Curso sobre o programa computacional R

Paulo Justiniano Ribeiro Junior

Última atualização: 16 de janeiro de 2005

Sobre o ministrante do curso

Paulo Justiniano Ribeiro Junior é Eng. Agrônomo pela ESAL, Lavras (atual UFLA), Mestre em Agronomia com área de concentração em estatística e experimentação agronômica pela ESALQ/USP. PhD em Estatística pela Lancaster University, UK.

PJRJr é professor do Departamento de Estatística da Universidade Federal do Paraná desde 1992 e tem usado o programa R em suas pesquisas desde 1999. É co-autor dos pacotes geoR e geoRglm contribuídos ao **CRAN** (*Compreheensive R Arquives Network*).

1 Uma primeira sessão com o R

Vamos começar "experimentando o R", para ter uma idéia de seus recursos e a forma de trabalhar. Para isto vamos rodar e estudar os comandos abaixo e seus resultados para nos familiarizar com o programa. Nas sessões seguintes iremos ver com mais detalhes o uso do programa 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.

```
# gerando dois vetores de coordenadas x e y de números pseudo-aleatórios
# e inspecionando os valores gerados
x <- rnorm(5)
x
y <- rnorm(x)
y

# colocando os pontos em um gráfico.
# Note que a janela gráfica se abrirá automaticamente
plot(x, y)

# verificando os objetos existentes na área de trabalho
ls()

# removendo objetos que não são mais necessários
rm(x, y)

# criando um vetor com uma sequência de números de 1 a 20
x <- 1:20

# um vetor de pesos com os desvios padrões de cada observação
w <- 1 + sqrt(x)/2

# montando um 'data-frame' de 2 colunas, x e y, e inspecionando o objeto
dummy <- data.frame(x=x, y= x + rnorm(x)*w)
dummy

# Ajustando uma regressão linear simples de y em x e examinando os resultados
fm <- lm(y ~ x, data=dummy)
summary(fm)

# como nós sabemos os pesos podemos fazer uma regressão ponderada
fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)
summary(fm1)
```

```
# tornando visíveis as colunas do data-frame
attach(dummy)

# fazendo uma regressão local não-paramétrica, e visualizando o resultado
lrf <- lowess(x, y)
plot(x, y)
lines(lrf)

# ... e a linha de regressão verdadeira (intercepto 0 e inclinação 1)
abline(0, 1, lty=3)

# a linha da regressão sem ponderação
abline(coef(fm))

# e a linha de regressão ponderada.
abline(coef(fm1), col = "red")

# removendo o objeto do caminho de procura
detach()

# O gráfico diagnóstico padrão para checar homocedasticidade.
plot(fitted(fm), resid(fm),
     xlab="Fitted values", ylab="Residuals",
     main="Residuals vs Fitted")

# gráficos de escores normais para checar assimetria, curtose e outliers (não muito útil a
qqnorm(resid(fm), main="Residuals Rankit Plot")

# ‘‘limpando’’ novamente (apagando objetos)
rm(fm, fm1, lrf, x, dummy)
```

Agora vamos inspecionar dados do experimento clássico de Michaelson e Morley para medir a velocidade da luz. Clique para ver o arquivo morley.tab de dados no formato texto. Gravar este arquivo no diretório `c:\temp`.

```
# para ver o arquivo digite:
file.show("c:\\temp\\morley.tab.txt")

# Lendo dados como um 'data-frame' e inspecionando  seu conteúdo.
# Há 5 experimentos (coluna Expt) e cada um com 20 ‘‘rodadas’’(coluna
# Run) e sl é o valor medido da velocidade da luz numa escala apropriada
mm <- read.table("c:\\temp\\morley.tab.txt")
mm

# definindo Expt e Run como fatores
mm$Expt <- factor(mm$Expt)
mm$Run <- factor(mm$Run)

# tornando o data-frame visível na posição 2 do caminho de procura (default)
attach(mm)
```

```
# comparando os 5 experimentos
plot(Expt, Speed, main="Speed of Light Data", xlab="Experiment No.")

# analisando como blocos ao acaso com ‘runs’ and ‘experiments’ como
fatores e inspecionando resultados
fm <- aov(Speed ~ Run + Expt, data=mm)
summary(fm)
names(fm)
fm$coef

# ajustando um sub-modelo sem ‘‘runs’’ e comparando via análise de variância
fm0 <- update(fm, . ~ . - Run)
anova(fm0, fm)

# desanexando o objeto e limpando novamente
detach()
rm(fm, fm0)
```

Vamos agora ver alguns gráficos gerados pelas funções `contour` e `image`.

```
# x é um vetor de 50 valores igualmente espaçados  no intervalo [-pi\, pi]. y idem.
x <- seq(-pi, pi, len=50)
y <- x

# f é uma matrix quadrada com linhas e colunas indexadas por x e y respectivamente
# com os valores da função cos(y)/(1 + x^2).
f <- outer(x, y, function(x, y) cos(y)/(1 + x^2))

# gravando parâmetros gráficos e definindo a região gráfica como quadrada
oldpar <- par(no.readonly = TRUE)
par(pty="s")

# fazendo um mapa de contorno de f e depois adicionando mais linhas para maiores detalhes
contour(x, y, f)
contour(x, y, f, nlevels=15, add=TRUE)

#  fa é a ‘‘parte assimétrica’’. (t() é transposição).
fa <- (f-t(f))/2

# fazendo um mapa de contorno
contour(x, y, fa, nlevels=15)

#    ...  e restaurando parâmetros gráficos iniciais
par(oldpar)

# Fazendo um gráfico de imagem
image(x, y, f)
image(x, y, fa)

# e apagando objetos novamente antes de prosseguir.
objects(); rm(x, y, f, fa)
```

Para encerrar esta sessão vajamos mais algumas funcionalidades do R.

```
# O R pode fazer operação com complexos
th <- seq(-pi, pi, len=100)
#  1i denota o número complexo  i.
z <- exp(1i*th)

# plotando complexos significa parte imaginária versus real
# Isto deve ser um círculo:
par(pty="s")
plot(z, type="l")

# Suponha que desejamos amostrar pontos dentro do círculo de raio unitário.
# uma forma simples de fazer isto é tomar números complexos com parte
# real e imaginária padrão
w <- rnorm(100) + rnorm(100)*1i

#  ...  e para mapear qualquer externo ao círculo no seu recíproco:
w <- ifelse(Mod(w) > 1, 1/w, w)

# todos os pontos estão dentro do círculo unitário, mas a distribuição
# não é uniforme.
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+",xlab="x", ylab="y")
lines(z)

# este segundo método usa a distribuição uniforme.
# os pontos devem estar melhor distribuídos sobre o círculo
w <- sqrt(runif(100))*exp(2*pi*runif(100)*1i)
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")
lines(z)

# apagando os objetos
rm(th, w, z)

# saindo do R
q()
```

2 Aritmética e Objetos

2.1 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	logarítmo base-10

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

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

2.2 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...
```

3 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.

3.1 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

3.2 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

3.3 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?

3.4 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

3.5 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)
```

3.6 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.

4 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.

4.1 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
```

4.2 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.

4.3 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.

4.4 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()`

5 Análise descritiva

5.1 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 (categóricas)**
  - 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)
> tcyl
> prop.table(tcyl)   # outra forma de obter freq. rel.
> 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)
```

5.2 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)
> prop.table(table(am, cyl))
> prop.table(table(am, cyl), margin=1)
> prop.table(table(am, cyl), margin=2)
> 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 abaixo 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))
```

5.3 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$Ozone                # a razão é que existem ‘‘dados perdidos’’ na variável O
> mean(airquality$Ozone, na.rm=T) # média do Ozone no período - retirando valores perdidos
```

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

5.4 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 5.4. 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 5.4. Lembre-se que se voce precisar corrigir algo na digitação voce pode fazê-lo abrindo a planilha novamente com o comando `fix(milsa)`.

Tabela 1: 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 2: 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
```

Atenção: Note que além de digitar os dados na planilha digitamos também o nome que escolhemos para cada variável. Para isto basta, na planilha, clicar no nome da variável e escolher a opção `CHANGE NAME` e informar o novo nome da 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.

Além disto precisamos definir uma variável única `idade` a partir das variáveis `ano` e `mes` que foram digitadas. Para gerar a variável idade (em anos) fazemos:

```
milsa$idade <- milsa$ano + milsa$mes/12
milsa$idade

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)
## ou então
prop.table(civil.tb)

## 1.2 Gráfico
## Para máquinas com baixa resolução gráfica (Sala A - LABEST)
```

```
## use o comando da próxima linha (sem os caracteres ##)
## 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)

## 2.1 Tabela:
instrucao.tb <- table(instrucao)
instrucao.tb
prop.table(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
```

5.5 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.

5.6 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
```

5.7 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.

5.8 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) encontre a mediana e quartis para cada crime.

   (c) encontre o número máximo e mínimo para cada crime.

   (d) faça um gráfico adequado para o número de assassinatos (*murder*).

   (e) faça um diagrama ramo-e-folhas para o número de estupros (*rape*).

   (f) verifique se há correlação entre os diferentes tipos de crime.

   (g) verifique se há correlação entre os crimes e a proporção de população urbana.

   (h) encontre os estados com maior e menor ocorrência de cada tipo de crime.

   (i) encontre os estados com maior e menor ocorrência per capta de cada tipo de crime.

   (j) encontre os estados com maior e menor ocorrência do total de crimes.

6 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.

6.1 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 $N(\mu = 0, \sigma^2 = 1)$.

```
> 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

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

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 visualizar 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
```

Note que a última probabilidade foi calculada de duas formas diferentes, a segunda usando o argumento `lower` que implementa um algorítmo de cálculo de probabilidades 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 99,73% da massa de probabilidade da distribuição normal.

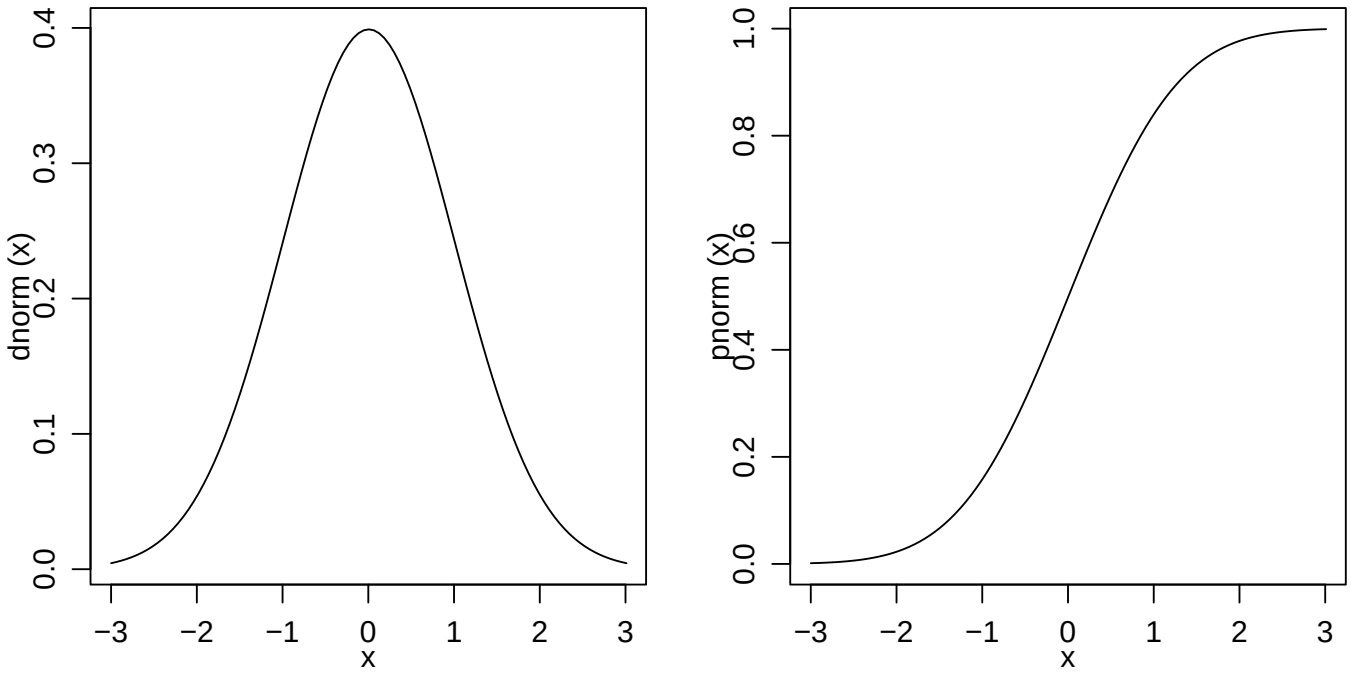

Figura 1: Funções de densidade e probabilidade da distribuição normal padrão.

```
> 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 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.

```
> 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)
```

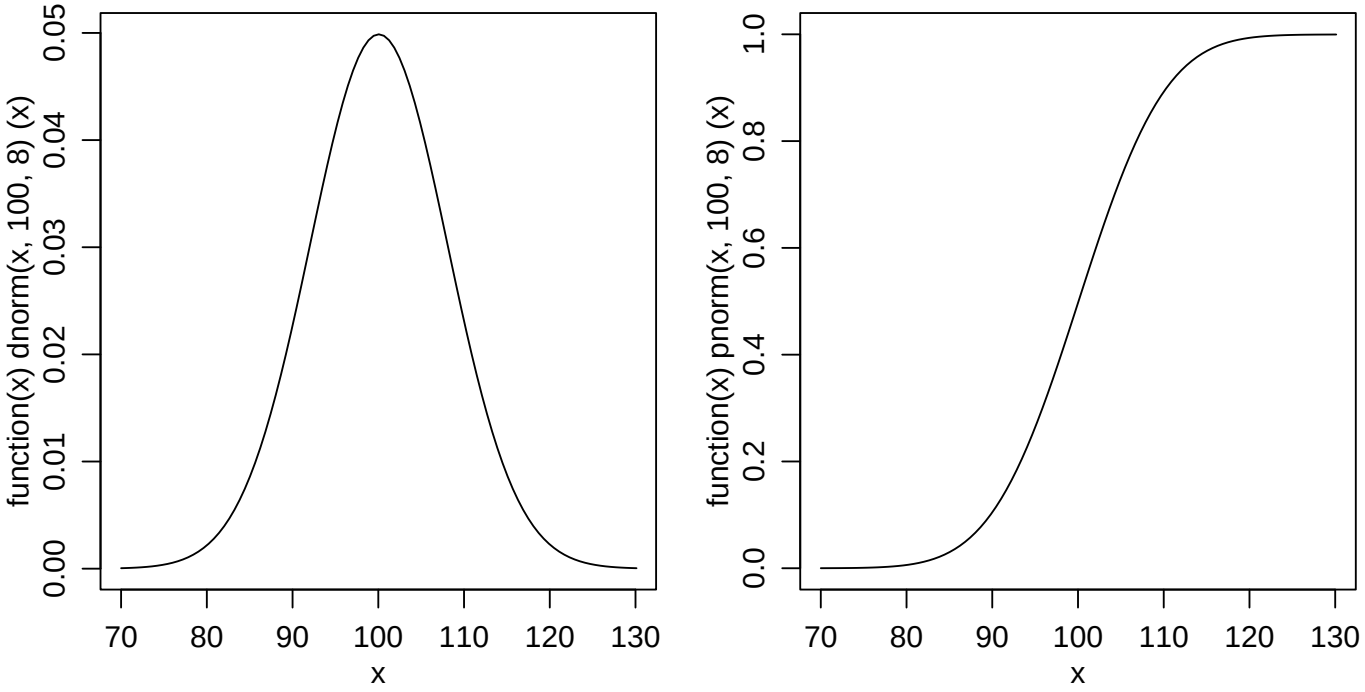

Figura 2: Funções de densidade de probabilidade (esquerda) e função de distribuição acumulada (direita) da $N(100, 64)$.

6.2 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 $P[X \geq 8] = P[X > 7]$
6. calcular $P[3 < X \leq 6] = P[4 \leq X < 7]$

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.

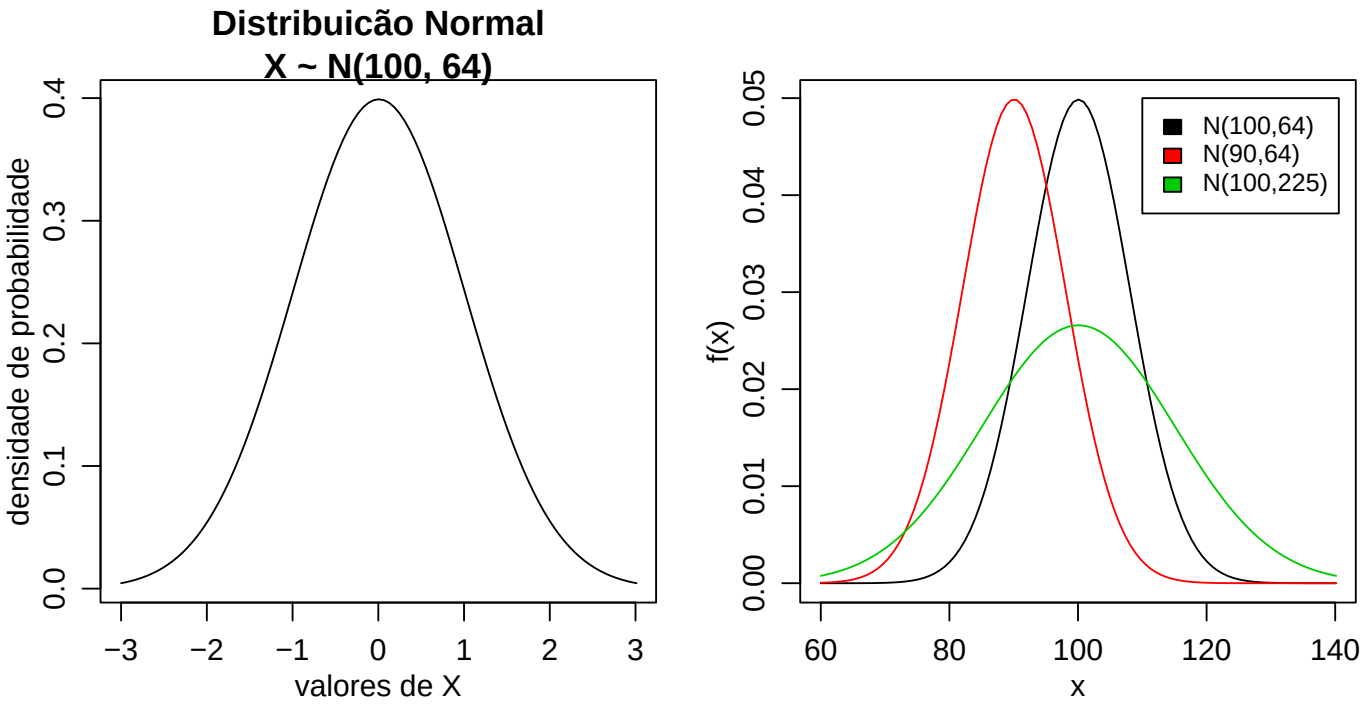

Figura 3: Gráfico com texto nos eixos e título (esquerda) e várias distribuições em um mesmo gráfico (direita).

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

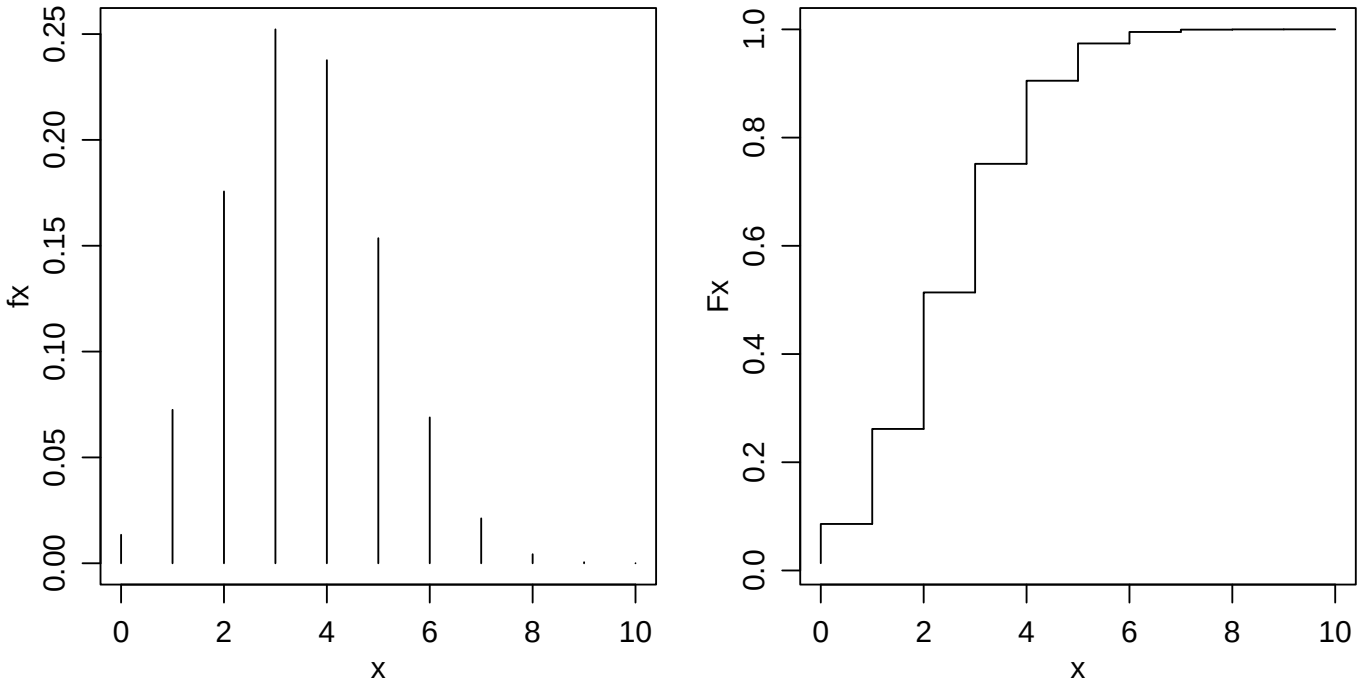

Figura 4: Funções de probabilidade (esquerda) e de distribuição acumulada (direita) da $B(10, 0.35)$.

6.3 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:

   - $P(X \leq 115)$
   - $P(X \geq 80)$
   - $P(X \leq 75)$

- $P(85 \leq X \leq 110)$
- $P(|X - 90| \leq 10)$
- O 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.

7 Conceitos básicos sobre distribuições de probabilidade

O objetivo desta sessão é mostrar o uso de funções do R em problemas de probabilidade. Exercícios que podem (e devem!) ser resolvidos analiticamente são usados para ilustrar o uso do programa e alguns de seus recursos para análises numéricas.

Os problemas nesta sessão foram retirados do livro:
Bussab, W.O. & Morettin, P.A. *Estatística Básica*. 4^a edição. Atual Editora. 1987.

EXEMPLO 1 (adaptado de Bussab & Morettin, página 132, exercício 1) Dada a função

$$f(x) = \begin{cases} 2\exp(-2x) & \text{, se } x \geq 0 \\ 0 & \text{, se } x < 0 \end{cases}$$

(a) mostre que está função é uma f.d.p.

(b) calcule a probabilidade de que $X > 1$

(c) calcule a probabilidade de que $0.2 < X < 0.8$

Para ser f.d.p. a função não deve ter valores negativos e deve integrar 1 em seu domínio. Vamos começar definindo esta função como uma *função* no R para qual daremos o nome de $f1$. A seguir fazemos o gráfico da função. Como a função tem valores positivos para x no intervalo de zero a infinito temos, na prática, para fazer o gráfico, que definir um limite em x até onde vai o gráfico da função. Vamos achar este limite tentando vários valores, conforme mostram os comandos abaixo. O gráfico escolhido foi o produzido pelo comando `plot(f1,0,5)` e mostrado na Figura 5.

