



# Tópicos de Estatística utilizando R



# Tópicos de Estatística utilizando R

Orientadores:  
Denise Aparecida Botter  
Mônica Carneiro Sandoval

Alunos:  
Fernando Itano  
Soane Mota dos Santos

# Sumário

Características do R .....	5
Importação de dados .....	6
Salvando arquivos separados por vírgula (CSV) no Microsoft Excel .....	6
Lendo arquivos do tipo CSV (separado por vírgulas) .....	6
Arquivos em banco de dados (ODBC) .....	6
Funções Estatísticas .....	7
Tabelas .....	7
Tabela de proporções .....	7
Resumo .....	8
Média .....	8
Variância .....	9
Desvio Padrão .....	9
Mediana .....	9
Aplica funções .....	10
Divisão de dados .....	10
lm .....	11
Parâmetros .....	11
Exemplos .....	11
Testes para a média populacional e para a comparação de duas médias .....	12
t.test() .....	12
Parâmetros .....	12
Opções .....	12
Testes para uma proporção populacional e para comparação de duas proporções .....	14
prop.test() .....	14
Parâmetros .....	14
Opções .....	14
fisher.test() .....	15
Parâmetros .....	15
Opções .....	15
Testes para Normalidade .....	16
Pacote: base .....	16
shapiro.test() .....	16
Parâmetros .....	16
Pacote opcional: nortest .....	16
ad.test() .....	16
Parâmetros .....	16
cvm.test() .....	17
Parâmetros .....	17
lillie.test() .....	17
Parâmetros .....	17
Parâmetros .....	17
Opções .....	17
Parâmetros .....	18
Testes para comparação de variâncias .....	19
Pacote: stats .....	19
bartlett.test() .....	19
Parâmetros .....	19
Pacote: car .....	20
levene.test() .....	20
Parâmetros .....	20
Funções Matemáticas .....	21
Combinatória .....	21
Fatorial .....	21

Raiz Quadrada.....	21
<b>Gráficos .....</b>	<b>22</b>
locator.....	22
text.....	22
Gráfico de barras.....	23
Histograma.....	23
Boxplot.....	24
Gráfico de Pizza.....	24
Gráfico de Dispersão.....	25
<b>Probabilidade .....</b>	<b>26</b>
Função Densidade (ou Probabilidade).....	26
Função Distribuição.....	26
Função Probabilidade.....	26
Gerador aleatório.....	26
<b>Exemplos .....</b>	<b>29</b>
Dividindo as observações em intervalos.....	29
Frequência de observações em cada intervalo.....	29
Rotulando os intervalos.....	29
Criando variáveis e exibindo-as em tabelas.....	29
Alterando os rótulos das variáveis.....	29
Histogramas com curvas teóricas.....	30
<b>Bibliografia .....</b>	<b>32</b>

## Características do R

- Não foi feito para manipulação de dados em larga escala.
- Forma mais fácil e direta de acessar os dados é convertê-los para texto e importar.
- Salva a sessão em um arquivo .RDATA, que armazena todos os objetos R, possibilitando que um projeto seja retomado posteriormente ou intercambiado com colaboradores.
- Acessa bancos de dados e planilhas Microsoft Excel via ODBC e outros bancos de dados por servidor SQL, ampliando a capacidade de trabalhar com dados em larga escala.
- A partir da versão 2.1.1 possui um editor de script, que facilita a execução de comandos diretamente de dentro do R.
- Possui pacotes com funções específicas que podem ser instalados pela Internet, através do próprio programa.
- Conta com inúmeros colaboradores no mundo inteiro que criam, testam e corrigem as funções que podem ser usadas por qualquer pessoa.
- Gera gráficos em diferentes formatos para as mais diversas utilizações.

# Importação de dados

## Salvando arquivos separados por vírgula (CSV) no Microsoft Excel

Para importar dados a partir de uma planilha do Microsoft Excel, é necessário antes salvar essa planilha no formato .CSV (valores separados por vírgula). Para salvá-la nesse formato, clique em **Arquivo** do menu, em seguida **Salvar como...**. Na parte inferior da caixa de diálogo, clique sobre as opções de **Salvar como tipo** e selecione **CSV (separado por vírgulas)**. Agora basta escolher a pasta de destino e clicar no botão **Salvar**. Surgirão algumas mensagens indicando a incompatibilidade de alguns recursos do Microsoft Excel com esse formato, mas como o que nos interessa são apenas os dados, clique em **OK** e em seguida em **SIM**.

## Lendo arquivos do tipo CSV (separado por vírgulas)

Podemos usar o comando **read.csv** para armazenar os dados desse arquivo em um objeto do R chamado **dados**:

sintaxe:

```
dados <- read.csv("caminho_e_nome_do_arquivo.csv", opções)
```

opções:

**sep**: caractere utilizado para separação dos campos e valores. Normalmente é utilizado o ponto e vírgula (;)  
**dec**: caractere utilizado para separar as casas decimais. Normalmente ponto (.) ou vírgula (,).  
**header**: TRUE, assume que a primeira linha da tabela contém rótulos das variáveis.  
FALSE, assume que os dados se iniciam na primeira linha.

exemplo:

Importa os dados provenientes do arquivo caries.csv, presente no drive C, pasta Analises descritivas, subpasta odonto, com vírgula como sinal decimal, ponto e vírgula como separador de campos e valores e armazena no objeto de nome caries:

```
> caries <- read.csv("C:/Analises descritivas/odonto/caries.csv", sep=";", dec=".", header=TRUE)
```

Visualiza os dados obtidos e armazenados no objeto caries:

```
> caries
  X1 X10
1  2  12
2  3  15
3  4  18
4  5  22
```

**Observação:** Pode-se optar por utilizar as **barras normais** como indicadoras do caminho do arquivo, ou **barras invertidas duplas**.

**Ex:** C:\\Analises descritivas\\odonto\\caries.csv tem o mesmo efeito de C:/Analises descritivas/odonto/caries.csv.

## Arquivos em banco de dados (ODBC)

É possível acessarmos bancos de dados previamente configurados através do ODBC (Open DataBase Connectivity). A grande vantagem é que podemos acessar bancos de dados extremamente grandes sem prejudicar o desempenho do processamento no R, pois os dados não serão armazenados na memória RAM, mas acessados diretamente do disco rígido sempre que necessário.

