CE-701 Bioestatística Avançada I

Primeiro Semestre de 2004

Paulo Justiniano Ribeiro Junior & James J Roper

Última atualização: 3 de junho de 2004

1 Usando o LINUX no LABEST

Nesta aula é feita uma introdução ao sistema operacional *LINUX* que vem sendo adotado no LABEST. É ainda mostrado como rodar o programa `R` neste sistema.

1.1 Comandos básicos do *LINUX*

Aqui estão alguns comandos básicos do *LINUX*:

Todos os comandos são documentados com `man` e possuem diversas outras opções.
Por exemplo para ver a documentação e opções do comando `tail` digite:
`man tail`
Para sair da tela de ajuda co comando basta digitar a tecla `q`

1.2 Praticando alguns comandos

Entre em sua conta, abra um terminal (clique no botão `xterm`) e faça o seguinte, utilizando os comandos da tabela acima.

1. inspecione o conteúdo do diretório com o comando `ls`
2. use o editor `nano` para criar um arquivo chamando `arquivo.txt`. Para abrir o editor digite no prompt do Linux:

   ```
   nano
   ```

   Digite o texto abaixo no editor:

   ```
   Este é um texto digitado no Linux usando o editor nano.
   ```

3. grave o arquivo e saia do editor. Para isto veja as opções na parte de baixo da tela do *nano*. Note que o caracter `^` corresponde à tecla `CTRL`. Portanto para gravar o arquivo voce vai precisar teclar `CTRL-O` (tecla "control" mais o caracter "O")
4. inspecione novamente o conteúdo do diretório com o comando `ls`
5. troque o nome do arquivo de `arquivo.txt` para `arq1.txt`
6. use o comando `more` para visualizar o conteúdo do arquivo

Tabela 1: Alguns comandos básicos do LINUX

who	mostra os usuários logados no sistema
w	também mostra os usuários logados no sistema
quota -v	mostra informações sobre cotas na área do usuário
du -hs *	mostra o espaço usado por cada arquivo/diretório de usuário
ls ls -l ls -a	lista conteúdo do diretório local mostra conteúdo detalhado mostra arquivos escondidos
mkdir	cria diretório
cp cp -r	copia arquivo copia recursivamente (para copiar diretórios)
mv	mover ou renomear arquivo/diretório
rm rm -r rm -rf	apaga arquivo apaga recursivamente apaga recursivamente sem confirmação (use com cuidado!)
cd	muda de diretório
pwd	mostra o diretório atual
cat, more ou less	mostram conteúdos de arquivo
tail head	mostra final de arquivo mostra começo de arquivo
zip e unzip	comprime/descomprime arquivos .zip
gzip e gunzip	comprime/descomprime arquivos .gz
gv xpdf	mostra arquivos postscript (.ps) mostra arquivos em ¨portable document format¨ (.pdf)
ssh scp	acessa outra máquina Linux via protocolo seguro SSH copia arquivos entre máquinas Linux via protocolo seguro
grep	procura por palavra ou expressão em um ou mais arquivos
rgrep	procura por palavra ou expressão recursivamente
chmod	muda permissão de arquivos e diretórios
locate	procura por um nome de arquivo/diretório
passwd	troca a senha
nano	abre o editor *nano*
emacs	abre o editor *emacs*
kile	abre o editor *kile* adequado para edição de textos em LaTeX
mozilla opera	abre o browser *Mozilla* abre o browser *Ópera*
ooffice	abre o *OpenOfficce*
R	abre o programa R
disquete*	abre programa para transferência de arquivos da área do usuário para disquete inserido em drive local

O símbolo * indica comando exclusivo para uso nos terminais do LABEST.

7. crie um diretório chamando `aula1`
8. copie o arquivo `arq1.txt` para dentro deste diretório
9. digite `pwd` e veja (e entenda) o que sai na tela
10. entre no diretório `aula1`

11. digite novamente `pwd` e veja o que sai na tela
12. volte para o seu diretório "raiz" usando o comando `cd`
13. digite `pwd` de novo e veja "onde voce está agora" (em qual diretório)
14. digite o comando `ls` e veja o resultado
15. apague o arquivo `arq1.txt`
16. digite novamente o comando `ls` e veja o resultado
17. entre no diretório `aula1`
18. use o comando `pwd` para ver se voce está no diretório correto
19. abra agora um novo arquivo chamando `arq2.R` usando o `emacs`
20. digite neste arquivo as seguinte linhas:

```
x <- rnorm(100)
summary(x)
hist(x)
sum(x > 0)
```

21. grave o arquivo e feche o editor `emacs`
22. veja o conteúdo do diretório com o comando `ls`
23. abra o editor `openoffice` e digite o seguinte texto

```
Este é um texto digitado no Linux usando o editor OpenOffice.

O Openoffice é uma alternativa ao MS-Office.
```

24. grave o texto num arquivo com o nome `arq3` no formato do openoffice
25. grave o texto num arquivo com o nome `arq3` no formato do MS-Word (extensão .doc)
26. feche o editor e retorne à linha de comando
27. liste os arquivos agora exitentes em seu diretório `aula1`
28. use o Openoffice para criar uma planilha com os seguinte dados

```
A   12
A   13
A   11
A   10
B   14
B   15
B   12
B   13
```

29. salve esta planilha num arquivo com o nome `arq4` no formato openoffice

30. salve esta planilha num arquivo com o nome `arq4` no formato do MS-Excel
31. feche o programa openoffice
32. liste os arquivos nos seu diretório
33. volte ao seu diretório raiz.

1.3 Alguns links

Alguns links com material introdutório sobre o *LINUX*:

- Apostila preparada por Stonebank é um excelente material introdutório.
- A Apostila preparada pelo PET-Informática é um excelente material introdutório.
- O Linux é um sítio copm muitas dicas e tutoriais.

Links para algumas distribuições *LINUX*:

- Kurimin Linux é um Linux que voce pode rodar a partir de um CD-ROM.
- Debian-Linux é a distribuição usada no LABEST.
- Documentação do Conectiva-Linux. O Conectiva é uma distribuição cuja a sede é em Curitiba-PR.
- e veja também a documentação do Mandrake Linux

1.4 Rodando o programa R no *LINUX*

O programa R pode ser rodado no *LINUX* de duas formas:

1. na linha do comando do LINUX (console) – basta digitar `R` na linha de comando do Linux.
2. dentro do editor *Xemacs (ou emacs)*, assim como é feito no Windows. Para isto inicie o editor com o comando `emacs &` e depois inicie o Rcom a combinação de teclas `ESC SHIFT-X SHIFT-R`.

Neste curso será dada preferência à segunda forma, i.e. rodar o R dentro do *Emacs*. Maiores detalhes sobre este mecanismo são fornecidos no Tutorial de Introdução ao R.

2 Introdução

Neste curso vamos utilizar um programa computational gratuito, de código aberto e livremente distribuído chamado R, que proporciona um ambiente para análises estatísticas.

2.1 O projeto R

O programa R é gratuito e de código aberto que propicia excelente ambiente para análises estatísticas e com recursos gráficos de alta qualidade. Detalhes sobre o projeto, colaboradores, documentação e diversas outras informações podem ser encontradas na página oficial do projeto em:
http://www.r-project.org.

O programa pode ser copiado livremente pela internet. Há um espelho (*mirror*) brasileiro da área de *downloads* do programa no *Departamento de Estatística da UFPR*:
http://www.est.ufpr.br/R
ou então via FTP:
ftp://est.ufpr.br/R

Será feita uma apresentação rápida da página do R durante o curso onde os principais recursos serão comentados assim como as idéias principais que governam o projeto e suas direções futuras.

2.2 Um tutorial sobre o R

Além dos materiais disponíveis na página do programa há também um *Tutorial de Introdução ao* R disponível em http://www.est.ufpr.br/Rtutorial.

Sugerimos aos participantes deste curso que percorram todo o conteúdo deste tutorial e retornem a ele sempre que necessário no decorrer do curso.

2.3 Cartão de referência

Para operar o R é necessário conhecer e digitar comandos. Isto pode trazer alguma dificuldade no inicio até que o usuário de familiarize com os comandos mais comuns. Uma boa forma de aprender e memorizar os comandos básicos é utilizar o Cartão de Referência que contém os comandos mais frequentemente utilizados.

2.4 Utilizando o R

Siga os seguintes passos.

1. inicie o R em seu computador.

2. voce verá uma janela de comandos com o símbolo >.
   Este é o *prompt* do R indicando que o programa está pronto para receber comandos.

3. a seguir digite (ou "recorte e cole") os comandos mostrados abaixo.
   No restante deste texto vamos seguir as seguintes convenções.

   - comandos do R são sempre mostrados em fontes do tipo `typewriter como esta`,
   - linhas iniciadas pelo símbolo # são comentários e são ignoradas pelo R.

3 Aritmética e Objetos

3.1 Operações aritméticas

Voce pode usar o R para avaliar algumas expressões aritméticas simples. Por exemplo:

```
> 1+2+3         # somando estes números ...
[1] 6           # obtem-se a resposta marcada com [1]

> 2+3*4         # um pouquinho mais complexo
[1] 14          # prioridade de operações (multiplicação primeiro)

> 3/2+1
[1] 2.5         # assim como divisão

> 4*3**3        # potências são indicadas por  ** ou ^
[1] 108         # e tem prioridade sobre multiplicação e divisão
```

O símbolo `[1]` pode parecer estranho e será explicado mais adiante.
O R também disponibiliza funções como as que voce encontra em uma calculadora:

```
> sqrt(2)
[1] 1.414214

> sin(3.14159)       # seno(Pi radianos) é zero
[1] 2.65359e-06      # e a resposta é bem próxima ...
```

O valor Pi está disponível como uma constante. Tente isto:

```
> sin(pi)
[1] 1.224606e-16     bem mais próximo de zero ...
```

Aqui está uma lista resumida de algumas funções aritméticas no R:

sqrt	raiz quadrada
abs	valor absoluto (positivo)
sin cos tan	funções trigonométricas
asin acos atan	funções trigonométricas inversas
sinh cosh tanh	funções hiperbólicas
asinh acosh atanh	funções hiperbólicas inversas
exp log	exponencial e logarítmo natural
log10	logarítimo base-10

Estas expressões podem ser agrupadas e combinadas em expressões mais complexas:

```
> sqrt(sin(45*pi/180))
[1] 0.8408964
```

3.2 Objetos

O R é uma linguagem orientada à objetos: variáveis, dados, matrizes, funções, etc são armazenados na memória ativa do computador na forma de objetos. Por exemplo, se um objeto x tem o valor 10 ao digitarmos e seu nome e programa exibe o valor do objeto:

```
> x
[1] 10
```

O dígito 1 entre colchetes indica que o conteúdo exibido inicia-se com o primeiro elemento de x. Voce pode armazenar um valor em um objeto com certo nome usando o símbolo ¡- (ou -¿). Exemplos:

```
> x <- sqrt(2)      # armazena a raiz quadrada de 2 em x
> x                 # digite o nome do objeto para ver seu conteúdo
[1] 1.414214
```

Alternativamente podem-se usar o símbolos `->`, `=` ou _. As linhas a seguir produzem o mesmo resultado.

```
> x <- sin(pi)    # este é o formato ‘‘tradicional’’
> sin(pi) -> x
> x = sin(pi)     # este formato foi introduzido em versões mais recentes
```

Neste material será dada preferência ao primeiro símbolo. Usuários pronunciam o comando dizendo que o objeto recebe um certo valor. Por exemplo em `x <- sqrt(2)` dizemos que "x recebe a raiz quadrada de 2". Como pode ser esperado voce pode fazer operações aritméticas com os objetos.

```
> y <- sqrt(5)      # uma nova variável chamada y
> y+x               # somando valores de x e y
[1]  2.236068
```

Note que ao atribuir um valor a um objeto o programa não imprime nada na tela. Digitando o nome do objeto o programa imprime seu conteúdo na tela. Digitando uma operação aritmética, sem atribuir o resultado a um objeto, faz com que o programa imprima o resultado na tela. Nomes de variáveis devem começar com uma letra e podem conter letras, números e pontos. Maiúsculas e minúsculas são consideradas diferentes. DICA: tente atribuir nomes que tenham um significado lógico. Isto facilita lidar com um grande número de objetos. Ter nomes como a1 até a20 pode causar confusão ... Aqui estão alguns exemplo válidos

```
> x <- 25
> x * sqrt(x) -> x1
> x2.1 <- sin(x1)
> xsq <- x2.1**2 + x2.2**2
```

E alguns que NÃO são válidos

```
> 99a <- 10      #‘99a’ não começa com letra
> a1 <- sqrt 10  # Faltou o parêntesis em sqrt
> a1_1 <- 10     # Não pode usar o ’underscore’ em um nome
> a-1 <- 99      # hífens também não podem ser usados...
> sqrt(x) <- 10  # não faz sentido...
```

4 Tipos de objetos

Os tipos básicos de objetos do Rsão:

- vetores
- matrizes e arrays
- data-frames
- listas
- funções

Experimente os comandos listados para se familiarizar com estas estruturas.