```
f1 <- function(x){
  fx <- ifelse(x < 0, 0, 2*exp(-2*x))
  return(fx)
}

plot(f1)
plot(f1,0,10)
plot(f1,0,5)
```

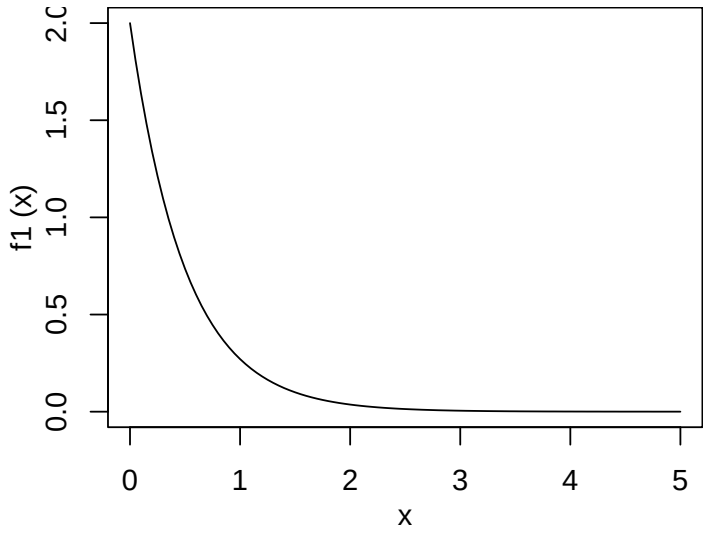

Figura 5: Gráfico da função de probabilidade do Exemplo 1.

Para verificar que a a integral da função é igual a 1 podemos usar a função `integrate` que efetua integração numérica. A função recebe como argumentos o objeto com a função a ser integrada e os limites de integração. Neste exemplo o objeto é `f1` definido acima e o domínio da função é $[0, Inf]$. A saída da função mostra o valor da integral (1) e o erro máximo da aproximação numérica.

```
> integrate(f1, 0, Inf)
1 with absolute error < 5e-07
```

Para fazer cálculos pedidos nos itens (b) e (c) lembramos que a probabilidade é dada pela área sob a curva da função no intervalo pedido. Desta forma as soluções seriam dadas pelas expressões

$$\begin{aligned} p_b &= P(X > 1) = \int_1^{\infty} f(x)dx = \int_1^{\infty} 2\,e^{-2x}dx \\ p_c &= P(0,2 < X < 0,8) = \int_{0,2}^{0,8} f(x)dx = \int_{0.2}^{0.8} 2\,e^{-2x}dx \end{aligned}$$

cuja representação gráfica é mostrada na Figura 6. Os comandos do R a seguir mostram como fazer o gráfico de função. O comando `plot` desenha o gráfico da função. Para destacar as áreas que correspondem às probabilidades pedidas vamos usar a função `polygon`. Esta função adiciona a um gráfico um polígono que é definido pelas coordenadas de seus vértices. Para sombrear a área usa-se o argumento `density`. Finalmente, para escrever um texto no gráfico usamos a função `text` com as coordenadas de posição do texto.

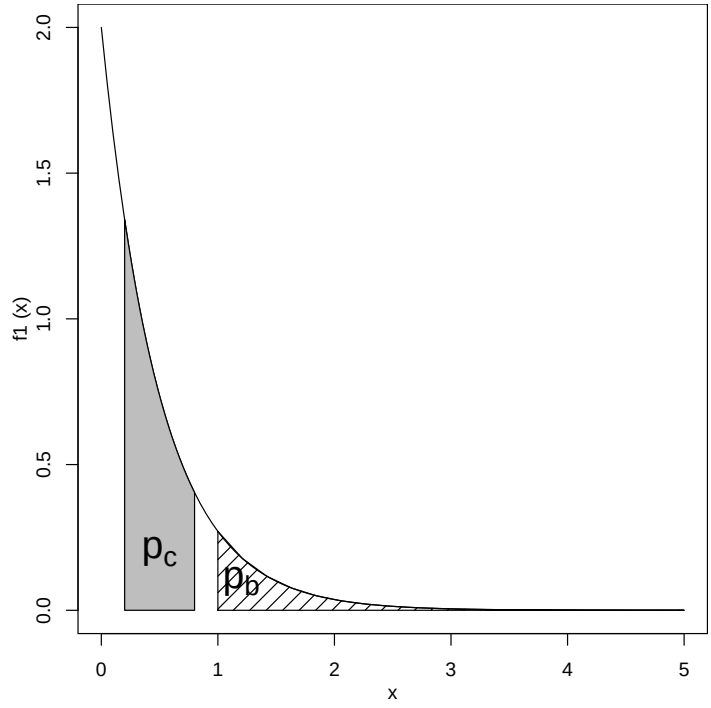

Figura 6: Probabilidades pedidas nos itens (b) e (c) do Exemplo 1.

```
> plot(f1,0,5)
> polygon(x=c(1,seq(1,5,l=20)), y=c(0,f1(seq(1,5,l=20))), density=10)
> polygon(x=c(0.2,seq(0.2,0.8,l=20),0.8), y=c(0,f1(seq(0.2,0.8,l=20)), 0), col="gray")
> text(c(1.2, 0.5), c(0.1, 0.2), c(expression(p[b],p[c])))
```

e como obter as probabilidades pedidas.

```
> integrate(f1, 1, Inf)
0.1353353 with absolute error < 2.1e-05
> integrate(f1, 0.2, 0.8)
0.4684235 with absolute error < 5.2e-15
```

EXEMPLO 2 (Bussab & Morettin, página 139, exercício 10) A demanda diária de arroz em um supermercado, em centenas de quilos, é uma v.a. X com f.d.p.

$$f(x) = \begin{cases} \frac{2}{3}x \text{ , se } 0 \leq x < 1 \\ -\frac{x}{3} + 1 \text{ , se } 1 \leq x < 3 \\ 0 \text{ , se } x < 0 \text{ ou } x \geq 3 \end{cases} \quad (1)$$

(a) Calcular a probabilidade de que sejam vendidos mais que 150 kg.

(b) Calcular a venda esperada em 30 dias.

(c) Qual a quantidade que deve ser deixada à disposição para que não falte o produto em 95% dos dias?

Novamente começamos definindo um objeto do R que contém a função dada em 1.

Neste caso definimos um vetor do mesmo tamanho do argumento `x` para armazenar os valores de $f(x)$ e a seguir preenchemos os valores deste vetor para cada faixa de valor de x. A seguir verificamos que a integral da função é 1 e fazemos o seu gráfico mostrado na Figura 7.

```
> f2 <- function(x){
+  fx <- numeric(length(x))
+  fx[x < 0] <- 0
+  fx[x >= 0 & x < 1] <- 2*x[x >= 0 & x < 1]/3
+  fx[x >= 1 & x <= 3] <- (-x[x >= 1 & x <= 3]/3) + 1
+  fx[x > 3] <- 0
+  return(fx)
+ }

> integrate(f2, 0, 3) ## verificando que a integral vale 1
1 with absolute error < 1.1e-15

> plot(f2, -1, 4)     ## fazendo o gráfico da função
```

Agora vamos responder às questões levantadas. Na questão (a) pede-se a probabilidade de que sejam vendidos mais que 150 kg (1,5 centenas de quilos), portanto a probabilidade $P[X > 1,5]$. A probabilidade corresponde à área sob a função no intervalo pedido ou seja $P[X > 1,5] = \int_{1,5}^{\infty} f(x)dx$ e esta integral pode ser resolvida numericamente com o comando:

```
> integrate(f2, 1.5, Inf)
0.3749999 with absolute error < 3.5e-05
```

A venda esperada em trinta dias é 30 vezes o valor esperado de venda em um dia. Para calcular a esperança $E[X] = \int xf(x)dx$ definimos uma nova função e resolvemos a integral. A função `integrate` retorna uma lista onde um dos elementos (`$value`) é o valor da integral.

```
## calculando a esperança da variável
> ef2 <- function(x){ x * f2(x) }
> integrate(ef2, 0, 3)
1.333333 with absolute error < 7.3e-05

> 30 * integrate(ef2, 0, 3)$value ## venda esperada em 30 dias
[1] 40
```

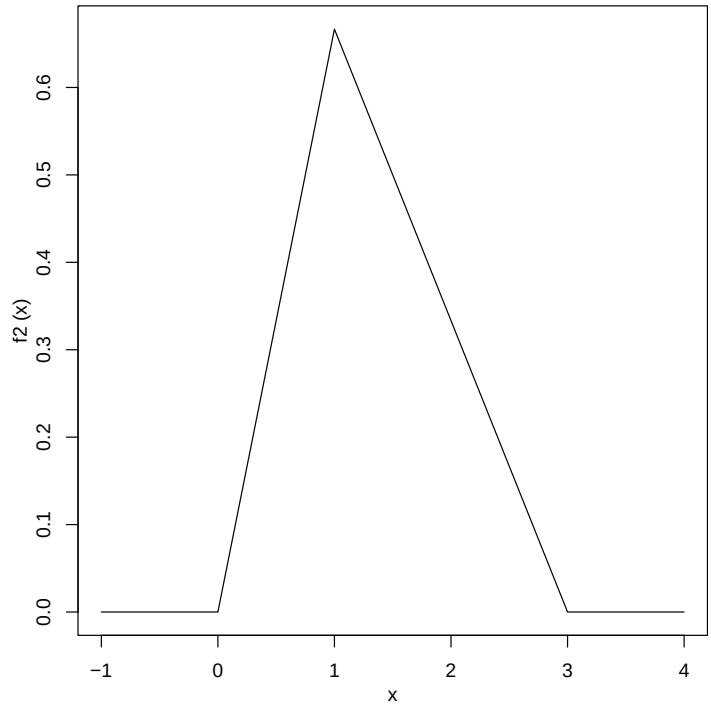

Figura 7: Gráfico da função densidade de probabilidade do Exemplo 2.

Na questão (c) estamos em busca do quantil 95% da distribuição de probabilidades, ou seja o valor de x que deixa 95% de massa de probabilidade abaixo dele. Este valor que vamos chamar de k é dado por:

$$\int_0^k f(x)dx = 0.95.$$

Para encontrar este valor vamos definir uma função que calcula a diferença (em valor absoluto) entre 0.95 e a probabilidade associada a um valor qualquer de x. O quantil será o valor que minimiza esta probabilidade. Este é portanto um problema de otimização numérica e para resolvê-lo vamos usar a função `optimize` do R, que recebe como argumentos a função a ser otimizada e o intervalo no qual deve procurar a solução. A resposta mostra o valor do quantil $x = 2.452278$ e a função objetivo com valor muito próximo de 0, que era o que desejávamos.

```
> f <- function(x) abs(0.95 - integrate(f2, 0, x)$value)
> optimise(f, c(0,3))
$minimum
[1] 2.452278

$objective
[1] 7.573257e-08
```

A Figura 8 ilustra as soluções dos itens (a) e (c) e os comandos abaixo foram utilizados para obtenção destes gráficos.

```
par(mfrow=c(1,2), mar=c(3,3,0,0), mgp=c(2,1,0))
plot(f2, -1, 4)
polygon(x=c(1.5, 1.5, 3), y=c(0,f2(1.5),0), dens=10)

k <- optimise(f, c(0,3))$min
plot(f2, -1, 4)
polygon(x=c(0, 1, k, k), y=c(0,f2(1), f2(k), 0), dens=10)
text(c(1.5, k), c(0.2, 0), c("0.95", "k"), cex=2.5)
```

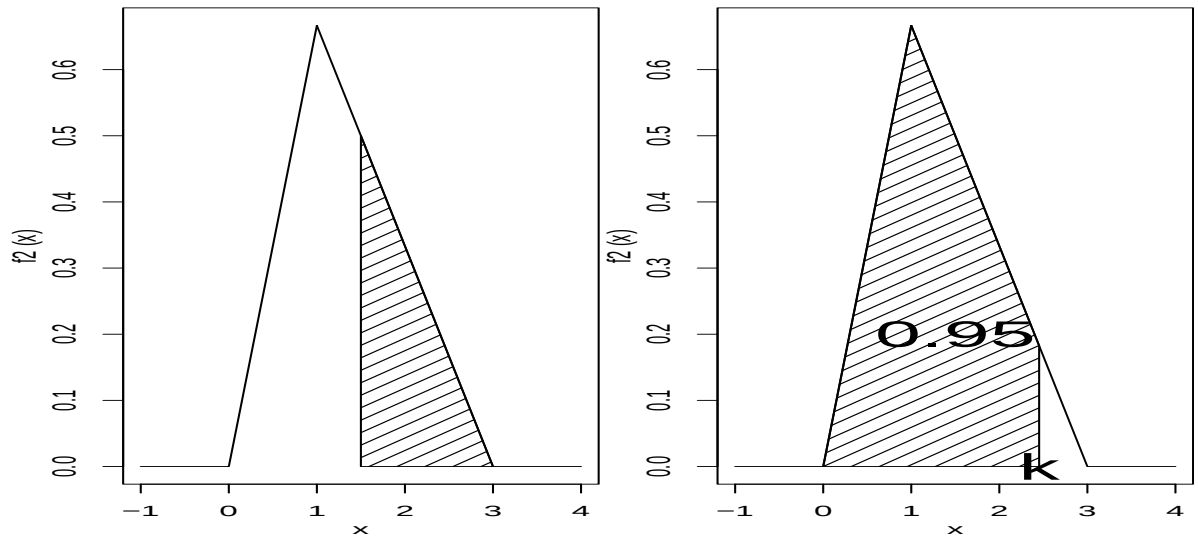

Figura 8: Gráficos indicando as soluções dos itens (a) e (c) do Exemplo 2.

Finalmente lembramos que os exemplos discutidos aqui são simples e não requerem soluções numéricas, devendo ser resolvidos analiticamente. Utilizamos estes exemplos somente para ilustrar a obtenção de soluções numéricas com o uso do R, que na prática deve ser utilizado em problemas mais complexos onde soluções analíticas não são triviais ou mesmo impossíveis.

7.1 Exercícios

1. (Bussab & Morettin, 5a edição, pag. 194, ex. 28)
   Em uma determinada localidade a distribuição de renda, em u.m. (unidade monetária) é uma variável aleatória X com função de distribuição de probabilidade:

$$f(x) = \begin{cases} \frac{1}{10}x + \frac{1}{10} & \text{se } 0 \leq x \leq 2 \\ -\frac{3}{40}x + \frac{9}{20} & \text{se } 2 < x \leq 6 \\ 0 & \text{se } x < 0 \text{ ou } x > 6 \end{cases}$$

   (a) mostre que $f(x)$ é uma f.d.p..

   (b) calcule os quartis da distribuição.

   (c) calcule a probabilidade de encontrar uma pessoa com renda acima de 4,5 u.m. e indique o resultado no gráfico da distribuição.

   (d) qual a renda média nesta localidade?

8 Complementos sobre distribuições de probabilidade

Agora que já nos familiarizamos com o uso das distribuições de probabilidade vamos ver alguns detalhes adicionais sobre seu funcionamento.

8.1 Probabilidades e integrais

A probabilidade de um evento em uma distribuição contínua é uma área sob a curva da distribuição. Vamos reforçar esta idéia revisitando um exemplo visto na aula anterior.

Seja X uma v.a. com distribuição $N(100, 100)$. Para calcular a probabilidade $P[X < 95]$ usamos o comando:

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

Vamos agora "esquecer" o comando `pnorm` e ver uma outra forma de resolver usando integração numérica. Lembrando que a normal tem a função de densidade dada por

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

vamos definir uma função no R para a densidade normal deste problema:

```
fn <- function(x){
   fx <- (1/sqrt(2*pi*100)) * exp((-1/200) * (x - 100)^2)
   return(fx)
}
```

Para obter o gráfico desta distribuição mostrado na Figura 9 usamos o fato que a maior parte da função está no intervalo entre a média +/- três desvios padrões, portanto entre 70 e 130. Podemos então fazer:

```
x <- seq(70, 130, l=200)
fx <- fn(x)
plot(x, fx, type='l')
```

Agora vamos marcar no gráfico a área que corresponde a probabilidade pedida. Para isto vamos criar um polígono com coordenadas `ax` e `ay` definindo o perímetro desta área

```
ax <- c(70, 70, x[x<95], 95,95)
ay <- c(0, fn(70), fx[x<95], fn(95),0)
polygon(ax,ay, dens=10)
```

Para calcular a área pedida sem usar a função `pnorm` podemos usar a função de integração numérica. Note que esta função, diferentemente da `pnorm` reporta ainda o erro de aproximação numérica.

```
integrate(fn, -Inf, 95)
0.3085375 with absolute error < 2.1e-06
```

Portanto para os demais ítens do problema $P[90 < X < 110]$ e $P[X > 95]$ fazemos:

```
> integrate(fn, 90, 110)
0.6826895 with absolute error < 7.6e-15
> integrate(fn, 95, +Inf)
0.6914625 with absolute error < 8.1e-05
```

e os resultados acima evidentemente coincidem com os obtidos na aula anterior usando `pnorm`.

Note ainda que na prática não precisamos definir e usar a função fn pois ela fornece o mesmo resultado que a função `dnorm`.

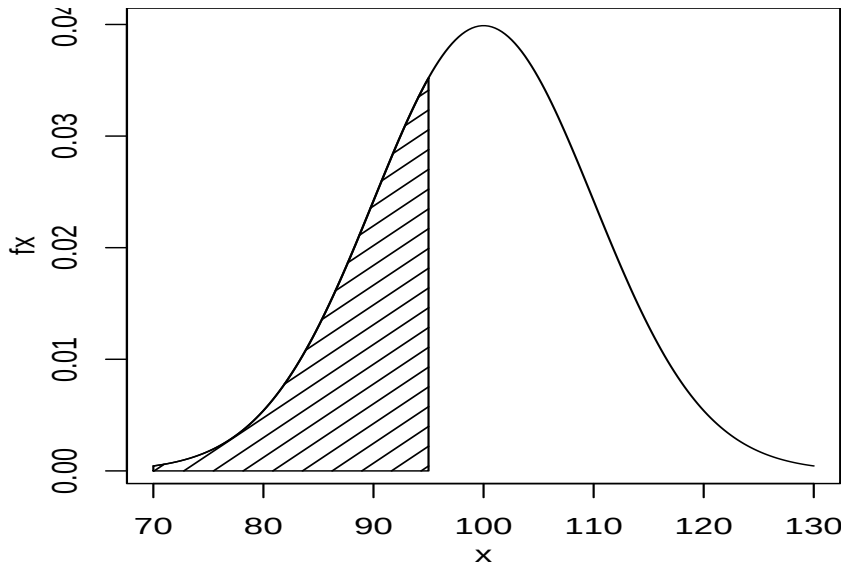

Figura 9: Funções de densidade da $N(100, 100)$ com a área correspondente à $P[X \leq 95]$.

8.2 Distribuição exponencial

A função de densidade de probabilidade da distribuição exponencial com parâmetro λ e denotada $Exp(\lambda)$ é dada por:

$$f(x) = \begin{cases} \frac{1}{\lambda}\, e^{-x/\lambda} & \text{para } x \geq 0 \\ 0 & \text{para } x < 0 \end{cases}$$

Seja uma variável X com distribuição exponencial de parâmetro $\lambda = 500$. Calcular a probabilidade $P[X \geq 400]$.

A solução analítica pode ser encontrada resolvendo

$$P[X \geq 400] = \int_{400}^{\infty} f(x)dx = \int_{400}^{\infty} \frac{1}{\lambda}\, e^{-x/\lambda}dx$$

que é uma integral que pode ser resolvida analiticamente. Fica como exercício encontrar o valor da integral acima.

Para ilustrar o uso do R vamos também obter a resposta usando integração numérica. Para isto vamos criar uma função com a expressão da exponencial e depois integrar no intervalo pedido

```
> fexp <- function(x, lambda=500){
     fx <- ifelse(x<0, 0, (1/lambda)*exp(-x/lambda))
     return(fx)
  }
> integrate(fexp, 400, Inf)
0.449329 with absolute error < 5e-06
```

e este resultado deve ser igual ao encontrado com a solução analítica.

Note ainda que poderíamos obter o mesmo resultado simplesmente usando a função `pexp` com o comando `pexp(400, rate=1/500, lower=F)`, onde o argumento corresponde a $1/\lambda$ na equação da exponencial.

A Figura 10 mostra o gráfico desta distribuição com indicação da área correspondente à probabilidade pedida. Note que a função é positiva no intervalo $(0, +\infty)$ mas para fazer o gráfico consideramos apenas o intervalo $(0, 2000)$.

```
x <- seq(0,2000, l=200)
fx <- dexp(x, rate=1/500)
```

```
plot(x, fx, type='l')

ax <- c(400, 400, x[x>400], 2000,2000)
ay <- c(0, dexp(c(400,x[x>400], 2000), 1/500),0)
polygon(ax,ay, dens=10)
```

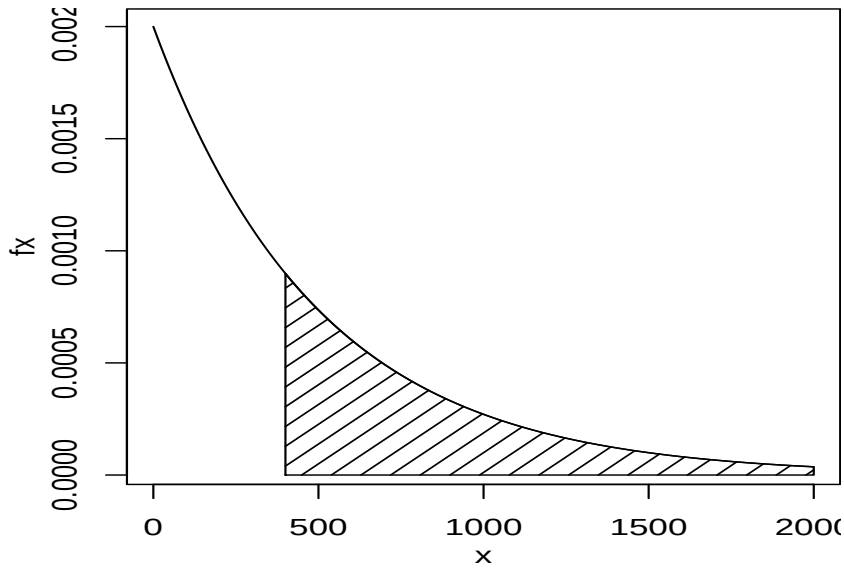

Figura 10: Função de densidade da $Exp(500)$ com a área correspondente à $P[X \geq 400]$.

8.3 Esperança e Variância

Sabemos que para a distribuição exponencial a esperança $E[X] = \int_0^\infty x f(x)dx = \lambda$ e a variância $Var[X] = \int_0^\infty (x - E[X])^2 f(x)dx = \lambda^2$ pois podem ser obtidos analiticamente.

Novamente para ilustrar o uso do R vamos "esquecer" que conhecemos estes resultados e vamos obtê-los numericamente. Para isto vamos definir funções para a esperança e variância e fazer a integração numérica.