# Funções Estatísticas

## Tabelas

### sintaxe:

```
table(dados)
```

### exemplo:

Dados utilizados da tabela 2.1 de Bussab e Morettin (2003).

```
table(dados$origem)
```

Capital	Interior	Outros
11	12	13

```
table(dados$origem,dados$estciv)
```

	Casado	Solteiro
Capital	7	4
Interior	8	4
Outros	5	8

```
table(dados$origem,dados$estciv,dados$educa)
```

```
, , = Ensino Fundamental
```

	Casado	Solteiro
Capital	2	2
Interior	1	2
Outros	2	3

```
, , = Ensino Médio
```

	Casado	Solteiro
Capital	4	1
Interior	6	1
Outros	2	4

```
, , = Superior
```

	Casado	Solteiro
Capital	1	1
Interior	1	1
Outros	1	1

## Tabela de proporções

Mostra os dados em formato de tabela usando proporções:

### sintaxe:

```
prop.table(tabela)
```

### Exemplo:

```
prop.table(table(dados$Educao))
```

Ensino Fundamental	Ensino Médio	Superior
0.3333333	0.5000000	0.1666667

## Resumo

Resume a variável quantitativa em: mínimo, máximo, média, mediana, 1º quartil, 3º quartil e dados não preenchidos. Caso a variável seja qualitativa, é informado o número de observações para cada nível.

### sintaxe:

```
summary(variável)
```

### Exemplo:

Resumo da variável salário

```
summary(dados$Salario)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.000  7.553 10.160 11.120 14.060 23.300
```

Resumo da variável salário apenas para casados

```
summary(dados$salario[dados$estciv=="Casado"])
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.560  8.743 11.930 12.120 15.030 23.300
```

Resumo da variável salário apenas para solteiros

```
summary(dados$salario[dados$estciv=="Solteiro"])
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.000  7.257  9.045  9.871 11.690 18.750
```

Resumo da variável qualitativa origem

```
summary(dados$Origem)
  Capital Interior  Outros
      11         12      13
```

**Observação:** Caso a variável desejada seja **qualitativa numérica**, é possível que o R interprete-a como sendo uma variável **quantitativa**. Para evitar que isso aconteça, utilize a função **as.factor()**.

**Ex:** `summary(as.factor(dados$sexo))`

## Média

### sintaxe:

```
mean(variável)
```

### Opções:

**na.rm:** TRUE, calcula a média considerando apenas os dados existentes, ignora os dados faltantes.

FALSE, calcula a média apenas se todos os valores estiverem preenchidos, caso contrário retorna NA.

### Exemplo:

```
mean(dados$Filhos)
[1] NA
```

```
mean(dados$Filhos, na.rm=TRUE)
[1] 1.65
```



## Variância

### sintaxe:

```
var(variável)
```

### Opções:

**na.rm:** TRUE, calcula a variância considerando apenas os dados existentes, ignora os dados faltantes.  
FALSE, calcula a variância apenas se todos os valores estiverem preenchidos, caso contrário retorna NA.

### Exemplo:

```
var(dados$Filhos)
Erro em var(dados$Filhos) : observações faltantes em cov/cor

var(dados$Filhos, na.rm=TRUE)
[1] 1.607895
```

## Desvio Padrão

### sintaxe:

```
sd(variável)
```

### Opções:

**na.rm:** TRUE, calcula o desvio padrão considerando apenas os dados existentes, ignora os dados faltantes.  
FALSE, calcula o desvio padrão apenas se todos os valores estiverem preenchidos, caso contrário retorna NA.

### Exemplo:

```
sd(dados$Filhos)
Erro em var(dados$Filhos) : observações faltantes em cov/cor

sd(dados$Filhos, na.rm=TRUE)
[1] 1.268028
```

## Mediana

Calcula a mediana do conjunto de dados.

```
median(variável)
```

### Opções:

**na.rm =** TRUE calcula a mediana considerando apenas os dados existentes, ignora os dados faltantes.  
FALSE calcula a mediana apenas se todos os valores estiverem preenchidos, caso contrário retorna NA.

### Exemplo:

```
median(dados$Filhos)
[1] NA

median(dados$Filhos, na.rm=TRUE)
[1] 2
```

## Aplica funções

Aplica a função desejada na variável escolhida segundo cada nível de um determinado fator.

### sintaxe:

```
tapply(variável, fator, função)
```

### Exemplo:

```
tapply(dados$Salario,dados$Estado,summary)
$Casado
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
4.560  8.743 11.930 12.120 15.030 23.300

$Solteiro
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
4.000  7.257  9.045  9.871 11.690 18.750

tapply(dados$Salario,dados$Estado,var)
Casado Solteiro
24.12800 15.53465
```

## Divisão de dados

Divide os dados em faixas determinadas.

### sintaxe:

```
cut(variável, faixas, rótulos, opções)
```

### Opções:

**right:** TRUE faz com que o intervalo seja fechado na direita e aberto na esquerda.  
FALSE faz com que o intervalo seja aberto na direita e fechado na esquerda.

### Exemplo:

```
cut(dados$Anos, breaks=c(20,30,40,50), labels=c("A","B","C"), right=FALSE)
[1] A B B A C A C C B A B A B C B B B B A B B B C A B B C A C B B B C B C C
Levels: A B C

table(cut(dados$Anos, breaks=c(20,30,40,50), labels=c("20|-30","30|-40","40|-50"),
right=F))

20|-30 30|-40 40|-50
   8      18      10

table(cut(dados$Salario, breaks=c(4,8,12,16,20,24), labels=c("4|-8","8|-12","12|-16",
"16|-20","20|-24"), right=F))

4|-8  8|-12 12|-16 16|-20 20|-24
 10    12    8      5      1
```

**Observação:** É possível a utilização da função cut() dentro da função table() diretamente. O resultado será uma tabela com a frequência de cada intervalo determinado pela função cut().

## lm

Cria um modelo linear que pode ser utilizado para regressão linear, análise de variância entre outros.

### sintaxe:

```
lm(formula, dados, opções)
```

### Parâmetros

**formula:** Especifica a relação entre a variável resposta e a variável explicativa. Ver exemplos.

**dados:** Objeto do tipo data.frame que contém os dados.

### Opções

**weights:** vetor opcional dos pesos a serem utilizados no processo de ajuste do modelo linear.

### Exemplos

Cria um modelo linear baseado na tabela 15.1 de Bussab e Morettin (2003), onde o objeto dados contém a tabela.

```
lm(Reação ~ Idade, dados)
```

#### Call:

```
lm(formula = Reação ~ Idade, data = dados)
```

#### Coefficients:

(Intercept)	Idade
80.5	0.9

A função `lm()` nos retorna a estimativa dos parâmetros do modelo linear da forma:  $E(Y | x) = \mu(x) = \alpha + \beta \cdot x$ , onde  $\alpha$  corresponde ao valor esperado para a variável dependente quando a variável explicativa assume o valor zero (também chamado de Intercepto ou coeficiente linear da reta) e  $\beta$  corresponde à variação média da variável dependente por unidade de variação da variável explicativa (ou coeficiente angular da reta). No exemplo acima, o modelo seria representado pela equação:  $\hat{E}(Y | x) = \hat{\mu}(x) = 80,5 + 0,9 \cdot x$ .

