sintaxe:  
> scan(n=número de elementos)  
#Exemplo:  
> x<-scan(n=2)  
1: 1  
2: 2  
Read 2 items
```

Por fim, o comando `scan()` pode ser utilizado para a leitura de strings e valores numéricos. Observe o exemplo abaixo:

```
#Exemplo:  
> scan(what=character())  
1: Curso Software R  
4: Edição 2  
6:  
Read 5 items  
[1] "Curso"      "Software"   "R"           "Edição"     "2"
```

Cada elemento introduzido é considerado um objeto. A função `scan()` tem ainda outros argumentos que podem ser facilmente acessados no help do R (`help(scan)`).

## 9.3 Estruturas de Controle da Linguagem R

A linguagem R, como qualquer linguagem de programação, possui várias instruções destinadas a alterar o curso sequencial normal da execução dos programas. Estudaremos a seguir cada uma dessas estruturas:

### 9.3.1 Instruções Condicionais

As instruções condicionais permitem ao programador explicitar diferentes alternativas a serem executadas dependendo de alguma condição a ser testada em dado ponto do código.

Podemos explicitar um exemplo em um bloco de instruções na linguagem R. Um bloco de instruções, em linguagem R, é indicado fornecendo essas instruções em linhas separadas e seguidas, delimitadas por chaves. Vamos ao exemplo:

```
#Exemplo:
> x<-2      #x é positivo
> z<-1
> if (x > 0) {
+ cat('x é positivo!\n')  #comando para escrever texto
+ y <- z / x
+ }else {
+ cat('x não é positivo!\n')
+ y <- z}
x é positivo!
```

A instrução *if* possui uma cláusula opcional, o *else*. Podemos usar em substituição a este *else* outro *if*. Podemos, ainda, usar várias instruções *if* aninhadas, como no exemplo seguinte:

```
#Exemplo:
> idade=60
> if (idade < 18) {
+ cat("Idade menor que 18\n")
+ } else if (idade < 35) {
+ cat("Idade menor que 35\n")
+ } else if (idade < 60) {
+ cat("Idade menor que 65\n")
+ } else {
+ cat("Idade maior ou igual a 60. BEM-VINDO À TERCEIRA IDADE!\n")}
Idade maior ou igual a 60. BEM-VINDO À TERCEIRA IDADE!
```

Embora só exista uma instrução em cada bloco, todas as instruções estão entre chaves como se tratassem de conjuntos de instruções. Caso não houvessemos colocado chaves originaríamos um erro de sintaxe. Uma boa regra para evitar problemas, é sempre usar as chaves para delimitar as duas partes de uma instrução *if*.

Temos, ainda, como instrução condicional a instrução *switch()* que é usada para escolher uma entre várias alternativas. Para facilitar, vejamos um exemplo em que o argumento é um texto:

```
#Exemplo:
> semáforo <- "verde"  #situação do semáforo entre aspas (string)
> switch(semáforo, verde = "continua", amarelo = "acelera",
+ vermelho = "pára")    #escolha entre as múltiplas opções
[1] "continua"
```

Por este exemplo, podemos ver que a instrução *switch* tem a seguinte sintaxe genérica:

```
#Sintaxe
> switch(valor,lista de componentes)
```

O resultado que obteremos da instrução nada mais é do que usar o conteúdo do primeiro argumento (*valor*) para indicar uma das componentes da lista. O conteúdo destas componentes é da mesma família do primeiro argumento. Se o primeiro argumento for um número então o resultado é a componente da lista com esse número. Já se o primeiro argumento for um nome, então será extraída a componente com esse nome.

### 9.3.2 Instruções Iterativas

O R tem várias instruções iterativas (ciclos) que nos permitem repetir blocos de instruções. Para o controle destes ciclos o R possui as instruções **repeat**, **for** e **while**.

A instrução while tem a seguinte estrutura genérica:

```
#Sintaxe
> while (condição)
{bloco de instruções}
```

Interpretamos a instrução while da seguinte forma: enquanto a condição for verdadeira, repetir o bloco de instruções do ciclo. Vejamos um pequeno exemplo:

```
#Exemplo:
#x recebe um número aleatório de uma distribuição normal
> x <- rnorm(1)
> while (x < 1) {      #enquanto x for menor que 1, faça:
#escreve que x vale um número aleatório menor que 1
+ cat("x=", x, "\t")
+ x <- rnorm(1)
+ if(x>=1){      #condição para nova linha
+ cat('\n')}
+ }      #o símbolo mais (+) espera o fechamento de chaves
x= 0.1663923    x= -0.7760292   x= -1.220098    x= -1.467930
x= 0.3146124    x= -1.424002
> x
[1] 1.972545 #primeiro número aleatório gerado que é maior que 1
```

Observe que as instruções do bloco dentro do ciclo while podem nunca ser executadas, uma vez que a condição seja falsa na primeira vez que o R "chega" ao ciclo. No exemplo acima se o primeiro número "sorteado" pela função *rnorm()* for superior ou igual a 1, as instruções do ciclo não serão executadas.

A instrução *repeat* permite que o R execute um bloco de instruções uma ou mais vezes. Sua forma genérica é:

```
#Sintaxe  
> repeat  
{bloco de instruções}
```

Repare que como não existe nenhuma condição lógica para governar a execução do bloco de instruções (como vimos no caso while). Logo, o R socorre-se em outras instruções que permitam parar a execução do processo iterativo. A instrução que controla o processo iterativo repeat é a instrução **break**. Sua função é fazer o R terminar a execução de um bloco de instruções. Entretanto, para evitar ciclos "infinitos" (loops), é conveniente garantir que a instrução break seja passível de ocorrer dentro do bloco de instruções. Vejamos um exemplo em que o R lê frases introduzidas pelo utilizador até que este introduza uma frase vazia:

```
#Exemplo:
> texto <- c() #variável texto recebe um vetor vazio
> repeat {      #repita até que tenhamos uma frase vazia
+ cat('Introduza uma frase (frase vazia termina): ')
+ fr <- readLines(n=1) #objeto ''fr'' recebe leitura da linha 1
+ if (fr == ''){       #condição de parada (frase vazia)
+ cat("Frase vazia inserida!")
+ cat("\nFIM\n")
+ break}
+ else texto <- c(texto,fr) #cria um vetor com as frases inseridas
+ cat("Frases inseridas:\n",texto,"\n")}
Introduza uma frase (frase vazia termina): Curso Software R
Frases inseridas:
Curso Software R
Introduza uma frase (frase vazia termina):
Frase vazia inserida!
FIM
>     #prompt espera novo comando (fim do repeat)
```

Conforme vemos, este ciclo irá terminar somente quando a variável "fr", que contém a frase que o utilizador acabou de introduzir, for vazia. Se isso não ocorrer, a instrução break não é executada e o ciclo continua se repetindo.

Além da instrução break, temos a instrução **next** que pode ser usada para controlar o que é feito num ciclo. A instrução next avança para a próxima iteração. O exemplo abaixo ilustra o uso desta instrução:

```
#Exemplo:
> ans=0      #eliminando buffer de memória
> repeat {    #comando repeat
+ cat('Introduza um número (zero termina): \n')
+ nro <- scan(n=1) #leitura do número introduzido
+ if (nro<0||nro>0) {
+ ans<-c(ans,nro) #criação do vetor com o histórico de números
+ next}           #reinicia uma nova iteração
+ if (nro==0) {
+ cat('Os números introduzidos foram:\n',ans,"\\n")#impressão do vetor
+ print('FIM'); cat('\\n')
+ break}}       #pára a execução do repeat
Introduza um número (zero termina): 1: 1
Read 1 item
Introduza um número (zero termina): 1: -2000
Read 1 item
Introduza um número (zero termina): 1: 0
Read 1 item
Os números introduzidos são:
 0 1 -2000
[1] "FIM"
```

No exemplo acima, pretende-se ler um conjunto de números positivos que são colecionados num vetor chamado “ans”. O utilizador deverá introduzir o número zero para terminar o ciclo. Caso seja introduzido um número diferente de zero, o código fará com que o R solicite um novo número (instrução `next`), reiniciando o ciclo `repeat`. Desta forma, o R ao encontrar a instrução `next` produz um efeito capaz de causar um salto no código para o início do ciclo que está sendo executado, sem executar as instruções subsequentes ao `next`.

Finalmente, trataremos da instrução `for` que permite controlar o número de vezes que um ciclo é executado. Consegue-se isso através de uma variável de controle que vai tomar uma série de valores pré-definidos em cada iteração do ciclo. A sua sintaxe genérica é:

```
#Sintaxe
for(<variável de controle> in <conjunto>)
{bloco de instruções}
```

Veja o exemplo abaixo:

```
#Exemplo:
#x recebe 10 números aleatórios de uma distribuição normal
> x <- rnorm(10)
> soma <- 0      #soma=0 para evitar resíduos de memória
> for (v in x) { #v será igual a cada um dos elementos do vetor x
+ if (v > 0) {   #caso em que os números são positivos
+ y <- v}       #y é uma variável auxiliar
+ else {         #caso em que os números não são positivos
+ y <- 0}       #para não alterar o resultado de "soma"
+ soma <- soma + y} #cálculo da soma dos números positivos
> x               #apresentação do vetor aleatório
[1] -2.3868397  0.3501884 -0.6210985  0.4928832  0.9672247
[7] 1.7185209   -0.6988870 -0.7885660  0.8171145  1.2330365
> soma            #resultado da soma dos números positivos
[1] 5.578968      #conferindo o resultado com a variável ''soma''
> 0.3501884+0.4928832+0.9672247+1.7185209+0.8171145+1.2330365
[1] 5.578968
```

O ciclo acima se inicia com a obtenção de um vetor com dez números aleatórios. Em seguida, é calculada a soma exclusiva dos números positivos do vetor aleatório. Através do ciclo for, a variável de controle (chamada de v) vai assumir, em cada iteração, um dos dez valores atribuídos a x. Isto é, na primeira iteração v assumirá o valor de  $x_1$ , que no exemplo é -2.3868397. Da mesma forma, na segunda iteração v assumirá o valor de  $x_2$  e assim sucessivamente até que se esgotem os valores de x.

Note que em vez da utilização deste ciclo para a soma dos números positivos de x, poderíamos ter calculado o valor da soma através do comando:

```
#Sintaxe
> soma = sum(x[x>0])      #para simples comparação
```

O que tornaria o cálculo mais simples e eficiente em termos de tempo, principalmente em se tratando de vetores maiores. Por exemplo, se tomarmos o exemplo abaixo:

```
#Exemplo:  
> x <- rnorm(1e+05)      #x recebe um vetor com 10^5 números  
#verifique a hora do computador em que é dado <ENTER> ao comando  
> t <- Sys.time()  
> soma <- 0  
> for (v in x) {  
+ if (v > 0) {  
+ y <- v}  
+ else {y <- 0}  
+ soma <- soma + y}  
#calculando o tempo decorrido para efetuar as iterações  
> Sys.time() - t  
Time difference of 0.5 secs  
> t <- Sys.time()  #verificando novamente o horário do <ENTER>  
> soma <- sum(x[x > 0]) #calculando a soma através da função  
#calculando o tempo decorrido para calcular a soma  
> Sys.time() - t  
Time difference of 0.04699993 secs
```

Por meio deste exemplo vemos que sempre que possível devemos evitar a execução de ciclos, pois, para um grande conjunto de dados, os resultados demorarão mais a aparecer do que na utilização de um comando específico. Entretanto, o tempo de execução varia de acordo com o desempenho (velocidade de processamento) de cada computador.

# 10

## Referências Bibliográficas

---

1. VENABLES, W.N.; SMITH, D.M; R Development Core Team. **An Introduction to R**. Version 2.8.1 (2008-12-22).
2. KUHNERT, P.; VENABLES, B. **An Introduction to R: Software for Statistical Modelling & Computing**. CSIRO Mathematical and Information Sciences: Cleveland, Australia, 2005.
3. HIEBELER, D. **MATLAB ©\ R References**. Dept. of Mathematics and Statistics, University of Maine. March 3, 2009.  
<http://www.math.umaine.edu/faculty/hiebeler>
4. PETERNELLI, L.A.; MELLO, M.P. de. **Conhecendo o R: Uma visão Estatística**. Editora UFV: Universidade Federal de Viçosa, 2007.
5. MONTGOMERY, Douglas C.; RUNGER, George C. **Estatística Aplicada e Probabilidade para Engenheiros**. 2 ed. Rio de Janeiro: LTC, 2003.
6. <http://www.r-project.org/>