4.1 Vetores

```
x1 <- 10
x1

x2 <- c(1, 3, 6)
x2
x2[1]
x2[2]
length(x2)
is.vector(x2)
is.matrix(x2)
is.numeric(x2)
is.character(x2)

x3 <- 1:10
x3

x4 <- seq(0,1, by=0.1)
x4
x4[x4 > 0.5]
x4 > 0.5

x5 <- seq(0,1,  len=11)
x5

x6 <- rep(1, 5)
x6

x7 <- rep(c(1, 2), c(3, 5))
x7

x8 <- rep(1:3, rep(5,3))
x8

x9 <- rnorm(10, mean=70, sd=10)
```

```
x9
sum(x9)
mean(x9)
var(x9)
min(x9)
max(x9)
summary(1:10)

x10 <- x9[x9 > 72]
```

Para mais detalhes sobre vetores voce pode consultar as seguinte páginas:

- Vetores
- Aritmética de vetores
- Caracteres e fatores
- Vetores Lógicos
- Índices

4.2 Matrizes

```
m1 <- matrix(1:12, ncol=3)
m1
length(m1)
dim(m1)
nrow(m1)
ncol(m1)
m1[1,2]
m1[2,2]
m1[,2]
m1[3,]
dimnames(m1)
dimnames(m1) <- list(c("L1", "L2", "L3","L4"), c("C1","C2","C3"))
dimnames(m1)
m1[c("L1","L3"),]
m1[c(1,3),]

m2 <- cbind(1:5, 6:10)
m2

m3 <- cbind(1:5, 6)
m3
```

Para mais detalhes sobre matrizes consulte a página:

- Matrizes

4.3 Arrays

O conceito de *array* generaliza a idéia de matrix. Enquanto em uma matrix os elementos são organizados em duas dimensões (linhas e colunas), em um array os elementos podem ser organizados em um número arbitrário de dimensões.

No R um *array* é definido utilizando a função array().

1. Defina um array com o comando a seguir e inspecione o objeto certificando-se que voce entendeu como *arrays* são criados.

   ```
   ar1 <- array(1:24, dim=c(3,4,2))
   ar1
   ```

   Examine agora os seguinte comandos:

   ```
   ar1 <- array(1:24, dim=c(3,4,2))
   ar1[,2:3,]
   ar1[2,,1]
   sum(ar1[,,1])
   sum(ar1[1:2,,1])
   ```

2. Inspecione o "help" da função array (digite `help(array)`), rode e inspecione os exemplos contidos na documentação.

Veja agora um exemplo de dados já incluido no R no formato de array. Para "carregar" e visualizar os dados digite:

```
data(Titanic)
Titanic
```

Para maiores informações sobre estes dados digite:

```
help(Titanic)
```

Agora responda às seguintes perguntas, mostrando os comandos do R utilizados:

1. quantas pessoas havia no total?
2. quantas pessoas havia na tripulação (crew)?
3. quantas crianças sobreviveram?
4. qual a proporção (em %) entre pessoas do sexo masculino e feminino entre os passageiros da primeira classe?
5. quais sao as proporções de sobreviventes entre homens e mulheres?

4.4 Data-frames

```
d1 <- data.frame(X = 1:10, Y = c(51, 54, 61, 67, 68, 75, 77, 75, 80, 82))
d1
names(d1)
d1$X
d1$Y
plot(d1)
plot(d1$X, d1$Y)

d2 <- data.frame(Y= c(10+rnorm(5, sd=2), 16+rnorm(5, sd=2), 14+rnorm(5, sd=2)))
d2$lev <- gl(3,5)
d2
by(d2$Y, d2$lev, summary)

d3 <- expand.grid(1:3, 4:5)
d3
```

Para mais detalhes sobre data-frame consulte a página:

- Data-frames

4.5 Listas

Listas são estruturas genéricas e flexíveis que permitem armazenar diversos formatos em um único objeto.

```
lis1 <- list(A=1:10, B="THIS IS A MESSAGE", C=matrix(1:9, ncol=3))
lis1

lis2 <- lm(Y ~ X, data=d1)
lis2
is.list(lis2)
class(lis2)
summary(lis2)
anova(lis2)
names(lis2)
lis2$pred
lis2$res
plot(lis2)

lis3 <- aov(Y ~ lev, data=d2)
lis3
summary(lis3)
```

4.6 Funções

O conteúdo das funções podem ser vistos digitando o nome da função (sem os parênteses).

```
lm
glm
plot
plot.default
```

Entretanto isto não é disponível desta forma para todas as funções como por exemplo em:

```
min
max
rnorm
lines
```

Nestes casos as funções não são escritas em linguagem R (em geral estão escritas em C) e voce tem que examinar o código fonte do R para visualizar o conteúdo das funções.

5 Entrando com dados

Pode-se entrar com dados no R de diferentes formas. O formato mais adequado vai depender do tamanho do conjunto de dados, e se os dados já existem em outro formato para serem importados ou se serão digitados diretamente no R.

A seguir são descritas 4 formas de entrada de dados com indicação de quando cada uma das formas deve ser usada. Os três primeiros casos são adequados para entrada de dados diretamente no R, enquanto o último descreve como importar dados já disponíveis eletronicamente.

5.1 Definindo vetores

Podemos entrar com dados definindo vetores com o comando `c()` ("c" corresponde a *concatenate*) ou usando funções que criam vetores. Veja e experimente com os seguinte exemplos.

```
a1 <- c(2,5,8)  # cria vetor a1 com os dados 2, 5 e 8
a1              # exibe os elementos de a1

a2 <- c(23,56,34,23,12,56)
a2
```

Esta forma de entrada de dados é conveniente quando se tem um pequeno número de dados.

Quando os dados tem algum "padrão" tal como elementos repetidos, números sequenciais pode-se usar mecanismos do R para facilitar a entrada dos dados como vetores. Examine os seguintes exemplos.

```
a3 <- 1:10    # cria vetor com números sequenciais de 1 a 10
a3

a4 <- (1:10)*10  # cria vetor com elementos 10, 20, ..., 100
a4

a5 <- rep(3, 5)  # cria vetor com elemento 3 repetido 5 vezes
a5

a6 <- rep(c(5,8), 3)  # cria vetor repetindo 3 vezes 5 e 8 alternadamente
a6

a7 <- rep(c(5,8), each=3)  # cria vetor repetindo 3 vezes 5 e depois 8
a7
```

5.2 Usando a função `scan`

Está função coloca o Rem modo *prompt* onde o usuário deve digitar cada dado seguido da tecla ¡ENTER¿. Para encerrar a entrada de dados basta digitar ¡ENTER¿ duas vezes consecutivas. Veja o seguinte resultado:

```
y <- scan()
#1: 11
#2: 24
#3: 35
#4: 29
#5: 39
```

```
#6: 47
#7:
#Read 6 items

y
#[1] 11 24 35 29 39 47
```

Este formato é maais ágil que o anterior e é conveniente para digitar vetores longos.

5.3 Usando a função `edit`

O comando `edit(data.frame())` abre uma planilha para digitação de dados que são armazanados como *data-frames*. Data-frames são o análogo no R a uma planilha.

Portanto digitando

```
a8 <- edit(data.frame())
```

será aberta uma planilha na qual os dados devem ser digitados. Quando terminar de entrar com os dados note que no canto superior direito da planilha existe um botão ¡QUIT¿. Pressionando este botão a planilha será fechada e os dados serão gravados no objeto indicado (no exemplo acima no objeto `a8`).

Se voce precisar abrir novamente planilha com os dados, para fazer correções e/ou inserir mais dados use o comando `fix`. No exemplo acima voce digitaria `fix(a8)`.

Esta forma de entrada de dados é adequada quando voce tem dados que não podem ser armazenados em um único vetor, por exemplo quando há dados de mais de uma variável para serem digitados.

5.4 Lendo dados de um arquivo texto

Se os dados já estão disponíveis em formato eletrônico, isto é, já foram digitados em outro programa, voce pode importar os dados para o R sem a necessidade de digitá-los novamente.

A forma mais fácil de fazer isto é usar dados em formato texto (arquivo do tipo ASCII). Por exemplo, se seus dados estão disponíveis em uma planilha eletrônica como EXCEL ou similar, voce pode na planilha escolher a opção ¡SALVAR COMO¿ e gravar os dados em um arquivo em formato texto.

No R usa-se a função `read.table` para ler os dados de um arquivo texto e armazenar no formato de *data-frame*.

Exemplo 1 Como primeiro exemplo considere importar para o R os dados deste arquivo texto. Clique no link para visualizar o arquivo. Agora copie o arquivo para sua área de trabalho (*working directory* do R). Para importar este arquivo usamos:

```
ex01 <- read.table(``gam01.txt'')
ex01
```

Exemplo 2 Como primeiro exemplo considere importar para o R os dados deste arquivo texto. Clique no link para visualizar o arquivo. Agora copie o arquivo para sua área de trabalho (*working directory* do R).

Note que este arquivo difere do anterior em um aspecto: os nomes das variáveis estão na primeira linha. Para que o R considere isto corretamente temos que informá-lo disto com o argumento `head=T`. Portanto para importar este arquivo usamos:

```
ex02 <- read.table(‘‘exemplo02.txt’’, head=T)
ex02
```

Exemplo 3 Como primeiro exemplo considere importar para o R os dados deste arquivo texto. Clique no link para visualizar o arquivo. Agora copie o arquivo para sua área de trabalho (*working directory* do R).

Note que este arquivo difere do primeiro em outros aspectos: além dos nomes das variáveis estarem na primeira linha, os campos agora não são mais separados por tabulação e sim por `:`. Alm disto os caracteres decimais estão separados por vírgula, sendo que o R usa ponto pois é um programa escrito em língua inglesa. Portanto para importar corretamente este arquivo usamos então os argumentos `sep` e `dec`:

```
ex03 <- read.table(‘‘dadosfic.csv’’, head=T, sep=’’:’’, dec=’’,’’)
ex03
```

Pra maiores informações consulte a documentação desta função com `?read.table`.

É possível ler dados diretamente de outros formatos que não seja texto (ASCII). Para mais detalhes consulte o manual *R data import/export*.

Para carregar conjuntos de dados que são já disponibilizados com o R use o comando `data()`

6 Análise descritiva

6.1 Descrição univariada

Nesta sessão vamos ver alguns (mas não todos!) comandos do R para fazer uma análise descritiva de um conjunto de dados.

Uma boa forma de iniciar uma análise descritiva adequada é verificar os tipode de variáveis disponíveis. Variáveis podem ser classificadas da seguinte forma:

- **qualitativas**
  - nominais
  - ordinais
- **quantitativas**
  - discretas
  - contínuas

e podem ser resumidas por tabelas, gráficos e/ou medidas.

Vamos ilustrar estes conceitos com um conjunto de dados já incluído no R, o conjunto `mtcars` que descreve características de diferentes modelos de automóvel.

Primeiro vamos carregar e inspecionar os dados.

```
> data(mtcars)
> mtcars         # mostra todo o conjunto de dados
> dim(mtcars)    # mostra a dimensão dos dados
> mtcars[1:5,]   # mostra as 5 primeiras linhas
> names(mtcars)  # mostra os nomes das variáveis
> help(mtcars)   # mostra documentação do conjunto de dados
```

Vamos agora, por simplicidade, selecionar um subconjunto destes dados com apenas algumas das variáveis. Para isto vamos criar um objeto chamado `mtc` que contém apenas as variáveis desejadas. Para selecioná-las indicamos os números das colunas correspondentes à estas variáveis.

```
> mtc <- mtcars[,c(1,2,4,6,9,10)]
> mtc[1:5,]
> names(mtc)
```

Vamos anexar o objeto para facilitar a digitação com o comando abaixo. O uso e sentido deste comando será explicado mais adiante.

```
> attach(mtc)
```

Vamos agora ver uma descrição da variável número de cilindros. Vamos fazer uma tabela de frequências absolutas e gráficos de barrase do tipo "torta". Depois fazemos o mesmo para frequências relativas.

```
> tcyl <- table(cyl)
> barplot(tcyl)
> pie(tcyl)

> tcyl <- 100* table(cyl)/length(cyl)
> barplot(tcyl)
> pie(tcyl)
```

Passando agora para uma variável quantitativa contínua vamos ver o comportamento da variável que mede o rendimento dos carros (em mpg – milhas por galão). Primeiro fazemos uma tabela de frequências, depois gráficos (histograma, box-plot e diagrama ramos-e-folhas) e finalmente obtemos algumas medidas que resumem os dados.

```
> table(cut(mpg, br=seq(10,35, 5)))

> hist(mpg)
> boxplot(mpg)
> stem(mpg)

> summary(mpg)
```

6.2 Descrição bivariada

Vamos primeiro ver o resumo de duas variáveis categóricas: o tipo de marcha e o número de cilindros. Os comandos abaixo mostram como obter uma tabela com o cruzamento destas variáveis e gráficos.

```
> table(am, cyl)
> plot(table(am, cyl))
> barplot(table(am, cyl), leg=T)
> barplot(table(am, cyl), beside=T, leg=T)
```

Agora vamos relacionar uma categórica (tipo de câmbio) com uma contínua (rendimento). O primeiro comando absixo mostra como obter medidas resumo do rendimento para cada tipo de câmbio. A seguir são mostrados alguns tipos de gráficos que podem ser obtidos para descrever o comportamento e associação destas variáveis.

```
> tapply(mpg, am, summary)

> plot(am, mpg)

> m0 <- mean(mpg[am==0])   # média de rendimento para cambio automático
> m0
> m1 <- mean(mpg[am==1])   # média de rendimento para cambio manual
> m1

> points(c(0,1), c(m0, m1), cex=2,col=2, pch=20)

> par(mfrow=c(1,2))
> by(hp, am, hist)
> par(mfrow=c(1,1))
```

Pode-se fazer um teste estatístico (usando o teste t) para comparar os redimentos de carros com diferentes tipos de câmbio e/ou com diferentes números de cilindros (usando a análise de variância).

```
> t.test(mpg[am==0], mpg[am==1])

> tapply(mpg, cyl, mean)
> plot(cyl,mpg)
> anova(aov(mpg ~ cyl))
```

Passamos agora para a relação entre duas contínuas (peso e rendimento) que pode ser ilustrada como se segue.

```
> plot(wt, mpg)   # gráfico de rendimento versus peso
> cor(wt, mpg)    # coeficiente de correlação linear de Pearson
```

Podemos ainda usar recusos gráficos para visualizar três variáveis ao mesmo tempo. Veja os gráficos produzidos com os comandos abaixo.

```
> points(wt[cyl==4], mpg[cyl==4], col=2, pch=19)
> points(wt[cyl==6], mpg[cyl==6], col=3, pch=19)
> points(wt[cyl==8], mpg[cyl==8], col=4, pch=19)

> plot(wt, mpg, pch=21, bg=(2:4)[codes(factor(cyl))])
> plot(wt, mpg, pch=21, bg=(2:4)[codes(factor(am))])

> plot(hp, mpg)
> plot(hp, mpg, pch=21, bg=c(2,4)[codes(factor(am))])

> par(mfrow=c(1,2))
> plot(hp[am==0], mpg[am == 0])
> plot(hp[am==1], mpg[am == 1])
> par(mfrow=c(1,1))
```

6.3 Descrevendo um outro conjunto de dados

Vamos agora utilizar um outro conjunto de dados que já vem disponível com o R – o conjunto `airquality`.