```
e.exp <- function(x, lambda = 500){
   ex <- x * (1/lambda) * exp(-x/lambda)
   return(ex)
}

> integrate(e.exp, 0, Inf)
500 with absolute error < 0.00088

> ex <- integrate(e.exp, 0, Inf)$value
> ex
[1] 500
v.exp <- function(x, lambda = 500, exp.x){
  vx <- ((x-exp.x)^2) * (1/lambda) * exp(-x/lambda)
  return(vx)
}

> integrate(v.exp, 0, Inf, exp.x=ex)
250000 with absolute error < 6.9
```

8.4 Gerador de números aleatórios

A geração da amostra depende de um *gerador de números aleatórios* que é controlado por uma *semente* (*seed* em inglês). Cada vez que o comando `rnorm` é chamado diferentes elementos da amostra são produzidos, porque a *semente* do gerador é automaticamente modificada pela função. Em geral o usuário não precisa se preocupar com este mecanismo. Mas caso necessário a função `set.seed` pode ser usada para controlar o comportamento do gerador de números aleatórios. Esta função define o valor inicial da semente que é mudado a cada geração subsequente de números aleatórios. Portanto para gerar duas amostras idênticas basta usar o comando `set.seed` conforme ilustrado abaixo.

```
> set.seed(214)     # define o valor da semente
> rnorm(5)          # amostra de 5 elementos
[1] -0.46774980  0.04088223  1.00335193  2.02522505  0.30640096
> rnorm(5)          # outra amostra de 5 elementos
[1]  0.4257775  0.7488927  0.4464515 -2.2051418  1.9818137
> set.seed(214)     # retorna o valor da semente ao valor inicial
> rnorm(5)          # gera novamente a primeira amostra de 5 elementos
[1] -0.46774980  0.04088223  1.00335193  2.02522505  0.30640096
```

No comando acima mostramos que depois da primeira amostra ser retirada a semente é mudada e por isto os elementos da segunda amostra são diferentes dos da primeira. Depois retornamos a semente ao seu estado original a a próxima amostra tem portanto os mesmos elementos da primeira.

Para saber mais sobre geração de números aleatórios no R veja `help(.Random.seed)` e `help(set.seed)`

8.5 Argumentos vetoriais, reciclagem

As funções de probabilidades aceitam também vetores em seus argumentos conforme ilustrado nos exemplo abaixo.

```
> qnorm(c(0.05, 0.95))
[1] -1.644854  1.644854
> rnorm(4, mean=c(0, 10, 100, 1000))
[1]   0.1599628   9.0957340 100.5595095 999.9129392
> rnorm(4, mean=c(10, 20, 30, 40), sd=c(2, 5))
[1] 10.58318 21.92976 29.62843 42.71741
```

Note que no último exemplo a *lei da reciclagem* foi utilizada no vetor de desvios padrão, i.e. os desvios padrão utilizados foram $(2, 5, 2, 5)$.

8.6 Aproximação pela Normal

Nos livros texto de estatística voce vai ver que as distribuições binomial e Poisson podem ser aproximadas pela normal. Isto significa que podemos usar a distribuição normal para calcular probabilidades *aproximadas* em casos em que seria "trabalhoso" calcular as probabilidades *exatas* pala binomial ou Poisson. Isto é especialmente importante no caso de usarmos calculadoras e/ou tabelas para calcular probabilidades. Quando usamos um computador esta aproximação é menos importante, visto que é fácil calcular as probabilidades exatas com o auxílio do computador. De toda forma vamos ilustrar aqui este resultado.

Vejamos como fica a aproximação no caso da distribuição binomial. Seja $X \sim B(n, p)$. Na prática, em geral a aproximação é considerada aceitável quando $np \geq 5$ e $n(1-p) \geq 5$ e sendo tanto melhor quanto maior for o valor de n. A aproximação neste caso é de que $X \sim B(n, p) \approx N(np, np(1-p))$.

Seja $X \sim B(10, 1/2)$ e portanto com a aproximação $X \approx N(5, 2.5)$. A Figura 11 mostra o gráfico da distribuição binomial e da aproximação pela normal.

```
xb <- 0:10
px <- dbinom(xb, 10, 0.5)
plot(xb, px, type='h')

xn <- seq(0, 10, len=100)
fx <- dnorm(xn, 5, sqrt(2.5))
lines(xn, fx)
```

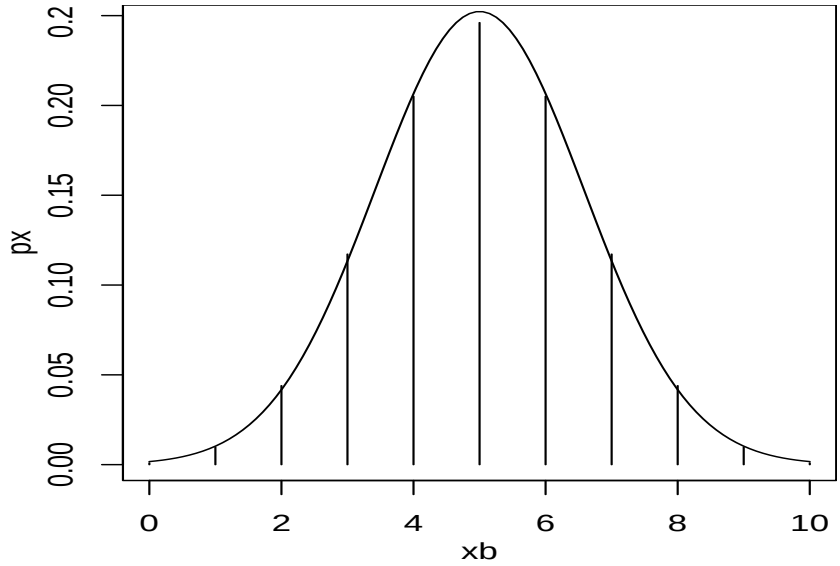

Figura 11: Função de probabilidade da $B(10, 1/2)$ e a aproximação pela $N(5, 2.5)$.

Vamos também calcular as seguintes probabilidades exatas e aproximadas, lembrando que ao usar a aproximação pela normal devemos usar a correção de continuidade e/ou somando e subtraindo 0.5 ao valor pedido.

- $P[X < 6]$
  Neste caso $P[X_B < 6] = P[X_B \leq 5] \approx P[X_N \leq 5.5]$

  ```
  > pbinom(5, 10, 0.5)
  [1] 0.6230469
  > pnorm(5.5, 5, sqrt(2.5))
  [1] 0.6240852
  ```

- $P[X \leq 6]$
  Neste caso $P[X_B \leq 6] \approx P[X_N \leq 6.5]$

  ```
  > pbinom(6, 10, 0.5)
  [1] 0.828125
  > pnorm(6.5, 5, sqrt(2.5))
  [1] 0.8286091
  ```

- $P[X > 2]$
  Neste caso $P[X_B > 2] = 1 - P[X_B \leq 2] \approx 1 - P[X_N \leq 2.5]$

  ```
  > 1-pbinom(2, 10, 0.5)
  [1] 0.9453125
  > 1-pnorm(2.5, 5, sqrt(2.5))
  [1] 0.9430769
  ```

- $P[X \geq 2]$
  Neste caso $P[X_B \geq 2] = 1 - P[X_B \leq 1] \approx P[X_N \leq 1.5]$

```
> 1-pbinom(1, 10, 0.5)
[1] 0.9892578
> 1-pnorm(1.5, 5, sqrt(2.5))
[1] 0.9865717
```

- $P[X = 7]$
  Neste caso $P[X_B = 7] \approx P[6.5 \leq X_N \leq 7.5]$

```
> dbinom(7, 10, 0.5)
[1] 0.1171875
> pnorm(7.5, 5, sqrt(2.5)) - pnorm(6.5, 5, sqrt(2.5))
[1] 0.1144677
```

- $P[3 < X \leq 8]$
  Neste caso $P[3 < X_B \leq 8] = P[X_B \leq 8] - P[X_B \leq 3] \approx P[X_N \leq 8.5] - P[X_N \leq 3.5]$

```
> pbinom(8, 10, 0.5) - pbinom(3, 10, 0.5)
[1] 0.8173828
> pnorm(8.5, 5, sqrt(2.5)) - pnorm(3.5, 5, sqrt(2.5))
[1] 0.8151808
```

- $P[1 \leq X \leq 5]$
  Neste caso $P[1 \leq X_B \leq 5] = P[X_B \leq 5] - P[X_B \leq 0] \approx P[X_N \leq 5.5] - P[X_N \leq 0.5]$

```
> pbinom(5, 10, 0.5) - pbinom(0, 10, 0.5)
[1] 0.6220703
> pnorm(5.5, 5, sqrt(2.5)) - pnorm(0.5, 5, sqrt(2.5))
[1] 0.6218719
```

8.7 Exercícios

1. (Bussab & Morettin, pag. 198, ex. 51)
   A função de densidade de probabilidade de distribuição Weibull é dada por:

$$f(x) = \begin{cases} \lambda\, x^{\lambda-1}\, e^{-x^{\lambda}} & \text{para } x \geq 0 \\ 0 & \text{para } x < 0 \end{cases}$$

   (a) Obter $E[X]$ para $\lambda = 2$. Obter o resultado analitica e computacionalmente.
   Dica: para resolver você vai precisar da definição da função Gama:

$$\Gamma(a) = \int_0^{\infty} x^{a-1}\, e^{-x} dx$$

   (b) Obter $E[X]$ para $\lambda = 5$.

   (c) Obter as probabilidades:
   - $P[X > 2]$
   - $P[1.5 < X < 6]$
   - $P[X < 8]$

9 Miscelânia de funcionalidades do R

9.1 O R como calculadora

Podemos fazer algumas operações matemáticas simples utilizando o R. Vejamos alguns exemplos calculando as seguinte somas:

(a) $10^2 + 11^2 + \ldots + 20^2$ Para obter a resposta devemos

- criar uma sequência de números de 10 a 20
- elevar ao quadrado cada valor deste vetor
- somar os elementos do vetor

E estes passos correspondem aos seguintes comandos

```
> (10:20)
> (10:20)^2
> sum((10:20)^2)
```

Note que só precisamos do último comando para obter a resposta, mas é sempre útil entender os comandos passo a passo!

(b) $\sqrt{\log(1)} + \sqrt{\log(10)} + \sqrt{\log(100)} + \ldots + \sqrt{\log(1000000)}$,
onde log é o logarítmo neperiano. Agora vamos resolver com apenas um comando:

```
> sum(sqrt(log(10^(0:6))))
```

9.2 Gráficos de funções

Para ilustrar como podemos fazer gráficos de funções vamos considerar cada uma das funções a seguir cujos gráficos são mostrados na Figura 12.

(a) $f(x) = 1 - \frac{1}{x} sin(x)$ para $0 \leq x \leq 50$

(b) $f(x) = \frac{1}{\sqrt{50\pi}} \exp[-\frac{1}{50}(x - 100)^2)]$ para $85 \leq x \leq 115$

A idéia básica é criar um vetor com valores das abscissas (valores de x) e calcular o valor da função (valores de $f(x)$) para cada elemento da função e depois fazer o gráfico unindo os pares de pontos. Vejamos os comandos para o primeiro exemplo.

```
> x1 <- seq(0,50, l=101)
> y1 <- 1 - (1/x1) * sin(x1)
> plot(x1, y1, type="l")
```

Note que este procedimento é o mesmo que aprendemos para fazer esboços de gráficos a mão em uma folha de papel!
Há uma outra maneira de fazer isto no R utilizando `plot.function` conforme pode ser visto no comando abaixo que nada mais faz que combinar os três comandos acima em apenas um.

```
> plot(function(x) 1 - (1/x) * sin(x), 0, 50)
```

Agora vamos ver o gráfico para o segundo exemplo.

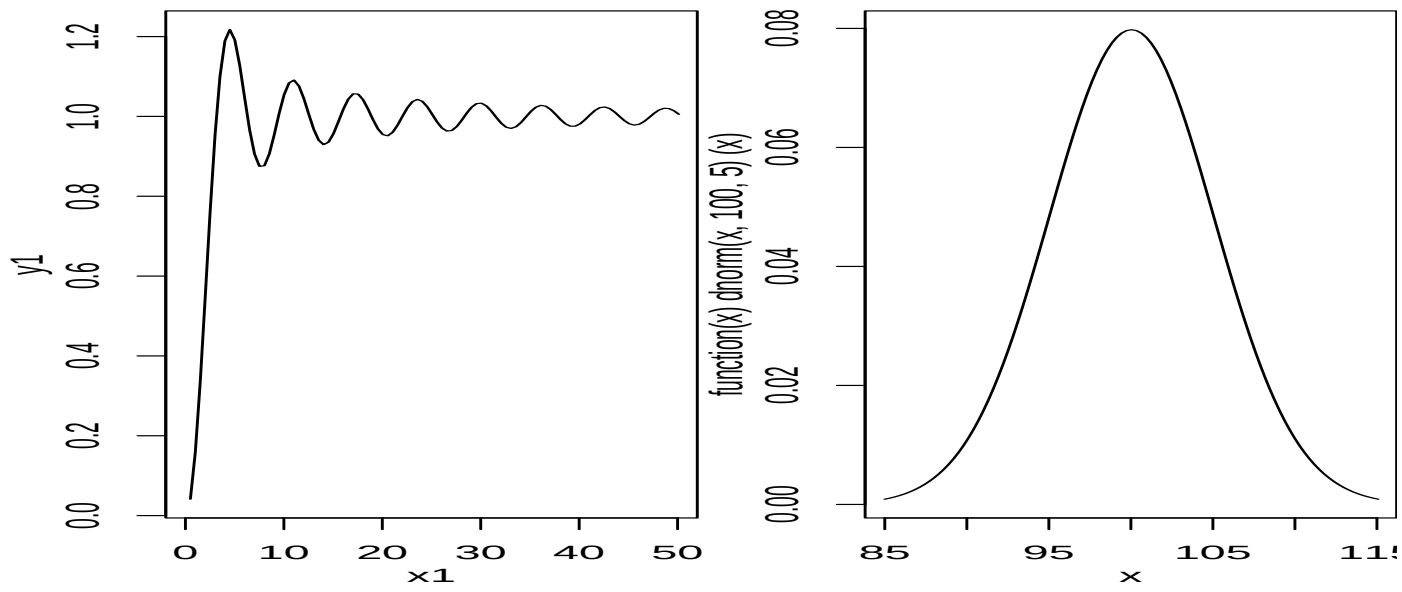

Figura 12: Gráficos das funções mostradas em (a) e (b).

```
> x2 <- seq(80, 120, l=101)
> y2 <- (1/sqrt(50*pi)) * exp(-0.02 * (x2-100)^2)
> plot(x2, y2, type="l")
```

Note que esta função é a densidade da distribuição normal o o gráfico pode também ser obtido com:

```
> y2 <- dnorm(x2, 100, 5)
> plot(x2, y2, type="l")
## ou ainda:
> plot(function(x) dnorm(x, 100, 5), 85, 115)
```

9.3 Integração numérica

A função `integrate` é usada para integração numérica em uma dimensão. Como exemplo vamos considerar resolver a seguinte integral:

$$I = \int_{-3}^{3} x^2 dx. \tag{2}$$

Para resolver a integral devemos criar uma *função* no R com a expressão da função que vamos integrar e esta deve ser passada para `integrate` conforme este exemplo:

```
> fx <- function(x) x^2
> integrate(fx, -3, 3)
18 with absolute error < 2e-13
```

A integral acima corresponde à área mostrada no gráfico da Figura 13.

Esta figura é obtida com os seguinte comandos:

```
> x <- seq(-4, 4, l=100)
> x2 <- x^2
> plot(x, x^2, ty='l')
> x <- seq(-3, 3, l=100)
> x2 <- x^2
> polygon(rbind(cbind(rev(x),0),cbind(x,x2)), col='gray')
```

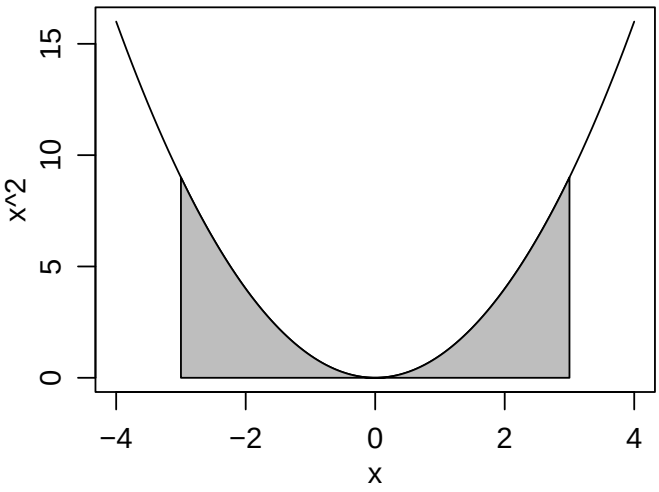

Figura 13: Gráfico onde a área indicada corresponde à integral definida na equação 2.

Vejamos mais um exemplo. Sabemos que para distribuições contínuas de probabilidades a integral está associada a probabilidade em um intervalo. Seja $f(x)$ uma f.d.p. de uma variável contínua, então $P(a < X < b) = \int_a^b f(x)dx$. Por exemplo, seja X v.a. com distribuição $N(100, 81)$ e portanto $f(x) = \frac{1}{9\sqrt{2\pi}} \exp\{-\frac{1}{162}(x-100)^2\}$. A probabilidade $P(85 < X < 105)$ pode ser calculada das três formas diferentes mostradas a seguir.

```
> fx <- function(x){(1/(9*sqrt(2*pi))) * exp(-(1/162)*(x-100)^2)}
> integrate(fx, 85, 105)
0.6629523 with absolute error < 7.4e-15

> integrate(function(x) dnorm(x, 100, 9), 85, 105)
0.6629523 with absolute error < 7.4e-15

> pnorm(105, 100, 9) - pnorm(85, 100, 9)
[1] 0.6629523
```

9.4 Criando vetores com elementos repetidos

As funções `rep` e `seq` do R são úteis para criar vetores de dados que seguem um certo padrão.
Clique aqui para ver um arquivo de dados.
vamos ver os comandos que podem ser usados para criar vetores para cada uma das três colunas iniciais deste arquivo.

```
## Primeira coluna
> rep(1:4, each=12)
## ou
> rep(1:4, rep(12,4))

## Segunda coluna
> rep(rep(1:3, each=4),4)

## Terceira coluna
> rep(1:4, 12)
```

9.5 Exercícios

1. Calcule o valor das expressões abaixo

   (a) Seja $x = (12, 11, 14, 15, 10, 11, 14, 11)$.
   Calcule $E = -n\lambda + (\sum_1^n x_i)\log(\lambda) - \sum_1^n \log(x_i!)$, onde n é o número de elementos do vetor x e $\lambda = 10$.
   Dica: o fatorial de um número pode ser obtido utilizando a função `prod`. Por exemplo o valor de 5! é obtido com o comando `prod(1:5)`.
   Há ainda uma outra forma usando a função Gama e lembrando que para a inteiro, $\Gamma(a+1) = a!$. Portanto podemos obter o valor de 5! com o comando `gamma(6)`.

   (b) $E = (\pi)^2 + (2\pi)^2 + (3\pi)^2 + ... + (10\pi)^2$

   (c) $E = \log(x+1) + \log(\frac{x+2}{2}) + \log(\frac{x+3}{3}) + \ldots + \log(\frac{x+20}{20})$, para $x = 10$

2. Obtenha o gráfico das seguintes funções:

   (a) $f(x) = x^{12}(1-x)^8$ para $0 < x < 1$

   (b) Para $\phi = 4$,

$$\rho(h) = \begin{cases} 1 - 1.5\frac{h}{\phi} + 0.5(\frac{h}{\phi})^3 \text{ , se } h < \phi \\ 0 \text{ , caso contrário} \end{cases}$$

3. Considerando as funções acima calcule as integrais a seguir e indique a área correspondente nos gráficos das funções.

   (a) $I_1 = \int_{0.2}^{0.6} f(x)dx$

   (b) $I_2 = \int_{1.5}^{3.5} \rho(h)dh$

4. Mostre os comandos para obter as seguintes sequências de números

   (a) 1 11 21 31 41 51 61 71 81 91

   (b) 1 1 2 2 2 2 2 3 3 3

   (c) 1.5 2.0 2.5 3.0 3.5 1.5 2.0 2.5 3.0 3.5 1.5 2.0 2.5 3.0 3.5

10 Intervalos de confiança – I

Nesta sessão vamos verificar como utilizar o R para obter intervalos de confiança para parâmetros de distribuições de probabilidade.

Para fins didáticos mostrando os recursos do R vamos mostrar três possíveis soluções:

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

10.1 Média de uma distribuição normal com variância desconhecida

Considere 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

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 variância desconhecida, para uma amostra de tamanho n é dado por:

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

onde t_t é o quantil de ordem $1 - \alpha/2$ da distribuição t de Student, com $n - 1$ graus de liberdade.

Vamos agora obter a resposta das três formas diferentes mencionadas acima.

10.1.1 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 distribuição t, para obter um IC a 95% de confiança.

```
> 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
```

10.1.2 Escrevendo uma função

Podemos generalizar a solução acima agrupando os comandos em uma função. Nos comandos primeiro definimos a função e a seguir utilizamos a função criada definindo intervalos a 95% e 99%.

```
> ic.m <- function(x, conf = 0.95){
+   n <- length(x)
+   media <- mean(x)
+   variancia <- var(x)
+   quantis <- qt(c((1-conf)/2, 1 - (1-conf)/2), df = n-1)
+   ic <- media + quantis * sqrt(variancia/n)
+   return(ic)
+ }
> ic.m(tempo)
[1] 4.278843 5.211157

> ic.m(tempo, conf=0.99)
[1] 4.107814 5.382186
```

Escrever uma função é particularmente útil quando um procedimento vai ser utilizados várias vezes.

10.1.3 Usando a função `t.test`

Mostramos as soluções 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. Neste caso 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
```

10.2 Exercícios

Em cada um dos exercícios abaixo tente obter os intervalos das três formas mostradas acima.

1. Pretende-se estimar a proporção p de cura, através de uso de um certo medicamento em doentes contaminados com cercária, que é uma das formas do verme da esquitosomose. Um experimento consistiu em aplicar o medicamento em 200 pacientes, escolhidos ao acaso, e observar que 160 deles foram curados. Montar o intervalo de confiança para a proporção de curados.
   Note que há duas expressões possíveis para este IC: o "otimista"e o "conservativo". Encontre ambos intervalos.