Caso necessário, pode-se remover o intercepto do modelo alterando a fórmula da seguinte maneira:

```
lm(Reação ~ Idade - 1, dados)
```

#### Call:

```
lm(formula = Reação ~ Idade - 1, data = dados)
```

#### Coefficients:

Idade
3.442

# Testes para a média populacional e para a comparação de duas médias

## **t.test()**

Realiza o teste t-Student para uma ou duas amostras.

### **sintaxe:**

```
t.test(amostra1, amostra2, opções)
```

### **Parâmetros**

**amostra1**: Vetor contendo a amostra da qual se quer testar a média populacional, ou comparar a média populacional com a média populacional da amostra 2.

**amostra2**: Vetor contendo a amostra 2 para comparação da média populacional com a média populacional da amostra 1.

### **Opções**

**alternative**: string indicando a hipótese alternativa desejada.  
Valores possíveis: "two-sided", "less" ou "greater".

**mu**: valor indicando o verdadeiro valor da média populacional para o caso de uma amostra, ou a diferença entre as médias para o caso de duas amostras.

**paired**: TRUE - realiza o teste t pareado.  
FALSE - realiza o teste t não pareado.

**var.equal**: TRUE - indica que a variância populacional é a igual nas duas amostras.  
FALSE - indica que a variância populacional de cada amostra é diferente.

**conf.level**: coeficiente de confiança do intervalo.

### **Exemplo:**

#### **Teste t para média populacional**

```
amostra1 = c(14.9,13.4,14.5,13.5,15.0,13.9,14.9,16.4,14.6,15.4)
t.test(amostra1,mu=15)
```

One Sample t-test

```
data: amostra1
t = -1.2252, df = 9, p-value = 0.2516
alternative hypothesis: true mean is not equal to 15
95 percent confidence interval:
 14.00375 15.29625
sample estimates:
mean of x
 14.65
```

#### **Teste t para comparação de duas médias com variâncias iguais**

```
amostra1 = c(16.6,13.4,14.6,15.1,12.9,15.2,14.0,16.6,15.4,13.0)
amostra2 = c(15.8,17.9,18.2,20.2,18.1,17.8,18.3,18.6,17.0,18.4)
t.test(amostra1, amostra2, var.equal = TRUE)
```

Two Sample t-test

```
data: amostra1 and amostra2
t = -6.0257, df = 18, p-value = 1.069e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.518003 -2.181997
sample estimates:
mean of x mean of y
 14.68      18.03
```

### Teste t pareado

```
antes = c(16.6,13.4,14.6,15.1,12.9,15.2,14.0,16.6,15.4,13.0)
depois = c(15.8,17.9,18.2,20.2,18.1,17.8,18.3,18.6,17.0,18.4)
t.test(antes,depois,paired=TRUE)
```

```
      Paired t-test
data:  antes and depois
t = -5.3231, df = 9, p-value = 0.000479
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.773642 -1.926358
sample estimates:
mean of the differences
      -3.35
```

# Testes para uma proporção populacional e para comparação de duas proporções

## `prop.test()`

Realiza o teste de proporções para uma ou duas amostras.

### sintaxe:

```
prop.test(x, n, p, opções)
```

### Parâmetros

**x**: Vetor contendo o número de sucessos em cada amostra.

**n**: Vetor contendo o número de realizações de cada amostra.

**p**: Vetor contendo as probabilidades de sucesso de cada amostra.

### Opções

**alternative**: string indicando a hipótese alternativa desejada.  
Valores possíveis: "two-sided", "less" ou "greater".

**conf.level**: coeficiente de confiança do intervalo.

**correct**: TRUE - indica que a correção de continuidade de Yates será aplicada.  
FALSE - indica que a correção de continuidade não será aplicada.

### Exemplo:

#### Teste para uma proporção populacional

```
prop.test(104,200,0.6,correct=F)
```

```
1-sample proportions test without continuity correction
```

```
data: 104 out of 200, null probability 0.6
X-squared = 5.3333, df = 1, p-value = 0.02092
alternative hypothesis: true p is not equal to 0.6
95 percent confidence interval:
 0.4510379 0.5882083
sample estimates:
      p
0.52
```

#### Teste para comparação de duas proporções

```
prop.test(c(104,50),c(200,95),correct=F)
```

```
2-sample test for equality of proportions without continuity
correction
```

```
data: c(104, 50) out of c(200, 95)
X-squared = 0.0103, df = 1, p-value = 0.9192
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.1282799 0.1156483
sample estimates:
 prop 1    prop 2
0.5200000 0.5263158
```

## **fisher.test()**

Realiza o teste exato de independência de linhas e colunas em uma tabela de contingência com as marginais fixas.

### **Sintaxe:**

```
fisher.test(x, opções)
```

### **Parâmetros**

**x**: Matriz (tabela) contendo a frequência de observações em cada casela.

### **Opções**

**alternative**: string indicando a hipótese alternativa desejada.  
Valores possíveis: "two-sided", "less" ou "greater".

**conf.int**: TRUE: calcula o intervalo de confiança para a razão de chances em tabelas de dimensão 2x2.

**conf.level**: coeficiente de confiança do intervalo.

### **Exemplo:**

Teste de independência do exemplo: Satisfação no trabalho, Agresti(2002, p.57)

Receita	Satisfação			
	Muito Insatisfeito	Pouco Insatisfeito	Moderadamente Satisfeito	Muito Satisfeito
< 15 mil	1	3	10	6
15-25 mil	2	3	10	7
25-40 mil	1	6	14	12
> 40 mil	0	1	9	11

```
Trabalho = matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4,  
  dimnames = list(Receita=c("< 15mil", "15-25mil", "25-40mil", "> 40mil"),  
  Satisfação=c("M.Insatisfeito", "P.Insatisfeito", "Mod.Satisfeito", "M.Satisfeito")))
```

```
fisher.test(Trabalho)
```

Fisher's Exact Test for Count Data

```
data: Trabalho  
p-value = 0.7827  
alternative hypothesis: two.sided
```

# Testes para Normalidade

Estes pacotes contém diversos testes que verificam se os dados amostrais contém evidências de serem oriundos de uma população com distribuição Normal.

## Pacote: base

Este pacote já está instalado.

### **shapiro.test()**

Realiza o teste de Shapiro-Wilk para normalidade.

#### **sintaxe:**

```
shapiro.test(amostra)
```

#### **Parâmetros**

**amostra**: Vetor contendo a amostra da qual se quer testar normalidade.  
Deve conter uma amostra de tamanho entre 3 e 5000.  
São permitidos *missing values*.