Estes dados são medidas de: concentração de ozônio (*Ozone*), radiação solar (*Solar.R*), velocidade de vento (*Wind*) e temperatura (*Temp*) coletados diariamente (*Day*) por cinco meses (*Month*).

Primeiramente vamos carregar e visualisar os dados com os comandos:

```
> data(airquality)       # carrega os dados
> airquality             # mostra os dados
```

Vamos agora usar alguns comandos para "conhecer melhor"os dados:

```
> is.data.frame(airquality)    # verifica se é um data-frame
> names(airquality)            # nome das colunas (variáveis)
> dim(airquality)              # dimensões do data-frame
> help(airquality)             # mostra o ‘‘help’’que explica os dados
```

Bem, agora que conhecemos melhor o conjunto `airquality`, sabemos o número de dados, seu formato, o número de nome das variáveis podemos começar a analisá-los.

Veja por exemplo alguns comandos:

```
> summary(airquality)             # rápido sumário das variáveis
> summary(airquality[,1:4])       # rápido sumário apenas das 4 primeiras variáveis
> mean(airquality$Temp)           # média das temperaturas no período
> mean(airquality$Ozone)          # média do Ozone no período - note a resposta NA
> airquality$Ozone                # a razão é que existem ‘‘dados perdidos’’ na variável
> mean(airquality$Ozone, na.rm=T) # média do Ozone no período - retirando valores perdido
```

Note que os útimos tres comandos são trabalhosos de serem digitados pois temos que digitar `airquality` a cada vez!
Mas há um mecanismo no R para facilitar isto: o *caminho de procura* ("search path"). Começe digitando e vendo s saída de:
`search()`

O programa vai mostrar o caminho de procura dos objetos. Ou seja, quando voce usa um nome do objeto o R vai procurar este objeto nos caminhos indicado, na ordem apresentada.

Pois bem, podemos "adicionar" um novo local neste caminho de procura e este novo local pode ser o nosso objeto `airquality`. Digite o seguinte e compare com o anterior:

```
> attach(airquality)   # anexando o objeto airquality no caminho de procura.
> search()             # mostra o caminho agora com o airquality incluído
> mean(Temp)           # e ... a digitação fica mais fácil e rápida !!!!
> mean(Ozone, na.rm=T) # pois com o airquality anexado o R acha as variáveis
```

NOTA: Para retirar o objeto do caminho de procura basta digitar `detach(airquality)`.

Bem, agora é com voce!
Reflita sobre os dados e use seus conhecimentos de estatística para fazer uma análise descritiva interessante destes dados.

Pense em questões relevantes e veja como usar medidas e gráficos para respondê-las. Use os comandos mostrados anteriormente. Por exemplo:

- as médias mensais variam entre si?
- como mostrar a evolução das variáveis no tempo?
- as variáveis estão relacionadas?
- etc, etc, etc

6.4 Descrevendo o conjunto de dados "Milsa" de Bussab & Morettin

O livro *Estatística Básica* de W. Bussab e P. Morettin traz no primeiro capítulo um conjunto de dados hipotético de atributos de 36 funcionários da companhia "Milsa". Os dados estão reproduzidos na tabela 6.4. Veja o livro para mais detalhes sobre este dados.

O que queremos aqui é ver como, no programa R:

- entrar com os dados
- fazer uma análise descritiva

Estes são dados no "estilo planilha", com variáveis de diferentes tipos: categóricas e numéricas (qualitativas e quantitativas). Portanto o formato ideal de armazanamento destes dados no R é o *data.frame*. Para entrar com estes dados no diretamente no Rpodemos usar o editor que vem com o programa. Para digitar rapidamente estes dados é mais fácil usar códigos para as variáveis categóricas. Desta forma, na coluna de estado civil vamos digitar o código 1 para *solteiro* e 2 para *casado*. Fazemos de maneira similar com as colunas *Grau de Instrução* e *Região de Procedência*. No comando a seguir invocamos o editor, entramos com os dados na janela que vai aparecer na sua tela e quanto saímos do editor (pressionando o botão `QUIT`) os dados ficam armazenados no objeto `milsa`. Após isto digitamos o nome do objeto (`milsa`) e podemos ver o conteúdo digitado, como mostra a tabela 6.4. Lembre-se que se voce precisar corrigir algo na digitação voce pode fazê-lo abrindo a planilha novamente com o comando `fix(milsa)`.

Tabela 2: Dados de Bussab & Morettin

Funcionário	Est. Civil	Instrução	N^o Filhos	Salário	Ano	Mês	Região
1	solteiro	1o Grau	-	4.00	26	3	interior
2	casado	1o Grau	1	4.56	32	10	capital
3	casado	1o Grau	2	5.25	36	5	capital
4	solteiro	2o Grau	-	5.73	20	10	outro
5	solteiro	1o Grau	-	6.26	40	7	outro
6	casado	1o Grau	0	6.66	28	0	interior
7	solteiro	1o Grau	-	6.86	41	0	interior
8	solteiro	1o Grau	-	7.39	43	4	capital
9	casado	2o Grau	1	7.59	34	10	capital
10	solteiro	2o Grau	-	7.44	23	6	outro
11	casado	2o Grau	2	8.12	33	6	interior
12	solteiro	1o Grau	-	8.46	27	11	capital
13	solteiro	2o Grau	-	8.74	37	5	outro
14	casado	1o Grau	3	8.95	44	2	outro
15	casado	2o Grau	0	9.13	30	5	interior
16	solteiro	2o Grau	-	9.35	38	8	outro
17	casado	2o Grau	1	9.77	31	7	capital
18	casado	1o Grau	2	9.80	39	7	outro
19	solteiro	Superior	-	10.53	25	8	interior
20	solteiro	2o Grau	-	10.76	37	4	interior
21	casado	2o Grau	1	11.06	30	9	outro
22	solteiro	2o Grau	-	11.59	34	2	capital
23	solteiro	1o Grau	-	12.00	41	0	outro
24	casado	Superior	0	12.79	26	1	outro
25	casado	2o Grau	2	13.23	32	5	interior
26	casado	2o Grau	2	13.60	35	0	outro
27	solteiro	1o Grau	-	13.85	46	7	outro
28	casado	2o Grau	0	14.69	29	8	interior
29	casado	2o Grau	5	14.71	40	6	interior
30	casado	2o Grau	2	15.99	35	10	capital
31	solteiro	Superior	-	16.22	31	5	outro
32	casado	2o Grau	1	16.61	36	4	interior
33	casado	Superior	3	17.26	43	7	capital
34	solteiro	Superior	-	18.75	33	7	capital
35	casado	2o Grau	2	19.40	48	11	capital
36	casado	Superior	3	23.30	42	2	interior

Tabela 3: Dados digitados usando códigos para variáveis

	civil	instrucao	filhos	salario	ano	mes	regiao
1	1	1	NA	4.00	26	3	1
2	2	1	1	4.56	32	10	2
3	2	1	2	5.25	36	5	2
4	1	2	NA	5.73	20	10	3
5	1	1	NA	6.26	40	7	3
6	2	1	0	6.66	28	0	1
7	1	1	NA	6.86	41	0	1
8	1	1	NA	7.39	43	4	2
9	2	2	1	7.59	34	10	2
10	1	2	NA	7.44	23	6	3
11	2	2	2	8.12	33	6	1
12	1	1	NA	8.46	27	11	2
13	1	2	NA	8.74	37	5	3
14	2	1	3	8.95	44	2	3
15	2	2	0	9.13	30	5	1
16	1	2	NA	9.35	38	8	3
17	2	2	1	9.77	31	7	2
18	2	1	2	9.80	39	7	3
19	1	3	NA	10.53	25	8	1
20	1	2	NA	10.76	37	4	1
21	2	2	1	11.06	30	9	3
22	1	2	NA	11.59	34	2	2
23	1	1	NA	12.00	41	0	3
24	2	3	0	12.79	26	1	3
25	2	2	2	13.23	32	5	1
26	2	2	2	13.60	35	0	3
27	1	1	NA	13.85	46	7	3
28	2	2	0	14.69	29	8	1
29	2	2	5	14.71	40	6	1
30	2	2	2	15.99	35	10	2
31	1	3	NA	16.22	31	5	3
32	2	2	1	16.61	36	4	1
33	2	3	3	17.26	43	7	2
34	1	3	NA	18.75	33	7	2
35	2	2	2	19.40	48	11	2
36	2	3	3	23.30	42	2	1

```
> milsa <- edit(data.frame())   # abra a planilha para entrada dos dados
> milsa                         # visualiza os dados digitados
> fix(milsa)                    # comando a ser usado para correções, se necessário
```

Note que além de digitar os dados na planilha digitamos também o nome que escolhemos para cada variável. A planilha digitada como está ainda não está pronta. Precisamos informar para o programa que as variáveis `civil`, `instrucao` e `regiao`, NÃO são numéricas e sim categóricas. No R variáveis categóricas são definidas usando o comando `factor()`, que vamos

usar para redefinir nossas variáveis conforme os comandos a seguir. Primeiro redefinimos a variável `civil` com os *rótulos (labels)* solteiro e casado associados aos *níveis (levels)* 1 e 2. Para variável `intrução` usamos o argumento adicional `ordered = TRUE` para indicar que é uma variável ordinal. Na variável `regiao` codificamos assim: 2=capital, 1=interior, 3=outro. Ao final inspecionamos os dados digitando o nome do objeto.

```
milsa$civil <- factor(milsa$civil, label=c("solteiro", "casado"), levels=1:2)
milsa$instrucao <- factor(milsa$instrucao, label=c("1oGrau", "2oGrau", "Superior"), lev=1
milsa$regiao <- factor(milsa$regiao, label=c("capital", "interior", "outro"), lev=c(2,1,3
milsa
```

Agora que os dados estão prontos podemos começar a análise descritiva. Inspecionem os comandos a seguir. Sugerimos que o leitor use o R para reproduzir os resultados mostrados no texto dos capítulos 1 a 3 do livro de Bussab & Morettin relacionados com este exemplo.

```
is.data.frame(milsa) # conferindo se é um data-frame
names(milsa)         # vendo o nome das variáveis
dim(milsa)           # vendo as dimensões do data-frame

attach(milsa)        # anexando ao caminho de procura

##
## Análise Univariada
##

## 1. Variável Qualitativa Nominal
civil
is.factor(civil)
## 1.1 Tabela:
civil.tb <- table(civil)
civil.tb
## ou em porcentagem
100 * table(civil)/length(civil)

## 1.2 Gráfico
## Para máquinas com baixa resolução gráfica (Sala A - LABEST)
## use o próximo comando
X11(colortype="pseudo.cube")
pie(table(civil))

## 1.3 Medidas
## encontrando a moda
civil.mo <- names(civil.tb)[civil.tb == max(civil.tb)]
civil.mo

## 2 Qualitativa Ordinal
instrucao
is.factor(instrucao)
detach(milsa)
milsa$instrucao <- factor(milsa$instrucao)
attach(milsa)
```

```
is.factor(instrucao)

## 2.1 Tabela:
instrucao.tb <- table(instrucao)
instrucao.tb
## 2.2 Gráfico:
barplot(instrucao.tb)
## 2.3 Medidas
instrucao.mo <- names(instrucao.tb)[instrucao.tb == max(instrucao.tb)]
instrucao.mo

median(as.numeric(instrucao)) # só calcula mediana de variáveis numéricas
levels(milsa$instrucao)[median(as.numeric(milsa$instrucao))]

## 3 Quantitativa discreta
filhos

## 3.1 Tabela:
filhos.tb <- table(filhos)
filhos.tb
filhos.tb/sum(filhos.tb) # frequências relativas

## 3.2 Gráfico:
plot(filhos.tb)            # gráfico das frequências absolutas
filhos.fac <- cumsum(filhos.tb)
filhos.fac                 # frequências acumuladas
plot(filhos.fac, type="s") # gráfico das frequências acumuladas

## 3.3 Medidas
## De posição
filhos.mo <- names(filhos.tb)[filhos.tb == max(filhos.tb)]
filhos.mo                  # moda

filhos.md <- median(filhos, na.rm=T)
filhos.md                  # mediana

filhos.me <- mean(filhos, na.rm=T)
filhos.me                  # média

## Medida de dispersão
range(filhos, na.rm=T)
diff(range(filhos, na.rm=T)) # amplitude

filhos.dp <- sd(filhos, na.rm=T)   # desvio padrão
filhos.dp
var(filhos, na.rm=T)       # variância

100 * filhos.dp/filhos.me  # coeficiente de variação

filhos.qt <- quantile(filhos, na.rm=T)
```

```
filhos.qt[4] - filhos.qt[2] # amplitude interquartílica

summary(filhos)            # várias medidas

## 4. Quantitativa Contínua
salario
## 4.1 Tabela
range(salario)          # máximo e mínimo
nclass.Sturges(salario) # número de classes pelo critério de Sturges
args(cut)
args(cut.default)
table(cut(salario, seq(3.5,23.5,l=8)))

## 4.2 Gráfico
hist(salario)
hist(salario, br=seq(3.5,23.5,l=8))
boxplot(salario)
stem(salario)
## 4.3 Medidas
## De posição
salario.md <- median(salario, na.rm=T)
salario.md                # mediana

salario.me <- mean(salario, na.rm=T)
salario.me                # média

## Medida de dispersão
range(salario, na.rm=T)
diff(range(salario, na.rm=T)) # amplitude

salario.dp <- sd(salario, na.rm=T)   # desvio padrão
salario.dp
var(salario, na.rm=T)     # variância

100 * salario.dp/salario.me  # coeficiente de variação

salario.qt <- quantile(salario, na.rm=T)
salario.qt[4] - salario.qt[2] # amplitude interquartílica

summary(salario)          # várias medidas

##
## Análise Bivariada
##
## 1. Qualitativa vs Qualitativa
## Ex. estado civil e grau de instrução

## 1.1 Tabela
civ.gi.tb <- table(civil, instrucao) # frequências absolutas
civ.gi.tb
```

```
civ.gi.tb/as.vector(table(civil))    # frequências por linha

## 1.2 Gráfico
plot(civ.gi.tb)
barplot(civ.gi.tb)
barplot(t(civ.gi.tb))

## 1.3. Medida de associação
summary(civ.gi.tb)   # resumo incluindo o teste Chi-quadrado
## criando uma nova variável para agrupar 2o Grau e Superior
instrucao1 <- ifelse(instrucao == 1, 1, 2)
table(instrucao)
table(instrucao1)
table(civil, instrucao1)
summary(table(civil, instrucao1))

## 2. Qualitativa vs Quantitativa
## Ex. grau de instrução vs salário

## 2.1 Tabela
quantile(salario)
ins.sal.tb <- table(instrucao, cut(salario, quantile(salario)))
ins.sal.tb

## 2.2 Gráfico
plot(instrucao, salario)
plot(salario, instrucao)

## 2.3 Medidas
## calculando as média para cada grau de instrução
tapply(salario, instrucao, mean)
## e as variâncias
tapply(salario, instrucao, var)
## e ainda os mínimo, máximo e quartis
tapply(salario, instrucao, quantile)

## 3. Quantitativa vs Quantitativa
## Ex. salário e idade

## 3.1 Tabela
table(cut(idade, quantile(idade)), cut(salario, quantile(salario)))
table(cut(idade, quantile(idade, seq(0,1,len=4))), cut(salario, quantile(salario, seq(0,1

## 3.2 Gráfico
plot(idade, salario)

## 3.3 Medidas
cor(idade, salario)

detach(milsa)             # desanexando do caminha de procura
```