2. Os dados abaixo são uma amostra aleatória da distribuição $Bernoulli(p)$. Obter IC's a 90% e 99%.

```
0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1
```

3. Encontre intervalos de confiança de 95% para a média de uma distribuição Normal com variância 1 dada a amostra abaixo

```
9.5 10.8 9.3 10.7 10.9 10.5 10.7 9.0 11.0 8.4
10.9 9.8 11.4 10.6 9.2 9.7 8.3 10.8 9.8 9.0
```

4. Queremos verificar se duas máquinas produzem peças com a mesma homogeneidade quanto a resistência à tensão. Para isso, sorteamos dias amostras de 6 peças de cada máquina, e obtivemos as seguintes resistências:

Máquina A	145	127	136	142	141	137
Máquina B	143	128	132	138	142	132

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

11 Testes de hipótese

Os exercícios abaixo são referentes ao conteúdo de *Testes de Hipóteses* conforme visto na disciplina de Estatística Geral II.

Eles devem ser resolvidos usando como referência qualquer texto de Estatística Básica.
Procure resolver primeiramente sem o uso de programa estatístico.

A idéia é relembrar como são feitos alguns testes de hipótese básicos e corriqueiros em estatística.

Nesta sessão vamos verificar como utilizar o R para fazer teste de hipóteses sobre parâmetros de distribuições para as quais os resultados são bem conhecidos.

Os comandos e cálculos são bastante parecidos com os vistos em intervalos de confiança e isto nem poderia ser diferente visto que intervalos de confiança e testes de hipótese são relacionados.

Assim como fizemos com intervalos de confiança, aqui sempre que possível e para fins didáticos mostrando os recursos do R vamos mostrar três possíveis soluções:

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

11.1 Comparação de variâncias de uma distribuição normal

Queremos verificar se duas máquinas produzem peças com a mesma homogeneidade quanto a resistência à tensão. Para isso, sorteamos dias amostras de 6 peças de cada máquina, e obtivemos as seguintes resistências:

Máquina A	145	127	136	142	141	137
Máquina B	143	128	132	138	142	132

O que se pode concluir fazendo um teste de hipótese adequado?

Solução:

Da teoria de testes de hipótese sabemos que, assumindo a distribuição normal, o teste para a hipótese:

$$H_0 : \sigma_A^2 = \sigma_B^2 \quad versus \quad H_a : \sigma_A^2 \neq \sigma_B^2$$

que é equivalente à

$$H_0 : \frac{\sigma_A^2}{\sigma_B^2} = 1 \quad versus \quad H_a : \frac{\sigma_A^2}{\sigma_B^2} \neq 1$$

é feito calculando-se a estatística de teste:

$$F_{calc} = \frac{S_A^2}{S_B^2}$$

e em seguida comparando-se este valor com um valor da tabela de F e/ou calculando-se o p-valor associado com $n_A - 1$ e $n_B - 1$ graus de liberdade. Devemos também fixar o nível de significância do teste, que neste caso vamos definir como sendo 5%.

Para efetuar as análises no R vamos primeiro entrar com os dados nos objetos que vamos chamar de `ma` e `mb` e calcular os tamanhos das amostras que vão ser armazenados nos objetos `na` e `nb`.

```
> ma <- c(145, 127, 136, 142, 141, 137)
> na <- length(ma)
> na
[1] 6
```

```
> mb <- c(143, 128, 132, 138, 142, 132)
> nb <- length(mb)
> nb
[1] 6
```

11.1.1 Fazendo as contas passo a passo

Vamos calcular a estatística de teste. Como temos o computador a disposição não precisamos de da tabela da distribuição F e podemos calcular o p-valor diretamente.

```
> ma.v <- var(ma)
> ma.v
[1] 40
> mb.v <- var(mb)
> mb.v
[1] 36.96667
> fcalc <- ma.v/mb.v
> fcalc
[1] 1.082056
> pval <- 2 * pf(fcalc, na-1, nb-1, lower=F)
> pval
[1] 0.9331458
```

No cálculo do P-valor acima multiplicamos o valor encontrado por 2 porque estamos realizando um teste bilateral.

11.1.2 Escrevendo uma função

Esta fica por sua conta!
Escreva a sua própria função para testar hipóteses sobre variâncias de duas distribuições normais.

11.1.3 Usando uma função do R

O R já tem implementadas funções para a maioria dos procedimentos estatísticos "usuais". Por exemplo, para testar variâncias neste exemplo utilizamos a função `var.test`. Vamos verificar os argumentos da função.

```
> args(var.test)
function (x, ...)
NULL
```

Note que esta saída não é muito informativa. Este tipo de resultado indica que `var.test` é um método com mais de uma função associada. Portanto devemos pedir os argumentos da função "default".

```
> args(var.test.default)
function (x, y, ratio = 1, alternative = c("two.sided", "less",
    "greater"), conf.level = 0.95, ...)
NULL
```

Neste argumentos vemos que a função recebe dois vertores de de dados (x e y), que por "default" a hipótese nula é que o quociente das variâncias é 1 e que a alternativa pode ser bilateral ou unilateral. Como `''two.sided''` é a primeira opção o "default" é o teste bilateral. Finalmente o nível de confiança é 95% ao menos que o último argumento seja modificado pelo usuário. Para aplicar esta função nos nossos dados basta digitar:

```
> var.test(ma, mb)

        F test to compare two variances

data:  ma and mb
F = 1.0821, num df = 5, denom df = 5, p-value = 0.9331
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1514131 7.7327847
sample estimates:
ratio of variances
          1.082056
```

e note que a saída inclui os resultados do teste de hipótese bem como o intervalo de confiança. A decisão baseia-se em verificar se o P-valor é menor que o definido inicialmente.

11.2 Exercícios

Os exercícios a seguir foram retirados do libro de Bussab & Morettin.

Note que nos exercícios abaixo nem sempre voce poderá usar funções de teste do R porque em alguns casos os dados brutos não estão disponíveis. Nestes casos voce deverá fazer os cálculos usando o R como calculadora.

1. Uma máquina automática de encher pacotes de café enche-os segundo uma distribuição normal, com média μ e variância $400g^2$. O valor de μ pode ser fixado num mostrador situado numa posição um pouco inacessível dessa máquina. A máquina foi regulada para $\mu = 500g$. Desejamos, de meia em meia hora, colher uma amostra de 16 pacotes e verificar se a produção está sob controle, isto é, se $\mu = 500g$ ou não. Se uma dessas amostras apresentasse uma média $\bar{x} = 492g$, voce pararia ou não a produção para verificar se o mostrador está na posição correta?

2. Uma companhia de cigarros anuncia que o índice médio de nicotina dos cigarros que fabrica apresenta-se abaixo de $23mg$ por cigarro. Um laboratório realiza 6 análises desse índice, obtendo: 27, 24, 21, 25, 26, 22. Sabe-se que o índice de nicotina se distribui normalmente, com variância igual a $4,86mg^2$. Pode-se aceitar, ao nível de 10%, a afirmação do fabricante.

3. Uma estação de televisão afirma que 60% dos televisores estavam ligados no seu programa especial de última segunda feira. Uma rede competidora deseja contestar essa afirmação, e decide, para isso, usar uma amostra de 200 famílias obtendo 104 respostas afirmativas. Qual a conclusão ao nível de 5% de significância?

4. O tempo médio, por operário, para executar uma tarefa, tem sido 100 minutos, com um desvio padrão de 15 minutos. Introduziu-se uma modificação para diminuir esse tempo, e, após certo período, sorteou-se uma amostra de 16 operários, medindo-se o tempo de execução de cada um. O tempo médio da amostra foi de 85 minutos, o o desvio padrão foi 12 minutos. Estes resultados trazem evidências estatísticas da melhora desejada?

5. Num estudo comparativo do tempo médio de adaptação, uma amostra aleatória, de 50 homens e 50 mulheres de um grande complexo industrial, produziu os seguintes resultados:

Estatísticas	Homens	Mulheres
Médias	3,2 anos	3,7 anos
Desvios Padrões	0,8 anos	0,9 anos

Pode-se dizer que existe diferença significativa entre o tempo de adaptação de homens e mulheres? A sua conclusão seria diferente se as amostras tivessem sido de 5 homens e 5 mulheres?

12 Explorando distribuições de probabilidade empíricas

Na Sessão 6 vimos com usar distribuições de probabilidade no R. Estas distribuições tem expressões conhecidas e são indexadas por um ou mais parâmetros. Portanto, conhecer a distribuição e seu(s) parâmetro(s) é suficiente para caracterizar completamente o comportamento distribuição e extrair resultados de interesse.

Na prática em estatística em geral somente temos disponível uma amostra e não conhecemos o mecanismo (distribuição) que gerou os dados. Muitas vezes o que se faz é: (i) assumir que os dados são provenientes de certa distribuição, (ii) estimar o(s) parâmetro(s) a partir dos dados. Depois disto procura-se verificar se o ajuste foi "bom o suficiente", caso contrário tenta-se usar uma outra distribuição e recomeça-se o processo.

A necessidade de estudar fenômenos cada vez mais complexos levou ao desenvolvimento de métodos estatísticos que às vezes requerem um flexibilidade maior do que a fornecida pelas distribuições de probabilidade de forma conhecida. Em particular, métodos estatísticos baseados em simulação podem gerar amostras de quantidades de interesse que não seguem uma distribuição de probabilidade de forma conhecida. Isto ocorre com frequência em métodos de *inferência Bayesiana* e métodos computacionalmente intensivos como *bootstrap*, *testes Monte Carlo*, dentre outros.

Nesta sessão vamos ver como podemos, a partir de um conjunto de dados explorar os possíveis formatos da distribuição geradora sem impor nenhuma forma paramétrica para função de densidade.

12.1 Estimação de densidades

A estimação de densidades é implementada no R pela função `density`. O resultado desta função é bem simples e claro: ela produz uma função de densidade obtida a partir dos dados sem forma paramétrica conhecida. Veja este primeiro exemplo que utiliza o conjunto de dados `precip` que já vem com o R e contém valores médios de precipitação em 70 cidades americanas. Nos comandos a seguir vamos carregar o conjunto de dados, fazer um histograma de frequências relativas e depois adicionar a este histograma a linha de densidade estimada, conforma mostra a Figura 14.

```
> data(precip)
> hist(precip, prob=T)
> precip.d <- density(precip)
> lines(precip.d)
```

Portanto podemos ver que a função `density` "suaviza" o histograma, capturando e concentrando-se nos principais aspectos dos dados disponíveis. Vamos ver na Figura 15 uma outra forma de visualizar os dados e sua densidade estimada, agora sem fazer o histograma.

```
> plot(precip.d)
> rug(precip)
```

Embora os resultados mostrados acima seja simples e fáceis de entender, há muita coisa por trás deles! Não vamos aqui estudar com detalhes esta função e os fundamentos teóricos nos quais se baseiam esta implementação computacional pois isto estaria muito além dos objetivos e escopo deste curso. Vamos nos ater às informações principais que nos permitam compreender o básico necessário sobre o uso da função. Para maiores detalhes veja as referências na documentação da função, que pode ser vista digitando `help(density)`

Basicamente, a função `density` produz o resultado visto anteriormente criando uma sequência de valores no eixo-X e estimando a densidade em cada ponto usando os dados ao redor deste ponto. Podem ser dados pesos aos dados vizinhos de acordo com sua proximidade ao ponto a ser estimado. Vamos examinar os argumentos da função.

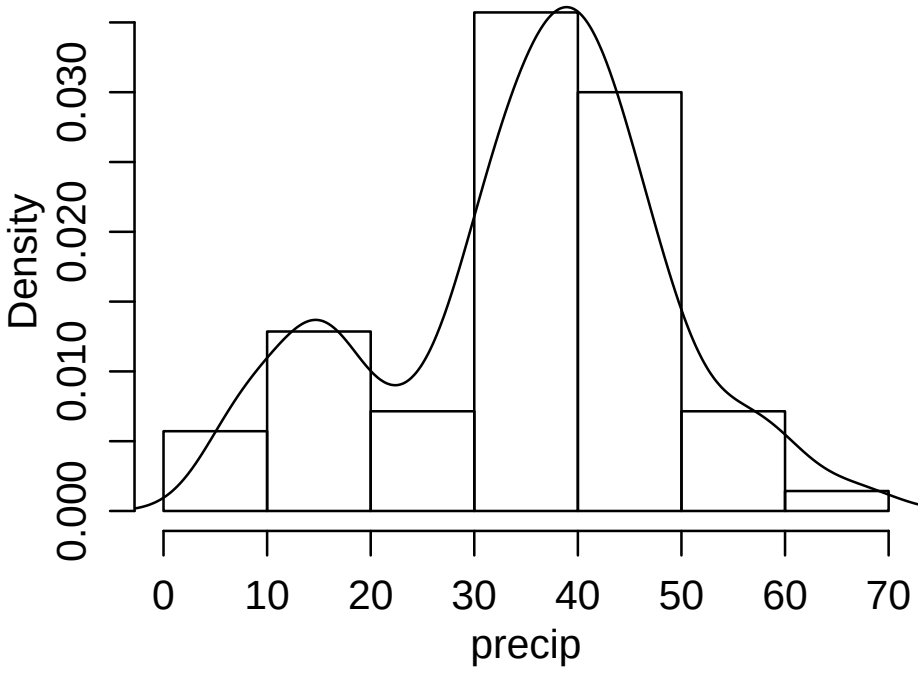

Figura 14: Histograma para os dados `precip` e a densidade estimada usando a função `density`.

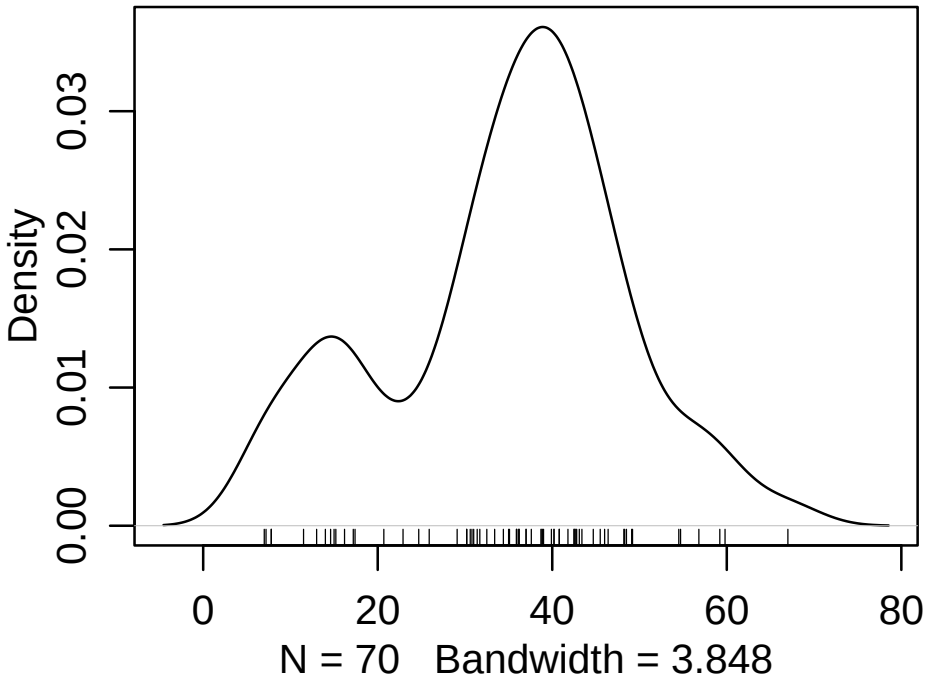

Figura 15: Dados `precip` e a densidade estimada usando a função `density`.

```
> args(density)
  function (x, bw = "nrd0", adjust = 1, kernel = c("gaussian",
      "epanechnikov", "rectangular", "triangular", "biweight",
      "cosine", "optcosine"), window = kernel, width, give.Rkern = FALSE,
      n = 512, from, to, cut = 3, na.rm = FALSE)
```

Os dois argumentos chave são portanto `bw` e `kernel` que controlam a distância na qual se procuram vizinhos e o peso a ser dado a cada vizinho, respectivamente. Para ilustrar isto vamos experimentar a função com diferentes valores para o argumento `bw`. Os resultados estão na Figura 16. Podemos notar que o grau de suavização aumenta a medida de aumentamos os valores deste argumento e as densidades estimadas podem ser bastante diferentes!

```
plot(density(precip, bw=1), main='')
rug(precip)
lines(density(precip, bw=5), lty=2)
lines(density(precip, bw=10), lty=3)
legend(5, 0.045, c('bw=1', 'bw=5', 'bw=10'), lty=1:3)
```

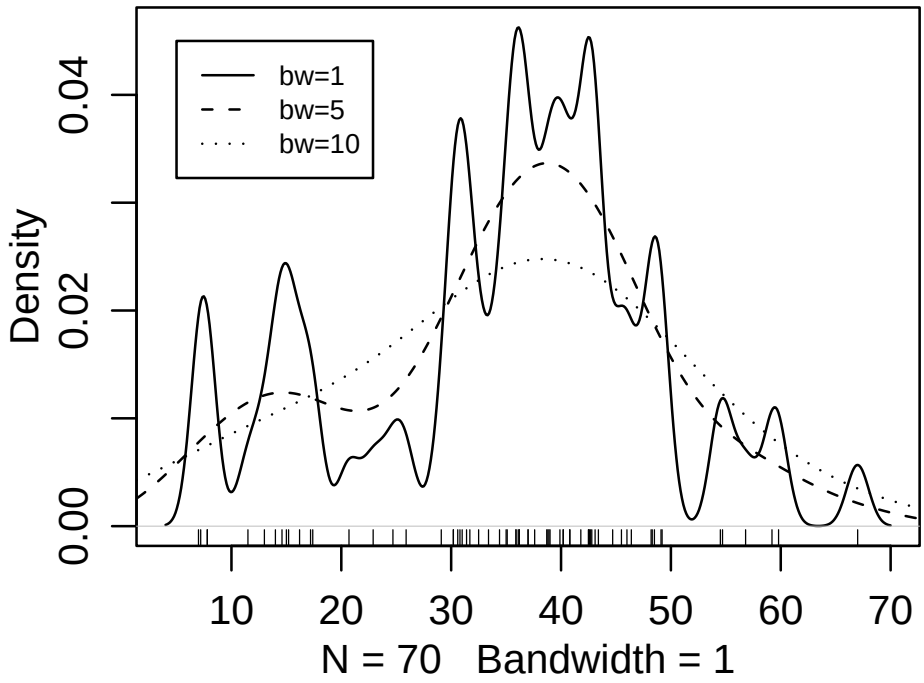

Figura 16: Densidade estimada usando a função `density` com diferentes valores para o argumento `bw`.

O outro argumento importante é tipo de função de pesos, ao que chamamos de núcleo (*kernel*). O R implementa vários núcleos diferentes cujos formatos são mostrados na Figura 17.

```
(kernels <- eval(formals(density)$kernel))
plot (density(0, bw = 1), xlab = "", main="kernels com bw = 1")
for(i in 2:length(kernels))
   lines(density(0, bw = 1, kern =  kernels[i]), col = i)
legend(1.5,.4, legend = kernels, col = seq(kernels),
       lty = 1, cex = .8, y.int = 1)
```

Utilizando diferentes núcleos no conjunto de dados `precip` obtemos os resultados mostrados na Figura 18. Note que as densidades estimadas utilizando os diferentes núcleos são bastante similares!

```
plot(density(precip), main='')
rug(precip)
lines(density(precip, ker='epa'), lty=2)
lines(density(precip, ker='rec'), col=2)
lines(density(precip, ker='tri'), lty=2, col=2)
```

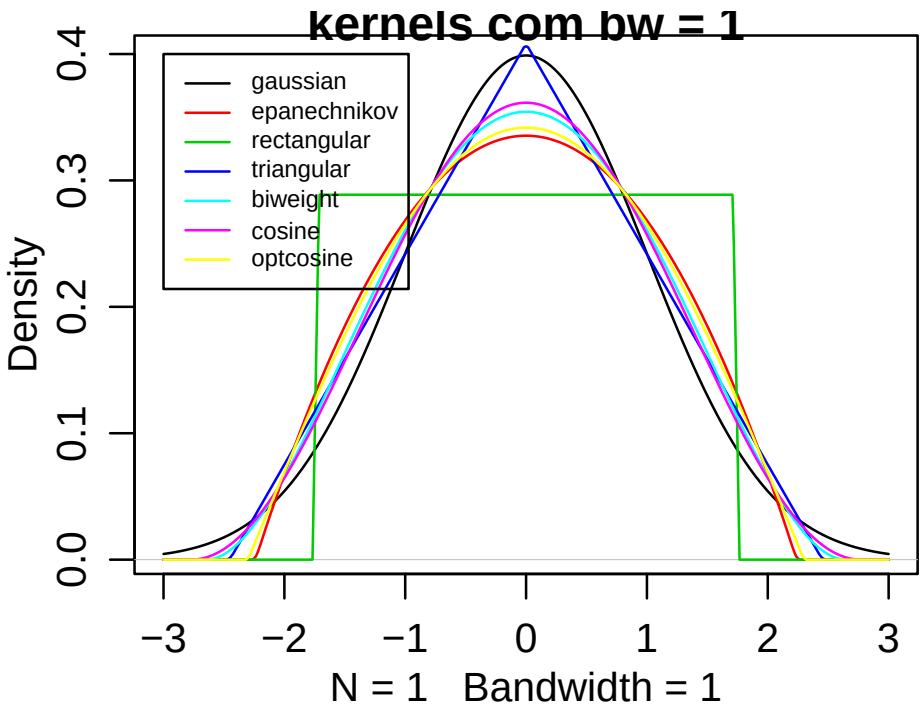

Figura 17: Diferentes núcleos implementados pela função `density`.

```
lines(density(precip, ker='biw'), col=3)
lines(density(precip, ker='cos'), lty=3, col=3)
legend(0, 0.035,
       legend = c("gaussian", "epanechnikov", "rectangular",
         "triangular", "biweight", "cosine"),
       lty = rep(1:2,3), col = rep(1:3, each=2))
```

Portanto, inspecionando os resultados anteriores podemos concluir que a *largura de banda* (*bandwidth* – `bw`) é o que mais influencia a estimação de densidade, isto é, é o argumento mais importante. O tipo de núcleo (`kernel`) é de importância secundária.