#### **Exemplo:**

```
shapiro.test(rnorm(10, mean=10, sd=4))  
Shapiro-Wilk normality test
```

```
data:  rnorm(10, mean = 10, sd = 4)  
W = 0.9779, p-value = 0.9532
```

## Pacote opcional: nortest

Este pacote precisa ser instalado. Ver guia primeiros contatos com R.

### **ad.test()**

Realiza o teste de Anderson-Darling para normalidade.

#### **sintaxe:**

```
ad.test(amostra)
```

#### **Parâmetros**

**amostra**: Vetor contendo a amostra da qual se quer testar normalidade.  
Deve conter uma amostra de tamanho maior ou igual a 7.  
São permitidos *missing values*.

#### **Exemplo:**

```
ad.test(rnorm(10, mean=10, sd=4))  
Anderson-Darling normality test
```

```
data:  rnorm(10, mean = 10, sd = 4)  
A = 0.2772, p-value = 0.5707
```



### **cvm.test()**

Realiza o teste de Cramer-von Mises para normalidade.

#### **sintaxe:**

```
cvm.test(amostra)
```

#### **Parâmetros**

**amostra:** Vetor contendo a amostra da qual se quer testar normalidade.  
Deve conter uma amostra de tamanho maior ou igual a 7.  
São permitidos missing values.

#### **Exemplo:**

```
cvm.test(rnorm(10, mean=10, sd=4))  
Cramer-von Mises normality test
```

```
data: rnorm(10, mean = 10, sd = 4)  
W = 0.0497, p-value = 0.4792
```

### **lillie.test()**

Realiza o teste de Lilliefors (Kolmogorov-Smirnov) para normalidade.

#### **sintaxe:**

```
lillie.test(amostra)
```

#### **Parâmetros**

**amostra:** Vetor contendo a amostra da qual se quer testar normalidade.  
Deve conter uma amostra de tamanho maior ou igual a 4.  
São permitidos missing values.

#### **Exemplo:**

```
lillie.test(rnorm(10, mean=10, sd=4))  
Lilliefors (Kolmogorov-Smirnov) normality test
```

```
data: rnorm(10, mean = 10, sd = 4)  
D = 0.2169, p-value = 0.2010
```

### **pearson.test()**

Realiza o teste Qui-quadrado de Pearson para normalidade.

#### **sintaxe:**

```
pearson.test(amostra)
```

#### **Parâmetros**

**amostra:** Vetor contendo a amostra da qual se quer testar normalidade.  
São permitidos missing values.

#### **Opções**

**n.classes:** Número de classes.  
São permitidos missing values.  
**adjust:** **TRUE:** o valor p é calculado de uma distribuição Qui-quadrado com o número de graus de liberdade igual ao número de classes - 3.  
**FALSE:** o valor p é calculado de uma distribuição Qui-quadrado com o número de graus de liberdade igual ao número de classes - 1.

#### **Exemplo:**

```
pearson.test(rnorm(10, mean=10, sd=4))  
Pearson chi-square normality test
```

```
data: rnorm(10, mean = 10, sd = 4)  
P = 2, p-value = 0.5724
```

## **sf.test()**

Realiza o teste de Shapiro-Francia para normalidade.

### **sintaxe:**

```
sf.test(amostra)
```

### **Parâmetros**

*amostra*: Vetor contendo a amostra da qual se quer testar normalidade.

Deve conter uma amostra de tamanho entre 5 e 5000.

São permitidos missing values.

### **Exemplo:**

```
sf.test(rnorm(10, mean=10, sd=4))
```

```
Shapiro-Francia normality test
```

```
data:  rnorm(10, mean = 10, sd = 4)
```

```
W = 0.935, p-value = 0.4419
```

# Testes para comparação de variâncias

## Pacote: stats

Este pacote já está instalado.

### `bartlett.test()`

Realiza o teste de Bartlett com a hipótese nula de que as variâncias dos grupos são iguais.

#### sintaxe:

```
bartlett.test(formula, dados)
```

#### Parâmetros

**formula**: Relação entre a variável dependente e o fator. Ex: "Vendas ~ Mês".

**dados**: Conjunto de dados onde será aplicada a formula.

#### Exemplo: Queremos comparar a variabilidade das vendas entre os meses

Loja	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
A	10	14	16	19	12	15	11	10	12	13	18	23
B	12	13	13	14	13	11	16	09	11	11	16	25
C	15	17	12	16	10	16	12	12	08	14	21	24

```
Vendas = c(10,12,15,14,13,17,16,13,12,19,14,16,12,
           13,10,15,11,16,11,16,12,10,9,12,12,
           11,8,13,11,14,18,16,21,23,25,24)
Mes = c("Jan", "Jan", "Jan", "Fev", "Fev", "Fev", "Mar", "Mar", "Mar", "Abr", "Abr", "Abr",
        "Mai", "Mai", "Mai", "Jun", "Jun", "Jun", "Jul", "Jul", "Jul", "Ago", "Ago", "Ago",
        "Set", "Set", "Set", "Out", "Out", "Out", "Nov", "Nov", "Nov", "Dez", "Dez", "Dez")
dados = data.frame(Vendas=Vendas, Mes=Mes)
```

```
bartlett.test(Vendas ~ Mes, data=dados)
```

Bartlett test of homogeneity of variances

data: Vendas by Mes

Bartlett's K-squared = 2.844, df = 11, p-value = 0.9926

**Observação:** Também é possível especificar os grupos da seguinte forma: `bartlett.test(list(GRUPO1, GRUPO2))`, onde GRUPO1 e GRUPO2 são vetores contendo os valores das observações de cada amostra.

#### Exemplo:

	Jogo 1	Jogo 2	Jogo 3	Jogo 4	Jogo 5	Jogo 6
Time A	30	25	32	22	19	26
Time B	18	24	31	28	29	30

```
TimeA = c(30,25,32,22,19,26)
```

```
TimeB = c(18,24,31,28,29,30)
```

```
bartlett.test(list(TimeA, TimeB))
```

Bartlett test of homogeneity of variances

data: list(TimeA, TimeB)

Bartlett's K-squared = 3e-04, df = 1, p-value = 0.9856

## Pacote: car

Este pacote precisa ser instalado.

### `levene.test()`

Realiza o teste de Bartlett com a hipótese nula de que as variâncias dos grupos são iguais.

#### sintaxe:

```
bartlett.test(formula, dados)
```

#### Parâmetros

*formula*: Relação entre a variável dependente e o fator. Ex: "Vendas ~ Mês".

*dados*: Conjunto de dados onde será aplicada a formula.