6.5 Uma demonstração de recursos gráficos do R

O R vem com algumas demonstrações (*demos*) de seus recursos "embutidas" no programa. Para listar as demos disponíveis digite na linha de comando:

```
demo()
```

Para rodar uma delas basta colocar o nome da escolhida entre os parênteses. As *demos* são úties para termos uma idéia dos recursos disponíveis no programa e para ver os comandos que devem ser utilizados.

Por exemplo, vamos rodar a *demo* de recursos gráficos. Note que os comandos vão aparecer na janela de comandos e os gráficos serão automaticamente produzidos na janela gráfica. A cada passo voce vai ter que teclar `ENTER` para ver o próximo gráfico.

- no "prompt" do programa R digite:

  ```
  demo(graphics)
  ```

- Voce vai ver a seguinte mensagem na tela:

  ```
  demo(graphics)
  ---- ~~~~~~~~

  Type  <Return>   to start :
  ```

- pressione a tecla ENTER
- a "demo" vai ser iniciada e uma tela gráfica irá se abrir. Na tela de comandos serão mostrados comandos que serão utilizados para gerar um gráfico seguidos da mensagem:

  ```
  Hit <Return> to see next plot:
  ```

- inspecione os comandos e depois pressione novamente a tecla ENTER.
  Agora voce pode visualizar na janela gráfica o gráfico produzido pelos comandos mostrados anteriormente. Inspecione o gráfico cuidadosamente verificando os recursos utilizados (título, legendas dos eixos, tipos de pontos, cores dos pontos, linhas, cores de fundo, etc).
- agora na tela de comandos apareceram novos comandos para produzir um novo gráfico e a mensagem:

  ```
  Hit <Return> to see next plot:
  ```

- inspecione os novos comandos e depois pressione novamente a tecla ENTER.
  Um novo gráfico surgirá ilustrando outros recursos do programa.
  Prossiga inspecionando os gráficos e comandos e pressionando ENTER até terminar a "demo".
  Experimente outras demos como `demo(pers)` e `demo(image)`, por exemplo.

6.6 Outros dados disponíveis no R

Assim como o conjunto `mtcars` usado acima, há vários conjuntos de dados incluídos no programa R. Estes conjuntos são todos documentados, isto é, voce pode usar a função `help` para obter uma descrição dos dados. Para ver a lista de conjuntos de dados disponíveis digite `data()`. Por exemplo tente os seguintes comandos:

```
> data()
> data(women)  # carrega o conjunto de dados women
> women        # mostra os dados
> help(woman)  # mostra a documentação destes dados
```

6.7 Mais detalhes sobre o uso de funções

As funções do R são documentadas e o uso é explicado e ilustrado usando a função `help`. Por exemplo, o comando `help(mean)` vai exibir e documentação da função `mean`. Note que no final da documentação há exemplos de uso da função que voce pode reproduzir para entendê-la melhor.

6.8 Exercícios

1. Experimente as funções `mean`, `var`, `sd`, `median`, `quantile` nos dados mostrados anteriormente. Veja a documentação das funções e as opções de uso.
2. Faça uma análise descritiva adequada do conjunto de dados `women`.
3. Carregue o conjunto de dados `USArrests` com o comando `data(USArrests)`. Examine a sua documentação com `help(USArrests)` e responda as perguntas a seguir.
   (a) qual o número médio e mediano de cada um dos crimes?
   (b) qual a média do total de crimes?
   (c) encontre a mediana e quartis para cada crime.
   (d) encontre o número máximo e mínimo para cada crime.
   (e) faça um gráfico adequado para o número de assassinatos (*murder*).
   (f) faça um diagrama ramo-e-folhas para o número de estupros (*rape*).
   (g) verifique se há correlação entre os diferentes tipos de crime.
   (h) verifique se há correlação entre os crimes e o tamanho da população.
   (i) encontre os estados com maior e menor ocorrência de cada tipo de crime.
   (j) encontre os estados com maior e menor ocorrência per capta de cada tipo de crime.
   (k) encontre os estados com maior e menor ocorrência do total de crimes.

]

7 Distribuições de Probabilidade

O programa R inclui funcionalidade para operações com distribuições de probabilidades. Para cada distribuição há 4 operações básicas indicadas pelas letras:

`d` calcula a densidade de probabilidade $f(x)$ no ponto

`p` calcula a função de probabilidade acumulada $F(x)$ no ponto

`q` calcula o quantil correspondente a uma dada probabilidade

`r` retira uma amostra da distribuição

Para usar os funções deve-se combinar uma das letras acima com uma abreviatura do nome da distribuição, por exemplo para calcular probabilidades usamos: `pnorm` para normal, `pexp` para exponencial, `pbinom` para binomial, `ppois` para Poisson e assim por diante.

Vamos ver com mais detalhes algumas distribuições de probabilidades.

7.1 Distribuição Normal

A funcionalidade para distribuição normal é implementada por argumentos que combinam as letras acima com o termo `norm`. Vamos ver alguns exemplos com a distribuição normal padrão. Por *default* as funções assumem a distribuição normal padrão ($(\mu = 0, \sigma^2 = 1)$).

```
> dnorm(-1)
[1] 0.2419707

> pnorm(-1)
[1] 0.1586553

> qnorm(0.975)
[1] 1.959964

> rnorm(10)
 [1] -0.0442493 -0.3604689  0.2608995 -0.8503701 -0.1255832  0.4337861
 [7] -1.0240673 -1.3205288  2.0273882 -1.7574165
```

O primeiro valor acima corresponde ao valor da densidade da normal

$$f(x) = \frac{1}{2\pi\sigma^2} \exp\{\frac{1}{2\sigma^2}(x-\mu)^2\}$$

com parâmetros $(\mu = 0, \sigma^2 = 1)$ no ponto -1. Portanto, o mesmo valor seria obtido substituindo x por -1 na expressão da normal padrão:

```
> (1/sqrt(2*pi)) * exp((-1/2)*(-1)^2)
[1] 0.2419707
```

A função `pnorm(-1)` calcula a probabilidade $P(X \leq -1)$.
O comando `qnorm(0.975)` calcula o valor de a tal que $P(X \leq a) = 0.975$.
Finalmente o comando `rnorm(10)` gera uma amostra de 10 elementos da normal padrão. Note que os valores que voce obtém rodando este comando podem ser diferentes dos mostrados acima.

As funções acima possuem argumentos adicionais, para os quais valores padrão (*default*) foram assumidos, e que podem ser modificados. Usamos `args` para ver os argumentos de uma função e `help` para visualisar a documentação detalhada:

```
> args(rnorm)
function (n, mean = 0, sd = 1)
```

As funções relacionadas à distribuição normal possuem os argumentos `mean` e `sd` para definir média e desvio padrão da distribuição que podem ser modificados como nos exemplos a seguir. Note nestes exemplos que os argumentos podem ser passados de diferentes formas.

```
> qnorm(0.975, mean = 100, sd = 8)
[1] 115.6797

> qnorm(0.975, m = 100, s = 8)
[1] 115.6797

> qnorm(0.975, 100, 8)
[1] 115.6797
```

Para informações mais detalhadas pode-se usar a função `help`. O comando

```
> help(rnorm)
```

irá exibir em uma janela a documentação da função que pode também ser chamada com `?rnorm`. Note que ao final da documentação são apresentados exemplos que podem ser rodados pelo usuário e que auxiliam na compreensão da funcionalidade.
Note também que as 4 funções relacionadas à distribuição normal são documentadas conjuntamente, portanto `help(rnorm)`, `help(qnorm)`, `help(dnorm)` e `help(pnorm)` irão exibir a mesma documentação.

Cálculos de probabilidades usuais, para os quais utilizávamos tabelas estatísticas podem ser facilmente obtidos como no exemplo a seguir.

Seja X uma v.a. com distribuição $N(100, 100)$. Calcular as probabilidades:

1. $P[X < 95]$
2. $P[90 < X < 110]$
3. $P[X > 95]$

Calcule estas probabilidades de forma usual, usando a tabela da normal. Depois compare com os resultados fornecidos pelo R. Os comandos do R para obter as probabilidades pedidas são:

```
> pnorm(95, 100, 10)
[1] 0.3085375

> pnorm(110, 100, 10) - pnorm(90, 100, 10)
[1] 0.6826895

> 1 - pnorm(95, 100, 10)
[1] 0.6914625
> pnorm(95, 100, 10, lower=F)
[1] 0.6914625
```

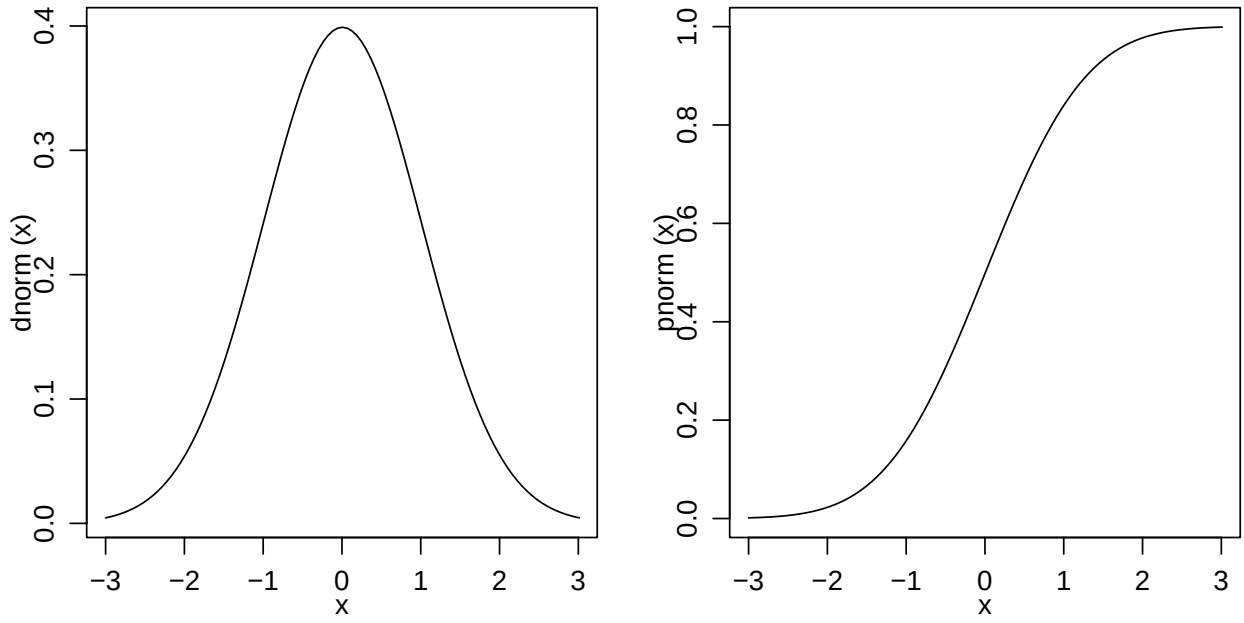

Figura 1: Funções de densidade e probabilidade da distribuição normal padrão.

Note que a última probabilidade foi calculada de duas formas diferentes, a segunda usando o argumento `lower` mais estável numericamente.