Bem, a esta altura voce deve estar se perguntando: mas como saber qual a largura de banda adequada? A princípio podemos tentar diferentes valores no argumento `bw` e inspecionar os resultados. O problema é que esta escolha é subjetiva. Felizmente para nós vários autores se debruçaram sobre este problema e descobriram métodos automáticos de seleção que que comportam bem na maioria das situações práticas. Estes métodos podem ser especificados no mesmo argumento `bw`, passando agora para este argumento caracteres que identificam o valor, ao invés de um valor numérico. No comando usado no início desta sessão onde não especificamos o argumento `bw` foi utilizado o valor "default" que é o método `nrd0` que implementa a regra prática de Silverman. Se quisermos mudar isto para o método de Sheather & Jones podemos fazer como nos comandos abaixo que produzem o resultado mostrado na Figura 19.

```
precip.dSJ <- density(precip, bw='sj')
plot(precip.dSJ)
rug(precip)
```

Os detalhes sobre os diferentes métodos implementados estão na documentação da função `bw.nrd`. Na Figura 20 ilustramos resultados obtidos com os diferentes métodos.

```
data(precip)
plot(density(precip, n=1000))
```

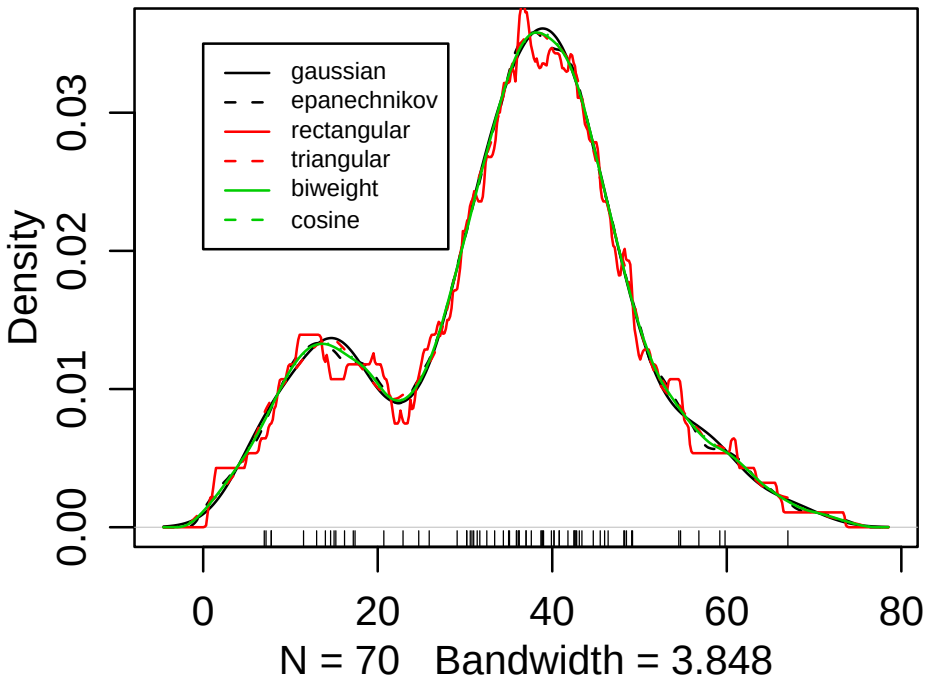

Figura 18: Densidade estimada usando a função `density` com diferentes valores para o argumento `kernel`.

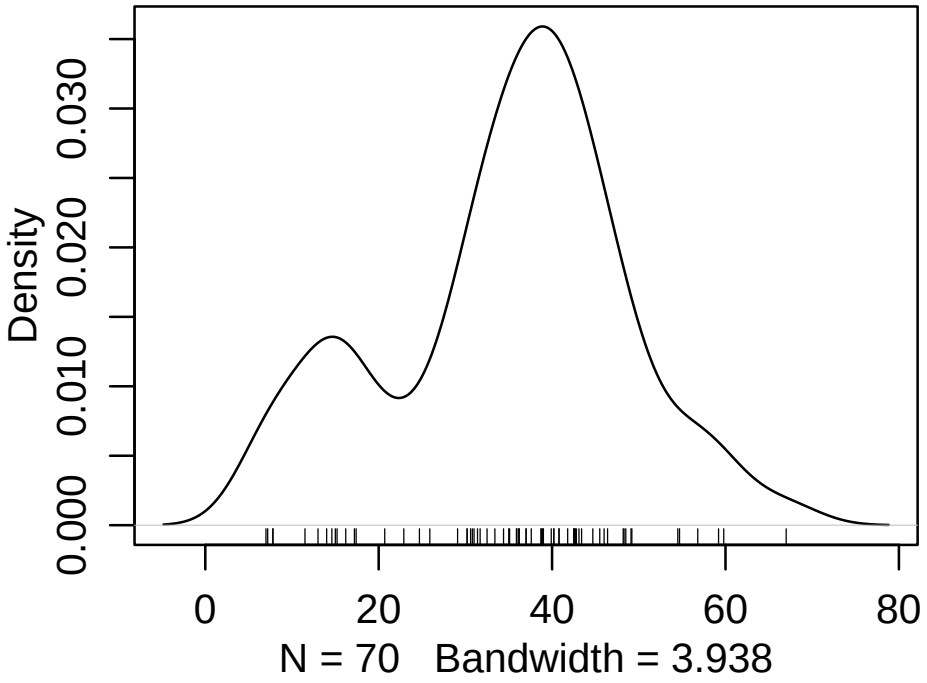

Figura 19: Densidade estimada para os dados `precip` usando a função `density` com critério de Sheather & Jones para seleção da largura de banda.

```
rug(precip)
lines(density(precip, bw="nrd"), col = 2)
lines(density(precip, bw="ucv"), col = 3)
lines(density(precip, bw="bcv"), col = 4)
lines(density(precip, bw="SJ-ste"), col = 5)
```

```
lines(density(precip, bw="SJ-dpi"), col = 6)
legend(55, 0.035,
       legend = c("nrd0", "nrd", "ucv", "bcv", "SJ-ste", "SJ-dpi"),
       col = 1:6, lty = 1)
```

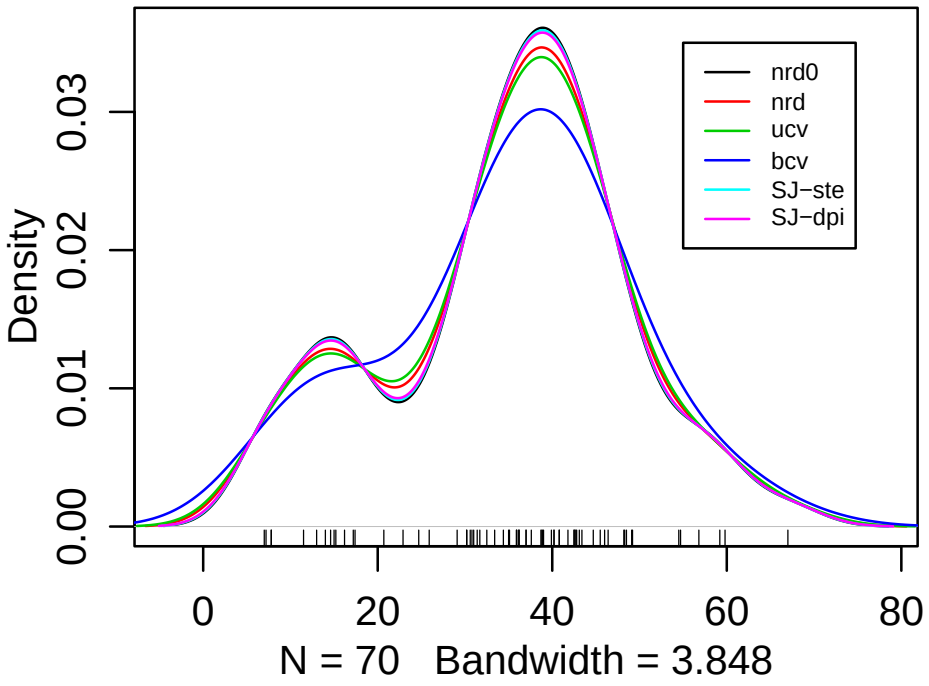

Figura 20: Diferentes métodos para largura de banda implementados pela função `density`.

12.2 Exercícios

1. Carregar o conjunto de dados `Faithful` e obter estimação de densidade para as variáveis 'tempo de erupção' e 'duração da erupção.

2. Carregar o conjunto `airquality` e densidades estimadas para as 4 variáveis medidas neste conjunto de dados.

3. Rodar e estudar os exemplos da sessão *examples* da documentação da função `density`.

13 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)
```

14 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 3: 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)
```

15 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 também 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 21.

```
> 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

```
> 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
```

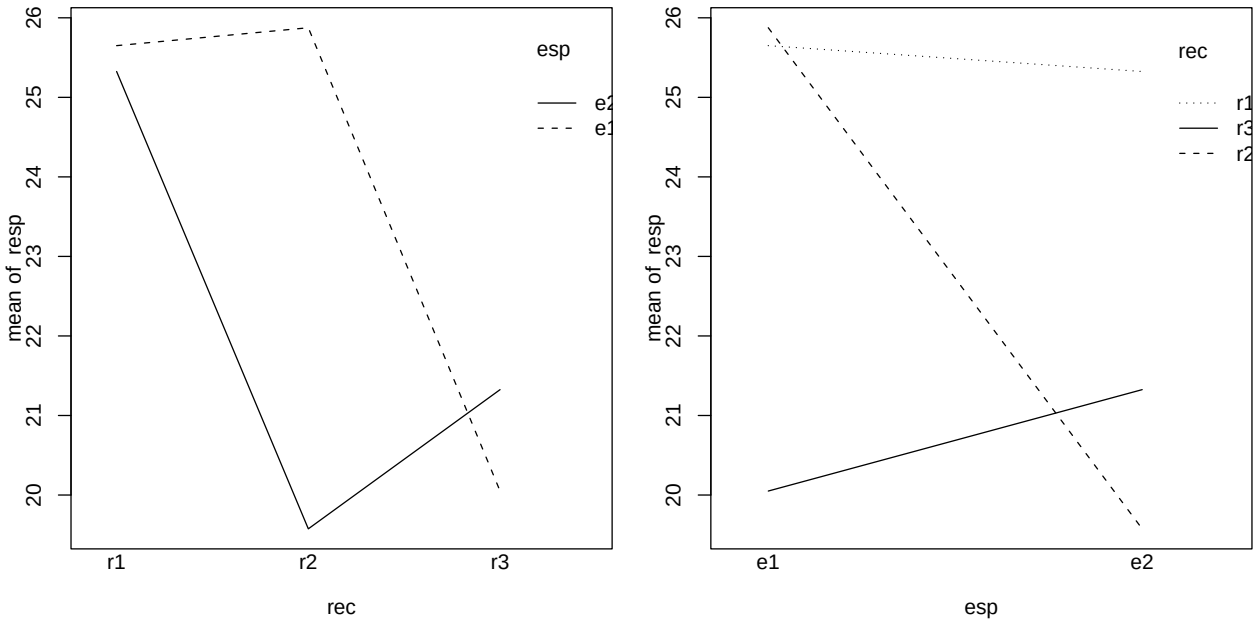

Figura 21: Gráficos de interação entre os fatores.

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 22 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)

> 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
```

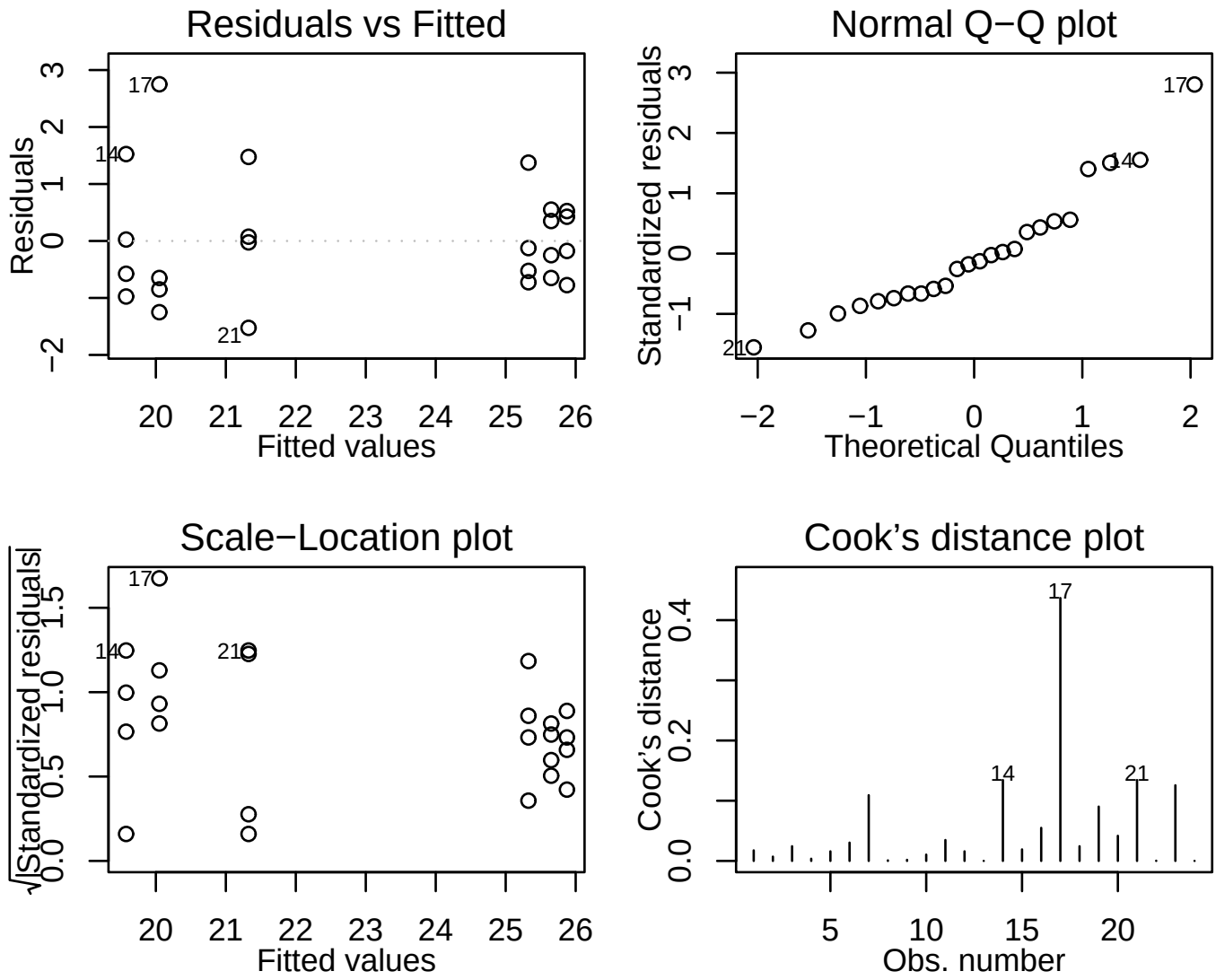

Figura 22: Gráficos de resíduos produzidos automaticamente pelo R.

```
> 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
```

16 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 4: 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)
```

17 Experimentos com fatores hierárquicos

Vamos considerar o exemplo da apostila retirado do livro de Montgomery. Clique aqui para ver e copiar o arquivo com conjunto de dados para sua área de trabalho.

O experimento estuda a variabilidade de lotes e fornecedores na puraza da matéria prima. A análise assume que os fornecedores são um efeito fixo enquanto que lotes são efeitos aleatórios.

Inicialmente vamos ler (importar) os dados para R com o comando `read.table` (certifique-se que o arquivo `exemplo06.txt` está na sua área de trabalho ou coloque o caminho do arquivo no comando abaixo). A seguir vamos examinar o objeto que contém os dados.

```
> ex06 <- read.table("exemplo06.txt", header=T)
> ex06

> dim(ex06)
[1] 36  3

> names(ex06)
[1] "forn" "lot"  "resp"

> is.factor(ex06$forn)
[1] FALSE
> is.factor(ex06$lot)
[1] FALSE

> ex06$forn <- as.factor(ex06$forn)
> ex06$lot <- as.factor(ex06$lot)

> is.factor(ex06$resp)
[1] FALSE
> is.numeric(ex06$resp)
[1] TRUE

> summary(ex06)
 forn   lot        resp
 1:12   1:9   Min.   :-4.0000
 2:12   2:9   1st Qu.:-1.0000
 3:12   3:9   Median : 0.0000
        4:9   Mean   : 0.3611
              3rd Qu.: 2.0000
              Max.   : 4.0000
```

Nos comandos acima verificamos que o objeto `ex06` possui 36 linhas correspondentes às observações e 3 colunas que correspondem às variáveis `forn` (fornecedor), `lot` (lote) e `resp` (a variável resposta).

A seguir verificamos que `forn` e `lot` não foram lidas como fatores. **NÃO** podemos seguir as análise desta forma pois o R leria os valores destas variáveis como quantidades numéricas e não como indicadores dos nívies dos fatores. Para corrigir isto usamos o comando `as.factor` para indicar ao R que estas variáveis são fatores.

Finalmente verificamos que a variável resposta é numérica e produzimos um rápido resumo dos dados.

Na sequência deveríamos fazer uma análise exploratória, alguns gráficos descritivos etc, como na análise dos experimentos mostrados anteriormente. Vamos deixar isto por conta do leitor e passar direto para a análise de variância.

A notação para indicar efeitos aninhados no modelo é `/`. Desta forma poderíamos ajustar o modelo da seguinte forma:

```
> ex06.av <- aov(resp ~ forn/lot, data=ex06)
> summary(ex06.av)
            Df Sum Sq Mean Sq F value  Pr(>F)
forn         2 15.056   7.528  2.8526 0.07736 .
forn:lot     9 69.917   7.769  2.9439 0.01667 *
Residuals   24 63.333   2.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Embora os elementos do quadro de análise de variância estejam corretos o teste F para efeito dos fornecedores está **ERRADO**. A análise acima considerou todos os efeitos como fixos e portanto dividiu os quadrados médios dos efeitos pelo quadrado médio do resíduo. Como `lotes` é um efeito aleatório deveríamos dividir o quadrado médio de to termo `lot` pelo quadrado médio de `forn:lot`

Uma forma de indicar a estrutura hierárquica ao R é especificar o modelo de forma que o termo de resíduo seja dividido de maneira adequada. Veja o resultado abaixo.

```
> ex06.av1 <- aov(resp ~ forn/lot + Error(forn) , data=ex06)
> summary(ex06.av1)

Error: forn
     Df  Sum Sq Mean Sq
forn  2 15.0556  7.5278

Error: Within
          Df Sum Sq Mean Sq F value  Pr(>F)
forn:lot   9 69.917   7.769  2.9439 0.01667 *
Residuals 24 63.333   2.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Agora o teste F errado não é mais mostrado, mas o teste correto também não foi feito! Isto não é problema porque podemos extrair os elementos que nos interessam e fazer o teste desejado. Primeiro vamos guardar verificamos que o comando `anova` produz uma lista que tem entre seus elementos os graus de liberdade `Df` e os quadrados médios (`Mean Sq`. A partir destes elementos podemos obtemos o valor da estatística F e o valor P associado.

```
> ex06.anova <- anova(ex06.av)
> is.list(ex06.anova)
[1] TRUE

> names(ex06.anova)
[1] "Df"      "Sum Sq"  "Mean Sq" "F value" "Pr(>F)"

> ex06.anova$Df
 1  2
 2  9 24
> ex06.anova$Mean
       1        2
7.527778 7.768519 2.638889
```

```
> Fcalc <- ex06.anova$Mean[1]/ex06.anova$Mean[2]
> Fcalc
        1
0.9690107
> pvalor <- 1 - pf(Fcalc, ex06.anova$Df[1], ex06.anova$Df[2])
> pvalor
        1
0.4157831
```

USANDO O PACOTE NLME

Uma outra possível e elegante solução no R para este problema é utilizar a função `lme` do pacote `nlme`. Note que a abordagem do problema por este pacote é um pouco diferente da forma apresentada no curso por se tratar de uma ferramente geral para modelos com efeitos aleatórios. Entretanto os todos os elementos relevantes da análise estão incluídos nos resultados. Vamos a seguir ver os comandos necessários comentar os resultados.

Inicialmente temos que carregar o pacote `nlme` com o comando `require`.
A seguir criamos uma variável para indicar o efeito aleatório que neste exemplo chamamos de `ex06$fa` utilizando a função `getGroups`.
Feito isto podemos rodar a função `lme` que faz o ajuste do modelo.

```
> require(nlme)
[1] TRUE
> ex06$fa <- getGroups(ex06, ~ 1|forn/lot, level=2)
> ex06.av <- lme(resp ~ forn, random=~1|fa, data=ex06)
> ex06.av
Linear mixed-effects model fit by REML
  Data: ex06
  Log-restricted-likelihood: -71.42198
  Fixed: resp ~ forn
(Intercept)       forn2       forn3
 -0.4166667   0.7500000   1.5833333

Random effects:
 Formula: ~1 | fa
        (Intercept) Residual
StdDev:    1.307561 1.624483

Number of Observations: 36
Number of Groups: 12
```

Este modelo tem a variável `forn` como efeito fixo e a variável `lot` como efeito aleatório com o componente de variância σ^2_{lote}. Além disto temos a variância residual σ^2. A saída acima mostra as estimativas destes componentes da variância como sendo $\hat{\sigma}^2_{lote} = (1.307)^2 = 1.71$ e $\hat{\sigma}^2 = (1.624)^2 = 2.64$.

O comando `anova` vai mostrar a análise de variância com apenas os efeitos principais. O fato do programa não incluir o efeito aleatório de lotes na saída não causa problema algum. O comando `intervals` mostra os intervalos de confiança para os componentes de variância. Portanto para verificar a significância do efeito de lotes basta ver se o intervalo para este componente de variância exclui o valor 0, o que é o caso neste exemplo conforme vamos abaixo.

```
> anova(ex06.av)
            numDF denDF   F-value p-value
```

```
(Intercept)     1    24 0.6043242  0.4445
forn            2     9 0.9690643  0.4158
> intervals(ex06.av)
Approximate 95% confidence intervals

 Fixed effects:
                 lower       est.    upper
(Intercept) -2.0772277 -0.4166667 1.243894
forn2       -1.8239746  0.7500000 3.323975
forn3       -0.9906412  1.5833333 4.157308

 Random Effects:
  Level: fa
                    lower     est.    upper
sd((Intercept)) 0.6397003 1.307561 2.672682

 Within-group standard error:
   lower     est.    upper
1.224202 1.624483 2.155644
```

Finalmente uma versão mais detalhada dos resultados pode ser obtida com o comando `summary`.

```
> summary(ex06.av)
Linear mixed-effects model fit by REML
 Data: ex06
       AIC      BIC    logLik
  152.8440 160.3265 -71.42198

Random effects:
 Formula: ~1 | fa
        (Intercept) Residual
StdDev:    1.307561 1.624483

Fixed effects: resp ~ forn
                 Value Std.Error DF    t-value p-value
(Intercept) -0.4166667 0.8045749 24 -0.5178718  0.6093
forn2        0.7500000 1.1378407  9  0.6591432  0.5263
forn3        1.5833333 1.1378407  9  1.3915246  0.1975
 Correlation:
      (Intr) forn2
forn2 -0.707
forn3 -0.707  0.500

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.4751376 -0.7500844  0.0812409  0.7060895  1.8720268

Number of Observations: 36
Number of Groups: 12
```

O próximo passo da seria fazer uma análise dos resíduos para verificar os pressupostos, semelhante ao que foi feito nos experimentos anteriormente analisados. Vamos deixar isto por conta do leitor.

18 Análise de Covariância

Vejamos agora a análise de covariância do exemplo da apostila do curso. Clique aqui para ver e copiar o arquivo com conjunto de dados para sua área de trabalho.

Começamos com a leitura e organização dos dados. Note que neste caso temos 2 variáveis numéricas, a resposta (`resp`) e a covariável (`cov`).