**Exemplo:** Utilizando o mesmo exemplo visto no teste de Bartlett:

Loja	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
A	10	14	16	19	12	15	11	10	12	13	18	23
B	12	13	13	14	13	11	16	09	11	11	16	25
C	15	17	12	16	10	16	12	12	08	14	21	24

```
levene.test(dados$Vendas, dados$Mes)
```

Levene's Test for Homogeneity of Variance

```
      Df F value Pr(>F)
group 11  0.1678 0.9981
      24
```

# Funções Matemáticas

## Combinatória

Calcula o número de combinações de **n** elementos em grupos de tamanho **k**.

### sintaxe:

```
choose(n,k)
```

### exemplo:

```
choose(0,0)  
[1] 1
```

```
choose(8,5)  
[1] 56
```

## Fatorial

Calcula o fatorial de **x**.

### sintaxe:

```
factorial(x)
```

### exemplo:

```
factorial(0)  
[1] 1
```

```
factorial(5)  
[1] 120
```

## Raiz Quadrada

Calcula a raiz quadrada de **x**.

### sintaxe:

```
sqrt(x)
```

### exemplo:

```
sqrt(0)  
[1] 0
```

```
sqrt(81)  
[1] 9
```

# Gráficos

Os gráficos nos permitem analisar uma grande quantidade de informações de forma rápida, sem que seja necessário olhar tabelas e medidas de resumo. O R possui uma enorme capacidade para gerar diversos tipos de gráficos de alta qualidade totalmente configuráveis, desde cores e tipos de linhas, até legendas e textos adicionais.

A grande maioria das funções gráficas faz uso de opções comuns, ou seja, é extremamente fácil personalizar qualquer tipo de gráfico pois muitas das opções são iguais. As opções comuns a todos os gráficos serão abordadas aqui, e em cada seção seguinte as opções específicas àquele determinado tipo de gráfico serão apresentadas.

## Opções:

**xlim:** (inicio,fim) dupla contendo os limites do eixo X.  
**ylim:** (inicio,fim) dupla contendo os limites do eixo Y.  
**xlab:** rótulo para o eixo X.  
**ylab:** rótulo para o eixo Y.  
**main:** título principal do gráfico.  
**col:** cor de preenchimento do gráfico, podendo ser um vetor. A lista das cores disponíveis pode ser obtida através do comando `colors()`.

## locator

Permite localizar uma coordenada clicando com o mouse no gráfico. Se não for definida a opção **type**, retorna apenas as coordenadas do ponto clicado. Útil para inserir textos e outros elementos em gráficos já prontos.

## sintaxe:

```
locator()
```

## Opções

**n:** Número máximo de pontos a localizar.  
**type:** p: cria pontos no gráfico com as coordenadas indicadas pelo mouse.  
l: cria linhas no gráfico com as coordenadas indicadas pelo mouse.

## text

Insere um texto nas coordenadas definidas.

## sintaxe:

```
text(x, y, labels, cex, col)
```

## Opções

**x:** Posição relativa a abscissa (eixo X).  
**y:** Posição relativa a ordenada (eixo Y).  
**labels:** Texto (ou vetor com textos) a ser inserido nas coordenadas definidas por x e y.  
**cex:** Proporção relativa ao tamanho dos caracteres do texto (padrão: 1).  
**col:** Cor do texto a ser inserido (padrão: preto).

## Gráfico de barras

Gráfico de freqüências para variáveis qualitativas.

**sintaxe:**

```
barplot(dados, opções)
```

**opções:**

**space:** espaço deixado antes de cada barra  
**width:** vetor contendo a largura relativa de cada barra com relação as demais. Valores iguais para todas as barras não terão efeito, pois a relação entre elas será 1.

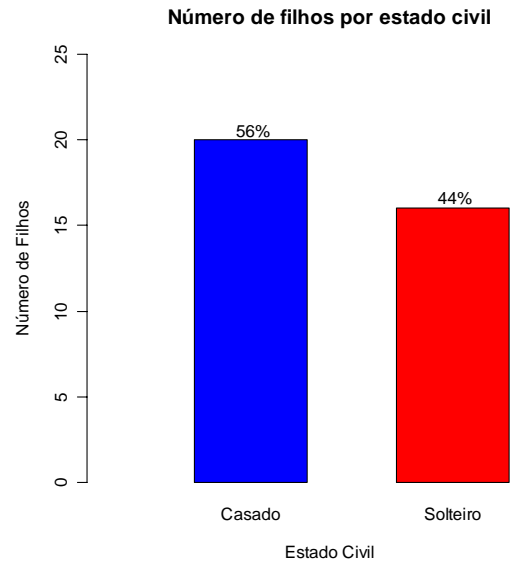
**exemplo:**

```
barplot(table(dados$estciv), col=c("blue","red"),  
        ylim=c(0,25),  
        space=.8, width=c(.2,.2),  
        main="Número de filhos por estado civil",  
        xlab="Estado Civil", ylab="Número de Filhos")
```

```
text(locator(n=2), c("56%", "44%"))
```

**Observação:**

- Caso não seja especificada a opção **xlim**, os valores da opção **width** não serão interpretados como **valores absolutos**, mas como **valores relativos** as demais barras.  
**Ex:** `barplot(table(dados$estciv), width(0.2,0.2))` tem o mesmo efeito que `barplot(table(dados$estciv), width(2,2))`
- Após o comando `text(locator(n=2), c("56%", "44%"))`, são necessários dois cliques em pontos do gráfico de barras onde serão inseridos os textos com os percentuais relativos a cada barra.



## Histograma

Gráfico de distribuição de freqüências para variáveis quantitativas.

**sintaxe:**

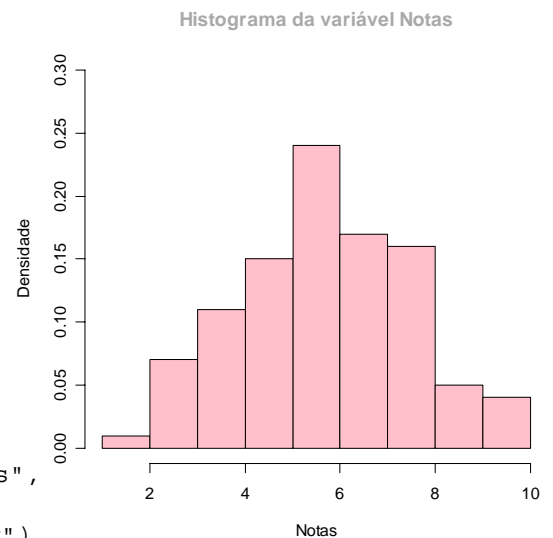
```
hist(dados, opções)
```

**opções:**

**prob:** T plota a densidade.  
F plota a freqüência absoluta.  
**breaks:** vetor contendo os pontos de definição das larguras das barra do histograma.