A seguir vamos ver comandos para fazer gráficos de distribuições de probabilidade. Vamos fazer gráficos de funções de densidade e de probabilidade acumulada. Estude cuidadosamente os comandos abaixo e verifique os gráficos por eles produzidos. A Figura 1 mostra gráficos da densidade (esquerda) e probabilidade acumulada (direita) da normal padrão, produzidos com os comandos a seguir. Para fazer o gráfico consideramos valores de X entre -3 e 3 que correspondem a +/- três desvios padrões da média, faixa que concentra mais de 99% da massa de probabilidade da distribuição normal.

```
> plot(dnorm, -3, 3)
> plot(pnorm, -3, 3)
```

A Figura 2 mostra gráficos da densidade (esquerda) e probabilidade acumulada (direita) da $N(100, 64)$. Para fazer estes gráficos tomamos uma sequência de valores de x e para cada um deles calculamos o valor da função $f(x)$ e depois unimos os pontos $(x, f(x))$ em um gráfico.

```
> x <- seq(70, 130, len=100)
> fx <- dnorm(x, 100, 8)
> plot(x, fx, type='l')
```

Note que, alternativamente, os mesmos gráficos poderiam ser produzidos com os comandos a seguir.

```
> plot(function(x) dnorm(x, 100, 8), 70, 130)
> plot(function(x) pnorm(x, 100, 8), 70, 130)
```

Comandos usuais de do R podem ser usados para modificar a aparência dos gráficos. Por exemplo, podemos incluir títulos e mudar texto dos eixos conforme mostrado na gráfico da esquerda da Figura 3 e nos dois primeiros comandos abaixo. Os demais comandos mostram como colocar diferentes densidades em um um mesmo gráfico como ilustrado à direita da mesma Figura.

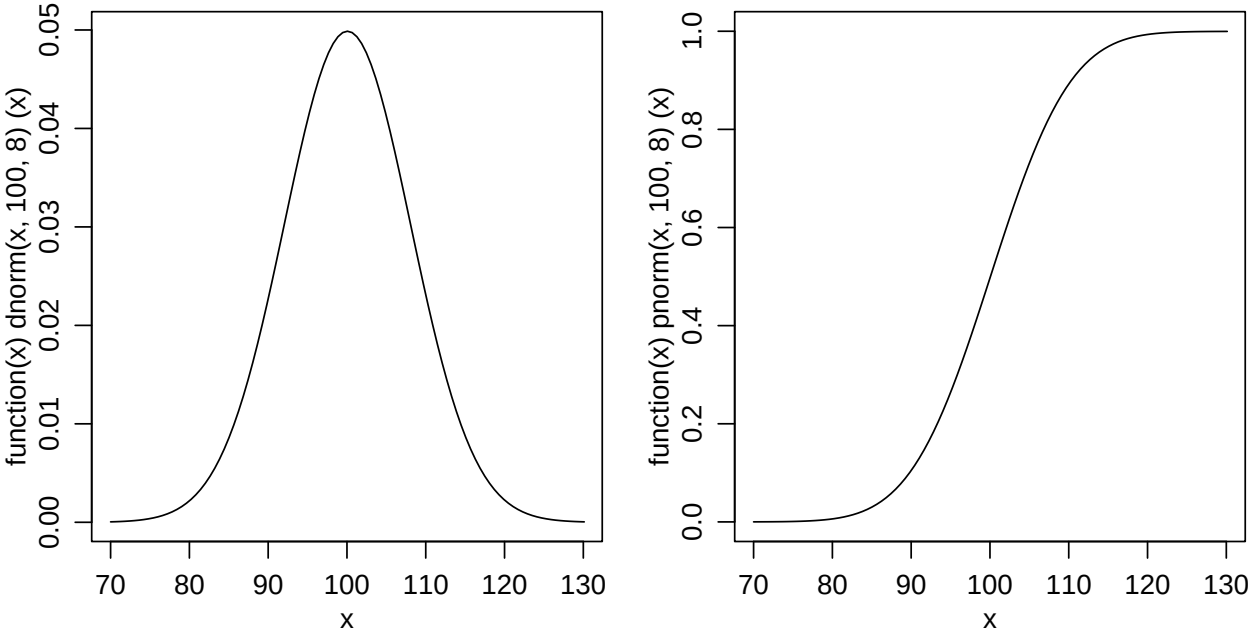

Figura 2: Funções de densidade e probabilidade da $N(100, 64)$.

```
> plot(dnorm, -3, 3, xlab='valores de X', ylab='densidade de probabilidade')
> title('Distribuicão Normal\nX ~ N(100, 64)')

> plot(function(x) dnorm(x, 100, 8), 60, 140, ylab='f(x)')
> plot(function(x) dnorm(x, 90, 8), 60, 140, add=T, col=2)
> plot(function(x) dnorm(x, 100, 15), 60, 140, add=T, col=3)
> legend(120, 0.05, c("N(100,64)","N(90,64)","N(100,225)"), fill=1:3)
```

7.2 Distribuição Binomial

Cálculos para a distribuição binomial são implementados combinando as *letras básicas* vistas acima com o termo `binom`. Vamos primeiro investigar argumentos e documentação com os comandos `args` e `binom`.

```
> args(dbinom)
function (x, size, prob, log = FALSE)

> help(dbinom)
```

Seja X uma v.a. com distribuição Binomial com $n = 10$ e $p = 0.35$. Vamos ver os comandos do R para:

1. fazer o gráfico das função de densidade
2. idem para a função de probabilidade
3. calcular $P[X = 7]$
4. calcular $P[X < 8] = P[X \leq 7]$
5. calcular $P[X \geq 8] = P[X > 7]$

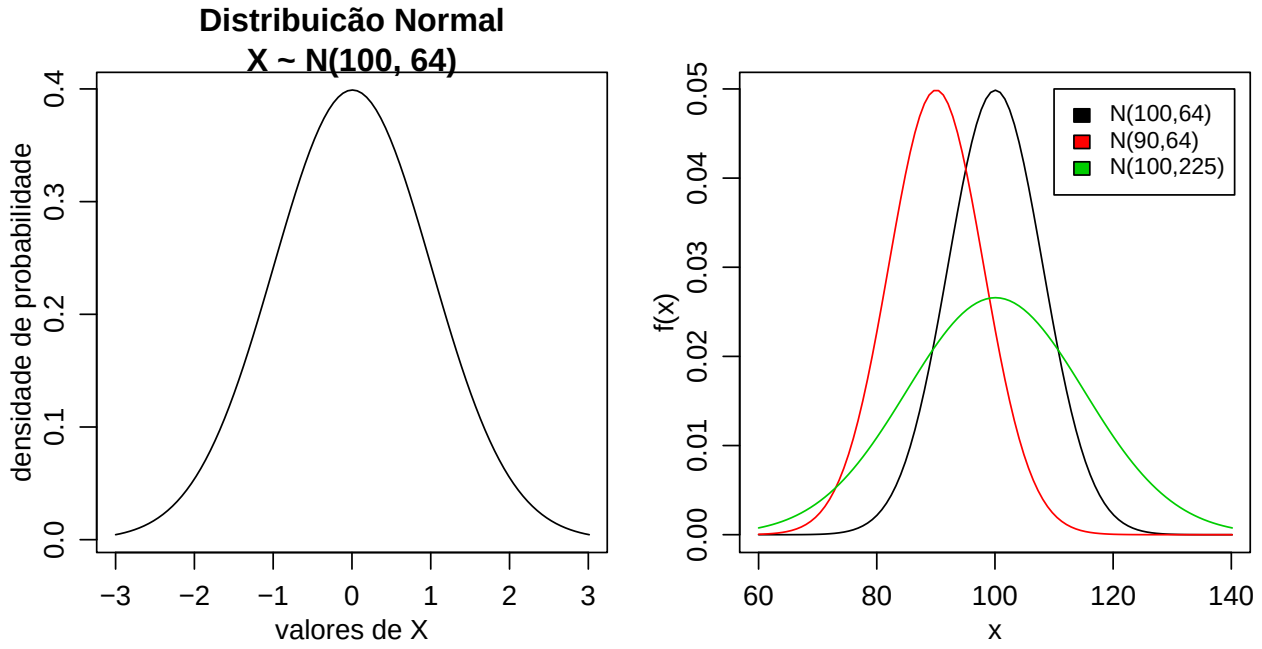

Figura 3: Gráfico com texto nos eixos e título (esquerda) e várias distribuições em um mesmo gráfico (direita).

6. calcular $P[3 < X \leq 6] = P[4 \geq X < 7]$

Note que sendo uma distribuição discreta de probabilidades os gráficos são diferentes dos obtidos para distribuição normal e os cálculos de probabilidades devem considerar as probabilidades nos pontos. Os gráficos das funções de densidade e probabilidade são mostrados na Figura 4.

```
> x <- 0:10

> fx <- dbinom(x, 10, 0.35)
> plot(x, fx, type='h')

> Fx <- pbinom(x, 10, 0.35)
> plot(x, Fx, type='S')

> dbinom(7, 10, 0.35)
[1] 0.02120302

> pbinom(7, 10, 0.35)
[1] 0.9951787
> sum(dbinom(0:7, 10, 0.35))
[1] 0.9951787

> 1-pbinom(7, 10, 0.35)
[1] 0.004821265
> pbinom(7, 10, 0.35, lower=F)
[1] 0.004821265

> pbinom(6, 10, 0.35) - pbinom(3, 10, 0.35)
```

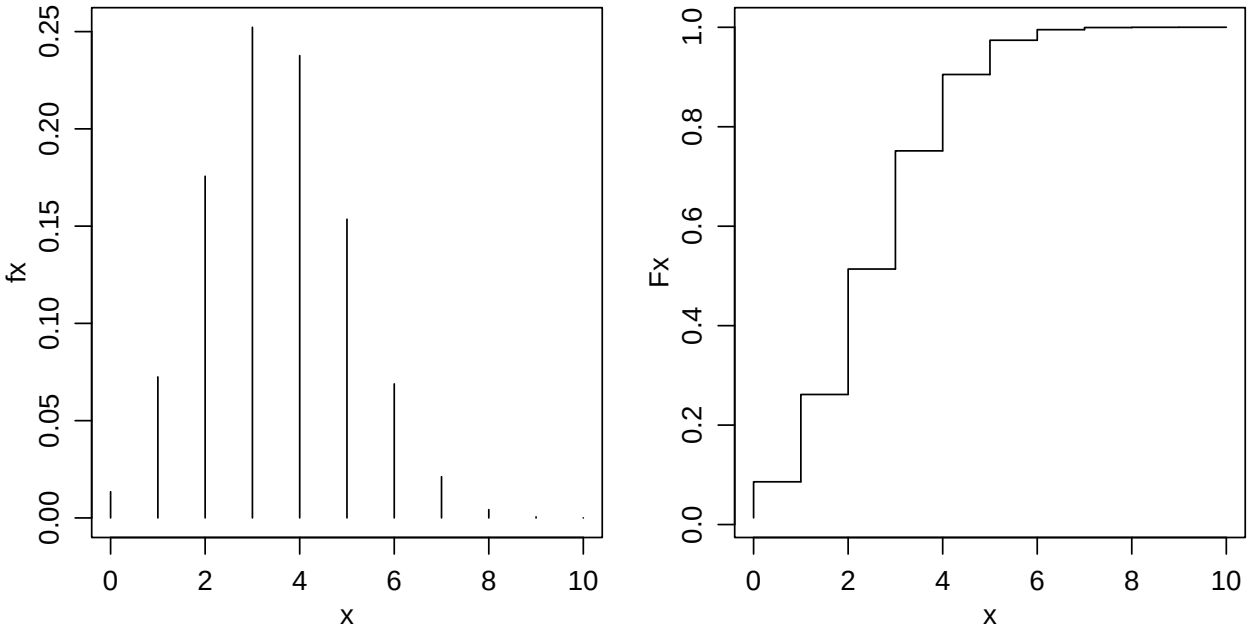

Figura 4: Funções de densidade (esquerda)e probabilidade (direita) da $B(10, 0.35)$.

```
[1] 0.4601487
> sum(dbinom(4:6, 10, 0.35))
[1] 0.4601487
```

7.3 Exercícios

Nos exercícios abaixo iremos também usar o R como uma calculadora estatística para resolver alguns exemplos/exercícios de probabilidade tipicamente apresentados em um curso de estatística básica.

Os exercícios abaixo com indicação de página foram retirados de:

Magalhães, M.N. & Lima, A.C.P. (2001) **Noções de Probabilidade e Estatística**. 3 ed. São Paulo, IME-USP. 392p.

1. (Ex 1, pag 67) Uma moeda viciada tem probabilidade de cara igual a 0.4. Para quatro lançamentos independentes dessa moeda, estude o comportamento da variável *número de caras* e faça um gráfico de sua função de distribuição.

2. (Ex 5, pag 77) Sendo X uma variável seguindo o modelo Binomial com parâmetro $n = 15$ e $p = 0.4$, pergunta-se:

   - $P(X \geq 14)$
   - $P(8 < X \leq 10)$
   - $P(X < 2 \text{ ou } X \geq 11)$
   - $P(X \geq 11 \text{ ou } X > 13))$
   - $P(X > 3 \text{ e } X < 6)$
   - $P(X \leq 13 \mid X \geq 11)$

3. (Ex 8, pag 193) Para $X \sim N(90, 100)$, obtenha:

   - $P(X \leq 115)$
   - $P(X \geq 80)$
   - $P(X \leq 75)$
   - $P(85 \leq X \leq 110)$
   - $P(|X - 90| \leq 10)$
   - P valor de a tal que $P(90 - a \leq X \leq 90 + a) = \gamma,\ \gamma = 0.95$

4. Faça os seguintes gráficos:

   - da função de densidade de uma variável com distribuição de Poisson com parâmetro $\lambda = 5$
   - da densidade de uma variável $X \sim N(90, 100)$
   - sobreponha ao gráfico anterior a densidade de uma variável $Y \sim N(90, 80)$ e outra $Z \sim N(85, 100)$
   - densidades de distribuições χ^2 com 1, 2 e 5 graus de liberdade.

5. A probabilidade de indivíduos nascerem com certa característica é de 0,3. Para o nascimento de 5 indivíduos e considerando os nascimentos como eventos independentes, estude o comportamento da variável *número de indivíduos com a característica* e faça um gráfico de sua função de distribuição.