```
> ex12 <- read.table("exemplo12.txt", header=T)
> ex12

> dim(ex12)
[1] 15  3
> names(ex12)
[1] "maq"  "cov"  "resp"
>
> ex12$maq <- as.factor(ex12$maq)
> is.numeric(ex12$cov)
[1] TRUE
> is.numeric(ex12$resp)
[1] TRUE
>
> summary(ex12)
 maq        cov             resp
 1:5   Min.   :15.00   Min.   :32.0
 2:5   1st Qu.:21.50   1st Qu.:36.5
 3:5   Median :24.00   Median :40.0
       Mean   :24.13   Mean   :40.2
       3rd Qu.:27.00   3rd Qu.:43.0
       Max.   :32.00   Max.   :49.0
```

Na análise de covariância os testes de significância tem que ser obtidos em ajustes separados. Isto é porque não temos ortogonalidade entre os fatores.

Primeiro vamos testar o intercepto (coeficiente β) da reta de regressão. Na análise de variância abaixo devemos considerar apenas o teste referente à variável `cov` que neste caso está corrigida para o efeito de `maq`. Note que para isto a variável `cov` tem que ser a última na especificação do modelo.

```
> ex12.av <- aov(resp ~ maq + cov, data=ex12)
> summary(ex12.av)
            Df  Sum Sq Mean Sq F value    Pr(>F)
maq          2 140.400  70.200  27.593 5.170e-05 ***
cov          1 178.014 178.014  69.969 4.264e-06 ***
Residuals   11  27.986   2.544
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A seguir testamos o efeito do fator `maq` corrigindo para o efeito da covariável. Para isto basta inverter a ordem dos termos na especificação do modelo.

```
> ex12.av <- aov(resp ~ cov + maq, data=ex12)
> summary(ex12.av)
            Df  Sum Sq Mean Sq  F value   Pr(>F)
cov          1 305.130 305.130 119.9330 2.96e-07 ***
maq          2  13.284   6.642   2.6106   0.1181
```

```
Residuals   11  27.986   2.544
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

19 Experimentos em Parcelas Subdivididas

Vamos mostrar aqui como especificar o modelo para análise de experimentos em parcelas subdivididas. Os comandos abaixo mostram a leitura e preparação dos dados e a obtenção da análise de variância. Deixamos por conta do leitor a análise exploratória e de resíduos, desdobramento das interações e testes de comparações múltiplas.

Considere o experimento em parcelas subdivididas de dados de produção de aveia descrito na apostila do curso. Clique aqui para ver e copiar o arquivo com conjunto de dados. A obtenção da análise de variância é ilustrada nos comandos e saídas abaixo.

```
> ex09 <- read.table("exemplo09.txt", header=T)
> ex09

> dim(ex09)
[1] 64  4
> names(ex09)
[1] "a"     "b"     "bloco" "resp"

> ex09$a <- as.factor(ex09$a)
> ex09$b <- as.factor(ex09$b)
> ex09$bloco <- as.factor(ex09$bloco)
> summary(ex09)
 a      b      bloco       resp
 1:16   1:16   1:16   Min.   :28.30
 2:16   2:16   2:16   1st Qu.:44.90
 3:16   3:16   3:16   Median :52.30
 4:16   4:16   4:16   Mean   :52.81
                      3rd Qu.:62.38
                      Max.   :75.40

> ex09.av <- aov(resp ~ bloco + a*b + Error(bloco/a), data=ex09)
> summary(ex09.av)

Error: bloco
      Df  Sum Sq Mean Sq
bloco  3 2842.87  947.62
Error: bloco:a
          Df  Sum Sq Mean Sq F value   Pr(>F)
a          3 2848.02  949.34  13.819 0.001022 **
Residuals  9  618.29   68.70
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Error: Within
          Df Sum Sq Mean Sq F value   Pr(>F)
b          3 170.54   56.85  2.7987 0.053859 .
a:b        9 586.47   65.16  3.2082 0.005945 **
Residuals 36 731.20   20.31
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

20 O pacote gsse401

Iremos examinar aqui um pacote que foi feito para ilustrar conceitos estatísticos utilizando o R. Este pacote serviu de material de apoio para um curso de estatística geral para estudantes de Pós-Graduação de diversos cursos na Universidade de Lancaster, Inglaterra.

Os objetivos são vários. Poderemos ver a flexibilidade do R para criar funções e materiais específicos. Alguns conceitos estatísticos serão revisados enquanto usamos as funções do pacote. Além disto podemos examinar as funções para ver como foram programadas no R.

O pacote se chama `gsse401` e tem uma página em http://www.est.ufpr.br/gsse401. O procedimento para instalação do pacote é descrito abaixo.

- **Usuários de LINUX** Na linha do Linux digite o comando abaixo para certificar-se que voce tem um diretório `Rpacks` para instalar o pacote
  `mkdir -p  /Rpacks`

  Depois, inicie o programa R e digite:

  ```
  > install.packages("gsse401",
                     contriburl="http://www.est.ufpr.br/gsse401",
                     lib = "~/Rpacks")
  > library(gsse401, lib= "~/Rpacks")
  ```

- **Usuários de WINDOWS** Inicie o R e digite os comandos

  ```
  > install.packages("gsse401", contriburl="http://www.est.ufpr.br/gsse401")
  > require(gsse401)
  ```

O pacote possui várias funções e conjuntos de dados. Para exibir os nomes dos conjuntos de dados e funções do pacote digite:

```
> gsse401.data()
> gsse401.functions()
```

Todas as funções e conjuntos de dados estão documentados, ou seja, voce pode usar a função `help()` para saber mais sobre cada um.

A versão atual possui 4 funções que serão mostradas em aula.

- `queue()` esta função simula o comportamento de uma fila com parâmetros de chegada e saída definidos pelo usuário.
- `clt()` esta funạão ilustra o Teorema Central do Limite.
- `mctest()` teste Monte Carlo para comparar 2 amostras.
- `reg()` função ilustrando convceitos de regressão linear.

Sugere-se as seguites atividades:

1. Carregue o conjunto de dados `ansc` e rode os exemplos de sua documentação. Discuta os resultados. Lembre-se que para carregar este conjunto de dados e ver sua documentação deve-se usar os comandos:

   ```
   > data(ansc)
   > help(asnc)
   ```

2. Explore a função `clt`. Veja a sua documentação, rode os exemplos e veja como foi programada digitando `clt` (sem os parênteses). Tente também usar a função digitando `clt()`.

3. Carregue o conjunto de dados `gravity`, veja sua documentação, rode e discuta os exemplos.

4. explore a função `mctest`, veja sua documentação e exemplos.

5. explore a função `queue`

6. explore a função `reg`. Tente também digitar `reg()` para o funcionamento interativo da função.

21 Usando simulação para ilustrar resultados

Podemos utilizar recursos computacionais e em particular simulações para inferir distribuições amostrais de quantidades de interesse. Na teoria de estatística existem vários resultados que podem ser ilustrados via simulação, o que ajuda na compreensão e visualização dos conceitos e resultados. Veremos alguns exemplos a seguir.

Este uso de simulações é apenas um ponto de partida pois estas são especialmente úteis para explorar situações onde resultados teóricos não são conhecidos ou não podem ser obtidos.

21.1 Relações entre a distribuição normal e a χ^2

Resultado 1: Se $Z \sim \mathrm{N}(0,1)$ então $Z^2 \sim \chi^2_{(1)}$.

Vejamos como ilustrar este resultado. Inicialmente vamos definir o valor da semente de números aleatórios para que os resultados possam ser reproduzidos. Vamos começar gerando uma amostra de 1000 números da distribuição normal padrão. A seguir vamos fazer um histograma dos dados obtidos e sobrepor a curva da distribuição teórica. Fazemos isto com os comando abaixo e o resultado está no gráfico da esquerda da Figura 23.

```
> set.seed(23)
> z <- rnorm(1000)
> hist(z, prob=T)
> curve(dnorm(x), -4, 4, add=T)
```

Note que, para fazer a comparação do histograma e da curva teórica é necessário que o histograma seja de frequências relativas e para isto usamos o argumento `prob = T`.

Agora vamos estudar o comportamento da variável ao quadrado. O gráfico da direita da Figura 23 mostra o histograma dos quadrados do valores da amostra e a curva da distribuição de $\chi^2_{(1)}$.

```
> hist(z^2, prob=T)
> curve(dchisq(x, df=1), 0, 10, add=T)
```

Nos gráficos anteriores comparamos o histograma da distribuição empírica obtida por simulação com a curva teórica da distribuição. Uma outra forma e mais eficaz forma de comparar distribuições empíricas e teóricas é comparar os quantis das distribuições e para isto utilizamos o *qq-plot*. O *qq-plot* é um gráfico dos dados ordenados contra os quantis esperados de uma certa distribuição. Quanto mais próximo os pontos estiverem da bissetriz do primeiro quadrante mais próximos os dados observados estão da distribuição considerada. Portanto para fazer o `qqplot` seguimos os passos:

1. obter os dados,
2. obter os quantis da distribuição teórica,
3. fazer um gráfico dos dados ordenados contra os quantis da distribuição.

Vamos ilustrar isto nos comandos abaixo. Primeiro vamos considerar como dados os quadrados da amostra da normal obtida acima. Depois obtemos os quantis teóricos da distribução χ^2 usando a função `qchisq` em um conjunto de probabilidades geradas pela função `ppoints`. Por fim usamos a função `qqplot` para obter o gráfico mostrado na Figura 24. Adicionamos neste gráfico a bissetriz do primeiro quadrante.

```
> quantis <- qchisq(ppoints(length(z)), df=1)
> qqplot(quantis, z^2)
> abline(0,1)
```

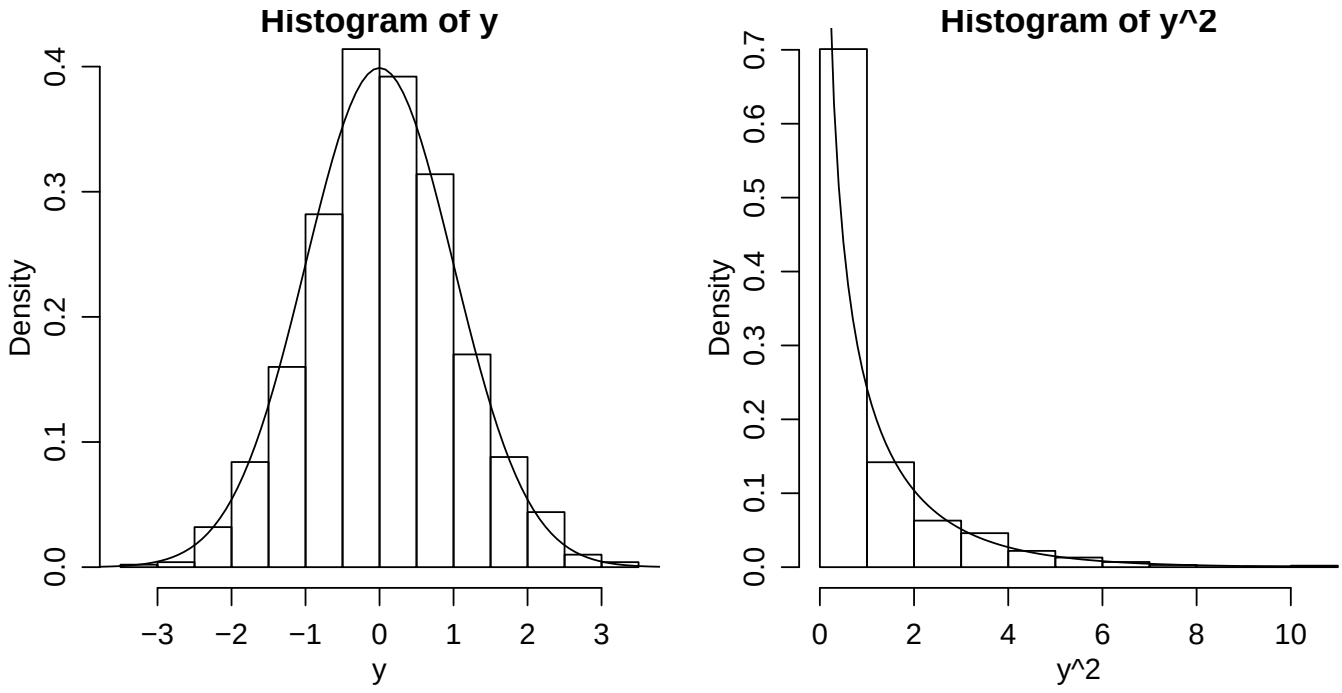

Figura 23: Histograma das amostra da e a curva teórica da distribuição normal padrão (esquerda) e histograma dos valores ao quadrado com a curva teórica da distribuição $\chi^2_{(1)}$ (direita).

Note que o comando `qchisq(ppoints(length(z)), df=1)` acima está concatenando 3 comandos e calcula os quantis da χ^2 a partir de uma sequência de valores de probabilidade gerada por `ppoints`. O número de elementos desta sequência deve igual ao número de dados e por isto usamos `length(z)`.

Resultado 2: Se $Z_1, Z_2, \ldots Z_n \sim \mathrm{N}(0,1)$ então $\sum_1^n Z_i^2 \sim \chi^2_{(n)}$.

Para ilustrar este resultado vamos gerar 10.000 amostras de 3 elementos cada da distribuição normal padrão, elevar os valores ao quadrado e, para cada amostra, somar os quadrados dos três números. Na Figura 25 mostramos o histograma dos valores obtidos com a curva da distribuição esperada e o *qq-plot*.

```
> z <- matrix(rnorm(30000), nc=3)
> sz2 <- apply(z^2, 1, sum)
> hist(sz2, prob=T, main="")
> curve(dchisq(x, df=3), 0, 30, add=T)
> qqplot(qchisq(ppoints(length(sz2)), df=3), sz2)
> abline(0,1)
```

21.2 Distribuição amostral da média de amostras da distribuição normal

Resultado 3: Se $Y_1, Y_2, \ldots Y_n \sim \mathrm{N}(\mu, \sigma^2)$ então $\bar{y} \sim \mathrm{N}(\mu, \sigma^2/n)$.

Neste exemplo vamos obter 1000 amostras de tamanho 20 de uma distribuição normal de média 100 e variância 30. Vamos organizar as amostras em uma matriz onde cada coluna corresponde a uma amostra. A seguir vamos calcular a média de cada amostra. Pelo **Resultado 3** acima esperamos que a média das médias amostrais seja 100 e a variância seja 1.5 (= 30/20), e que a distribuição das médias amostrais seja normal, valores bem próximos dos obtidos acima. Para completar vamos obter o gráfico com o histograma das médias das amostras e a distribuição teórica conforme Figura 26 e o respectivo `qq-plot`.

```
> y <- matrix(rnorm(20000, mean=100, sd=sqrt(30)), nc=1000)
> ybar <- apply(y, 2, mean)
```

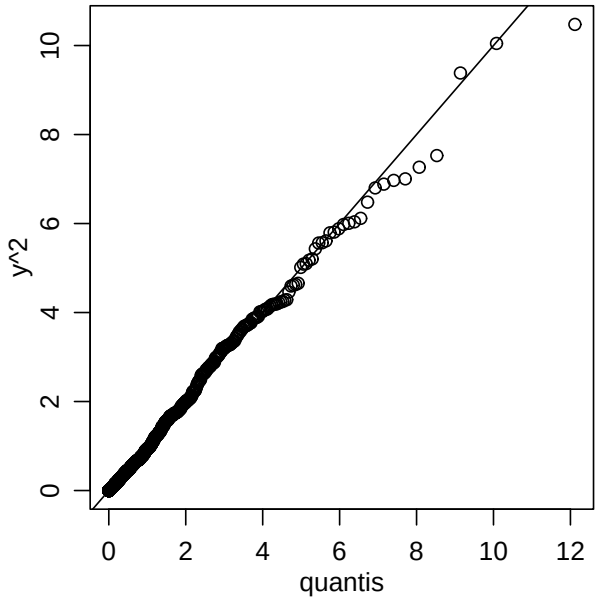

Figura 24: Comparando dados e quantis da χ^2 utilizando o *qq-plot*

```
> mean(ybar)
> [1] 99.96043
> var(ybar)
[1] 1.582839
> hist(ybar, prob = T)
> curve(dnorm(x, mean=100, sd=sqrt(30/20)), 95, 105, add=T)
> qqnorm(ybar)
> qqline(ybar)
```

Note que para obter o *qq-plot* neste exemplo utilizamos as funções `qqnorm` `qqline` já disponíveis no R para fazer *qq-plot* para distribuição normal.

21.3 Exercícios

1. Ilustrar usando simulação o resultado que afirma que o estimador $S^2 = \sum \frac{(x_i - \bar{x})^2}{n-1}$ da variância de uma distribuição normal tem distribuição χ^2_{n-1}.
   DICA: Voce pode começar pensando nos passos necessários para ilustrar este resultado:
   - escolha os parâmetros de uma distribuição normal,
   - escolha o tamanho de amostra n e o número de simulações N,
   - gere N amostras de tamanho n,
   - para cada amostra calcule S^2,
   - faça um histograma com os valores S^2 e compare com a curva de uma distrição χ^2_{n-1}.

2. Seja $X_1, \ldots, X_n$ a.a. de uma distribuição $\mathrm{N}(\mu, \sigma^2)$. Ilustrar o resultado que justifica o teste-t para média de uma amostra,
   $$\frac{\bar{x} - \mu}{S/\sqrt{n}} \sim t_{n-1}$$
   onde S é o desvio padrão da amostra e n o tamanho da amostra.
   DICA: começe verificando passo a passo, como no exercício anterior, o que é necessário para ilustrar este resultado.

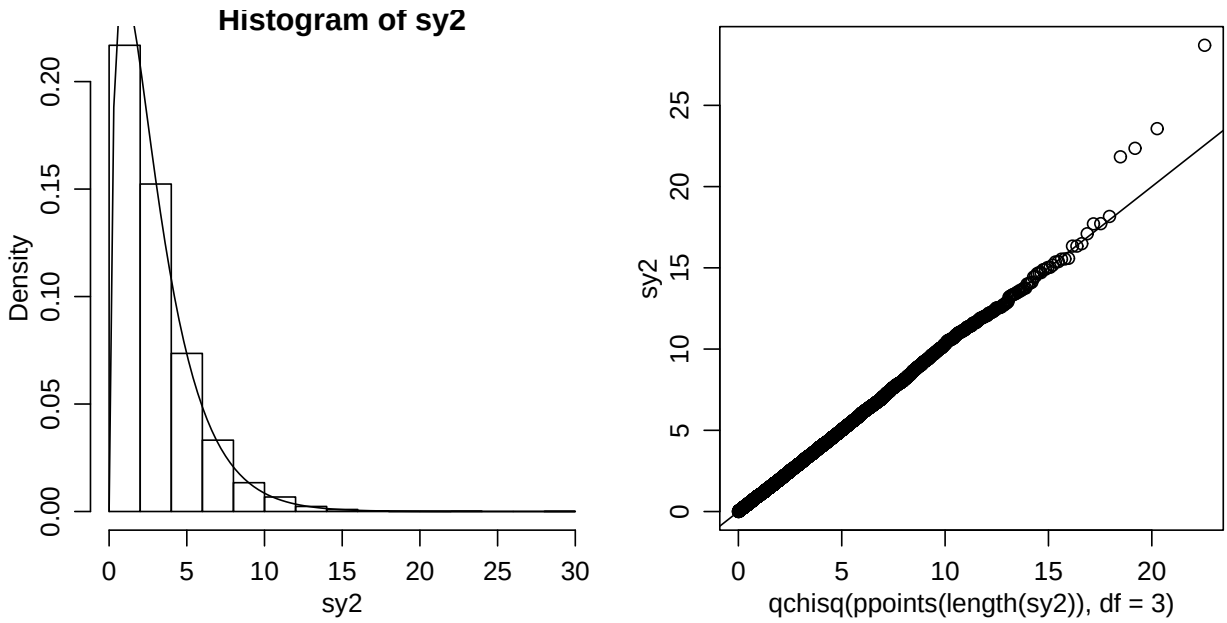

Figura 25: Histograma da uma amostra da soma dos quadrados de três valores da normal padrão e a curva teórica da distribuição de $\chi^2_{(3)}$ (esquerda) e o respectivo *qq-plot*.

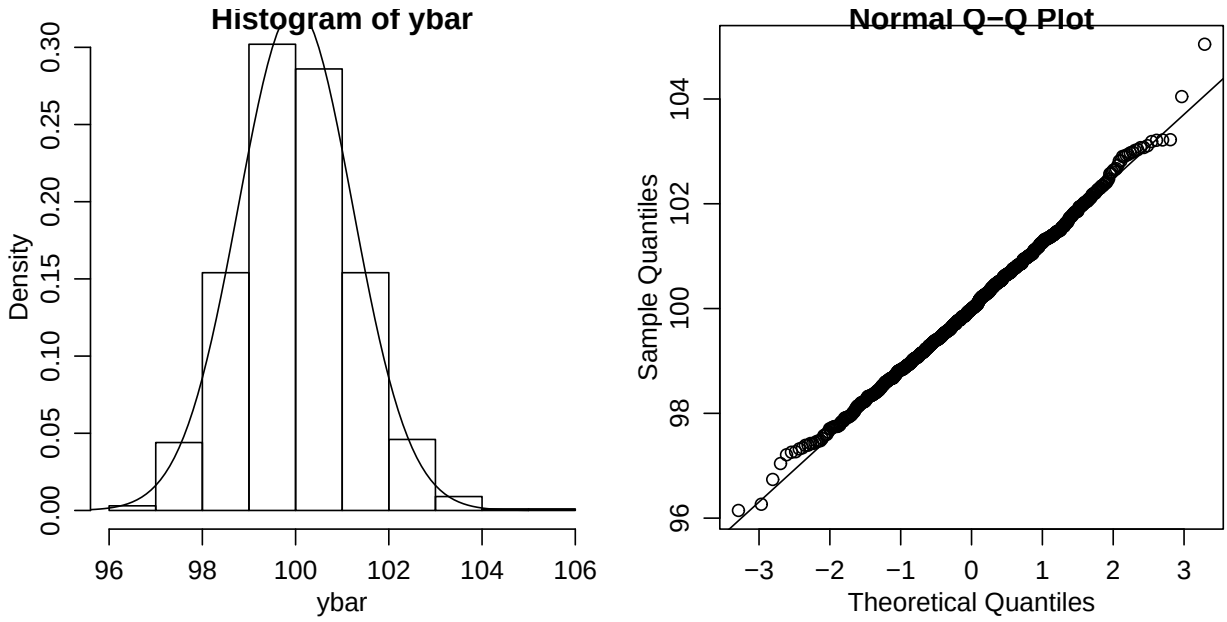

Figura 26: Histograma de uma amostra da distribuição amostral da média e a curva teórica da distribuição e o respectivo *qq-plot*.

3. Ilustrar o resultado que diz que o quociente de duas variáveis independentes com distribuição χ^2 tem distribuição F.

22 Ilustrando propriedades de estimadores

22.1 Consistência

Um estimador é consistente quando seu valor se aproxima do verdadeiro valor do parâmetro à medida que aumenta-se o tamanho da amostra. Vejamos como podemos ilustrar este resultado usando simulação. A idéia básica é a seguite:

1. escolher uma distribuição e seus parâmetros,
2. definir o estimador,
3. definir uma sequência crescente de valores de tamanho de amostras,
4. obter uma amostra de cada tamanho,
5. calcular a estatística para cada amostra,
6. fazer um gráfico dos valores das estimativas contra o tamanho de amostra, indicando neste gráfico o valor verdadeiro do parâmetro.