**exemplo:**

```
hist(cdnnotas$Notas, main="Histograma da variável Notas",  
     prob=T, xlab="Notas", ylab="Densidade",  
     col=c("pink"), ylim=c(0,0.3), col.main="darkgray")
```



## Boxplot

**sintaxe:**

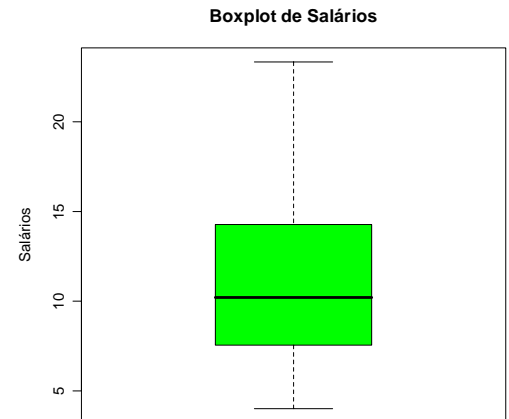
```
boxplot(dados, opções)
```

**opções:**

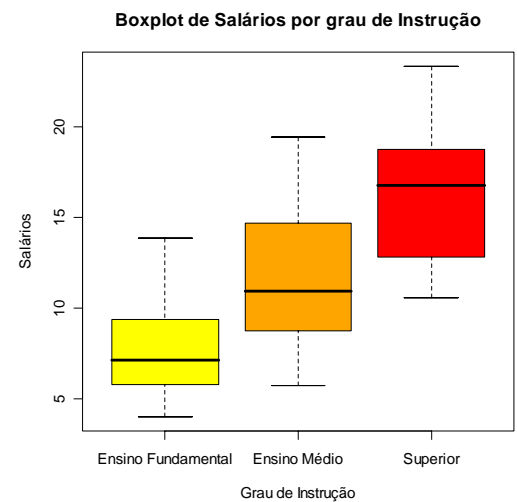
**outline:** T plota os outliers.  
F não plota os outliers.

**exemplo:**

```
boxplot(dados$salario, main="Boxplot de Salários",  
        ylab="Salários", col=("green"))
```



```
boxplot(dados$salario ~ dados$educa,  
        main="Boxplot de Salários por grau de Instrução",  
        xlab="Grau de Instrução", ylab="Salários",  
        col=c("yellow", "orange", "red"))
```



## Gráfico de Pizza

**sintaxe:**

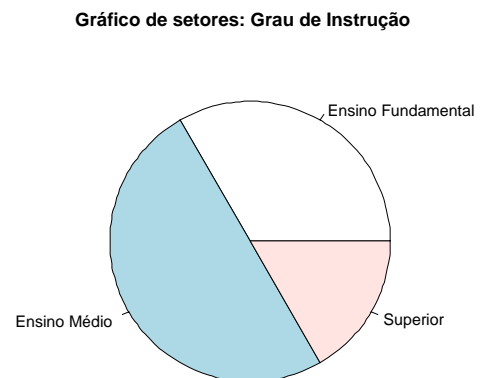
```
pie(dados, opções)
```

**opções:**

**labels:** vetor contendo os rótulos de cada fatia.  
**radius:** raio da circunferência da pizza. (padrão=1)  
**col:** vetor contendo as cores das fatias.

**exemplo:**

```
pie(table(dados$educa),  
    main="Gráfico de setores: Grau de Instrução")
```

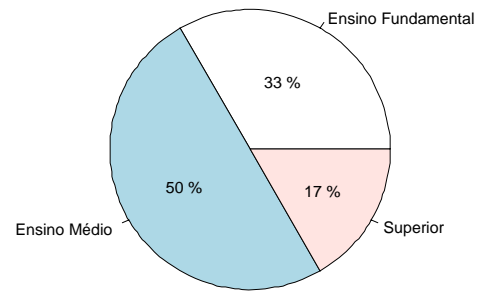




Para incluir as porcentagens dentro de cada fatia, execute as linhas abaixo clicando na fatia branca, azul e rosa, nesta sequência.

```
text(locator(n=1),
     paste(round(prop.table(table(dados$educa))[1],
                    digits=2)*100,"%"))
text(locator(n=1),
     paste(round(prop.table(table(dados$educa))[2],
                    digits=2)*100,"%"))
text(locator(n=1),
     paste(round(prop.table(table(dados$educa))[3],
                    digits=2)*100,"%"))
```

Gráfico de setores: Grau de Instrução



## Gráfico de Dispersão

Plota dados2 em função de dados1.

**sintaxe:**

```
plot(dados1, dados2, opções)
```

**opções:**

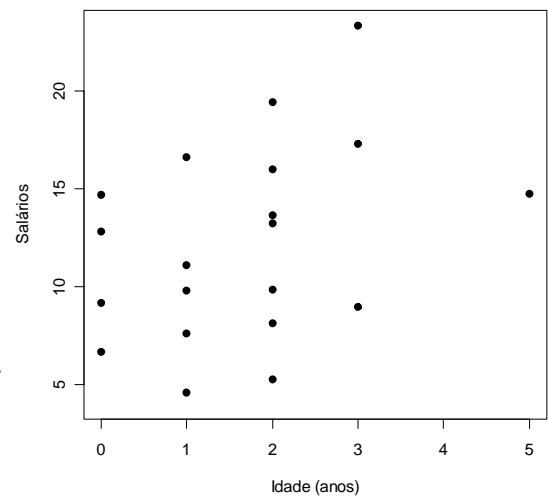
**pch:** Escolhe o tipo de caractere.

**lwd:** Espessura do caractere a ser plotado

**exemplo:**

```
plot(dados$filhos, dados$salario, pch=1, lwd=2,
     main="Gráfico de dispersão: Salário x Idade",
     xlab="Idade (anos)", ylab="Salários")
```

Gráfico de dispersão: Salário x Idade



# Probabilidade

## Função Densidade (ou Probabilidade)

Calcula o valor da densidade, no caso de distribuições contínuas, ou a probabilidade  $P(X=x)$ , no caso de distribuições discretas, para cada elemento do vetor  $x$ . No R o nome dessa função é iniciado pela letra **d** mais o nome da distribuição (ex: `dbinom`, `dpois` etc.).

## Função Distribuição

Calcula a distribuição acumulada:  $P(X \leq x)$ . O nome da função é iniciado pela letra **p** mais o nome da distribuição (ex: `pnorm`, `pexp` etc.).

## Função Probabilidade

Calcula o valor de  $x$  correspondente a probabilidade  $p$  acumulada. É o inverso da função Distribuição. O nome da função é iniciado pela letra **q** mais o nome da distribuição (ex: `qbeta`, `qcauchy` etc.).

## Gerador aleatório

Gera números aleatórios baseados na distribuição definida. O nome da função é iniciado pela letra **r** mais o nome da distribuição (ex: `rnorm`, `rbinom` etc.).