6. Sendo X uma variável seguindo o modelo Normal com média $\mu = 130$ e variância $\sigma^2 = 64$, pergunta-se: (a) $P(X \geq 120)$ (b) $P(135 < X \leq 145)$ (c) $P(X < 120 \text{ ou } X \geq 150)$

7. (Ex 3.6, pag 65) Num estudo sobre a incidência de câncer foi registrado, para cada paciente com este diagnóstico o número de casos de câncer em parentes próximos (pais, irmãos, tios, filhos e sobrinhos). Os dados de 26 pacientes são os seguintes:

Paciente	1	2	3	4	5	6	7	8	9	10	11	12	13
Incidência	2	5	0	2	1	5	3	3	3	2	0	1	1
Paciente	14	15	16	17	18	19	20	21	22	23	24	25	26
Incidência	4	5	2	2	3	2	1	5	4	0	0	3	3

   Estudos anteriores assumem que a incidência de câncer em parentes próximos pode ser modelada pela seguinte função discreta de probabilidades:

Incidência	0	1	2	3	4	5
p_i	0.1	0.1	0.3	0.3	0.1	0.1

   - os dados observados concordam com o modelo teórico?
   - faça um gráfico mostrando as frequências teóricas (esperadas) e observadas.

8 Intervalos de confiança e testes de hipótese

Nesta sessão vamos verificar como utilizar o R para obter intervalos de confiança e testar hipóteses sobre parâmetros de interesse.

8.1 Média de uma distribuição normal com variância desconhecida

Considere resolver o seguinte problema:

Exemplo

O tempo de reação de um novo medicamento pode ser considerado como tendo distribuição Normal e deseja-se fazer inferência sobre a média que é desconhecida obtendo um intervalo de confiança. Vinte pacientes foram sorteados e tiveram seu tempo de reação anotado. Os dados foram os seguintes (em minutos):

2.9	3.4	3.5	4.1	4.6	4.7	4.5	3.8	5.3	4.9
4.8	5.7	5.8	5.0	3.4	5.9	6.3	4.6	5.5	6.2

Neste primeiro exemplo, para fins didáticos vamos mostrar duas possíveis soluções:

1. fazendo as contas passo a passo, utilizando o R como uma calculadora
2. usando uma função já existente no R

Entramos com os dados com o comando

```
> tempo <- c(2.9, 3.4, 3.5, 4.1, 4.6, 4.7, 4.5, 3.8, 5.3, 4.9,
             4.8, 5.7, 5.8, 5.0, 3.4, 5.9, 6.3, 4.6, 5.5, 6.2)
```

Sabemos que o intervalo de confiança para média de uma distribuição normal com média desconhecida é dado por:

$$\left(\bar{x} - t_{\alpha/2}\sqrt{\frac{S^2}{n}} \quad , \quad \bar{x} + t_{1-\alpha/2}\sqrt{\frac{S^2}{n}} \right)$$

Vamos agora obter a resposta de duas formas diferentes.

8.1.1 Fazendo as contas passo a passo

Nos comandos a seguir calculamos o tamanho da amostra, a média e a variância amostral.

```
> n <- length(tempo)
> n
[1] 20
> t.m <- mean(tempo)
> t.m
[1] 4.745
> t.v <- var(tempo)
> t.v
[1] 0.992079
```

A seguir montamos o intervalo utilizando os quantis da dritribuição t.

```
> t.ic <- t.m + qt(c(0.025, 0.975), df = n-1) * sqrt(t.v/length(tempo))
> t.ic
[1] 4.278843 5.211157
```

8.1.2 Usando a função `t.test`

Mostramos a solução acima para ilustrar a flexibilidade e o uso do programa. Entretanto não precisamos fazer isto na maioria das vezes porque o R já vem com várias funções para procedimentos estatísticos já escritas.

Para este exemplo específico

a função `t.test` pode ser utilizada como vemos no resultado do comando a sequir que coincide com os obtidos anteriormente.

```
> t.test(tempo)

        One Sample t-test

data:  tempo
t = 21.3048, df = 19, p-value = 1.006e-14
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 4.278843 5.211157
sample estimates:
mean of x
    4.745
```

O resultado da função mostra a estimativa obtida da média (4.745), o intervalo de confiança a 95e testa a igualdade de média a zero ($p - value = 1.006e - 14$), em um teste bilateral.

Os valores definidos no IC e teste de Hipótese acima são "defaults" que podem ser modificados. Por exemplo, para obter um IC a 99

```
> t.test(tempo, alt = "greater", mu = 3, conf.level = 0.99)

        One Sample t-test

data:  tempo
t = 7.835, df = 19, p-value = 1.140e-07
alternative hypothesis: true mean is greater than 3
99 percent confidence interval:
 4.179408      Inf
sample estimates:
mean of x
    4.745
```

8.2 Teste χ^2 de independência

Quando estudamos a relação entre duas variáveis qualitativas em geral fazemos uma tabela com o resultado do cruzamento desta variáveis. Em geral existe interesse em verificar se as variáveis estão associadas e para isto calcula-se uma medida de associação tal como o χ^2, coeficiente de contingência C, ou similar. O passo seguinte é testar se existe evidência que a associação é significativa. Uma possível forma de fazer isto é utilizando o teste χ^2.

Para ilustrar o teste vamos utilizar o conjunto de dados `HairEyeColor` que já vem disponível com o R. Para carregar e visualizar os dados use os comando abaixo.

```
> data(HairEyeColor)
> HairEyeColor
> as.data.frame(HairEyeColor)
```

Para saber mais sobre estes dados veja `help(HairEyeColor)` Note que estes dados já vem "resumidos" na forma de uma tabela de frequências tri-dimensional, com cada uma das dimensões .correspondendo a um dos atributos - cor dos cabelos, olhos e sexo.

Para ilustrar aqui o teste χ^2 vamos verificar se existe associação entre 2 atributos: cor dos olhos e cabelos entre os indivíduos do sexo feminino. Vamnos adotar $\alpha = 5\%$ como nível de significância. Nos comandos abaixo primeiro isolamos apenas a tabela com os indivíduos do sexo masculino e depois aplicamos o teste sobre esta tabela.

```
> HairEyeColor[,,1]
       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    38   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8
> chisq.test(HairEyeColor[,,1])

        Pearson's Chi-squared test

data:  HairEyeColor[, , 1]
X-squared = 42.1633, df = 9, p-value = 3.068e-06

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(HairEyeColor[, , 1])
```

O $p-value$ sugere que a associação é significativa. Entretanto este resultado deve ser visto com cautela pois a mensagem de alerta (*Warning message*) emitida pelo programa chama atenção ao fato de que há várias caselas com baixa frequência na tabela e portanto as condições para a validade do teste não são perfeitamente satisfeitas.

Uma possibilidade neste caso é então usar o $p-value$ calculado por simulação, ao invés do resultado assintótico usado no teste tradicional.

```
> chisq.test(HairEyeColor[,,1], sim=T)

        Pearson's Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  HairEyeColor[, , 1]
X-squared = 42.1633, df = NA, p-value = 0.0004998
```

Note que agora a mensagem de alerta não é mais emitida e que a significância foi confirmada (P-valor ¡ 0.05). Note que se voce rodar este exemplo poderá obter um $p-value$ um pouco diferente porque as simulações não necessariamente serão as mesmas.

Lembre-se de inspecionar `help(chisq.test)` para mais detalhes sobre a implementação deste teste no R.

8.3 Teste para o coeficiente de correlação linear de Pearson

Quando temos duas variáveis quantitativas podemos utilizar o coeficiente de correlação linear para medir a associação entre as variáveis, se a relação entre elas for linear. Para ilustrar o teste para o coeficiente linear de Pearson vamos estudar a relação entre o peso e rendimento

de carros. Para isto vamos usar as variáveis `wt` (peso) e `mpg` (milhas por galão) do conjunto de dados `mtcars`.

```
> data(mtcars)
> attach(mtcars)
> cor(wt, mpg)
[1] -0.8676594
> cor.test(wt, mpg)

        Pearson's product-moment correlation

data:  wt and mpg
t = -9.559, df = 30, p-value = 1.294e-10
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9338264 -0.7440872
sample estimates:
       cor
-0.8676594

> detach(mtcars)
```

Portanto o p-valor acima mmostra que a correlação encontrada de -0.87 difere significativamente de zero. Note que uma análise mais cuidadosa deveria incluir o exame do gráfico entre estas duas variáveis para ver se o coeficiente de correlação linear é adequado para medir a associação.

8.4 Comparação de duas médias

Quando temos uma variável qualitativa com dois níveis e outra quantitativa a análise em geral recai em comparar as médias da quantitativa para cada grupo da qualitativa. Para isto podemos utilizar o *testeT*. Há diferentes tipos de teste T: para amostras independentes ou pareadas, variâncias iguais ou desiguais.

Considere o seguinte exemplo:

Os dados a seguir correpondem a teores de um elemento indicador da qualidade de um certo produto vegetal. Foram coletadas 2 amostras referentes a 2 métodos de produção e deseja-se comparar as médias dos métodos fazendo-se um teste t bilateral, ao nível de 5% de significância e considerando-se as variâncias iguais.

Método 1	0.9	2.5	9.2	3.2	3.7	1.3	1.2	2.4	3.6	8.3
Método 2	5.3	6.3	5.5	3.6	4.1	2.7	2.0	1.5	5.1	3.5

```
> m1 <- c(0.9, 2.5, 9.2, 3.2, 3.7, 1.3, 1.2, 2.4, 3.6, 8.3)
> m2 <- c(5.3, 6.3, 5.5, 3.6, 4.1, 2.7, 2.0, 1.5, 5.1, 3.5)

 t.test(m1,m2, var.eq=T)

        Two Sample t-test
```

```
data:  m1 and m2
t = -0.3172, df = 18, p-value = 0.7547
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.515419  1.855419
sample estimates:
mean of x mean of y
     3.63      3.96
```

Os resultados mostram que não á evidências para rejeitar a hipótese de igualdade entre as médias.

8.5 Exercícios

1. Revisite os dados `milsa` visto na aula de estatística descritiva e selecione pares de variáveis adequadas para efetuar:

   (a) um teste χ^2

   (b) um teste para o coeficiente de correlação

   (c) um teste t

2. Inspecione o conjunto de dados humanos.txt, selecione variáveis a aplique os testes vistos nesta Seção.

3. Queremos verificar se machos e fêmeas de uma mesma espécie possuem o mesmo comprimento (em *mm*) Para isso, foram medidos 6 exemplares de cada sexo e obtivemos os seguintes comprimentos:

Machos	145	127	136	142	141	137
Fêmeas	143	128	132	138	142	132

   Obtenha intervalos de confiança para a razão das variâncias e para a diferença das médias dos dois grupos.

   Dica: Use as funções `var.test` e `t.test`

4. Carregue o conjunto de dados *iris* usando o comando `data(iris)`.
   Veja a descrição dos dados em `help(iris)`.
   Use a função `cor.test` para testar a correlação entre o comprimento de sépalas e pétalas.

9 Experimentos com delineamento inteiramente casualizados

Nesta sessão iremos usar o R para analisar um experimento em delineamento inteiramente casualizado.
A seguir são apresentados os comandos para a análise do experimento. Inspecione-os cuidadosamente e discuta os resultados e a manipulação do programa R.

Primeiro lemos o arquivo de dados que deve ter sido copiado para o seu diretório de trabalho.

```
ex01 <- read.table("exemplo01.txt", head=T)
ex01
```

Caso o arquivo esteja em outro diretório deve-se colocar o caminho completo deste diretório no argumento de `read.table` acima.
A seguir vamos inspecionar o objeto que armazena os dados e suas componentes.

```
is.data.frame(ex01)
names(ex01)

ex01$resp
ex01$trat

is.factor(ex01$trat)
is.numeric(ex01$resp)
```

Portando concluímos que o objeto é um *data-frame* com duas variáveis, sendo uma delas um fator (a variável *trat*) e a outra uma variável numérica.

Vamos agora fazer uma rápida análise descritiva:

```
summary(ex01)
tapply(ex01$resp, ex01$trat, mean)
```

Há um mecanismo no R de "anexar"objetos ao caminho de procura que permite economizar um pouco de digitação. Veja os comandos abaixo e compara com o comando anterior.

```
search()

attach(ex01)
search()

tapply(resp, trat, mean)
```

Interessante não? Quando "anexamos"um objeto do tipo *list* ou *data.frame* no caminho de procura com o comando `attach()` fazemos com que os componentes deste objeto se tornem imediatamente disponíveis e portanto podemos, por exemplo, digitar somente `trat` ao invés de ex01$trat.

Vamos prosseguir com a análise exploratória, obtendo algumas medidas e gráficos.

```
ex01.m <- tapply(resp, trat, mean)
ex01.m
```

```
ex01.v <-  tapply(resp, trat, var)
ex01.v

plot(ex01)
points(ex01.m, pch="x", col=2, cex=1.5)

boxplot(resp ~ trat)
```

Além dos gráficos acima podemos também verificar a homogeneidade de variâncias com o Teste de Bartlett.

```
bartlett.test(resp, trat)
```

Agora vamos fazer a análise de variância. Vamos "desanexar"o objeto com os dados (embora isto não seja obrigatório).

```
detach(ex01)

ex01.av <- aov(resp ~ trat, data = ex01)
ex01.av

summary(ex01.av)
anova(ex01.av)
```

Portanto o objeto `ex01.av` guarda os resultados da análise. Vamos inspecionar este objeto mais cuidadosamente e fazer também uma análise dos resultados e resíduos:

```
names(ex01.av)
ex01.av$coef

ex01.av$res
residuals(ex01.av)

plot(ex01.av) # pressione a tecla enter para mudar o gráfico

par(mfrow=c(2,2))
plot(ex01.av)
par(mfrow=c(1,1))

plot(ex01.av$fit, ex01.av$res, xlab="valores ajustados", ylab="resíduos")
title("resíduos vs Preditos")

names(anova(ex01.av))
s2 <- anova(ex01.av)$Mean[2]   # estimativa da variância

res <- ex01.av$res        # extraindo resíduos
respad <- (res/sqrt(s2))  # resíduos padronizados
boxplot(respad)
title("Resíduos Padronizados" )

hist(respad, main=NULL)
```

```
title("Histograma dos resíduos padronizados")

stem(respad)
qqnorm(res,ylab="Residuos", main=NULL)
qqline(res)
title("Gráfico Normal de Probabilidade dos Resíduos")

shapiro.test(res)
```

E agora um teste Tukey de comparação múltipla

```
ex01.tu <- TukeyHSD(ex01.av)
plot(ex01.tu)
```

10 Experimentos com delineamento em blocos ao acaso

Vamos agora analisar o experimento em blocos ao acaso descrito na apostila do curso. Os dados estão reproduuzidos na tabela abaixo.