22.1.1 Média da distribuição normal

Seguindo os passos acima vamos:

1. tomar a distribuição Normal de média 10 e variância 4,
2. definir o estimador $\bar{X} = \sum_{i=1}^{n} \frac{x_i}{n}$,
3. escolhemos os tamanhos de amostra $n = 2, 5, 10, 15, 20, \ldots, 1000, 1010, 1020, \ldots, 5000$,
4. fazemos os cálculos e produzimos um gráfico como mostrado na 27 com os comandos a seguir.

```
> ns <- c(2, seq(5, 1000, by=5), seq(1010, 5000, by=10))
> estim <- numeric(length(ns))
> for (i in 1:length(ns)){
>   amostra <- rnorm(ns[i], 10, 4)
>   estim[i] <- mean(amostra)
> }
> plot(ns, estim)
> abline(h=10)
```

22.2 Momentos das distribuições amostrais de estimadores

Para inferência estatística é necessário conhecer a distribuição amostral dos estimadores. Em alguns casos estas distribuições são derivadas analiticamente. Isto se aplica a diversos resultados vistos em um curso de Inferência Estatística. Por exemplo o resultado visto na sessão 21: se $Y_1, Y_2, \ldots Y_n \sim \mathrm{N}(\mu, \sigma^2)$ então $\bar{y} \sim \mathrm{N}(\mu, \sigma^2/n)$. Resultados como estes podem ser ilustrados computacionalmente como visto na Sessão 21.

Além disto este procedimento permite investigar distribuições amostrais que são complicadas ou não podem ser obtidas analiticamente.

Vamos ver um exemplo: considere Y uma v.a. com distribuição normal $N(\mu, \sigma^2)$ e seja um parâmetro de interesse $\theta = \mu/\sigma^2$. Para obter por simulação a esperança e variância do estimador $T = \bar{Y}/S^2$ onde $\bar{Y}$ é a média e S^2 a variância de uma amostra seguimos os passos:

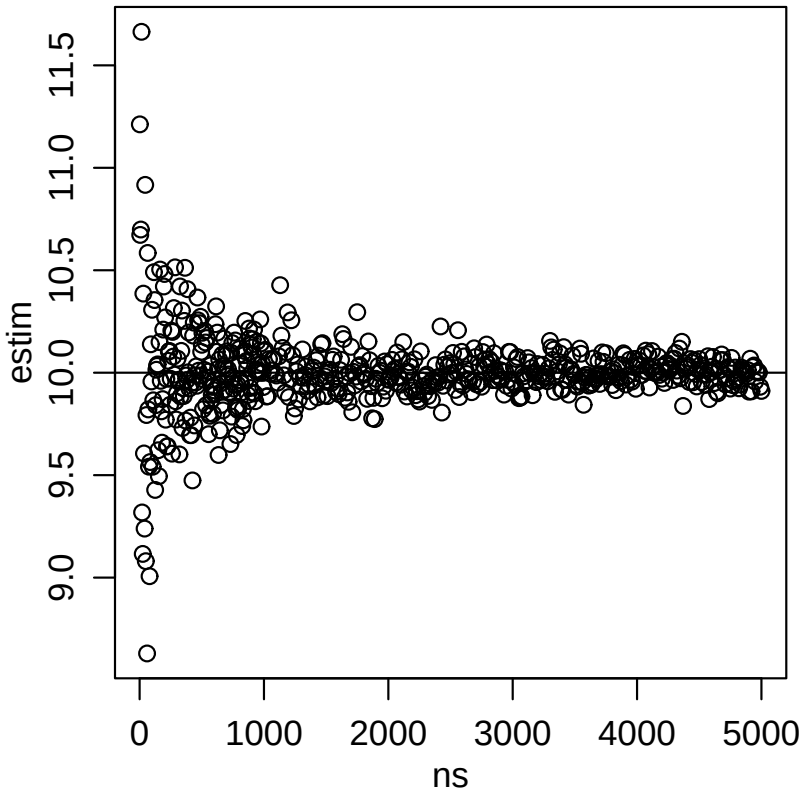

Figura 27: Médias de amostras de diferentes tamanhos.

1. escolher uma distribuição e seus parâmetros, no caso vamos escolher uma $N(180, 64)$,
2. definir um tamanho de amostra, no caso escolhemos $n = 20$,
3. obter por simulação um número N de amostras, vamos usar $N = 1000$,
4. calcular a estatística de interesse para cada amostra,
5. usar as amostras para obter as estimativas $\hat{E}[T]$ e $\hat{\text{Var}}[T]$.

Vamos ver agora comandos do R.

```
> amostras <- matrix(rnorm(20*1000, mean=180, sd=8), nc=1000)
> Tvals <- apply(amostras, 2, function(x) {mean(x)/var(x)})
> ET <- mean(Tvals)
> ET
[1] 3.134504
> VarT <- var(Tvals)
> VarT
[1] 1.179528
```

Nestes comandos primeiro obtemos 1000 amostras de tamanho 20 que armazenamos em uma matriz de dimensão 20×1000, onde cada coluna é uma amostra. A seguir usamos a função `apply` para calcular a quantidade desejada que definimos com `function(x) {mean(x)/var(x)}`. No caso anterior foi obtido $\hat{E}[T] \approx 3.13$ e $\hat{\text{Var}}[T] \approx 1.18$.

Se voce rodar os comandos acima deverá obter resultados um pouco diferentes (mas não muito!) pois nossas amostras da distribuição normal não são as mesmas.

22.3 Não-tendenciosidade

Fica como exercício.

22.4 Variância mínima

Fica como exercício.

22.5 Exercícios

1. Ilustre a consistência do estimador $\hat{\lambda} = 1/\bar{X}$ de uma distribuição exponencial $f(x) = \lambda \exp\{-\lambda x\}$.

2. No exemplo dos momentos das distribuições de estimadores visto em (22.2) ilustramos a obtenção dos momentos para um tamanho fixo de amostra $n = 20$. Repita o procedimento para vários tamanho de amostra e faça um gráfico mostrando o comportamento de $\hat{E}[T]$ e $\hat{\text{Var}}[T]$ em função de n.

3. Estime por simulação a esperança e variância do estimador $\hat{\lambda} = \bar{X}$ de uma distribuição de Poisson de parâmetro λ para um tamanho de amostra $n = 30$. Compare com os valores obtidos analiticamente. Mostre em um gráfico como os valores de $\hat{E}[\hat{\lambda}]$ e $\hat{\text{Var}}[\hat{\lambda}]$ variam em função de n.

4. Crie um exemplo para ilustrar a não tendenciosidade de estimadores. Sugestão: compare os estimadores $S^2 = \sum_{i=1}^{n}(X_1 - \bar{X})^2/(n-1)$ e $\hat{\sigma}^2 = \sum_{i=1}^{n}(X_1 - \bar{X})^2/n$ do parâmetro de variância σ^2 de uma distribuição normal.

5. Crie um exemplo para comparar a variância de dois estimadores. Por exemplo compare por simulação as variâncias dos estimadores $T_1 = \bar{X}$ e $T_2 = (X_{[1]} + X_{[n]})/2$ do parâmetro μ de uma distribuição $\text{N}(\mu, \sigma^2)$, onde $X_{[1]}$ e $X_{[n]}$ são os valores mínimo e máximo da amostra, respectivamente.

23 Funções de verossimilhança

A função de verossimilhança é central na inferência estatística. Nesta sessão vamos ver como traçar funções de verossimilhança utilizando o programa R.

23.1 Exemplo 1: Distribuição normal com variância conhecida

Seja o vetor $(12, 15, 9, 10, 17, 12, 11, 18, 15, 13)$ uma amostra aleatória de uma distribuição normal de média μ e variância conhecida e igual a 4. O objetivo é fazer um gráfico da função de log-verossimilhança.
Solução:
Vejamos primeiro os passos da solução analítica:

1. Temos que $X_1, \ldots, X_n$ onde, neste exemplo $n = 10$, é uma a.a. de $X \sim N(\mu, 4)$,
2. a densidade para cada observação é dada por $f(x_i) = \frac{1}{2\sqrt{2\pi}} \exp\{-\frac{1}{8}(x_i - \mu)^2\}$,
3. a verossimilhança é dada por $L(\mu) = \prod_1^{10} f(x_i)$,
4. e a log-verossimilhança é dada por

$$\begin{aligned} l(\mu) &= \sum_1^{10} \log(f(x_i)) \\ &= -5\log(8\pi) - \frac{1}{8}(\sum_1^{10} x_i^2 - 2\mu \sum_1^{10} x_i + 10\mu^2), \end{aligned} \tag{3}$$

5. que é uma função de μ e portanto devemos fazer um gráfico de $l(\mu)$ *versus* μ tomando vários valores de μ e calculando os valores de $l(\mu)$.

Vamos ver agora uma primeira possível forma de fazer a função de verossimilhança no R.

1. Primeiro entramos com os dados que armazenamos no vetor `x`

```
> x <- c(12, 15, 9, 10, 17, 12, 11, 18, 15, 13)
```

2. e calculamos as quantidades $\sum_1^{10} x_i^2$ e $\sum_1^{10} x_i$

```
> sx2 <- sum(x^2)
> sx <- sum(x)
```

3. agora tomamos uma sequência de valores para μ. Sabemos que o estimador de máxima verossimilhança neste caso é $\hat{\mu} = 13.2$ (este valor pode ser obtido com o comando `mean(x)`) e portanto vamos definir tomar valores ao redor deste ponto.

```
> mu.vals <- seq(11, 15, l=100)
```

4. e a seguir calculamos os valores de $l(\mu)$ de acordo com a equação acima

```
> lmu <- -5 * log(8*pi) - (sx2 - 2*mu.vals*sx + 10*(mu.vals^2))/8
```

5. e finalmente fazemos o gráfico visto na Figura 28

```
> plot(mu.vals, lmu, type='l', xlab=expression(mu), ylab=expression(l(mu)))
```

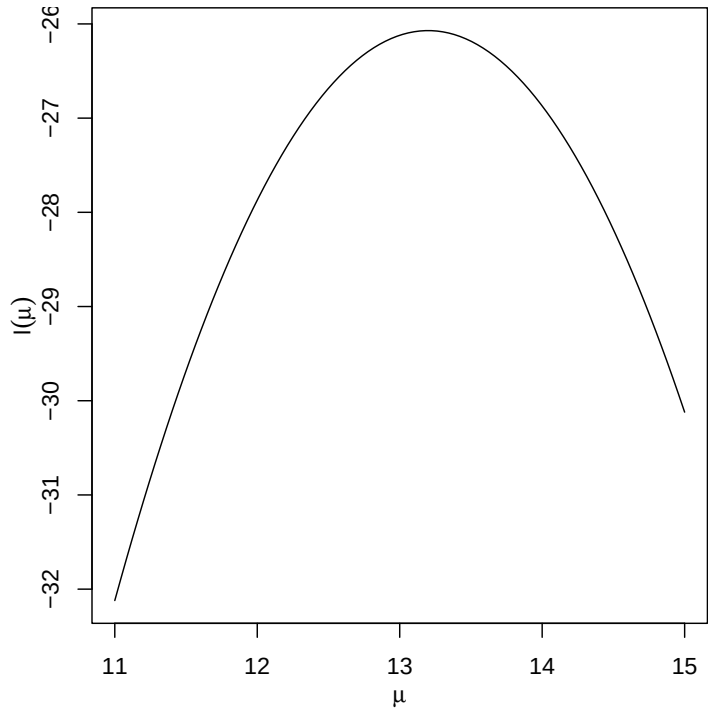

Figura 28: Função de verossimilhança para o parâmetro μ da distribuição normal com variância $\sigma^2 = 4$ com os dados do Exemplo 1.

Entretanto podemos obter a função de verossimilhança no R de outras forma mais geral e menos trabalhosas. Sabemos que a função `dnorm` calcula a densidade $f(x)$ da distribuição normal e podemos usar este fato para evitar a digitação da expressão acima.

- Primeiro vamos criar uma função que calcula o valor da log-verossimilhança para um certo valor do parâmetro e para um certo conjunto de dados,

```
> logvero <- function(mu, dados){
     sum(dnorm(dados, mean = mu, sd = 2, log = TRUE))
  }
```

- a seguir criamos uma sequência adequada de valores de μ e calculamos $l(\mu)$ para cada um dos valores

```
> mu.vals <- seq(11, 15, l=100)
> mu.vals
> lmu <- sapply(mu.vals, logvero, dados = x)
> lmu
```

  Note na sintaxe acima que a função `sapply` aplica a função `logvero` anteriormente definida em cada elemento do vetor `mu.vals`.

- Finalmente fazemos o gráfico.

```
> plot(mu.vals, lmu, type='l', xlab=expression(mu), ylab=expression(l(mu)))
```

Para encerrar este exemplo vamos apresentar uma solução ainda mais genérica que consiste em criar uma função que vamos chamar de `vero.norm.v4` para cálculo da verossimilhança de distribuições normais com σ^2=4. Esta função engloba os comandos acima e pode ser utilizada para obter o gráfico da log-verossimilhança para o parâmetro μ para qualquer amostra obtida desta distribuição.

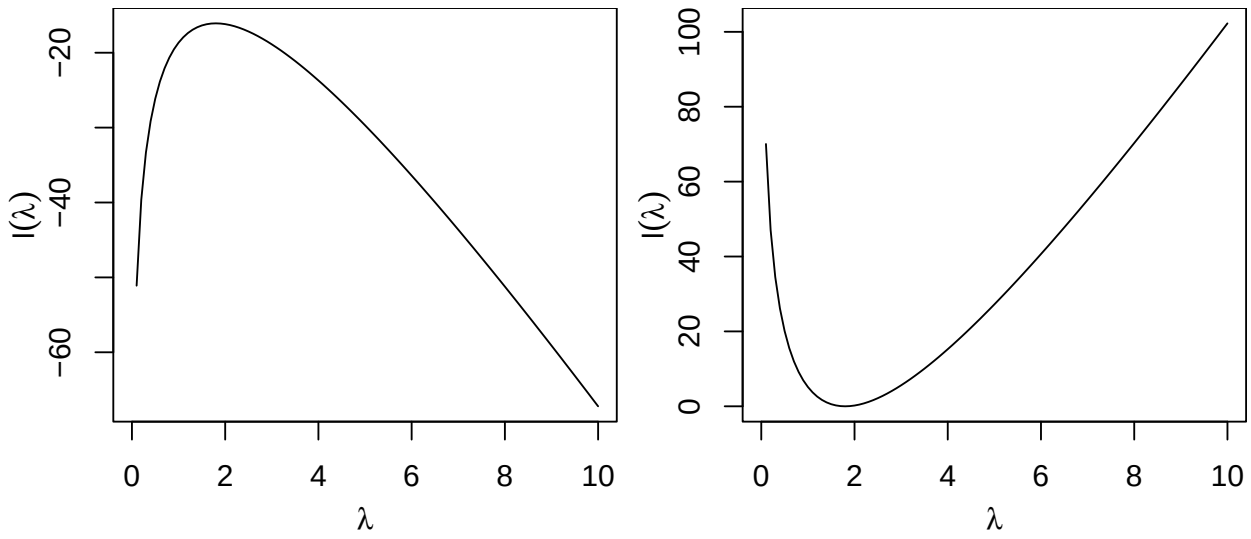

Figura 29: Função de verossimilhança e deviance para o parâmetro λ da distribuição Poisson.

```
> vero.normal.v4 <- function(mu, dados){
    logvero <- function(mu, dados)
      sum(dnorm(dados, mean = mu, sd = 2, log = TRUE))
    sapply(mu, logvero, dados = dados)
  }
> curve(vero.normal.v4(x, dados = x), 11, 15,
      xlab=expression(mu), ylab=expression(l(mu)))
```

23.2 Exemplo 2: Distribuição Poisson

Considere agora a amostra armazenada no vetor `y`:

```
> y <- c(5, 0, 3, 2, 1, 2, 1, 1, 2, 1)
```

de uma distribuição de Poisson de parâmetro λ. A função de verossimilhança pode ser definida por:

```
lik.pois <- function(lambda, dados){
  loglik <- function(l, dados){sum(dpois(dados, lambda = l, log = TRUE))}
  sapply(lambda, loglik, dados = dados)
}
```

E podemos usar esta função para fazer o gráfico da função de verossimilhança como visto à esquerda da Figura 29

```
> lambda.vals <- seq(0, 10, l=101)
> loglik <- sapply(lambda.vals, lik.pois, dados=y)
> plot(lambda.vals, loglik, ty = "l")
## ou mudando o texto do eixos
> plot(lambda.vals, loglik, type = "l", xlab=expression(lambda),
>      ylab=expression(l(lambda)))
## ou
> curve(lik.pois(x, dados=y), 0, 10)
```

Alternativamente pode-se fazer um gráfico da função deviance, como nos comandos abaixo.

```
> dev.pois <- function(lambda, dados){
>   lambda.est <- mean(dados)
>   lik.lambda.est <- lik.pois(lambda.est, dados = dados)
>   lik.lambda <- lik.pois(lambda, dados = dados)
>   return(-2 * (lik.lambda - lik.lambda.est))
> }
> curve(dev.pois(x, dados=y), 0, 10)
## fazendo novamente em um intervalo menor
> curve(dev.pois(x, dados=y), 0.5, 5)
```

O estimador de máxima verossimilhança é o valor que maximiza a função de verossimilhança que é o mesmo que minimiza a função deviance. Neste caso sabemos que o estimador tem expressão analítica fechada $\lambda = \bar{x}$ e portanto calculado com o comando.

```
> lambda.est
[1] 1.8
```

Caso o estimador não tenha expressão fechada pode-se usar maximização (ou minimização) numérica. Para ilustrar isto vamos encontrar a estimativa do parâmetro da Poisson e verificar que o valor obtido coincide com o valor dado pela expressão fechada do estimador. Usamos o função `optimise` para encontrar o ponto de mínimo da função deviance.

```
> optimise(dev.pois, int=c(0, 10), dados=y)
$minimum
[1] 1.800004

$objective
[1] 1.075406e-10
```

A função `optimise()` é adequada para minimizações envolvendo um único parâmetro. Para dois ou mais parâmetros deve-se usar a função `optim()`

Finalmente os comandos abaixo são usados para obter graficamente o intervalo de confiança baseado na verossimilhança.

```
corte <- qchisq(0.95, df=1)
abline(h=corte)

## obtendo os limites (aproximados) do IC
l.vals <- seq(0.5,5,l=1001)
dev.l <- dev.pois(l.vals, dados=y)
dif <- abs(dev.l - corte)
ind <- l.vals < lambda.est
ic2.lambda <- c(l.vals[ind][which.min(dif[ind])],
                l.vals[!ind][which.min(dif[!ind])])
ic2.lambda
## adicionando ao gráfico
curve(dev.pois(x, dados=y), 1, 3.5,
      xlab=expression(lambda), ylab=expression(l(lambda)))
segments(ic2.lambda, 0, ic2.lambda, corte)
```

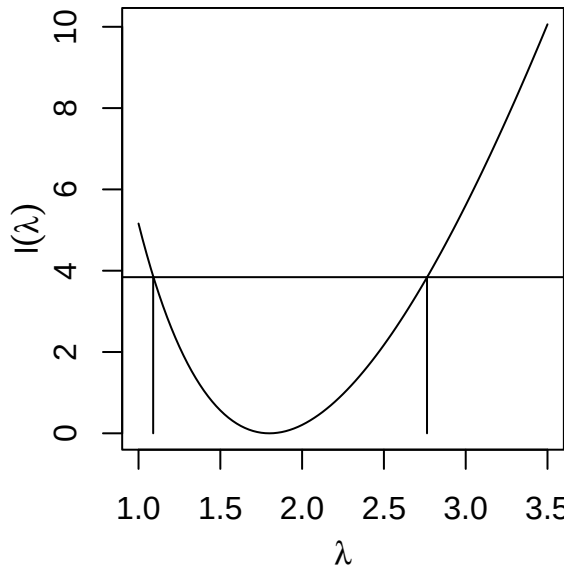

Figura 30: Função de verossimilhança e deviance para o parâmetro λ da distribuição Poisson.

23.3 Exercícios

1. Seja a amostra abaixo obtida de uma distribuição Poisson de parâmetro λ.
   5 4 6 2 2 4 5 3 3 0 1 7 6 5 3 6 5 3 7 2
   Obtenha o gráfico da função de log-verossimilhança.

2. Seja a amostra abaixo obtida de uma distribuição Binomial de parâmetro p e com $n = 10$.
   7 5 8 6 9 6 9 7 7 7 8 8 9 9 9
   Obtenha o gráfico da função de log-verossimilhança.

3. Seja a amostra abaixo obtida de uma distribuição χ^2 de parâmetro ν.
   8.9 10.1 12.1 6.4 12.4 16.9 10.5 9.9 10.8 11.4
   Obtenha o gráfico da função de log-verossimilhança.

24 Intervalos de confiança baseados na deviance

Neste sessão discutiremos a obtenção de intervalos de confiança baseado na deviance.

24.1 Média da distribuição normal com variância conhecida

Seja $X_1, \ldots, X_n$ a.a. de uma distribuição normal de média θ e variância 1. Vimos que:

1. A função de log-verossimilhança é dada por $l(\theta) = \text{cte} + \frac{1}{2}\sum_{i=1}^{n}(x_i - \theta)^2$;
2. o estimador de máxima verossimilhança é $\hat{\theta} = \frac{\sum_{i=1}^{n} X_i}{n} = \bar{X}$;
3. a função deviance é $D(\theta) = n(\bar{x} - \theta)^2$;
4. e neste caso a deviance tem distribuição exata $\chi^2_{(1)}$;
5. e os limites do intervalo são dados por $\bar{x} \stackrel{+}{-} \sqrt{c^*/n}$, onde c^* é o quantil $(1 - \alpha/2)$ da distribuição $\chi^2_{(1)}$.