Distribuição	Sintaxe	Opções
Binomial	<code>dbinom(x, n, p, opções)</code>	<b>x</b> : vetor contendo o número total de sucessos em $n$ ensaios de Bernoulli. <b>p</b> : probabilidade de sucesso. <b>q</b> : vetor contendo os quantis em $n$ ensaios de Bernoulli. <b>prob</b> : vetor contendo as probabilidades em $n$ ensaios de Bernoulli. <b>obs</b> : número de observações. <b>n</b> : número de ensaios de Bernoulli.
	<code>pbinom(q, n, p, opções)</code>	
	<code>qbinom(prob, n, p, opções)</code>	
	<code>rbinom(obs, n, p)</code>	
Poisson	<code>dpois(x, lambda, opções)</code>	<b>x</b> : vetor contendo o número de ocorrências. <b>lambda</b> : número médio de eventos ocorrendo no intervalo considerado. <b>q</b> : vetor contendo os quantis. <b>prob</b> : vetor contendo as probabilidades. <b>obs</b> : número de observações.
	<code>ppois(q, lambda, opções)</code>	
	<code>qpois(prob, lambda, opções)</code>	
	<code>rpois(obs, lambda)</code>	
Geométrica	<code>dgeom(x, prob, opções)</code>	<b>x</b> : vetor contendo o número de falhas ocorridas em uma sequência de Bernoulli antes do sucesso. <b>prob</b> : probabilidade de sucesso em cada tentativa. <b>q</b> : vetor contendo os quantis <b>p</b> : vetor contendo as probabilidades <b>n</b> : número de observações.
	<code>pgeom(q, prob, opções)</code>	
	<code>qgeom(p, prob, opções)</code>	
	<code>rgeom(n, prob)</code>	

Hipergeométrica	<code>dhyper(x, m, n, k, opções)</code>	<b>x</b> : vetor contendo o número de elementos com a característica A extraídos sem reposição de uma urna que contém elementos com características A e B. <b>m</b> : número de elementos com a característica A. <b>n</b> : número de elementos com a característica B. <b>k</b> : número de elementos extraídos da urna. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>nn</b> : número de observações.
	<code>phyper(q, m, n, k, opções)</code>	
	<code>qhyper(p, m, n, k, opções)</code>	
	<code>rhyper(nn, m, n, k)</code>	
Uniforme	<code>dunif(x, min=0, max=1, opções)</code>	<b>x</b> : vetor contendo os quantis. <b>min</b> : limite inferior da distribuição. <b>max</b> : limite superior da distribuição. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.
	<code>punif(q, min=0, max=1, opções)</code>	
	<code>qunif(p, min=0, max=1, opções)</code>	
	<code>runif(n, min=0, max=1)</code>	
Exponencial $E(X)=1/\text{rate}$ $\text{Var}(X)=1/\text{rate}^2$	<code>dexp(x, rate=1, opções)</code>	<b>x</b> : vetor contendo os quantis. <b>rate</b> : parâmetro da distribuição. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.
	<code>pexp(q, rate=1, opções)</code>	
	<code>qexp(p, rate=1, opções)</code>	
	<code>rexp(n, rate = 1)</code>	
Normal	<code>dnorm(x, mean=0, sd=1, opções)</code>	<b>x</b> : vetor contendo os quantis. <b>mean</b> : média da distribuição. <b>sd</b> : desvio padrão da distribuição. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.
	<code>pnorm(q, mean=0, sd=1, opções)</code>	
	<code>qnorm(p, mean=0, sd=1, opções)</code>	
	<code>rnorm(n, mean=0, sd=1)</code>	
Gama $E(X)=\text{shape}/\text{rate}$ $\text{Var}(X) = \text{shape}/\text{rate}^2$	<code>dgamma(x, shape, rate = 1, opções)</code>	<b>x</b> : vetor contendo os quantis. <b>shape</b> : parâmetro da distribuição. <b>rate</b> : parâmetro da distribuição. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.
	<code>pgamma(q, shape, rate = 1, opções)</code>	
	<code>qgamma(p, shape, rate = 1, opções)</code>	
	<code>rgamma(n, shape, rate = 1)</code>	
Qui-quadrado	<code>dchisq(x, df, ncp=0, opções)</code>	<b>x</b> : vetor contendo os quantis. <b>df</b> : graus de liberdade. <b>ncp</b> : parâmetro de não centralidade. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.
	<code>pchisq(q, df, ncp=0, opções)</code>	
	<code>qchisq(p, df, ncp=0, opções)</code>	
	<code>rchisq(n, df, ncp=0)</code>	
t-Student	<code>dt(x, df, ncp=0, opções)</code>	<b>x</b> : vetor contendo os quantis. <b>df</b> : graus de liberdade. <b>ncp</b> : parâmetro de não centralidade. <b>q</b> : vetor contendo os quantis. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.
	<code>pt(q, df, ncp=0, opções)</code>	
	<code>qt(p, df, opções)</code>	
	<code>rt(n, df)</code>	

F-Snedecor	<code>df(x, df1, df2, opções)</code>	<b>x</b> : vetor contendo os quantis.
	<code>pf(q, df1, df2, ncp=0, opções)</code>	<b>df1</b> : graus de liberdade do numerador. <b>df2</b> : graus de liberdade do denominador.
	<code>qf(p, df1, df2, opções)</code>	<b>q</b> : vetor contendo os quantis.
	<code>rf(n, df1, df2)</code>	<b>ncp</b> : parâmetro de não centralidade. <b>p</b> : vetor contendo as probabilidades. <b>n</b> : número de observações.

O termo `opções` que aparece em algumas funções pode ser substituído por:

`lower.tail = (TRUE)`

**TRUE**: Calcula as probabilidades da forma  $P(X \leq x)$ .

**FALSE**: Calcula as probabilidades da forma  $P(X > x)$ .

`log = (FALSE)`

`log.p = (FALSE)`

**TRUE**: As probabilidades  $p$  são dadas como o logaritmo natural de  $p$  (  $\log(p)$  ).

Em caso de dúvida, digite `?nome_da_função` para obter a ajuda necessária.

Para mais distribuições, utilize o comando: `help.search("distribution")` e em seguida utilize: `?Nome_da_distribuição` para mais informações sobre a distribuição escolhida.