Tabela 4: Conteúdo de óleo de *S. linicola*, em percentagem, em vários estágios de crescimento (Steel & Torrie, 1980, p.199).

Estágios	Blocos			
	I	II	III	IV
Estágio 1	4,4	5,9	6,0	4,1
Estágio 2	3,3	1,9	4,9	7,1
Estágio 3	4,4	4,0	4,5	3,1
Estágio 4	6,8	6,6	7,0	6,4
Estágio 5	6,3	4,9	5,9	7,1
Estágio 6	6,4	7,3	7,7	6,7

Inicialmente vamos entrar com os dados no R. Há várias possíveis maneiras de fazer isto. Vamos aqui usar a função `scan` e entrar com os dados por linha da tabela. Digitamos o comando abaixo e e função `scan` recebe os dados. Depois de digitar o último dado digitamos ENTER em um campo em branco e a função encerra a entrada de daods retornando para o *prompt* do programa.

OBS: Note que, sendo um programa escrito na língua inglesa, os decimais devem ser indicados por '.' e não por vírgulas.

```
> y <- scan()
1: 4.4
2: 5.9
3: 6.0
...
24: 6.7
25:
Read 24 items
```

Agora vamos montar um *data.frame* com os dados e os indicadores de blocos e tratamentos.

```
ex02 <- data.frame(estag = factor(rep(1:6, each=4)), bloco=factor(rep(1:4, 6)), resp=y)
```

Note que usamos a função `factor` para indicar que as variáveis `blocos` e `estag` são níveis de fatores e não valores numéricos.

Vamos agora explorar um pouco os dados.

```
names(ex02)
summary(ex02)

attach(ex02)

plot(resp ~ estag + bloco)

interaction.plot(estag, bloco, resp)
interaction.plot(bloco, estag, resp)
```

```
ex02.mt <- tapply(resp, estag, mean)
ex02.mt
ex02.mb <- tapply(resp, bloco, mean)
ex02.mb

plot.default(estag, resp)
points(ex02.mt, pch="x", col=2, cex=1.5)

plot.default(bloco, resp)
points(ex02.mb, pch="x", col=2, cex=1.5)
```

Nos gráficos e resultados acima procuramos captar os principais aspectos dos dados bem como verificar se não há interação entre blocos e tratamentos, o que não deve acontecer neste tipo de experimento.

A seguir vamos ajustar o modelo e obter outros resultados, incluindo a análise de resíduos e testes para verificar a validades dos pressupostos do modelo.

```
ex02.av <- aov(resp ~ bloco + estag)
anova(ex02.av)

names(ex02.av)

par(mfrow=c(2,2))
plot(ex02.av)

par(mfrow=c(2,1))
residuos <- (ex02.av$residuals)

plot(ex02$bloco,residuos)
title("Resíduos vs Blocos")

plot(ex02$estag,residuos)
title("Resíduos vs Estágios")

par(mfrow=c(2,2))
preditos <- (ex02.av$fitted.values)
plot(residuos,preditos)
title("Resíduos vs Preditos")
respad <- (residuos/sqrt(anova(ex02.av)$"Mean Sq"[2]))
boxplot(respad)
title("Resíduos Padronizados")
qqnorm(residuos,ylab="Residuos", main=NULL)
qqline(residuos)
title("Grafico Normal de \n Probabilidade dos Resíduos")

## teste para normalidade
shapiro.test(residuos)

## Testando a não aditividade
```

```
## primeiro vamos extrair coeficientes de tratamentos e blocos
ex02.av$coeff
bl <- c(0, ex02.av$coeff[2:4])
tr <- c(0, ex02.av$coeff[5:9])
bl
tr

## agora criar um novo termo e testar sua significancia na ANOVA
bltr <- rep(bl, 6) * rep(tr, rep(4,6))

ttna <- update(ex02.av, .~. + bltr)
anova(ttna)
```

Os resultados acima indicam que os pressupostos estão obedecidos para este conjunto de dados e a análise de variância é válida. Como foi detectado efeito de tratamentos vamos proceder fazendo um teste de comparações múltiplas e encerrar as análises desanexando o objeto do caminho de procura.

```
ex02.tk <- TukeyHSD(ex02.av, "estag", ord=T)
ex02.tk
plot(ex02.tk)

detach(ex02)
```

11 Experimentos em esquema fatorial

O experimento fatorial descrito na apostila do curso de Planejamento de Experimentos II comparou o crescimento de mudas de eucalipto considerando diferentes recipientes e espécies.

1. **Lendo os dados**

   Vamos considerar agora que os dados já esteajm digitados em um arquivo texto. Clique aqui para ver e copiar o arquivo com conjunto de dados para o seu diretório de trabalho.

   A seguir vamos ler (importar) os dados para R com o comando `read.table`:

```
> ex04 <- read.table("exemplo04.txt", header=T)
> ex04
```

   Antes de começar as análise vamos inspecionar o objeto que contém os dados para saber quantas observações e variáveis há no arquivo, bem como o nome das variáveis. Vamos tembém pedir o R que exiba um rápido resumo dos dados.

```
> dim(ex04)
[1] 24  3

> names(ex04)
[1] "rec"  "esp"  "resp"

> attach(ex04)

> is.factor(rec)
[1] TRUE
> is.factor(esp)
[1] TRUE
> is.factor(resp)
[1] FALSE
> is.numeric(resp)
[1] TRUE
```

   Nos resultados acima vemos que o objeto `ex04` que contém os dados tem 24 linhas (observações) e 3 colunas (variáveis). As variáveis tem nomes `rec`, `esp` e resp, sendo que as duas primeiras são *fatores* enquanto `resp` é uma variável numérica, que neste caso é a variável resposta. O objeto `ex04` foi incluído no caminho de procura usando o comando `attach` para facilitar a digitação.

2. **Análise exploratória**

   Inicialmente vamos obter um resumo de nosso conjunto de dados usando a função `summary`.

```
> summary(ex04)
 rec    esp          resp
 r1:8   e1:12   Min.   :18.60
 r2:8   e2:12   1st Qu.:19.75
 r3:8           Median :23.70
```

```
Mean   :22.97
3rd Qu.:25.48
Max.   :26.70
```

Note que para os fatores são exibidos o número de dados em cada nível do fator. Já para a variável numérica são mostrados algumas medidas estatísticas. Vamos explorar um pouco mais os dados

```
> ex04.m <- tapply(resp, list(rec,esp), mean)
> ex04.m
       e1     e2
r1 25.650 25.325
r2 25.875 19.575
r3 20.050 21.325

> ex04.mr <- tapply(resp, rec, mean)
> ex04.mr
     r1      r2      r3
25.4875 22.7250 20.6875

> ex04.me <- tapply(resp, esp, mean)
> ex04.me
      e1       e2
23.85833 22.07500
```

Nos comandos acima calculamos as médias para cada fator, assim como para os cruzamentos entre os fatores. Note que podemos calcular outros resumos além da média. Experimente nos comandos acima substituir `mean` por `var` para calcular a variância de cada grupo, e por `summary` para obter um outro resumo dos dados.

Em experimentos fatoriais é importante verificar se existe interação entre os fatores. Inicialmente vamos fazer isto graficamente e mais a frente faremos um teste formal para presença de interação. Os comandos a seguir são usados para produzir os gráficos exibidos na Figura 5.

```
> par(mfrow=c(1,2))
> interaction.plot(rec, esp, resp)
> interaction.plot(esp, rec, resp)
```

3. **Análise de variância**

   Seguindo o modelo adequado, o análise de variância para este experimento inteiramente casualizado em esquema fatorial pode ser obtida com o comando:

   ```
   > ex04.av <- aov(resp ~ rec + esp + rec * esp)
   ```

   Entretanto o comando acima pode ser simplificado produzindo os mesmos resultados com o comando

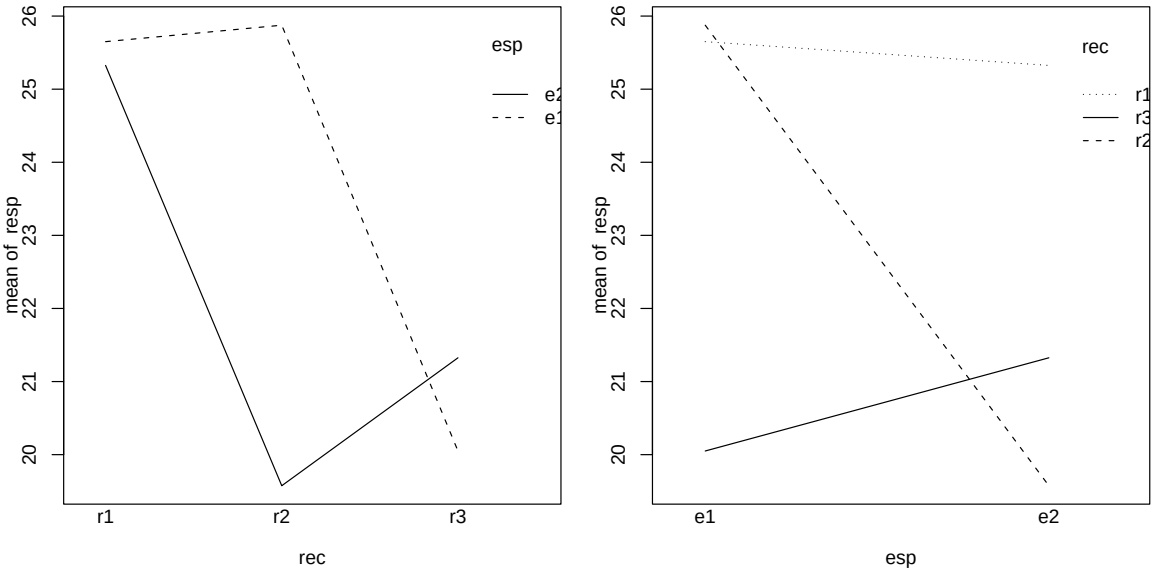

Figura 5: Gráficos de interação entre os fatores.

```
> ex04.av <- aov(resp ~ rec * esp)
> summary(ex04.av)
            Df Sum Sq Mean Sq F value    Pr(>F)
rec          2 92.861  46.430  36.195 4.924e-07 ***
esp          1 19.082  19.082  14.875  0.001155 **
rec:esp      2 63.761  31.880  24.853 6.635e-06 ***
Residuals   18 23.090   1.283
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Isto significa que no R, ao colocar uma interação no modelo, os efeitos principais são incluídos automaticamente. Note no quadro de análise de variância que a interação é denotada por `rec:esp`. A análise acima ostra que este efeito é significativo, confirmando o que verificamos nos gráficos de interação vistos anteriormente.

O objeto `ex04.av` guarda todos os resultados da análise e pode ser explorado por diversos comandos. Por exemplo a função `model.tables` aplicada a este objeto produz tabelas das médias definidas pelo modelo. O resultado mostra a média geral, médias de cada nível fatores e das combinações dos níveis dos fatores. Note que no resultado está incluído também o número de dados que gerou cada média.

```
> ex04.mt <- model.tables(ex04.av, ty="means")
> ex04.mt
Tables of means
Grand mean

22.96667

 rec
       r1    r2    r3
    25.49 22.73 20.69
rep  8.00  8.00  8.00
```

```
 esp
       e1    e2
    23.86 22.07
rep 12.00 12.00

 rec:esp
     esp
rec   e1     e2
  r1  25.650 25.325
  rep  4.000  4.000
  r2  25.875 19.575
  rep  4.000  4.000
  r3  20.050 21.325
  rep  4.000  4.000
```

Mas isto não é tudo! O objeto `ex04.av` possui vários elementos que guardam informações sobre o ajuste.

```
> names(ex04.av)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "contrasts"     "xlevels"       "call"          "terms"
[13] "model"

> class(ex04.av)
[1] "aov" "lm"
```

O comando `class` mostra que o objeto `ex04.av` pertence às classes `aov` e `lm`. Isto significa que devem haver *métodos* associados a este objeto que tornam a exploração do resultado mais fácil. Na verdade já usamos este fato acima quando digitamos o comando `summary(ex04.av)`. Existe uma função chamada `summary.aov` que foi utilizada já que o objeto é da classe `aov`. Iremos usar mais este mecanismo no próximo passo da análise.

4. **Análise de resíduos**

   Após ajustar o modelo devemos proceder a análise dos resíduos para verificar os pressupostos. O R produz automaticamente 4 gráficos básicos de resíduos conforme a Figura 6 com o comando `plot`.

```
> par(mfrow=c(2,2))
> plot(ex04.av)
```

   Os gráficos permitem uma análise dos resíduos que auxiliam no julgamento da adequacidade do modelo. Evidentemente voce não precisa se limitar os gráficos produzidos automaticamente pelo R – voce pode criar os seus próprios gráficos muito facilmente. Neste gráficos voce pode usar outras variáveis, mudar texto de eixos e títulos, etc, etc, etc. Examine os comandos abaixo e os gráficos por eles produzidos.