# Exemplos

## Dividindo as observações em intervalos

```
salario <- c(12, .4, 5, 2, 50, 8, 3, 1, 4, .25)
intervalo <- cut(salario,breaks=c(0,1,5,max(salario)))
intervalo
[1] (5,50] (0,1] (1,5] (1,5] (5,50] (5,50] (1,5] (0,1] (1,5] (0,1]
Levels: (0,1] (1,5] (5,50]
```

## Frequência de observações em cada intervalo

```
table(intervalo)
intervalo
(0,1] (1,5] (5,50]
      3      4      3
intervalo
[1] (5,50] (0,1] (1,5] (1,5] (5,50] (5,50] (1,5] (0,1] (1,5] (0,1]
Levels: (0,1] (1,5] (5,50]
```

## Rotulando os intervalos

```
levels(intervalo) <-c("pobre","rico","rolando na grana")
table(intervalo)
intervalo
      pobre      rico  rolando na grana
      3      4      3
```

## Criando variáveis e exibindo-as em tabelas

```
Peso= c(60, 75, 55, 68)
Altura = c(65, 61, 70, 65)
Genero = c("Fe", "Fe", "M", "Fe")
estudo = data.frame(Peso,Altura,Genero)
estudo
  Peso Altura Genero
1   60     65     Fe
2   75     61     Fe
3   55     70      M
4   68     65     Fe
```

## Alterando os rótulos das variáveis

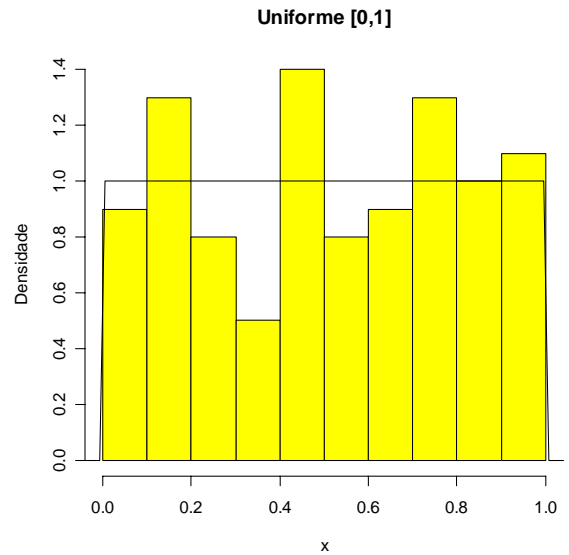
```
estudo = data.frame(P=Peso,H=Altura,G=Genero)
row.names(estudo)<-c("Maria","Alice","André","Julia")
estudo
      P  H  G
Maria 60 65 Fe
Alice 75 61 Fe
André 55 70  M
Julia 68 65 Fe
```

## Histogramas com curvas teóricas

```
x=runif(100)

hist(x,probability=TRUE,main="Uniforme
[0,1]",
ylab="Densidade",col="yellow")

curve(dunif(x,0,1),add=T)
```



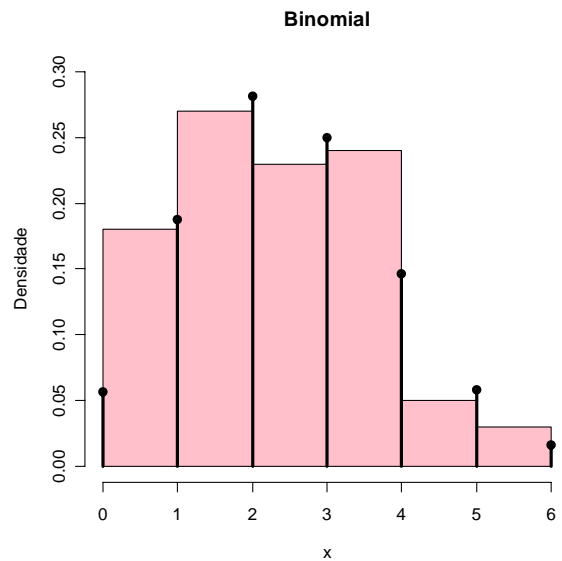
```
n=10;p=0.25
x=rbinom(100,n,p)

hist(x,probability=TRUE,ylab="Densidade",
col="pink", main="Binomial",ylim=c(0,0.30))

xvalores=0:n

points(xvalores,dbinom(xvalores,n,p),type="h",
lwd=3)

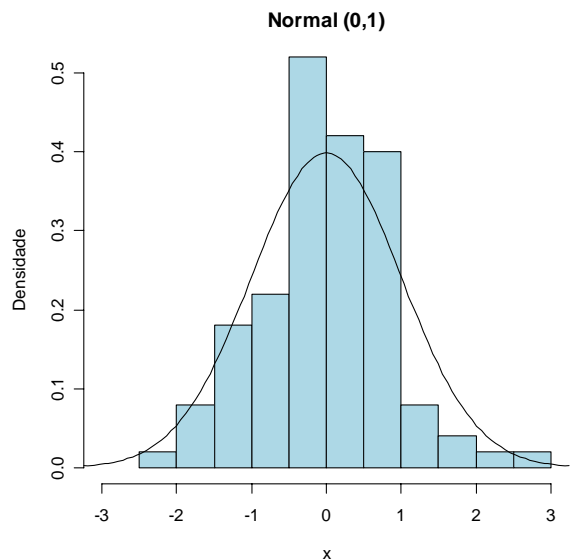
points(xvalores,dbinom(xvalores,n,p),type="p",
lwd=3)
```



```
x=rnorm(100)

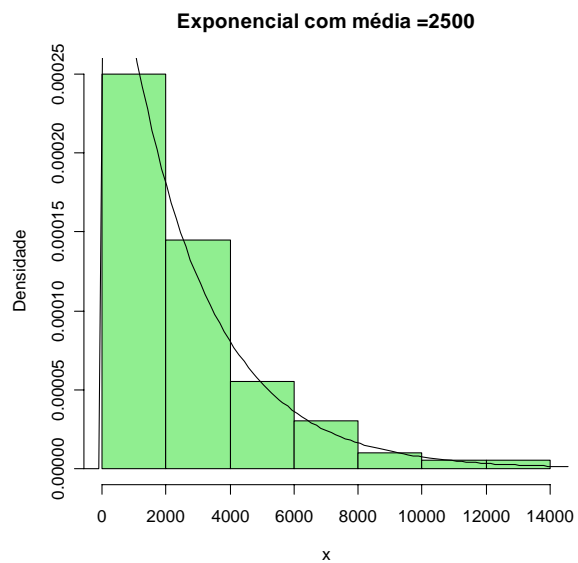
hist(x,probability=TRUE,col="lightblue",
main="Normal(0,1)",ylab="Densidade",
ylim=c(0,0.5),xlim=c(-3,3))

curve(dnorm(x),add=T)
```



```
x=rexp(100,1/2500)
hist(x,probability=TRUE,
col="lightgreen",main="Exponencial
com média=2500",ylab="Densidade")

curve(dexp(x,1/2500),add=T)
```



# Bibliografia

Bussab, W. de O. e Morettin, P. A. (2003). **Estatística Básica**, 5ª ed. São Paulo: Editora Saraiva.

The R Project for Statistical Computing (08/2006). **[www.r-project.org](http://www.r-project.org)**