```
> par(mfrow=c(2,1))
> residuos <- resid(ex04.av)
```

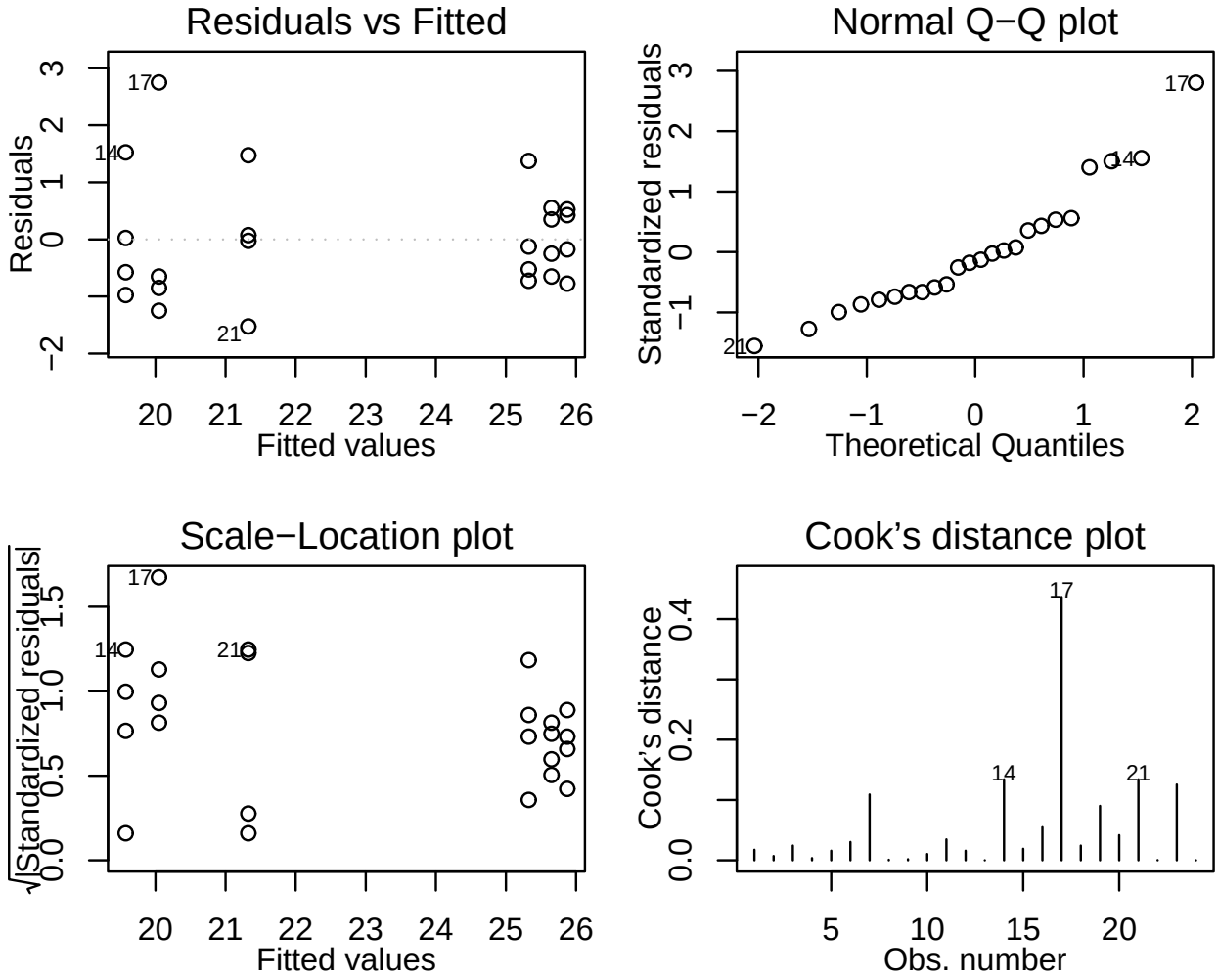

Figura 6: Gráficos de resíduos produzidos automaticamente pelo R.

```
> plot(ex04$rec, residuos)
> title("Resíduos vs Recipientes")

> plot(ex04$esp, residuos)
> title("Resíduos vs Espécies")

> par(mfrow=c(2,2))
> preditos <- (ex04.av$fitted.values)
> plot(residuos, preditos)
> title("Resíduos vs Preditos")
> s2 <- sum(resid(ex04.av)^2)/ex04.av$df.res
> respad <- residuos/sqrt(s2)
> boxplot(respad)
> title("Resíduos Padronizados")
> qqnorm(residuos,ylab="Residuos", main=NULL)
> qqline(residuos)
> title("Grafico Normal de \n Probabilidade dos Resíduos")
```

Além disto há alguns testes já programados. Como exemplo vejamos e teste de Shapiro-

Wilk para testar a normalidade dos resíduos.

```
> shapiro.test(residuos)

        Shapiro-Wilk normality test

data:  residuos
W = 0.9293, p-value = 0.09402
```

5. **Desdobrando interações**

   Conforma visto na apostila do curso, quando a interação entre os fatores é significativa podemos desdobrar os graus de liberdade de um fator dentro de cada nível do outro. A forma de fazer isto no R é reajustar o modelo utilizando a notação `/` que indica efeitos aninhados. Desta forma podemos desdobrar os efeitos de espécie dentro de cada recipiente e vice versa conforme mostrado a seguir.

```
> ex04.avr <- aov(resp ~ rec/esp)
> summary(ex04.avr, split=list("rec:esp"=list(r1=1, r2=2)))
              Df Sum Sq Mean Sq F value    Pr(>F)
rec            2 92.861  46.430 36.1952 4.924e-07 ***
rec:esp        3 82.842  27.614 21.5269 3.509e-06 ***
  rec:esp: r1  1  0.211   0.211  0.1647    0.6897
  rec:esp: r2  1 79.380  79.380 61.8813 3.112e-07 ***
  rec:esp: r3  1  3.251   3.251  2.5345    0.1288
Residuals     18 23.090   1.283
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


> ex04.ave <- aov(resp ~ esp/rec)
> summary(ex04.ave, split=list("esp:rec"=list(e1=c(1,3), e2=c(2,4))))
              Df  Sum Sq Mean Sq F value    Pr(>F)
esp            1  19.082  19.082  14.875  0.001155 **
esp:rec        4 156.622  39.155  30.524 8.438e-08 ***
  esp:rec: e1  2  87.122  43.561  33.958 7.776e-07 ***
  esp:rec: e2  2  69.500  34.750  27.090 3.730e-06 ***
Residuals     18  23.090   1.283
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

6. **Teste de Tukey para comparações múltiplas**

   Há vários testes de comparações múltiplas disponíveis na literatura, e muitos deles implementados no R. Os que não estão implementados podem ser facilmente calculados utilizando os recursos do R.

   Vejamos por exemplo duas formas de usar o *Teste de Tukey*, a primeira usando uma implementação com a função `TukeyHSD` e uma segunda fazendo ops cálculos necessários com o R.

   Poderíamos simplesmente digitar:

```
> ex04.tk <- TukeyHSD(ex04.av)
> plot(ex04.tk)
> ex04.tk
```

e obter diversos resultados. Entretanto nem todos nos interessam. Como a interação foi significativa na análise deste experimento a comparação dos níveis fatores principais não nos interessa.

Podemos então pedir a função que somente mostre a comparação de médias entre as combinações dos nívies dos fatores.

```
> ex04.tk <- TukeyHSD(ex04.ave, "esp:rec")
> plot(ex04.tk)
> ex04.tk
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = resp ~ esp/rec)

$"esp:rec"
        diff        lwr       upr
 [1,] -0.325 -2.8701851  2.220185
 [2,]  0.225 -2.3201851  2.770185
 [3,] -6.075 -8.6201851 -3.529815
 [4,] -5.600 -8.1451851 -3.054815
 [5,] -4.325 -6.8701851 -1.779815
 [6,]  0.550 -1.9951851  3.095185
 [7,] -5.750 -8.2951851 -3.204815
 [8,] -5.275 -7.8201851 -2.729815
 [9,] -4.000 -6.5451851 -1.454815
[10,] -6.300 -8.8451851 -3.754815
[11,] -5.825 -8.3701851 -3.279815
[12,] -4.550 -7.0951851 -2.004815
[13,]  0.475 -2.0701851  3.020185
[14,]  1.750 -0.7951851  4.295185
[15,]  1.275 -1.2701851  3.820185
```

Mas ainda assim temos resultados que não interessam. Mais especificamente estamos intessados nas comparações dos níveis de um fator dentro dos nívies de outro. Por exemplo, vamos fazer as comparações dos recipientes para cada uma das espécies.

Primeiro vamos obter

```
> s2 <- sum(resid(ex04.av)^2)/ex04.av$df.res
> dt <- qtukey(0.95, 3, 18) * sqrt(s2/4)
> dt
[1] 2.043945
>
> ex04.m
       e1     e2
r1 25.650 25.325
```

```
r2 25.875 19.575
r3 20.050 21.325
>
> m1 <- ex04.m[,1]
> m1
    r1     r2     r3
25.650 25.875 20.050
> m1d <- outer(m1,m1,"-")
> m1d
       r1     r2    r3
r1  0.000 -0.225 5.600
r2  0.225  0.000 5.825
r3 -5.600 -5.825 0.000
> m1d <- m1d[lower.tri(m1d)]
> m1d
    r2     r3   <NA>
 0.225 -5.600 -5.825
>
> m1n <- outer(names(m1),names(m1),paste, sep="-")
> names(m1d) <- m1n[lower.tri(m1n)]
> m1d
 r2-r1  r3-r1  r3-r2
 0.225 -5.600 -5.825
>
> data.frame(dif = m1d, sig = ifelse(abs(m1d) > dt, "*", "ns"))
         dif sig
r2-r1  0.225  ns
r3-r1 -5.600   *
r3-r2 -5.825   *
>
> m2 <- ex04.m[,2]
> m2d <- outer(m2,m2,"-")
> m2d <- m2d[lower.tri(m2d)]
> m2n <- outer(names(m2),names(m2),paste, sep="-")
> names(m2d) <- m2n[lower.tri(m2n)]
> data.frame(dif = m2d, sig = ifelse(abs(m2d) > dt, "*", "ns"))
        dif sig
r2-r1 -5.75   *
r3-r1 -4.00   *
r3-r2  1.75  ns
```

12 Transformação de dados

Tranformação de dados é uma das possíveis formas de contarnar o problema de dados que não obedecem os pressupostos da análise de variância. Vamos ver como isto poder ser feito com o programa R.

Considere o seguinte exemplo da apostila do curso.

Tabela 5: Número de reclamações em diferentes sistemas de atendimento

Trat	Repetições					
	1	2	3	4	5	6
1	2370	1687	2592	2283	2910	3020
2	1282	1527	871	1025	825	920
3	562	321	636	317	485	842
4	173	127	132	150	129	227
5	193	71	82	62	96	44

Inicialmente vamos entrar com os dados usando a função `scan` e montar um *data-frame*.

```
> y <- scan()
1: 2370
2: 1687
3: 2592
...
30: 44
31:
Read 30 items

> tr <- data.frame(trat = factor(rep(1:5, each=6)), resp = y)
> tr
```

A seguir vamos fazer ajustar o modelo e inspecionar os resíduos.

```
tr.av <- aov(resp ~ trat, data=tr)
plot(tr.av)
```

O gráfico de resíduos *vs* valores preditos mostra claramente uma heterogeneidade de variâncias e o $QQ - plot$ mostra um comportamento dos dados que se afasta muito da normal. A menssagem é clara mas podemos ainda fazer testes para verificar o desvio dos pressupostos.

```
> bartlett.test(tr$resp, tr$trat)

        Bartlett test for homogeneity of variances

data:  tr$resp and tr$trat
Bartlett's K-squared = 29.586, df = 4, p-value = 5.942e-06

> shapiro.test(tr.av$res)

        Shapiro-Wilk normality test

data:  tr.av$res
W = 0.939, p-value = 0.08535
```

Nos resultados acima vemos que a homogeneidade de variâncias foi rejeitada.

Para tentar contornar o problema vamos usar a transformação Box-Cox, que consiste em transformar os dados de acordo com a expressão

$$y' = \frac{y^\lambda - 1}{\lambda},$$

onde λ é um parâmeto a ser estimado dos dados. Se $\lambda = 0$ a equação acima se reduz a

$$y' = \log(y),$$

onde log é o logarítmo neperiano. Uma vez obtido o valor de λ encontramos os valores dos dados transformados conforme a equação acima e utilizamos estes dados transformados para efetuar as análises.

A função `boxcox` do pacote `MASS` calcula a verossimilhança perfilhada do parâmetro λ. Devemos escolher o valor que maximiza esta função. Nos comandos a seguir começamos carregando o pacote `MASS` e depois obtemos o gráfico da verossimilhança perfilhada. Como estamos interessados no máximo fazermos um novo gráfico com um *zoom* na região de interesse.

```
require(MASS)
boxcox(resp ~ trat, data=tr, plotit=T)
boxcox(resp ~ trat, data=tr, lam=seq(-1, 1, 1/10))
```

O gráfico mostra que o valor que maximiza a função é aproximadamente $\hat{\lambda} = 0.1$. Desta forma o próximo passo é obter os dados transformados e depois fazer as análise utilizando estes novos dados.

```
tr$respt <- (tr$resp^(0.1) - 1)/0.1
tr.avt <- aov(respt ~ trat, data=tr)
plot(tr.avt)
```

Note que os resíduos tem um comportamento bem melhor do que o observado para os dados originais. A análise deve prosseguir usando então os dados transformados.

NOTA: No gráfico da verossimilhança perfilhada notamos que é mostrado um intervalo de confiança para λ e que o valor 0 está contido neste intervalo. Isto indica que podemos utilizar a transformação logarítimica dos dados e os resultados serão bom próximos dos obtidos com a transformação préviamente adotada.

```
tr.avl <- aov(log(resp) ~ trat, data=tr)
plot(tr.avl)
```