Vamos considerar que temos uma amostra onde $n = 20$ e $\bar{x} = 32$. Neste caso a função deviance é como mostrada na Figura 31 que é obtida com os comandos abaixo onde primeiro definimos uma função para calcular a deviance que depois é mostrada em um gráfico para valores entre 30 e 34. Para obtermos um intervalo a 95% de confiança escolhemos o quantil correspondente na distribuição $\chi^2_{(1)}$ e mostrado pela linha tracejada no gráfico. Os pontos onde esta linha cortam a função são, neste exemplo, determinados analiticamente pela expressão dada acima e indicados pelos setas verticais no gráfico.

```
> dev.norm.v1 <- function(theta, n, xbar){n * (xbar - theta)^2}
> thetaN.vals <- seq(31, 33, l=101)
> dev.vals <- dev.norm.v1(thetaN.vals, n=20, xbar=32)
> plot(thetaN.vals, dev.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> corte <- qchisq(0.95, df = 1)
> abline(h = corte, lty=3)
> limites <- 32 + c(-1, 1) * sqrt(corte/20)
> limites
[1] 31.56174 32.43826
> segments(limites, rep(corte,2), limites, rep(0,2))
```

Vamos agora examinar o efeito do tamanho da amostra na função. A Figura 32 mostra as funções para três tamanhos de amostra, $n = 10, 20$ e 50 que são obtidas com os comandos abaixo. A linha horizontal mostra o efeito nas amplitudes dos IC's.

```
> dev10.vals <- dev.norm.v1(thetaN.vals, n=10, xbar=32)
> plot(thetaN.vals, dev10.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> dev20.vals <- dev.norm.v1(thetaN.vals, n=20, xbar=32)
> lines(thetaN.vals, dev20.vals, lty=2)
> dev50.vals <- dev.norm.v1(thetaN.vals, n=50, xbar=32)
> lines(thetaN.vals, dev50.vals, lwd=2)
> abline(h = qchisq(0.95, df = 1), lty=3)
> legend(31, 2, c('n=10', 'n=20', 'n=50'), lty=c(1,2,1), lwd=c(1,1,2), cex=0.7)
```

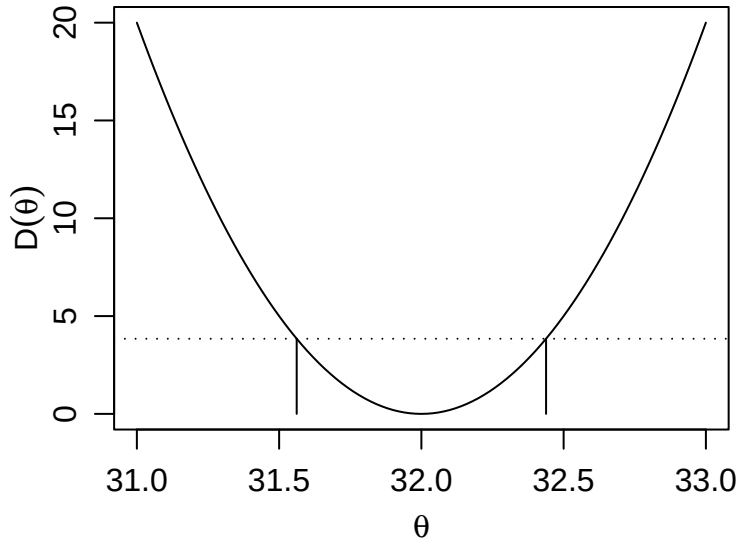

Figura 31: Função deviance para $\mathrm{N}(\theta, 1)$ para uma amostra de tamanho 20 e média 32.

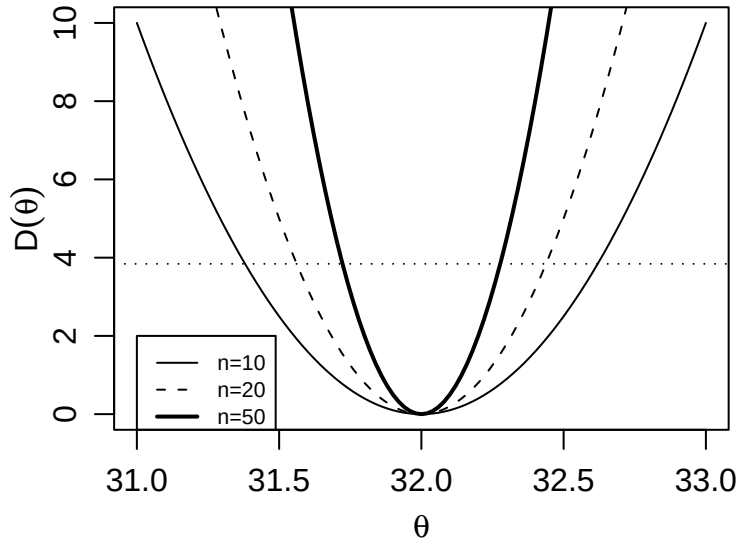

Figura 32: Funções deviance para o parâmetro θ da $\mathrm{N}(\theta, 1)$ para amostras de média 32 e tamanhos de amostra $n = 10, 20$ e 50.

24.2 IC para o parâmetro da distribuição exponencial

Seja $X_1, \ldots, X_n$ a.a. de uma distribuição exponencial de parâmetro θ com função de densidade $f(x) = \theta \exp\{-\theta x\}$. Vimos que:

1. A função de log-verossimilhança é dada por $l(\theta) = n \log(\theta) - \theta n \bar{x}$;
2. o estimador de máxima verossimilhança é $\hat{\theta} = \frac{n}{\sum_{i=1}^{n} X_i} = \frac{1}{\bar{X}}$;
3. a função deviance é $D(\theta) = 2n \left[\log(\hat{\theta}/\theta) + \bar{x}(\theta - \hat{\theta})\right]$;
4. e neste caso a deviance tem distribuição assintótica $\chi^2_{(1)}$;
5. e os limites do intervalo não podem ser obtidos analiticamente, devendo ser obtidos por:
   - métodos numéricos ou gráficos, ou,

- pela aproximação quadrática da verossimilhança por série de Taylor que neste caso fornece uma expressão da deviance aproximada dada por $D(\theta) \approx n\left(\frac{\theta-\hat{\theta}}{\hat{\theta}}\right)^2$.

A seguir vamos ilustrar a obtenção destes intervalos no R. Vamos considerar que temos uma amostra onde $n = 20$ e $\bar{x} = 10$ para a qual a função deviance é mostrada na Figura 33 e obtida de forma análoga ao exemplo anterior.

```
> dev.exp <- function(theta, n, xbar){
+   2*n*(log((1/xbar)/theta) + xbar*(theta-(1/xbar)))
+ }
> thetaE.vals <- seq(0.04,0.20, l=101)
> dev.vals <- dev.exp(thetaE.vals, n=20, xbar=10)
> plot(thetaE.vals, dev.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
```

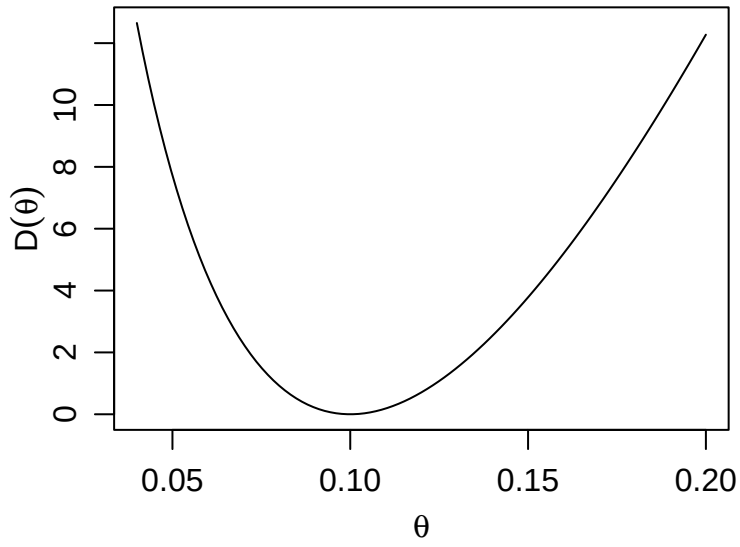

Figura 33: Função deviance da Exp(θ) para uma amostra de tamanho 20 e média 10.

Neste exemplo, diferentemente do anterior, não determinamos a distribuição exata da deviance e usamos a distribuição assintótica $\chi^2_{(1)}$ na qual se baseia a linha de corte tracejada mostrada no gráfico para definir o IC do parâmetro ao nível de 95% de confiança.

Para encontrar os limites do IC precisamos dos valores no eixo dos parâmetros nos pontos onde a linha de corte toca a função deviance o que corresponde a resolver a equação $D(\theta) = 2n\left[\log(\hat{\theta}/\theta) + \bar{x}(\theta - \hat{\theta})\right] = c^*$ onde c^* é quantil da distribuição da χ^2 com 1 grau de liberdade correspondente ao nível de confiança desejado. Por exemplo, para 95% o valor de $\chi^2_{1,0.95}$ é 3.84. Como esta equação não tem solução analítica (diferentemente do exemplo anterior) vamos examinar a seguir duas possíveis soluções para encontrar os limites do intervalo.

24.2.1 Solução numérica/gráfica simplificada

Iremos aqui considerar uma solução simples baseada no gráfico da função deviance para encontrar os limites do IC que consiste no seguinte: Para fazermos o gráfico da deviance criamos uma sequência de valores do parâmetro θ. A cada um destes valores corresponde um valor de $D(\theta)$. Vamos então localizar os valores de θ para os quais $D(\theta)$ é o mais próximo possível do ponto de corte. Isto é feito com o código abaixo e o resultado exibido na Figura 34.

```
> plot(thetaE.vals, dev.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> corte <- qchisq(0.95, df = 1)
> abline(h = corte, lty=3)
> dif <- abs(dev.vals - corte)
> linf <- thetaE.vals[thetaE.vals<(1/10)][which.min(dif[thetaE.vals<(1/10)])]
> lsup <- thetaE.vals[thetaE.vals>(1/10)][which.min(dif[thetaE.vals>(1/10)])]
> limites.dev <- c(linf, lsup)
> limites.dev
[1] 0.0624 0.1504
> segments(limites.dev, rep(corte,2), limites.dev, rep(0,2))
```

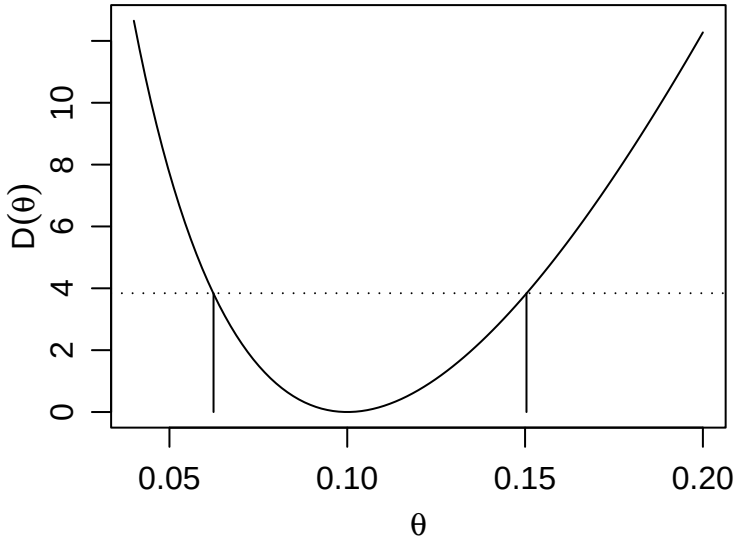

Figura 34: Obtenção gráfica do IC para o parâmetro θ da Exp(θ) para uma amostra de tamanho 20 e média 10.

Note que neste código procuramos primeiro e limite inferior entre os valores menores que a estimativa do parâmetro (1/10) e depois o limite superior entre os valores maiores que esta estimativa. Embora este procedimento bastante simples e sujeito a imprecisão podemos torná-lo quão preciso quanto quisermos bastando para isto definir um vetor com menor espaçamento para os valores para o parâmetro, por exemplo poderiamos usar `thetaE.vals <- seq(0.04,0.20,l=1001)`.

24.2.2 Aproximação quadrática da verossimilhança

Nesta abordagem aproximamos a função deviance por uma função quadrática obtida pela expansão por série de Taylor ao redor do estimador de máxima verossimilhança:

$$D(\theta) \approx n \left(\frac{\theta - \hat{\theta}}{\hat{\theta}} \right)^2 .$$

A Figura 35 obtida com os comandos mostra o gráfico desta função deviance aproximada. A Figura também mostra os IC's obtido com esta função. Para a aproximação quadrática os limites dos intervalos são facilmente determinados analiticamente e neste caso dados por:

$$\left(\hat{\theta}(1 - \sqrt{c^*/n}) \ , \ \hat{\theta}(1 + \sqrt{c^*/n}) \right).$$

```
> devap.exp <- function(theta, n, xbar){n * (xbar *(theta - (1/xbar)))^2}
> devap.vals <- devap.exp(thetaE.vals, n=20, xbar=10)
> plot(thetaE.vals, devap.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> corte <-  qchisq(0.95, df = 1)
> abline(h = corte, lty=3)
> limites.devap <- c((1/10)*(1 - sqrt(corte/20)), (1/10)*(1 + sqrt(corte/20)))
> limites.devap
[1] 0.05617387 0.14382613
> segments(limites.devap, rep(corte,2), limites.devap, rep(0,2))
```

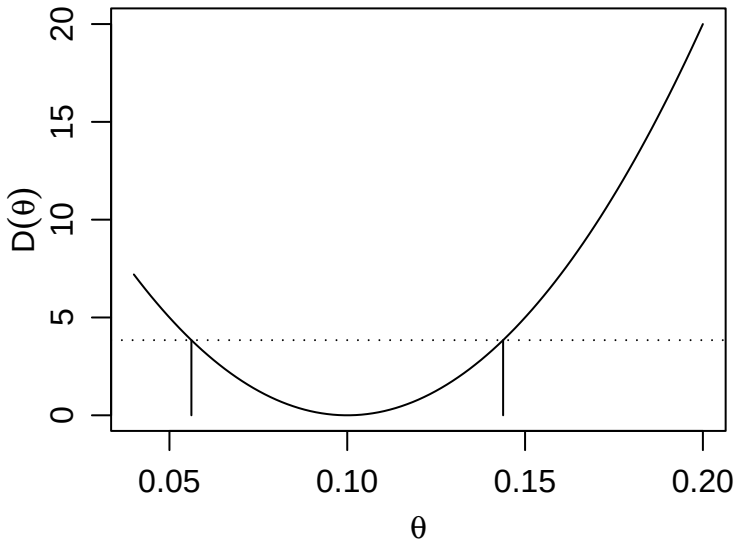

Figura 35: Função deviance obtida pela aproximação quadrática para Exp(θ) e uma amostra de tamanho 20 e média 10.

24.3 Comparando as duas estratégias

Examinando os limites dos intervalos encontrados anteriormente podemos ver que são diferentes. Vamos agora colocar os resultados pelos dois métodos em um mesmo gráfico (Figura 36) para comparar os resultados.

```
> plot(thetaE.vals, dev.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> lines(thetaE.vals, devap.vals, lty=2)
> abline(h = corte, lty=3)
> segments(limites.dev, rep(corte,2), limites.dev, rep(0,2))
> segments(limites.devap, rep(corte,2), limites.devap, rep(0,2), lty=2)
> legend(0.07, 12, c('deviance', 'aproximacão quadrática'), lty=c(1,2), cex=0.7)
```

Vamos agora examinar o efeito do tamanho da amostra na função deviance e sua aproximação quadrática. A Figura 32 mostra as funções para três tamanhos de amostra, $n = 10, 30$ e 100 que são obtidas com os comandos abaixo onde vemos que a aproximação fica cada vez melhor com o aumento do tamanho da amostra.

```
> thetaE.vals <- seq(0.04, 0.20, l=101)
> dev10.vals <- dev.exp(thetaE.vals, n=10, xbar=10)
```

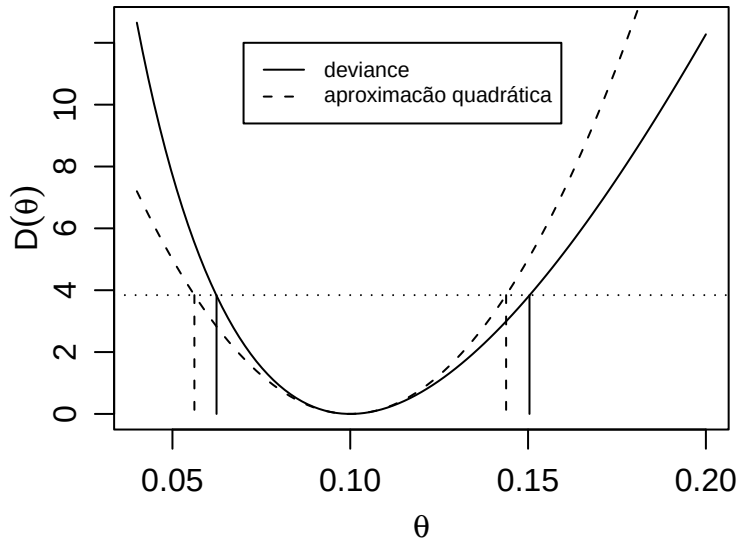

Figura 36: Comparação dos IC's de confiança obtidos pela solução gráfica/numérica (linha sólida) e pela aproximação quadrática (linha tracejada) para o parâmetro θ da Exp(θ) para uma amostra de tamanho 20 e média 10.

```
> plot(thetaE.vals, dev10.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> devap10.vals <- devap.exp(thetaE.vals, n=10, xbar=10)
> lines(thetaE.vals, devap10.vals, lty=2)
> abline(h = qchisq(0.95, df = 1), lty=3)
>
> dev30.vals <- dev.exp(thetaE.vals, n=30, xbar=10)
> plot(thetaE.vals, dev30.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> devap30.vals <- devap.exp(thetaE.vals, n=30, xbar=10)
> lines(thetaE.vals, devap30.vals, lty=2)
> abline(h = qchisq(0.95, df = 1), lty=3)
>
> dev100.vals <- dev.exp(thetaE.vals, n=100, xbar=10)
> plot(thetaE.vals, dev100.vals, ty='l',
+      xlab=expression(theta), ylab=expression(D(theta)))
> devap100.vals <- devap.exp(thetaE.vals, n=100, xbar=10)
> lines(thetaE.vals, devap100.vals, lty=2)
> abline(h = qchisq(0.95, df = 1), lty=3)
```

24.4 Exercícios

1. Seja 14.1, 30.0, 19.6, 28.2, 12.5, 15.2, 17.1, 11.0, 25.9, 13.2, 22.8, 22.1 a.a. de uma distribuição normal de média 20 e variância σ^2.

   (a) Obtenha a função deviance para σ^2 e faça o seu gráfico.

   (b) Obtenha a função deviance para σ e faça o seu gráfico.

   (c) Obtenha os IC's a 90% de confiança.

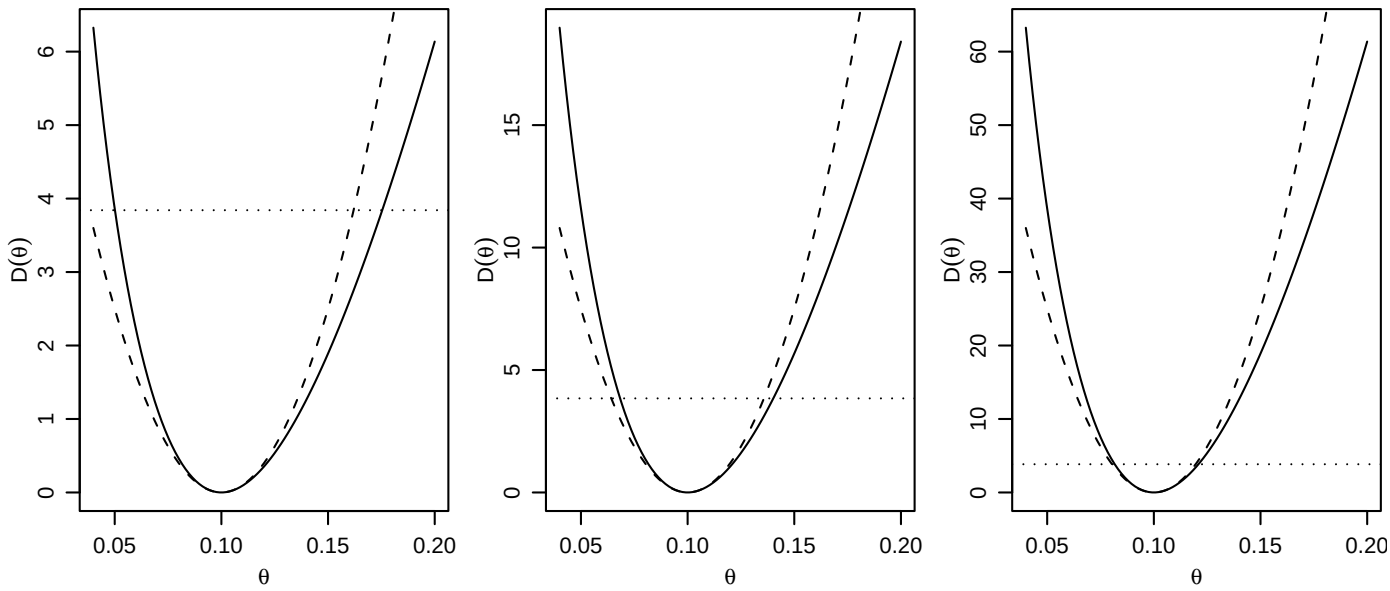

Figura 37: Funções deviance e deviance aproximada para o parâmetro θ da Exp(θ) em amostras de média 10 e tamanhos $n = 10$ (esquerda), 30 (centro) e 100 (direita).

2. Repita as análises mostradas no exemplo acima da distribuição exponencial mas agora utilizando a seguinte parametrização para a função de densidade:

$$f(x) = \frac{1}{\lambda} \exp\left(-x/\lambda\right) \quad x \geq 0.$$

Discuta as diferenças entre os resultados obtidos nas duas parametrizações.

25 Mais sobre intervalos de confiança

Nesta aula vamos nos aprofundar um pouco mais na teoria de intervalos de confiança. São ilustrados os conceitos de:

- obtenção de intervalos de confiança pelo método da quantidade pivotal,
- resultados diversos da teoria de verossimilhança,
- intervalos de cobertura.

Voce vai precisar conhecer de conceitos do método da quantidade pivotal, a propriedade de normalidade assintótica dos estimadores de máxima verossimilhança e a distribuição limite da função deviance.

25.1 Inferência para a distribuição Bernoulli

Os dados abaixo são uma amostra aleatória da distribuição $Bernoulli(p)$.

0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1

Desejamos obter:

(a) o gráfico da função de verossimilhança para p com base nestes dados

(b) o estimador de máxima verossimilhança de p, a informação observada e a informação de Fisher

(c) um intervalo de confiança de 95% para p baseado na normalidade assintótica de $\hat{p}$

(d) compare o intervalo obtido em (b) com um intervalo de confiança de 95% obtido com base na distribuição limite da função deviance

(e) a probabilidade de cobertura dos intervalos obtidos em (c) e (d). (O verdadeiro valor de p é 0.8)

Primeiramente vamos entrar com os dados na forma de um vetor.

```
> y <- c(0,0,0,1,1,0,1,1,1,1,0,1,1,0,1,1,1,1,0,1,1,1,1,1,1)
```

(a)
Vamos escrever uma função para obter a função de verossimilhança.

```
> vero.binom <- function(p, dados){
+   n <- length(dados)
+   x <- sum(dados)
+   return(dbinom(x, size = n, prob = p, log = TRUE))
+ }
```

Esta função exige dados do tipo 0 ou 1 da distribuição Bernoulli. Entretanto às vezes temos dados Binomiais do tipo n e x (número x de sucessos em n observações). Por exemplo, para os dados acima teríamos $n = 25$ e $x = 18$. Vamos então escrever a função acima de forma mais geral de forma que possamos utilizar dados disponíveis tanto em um quanto em ou outro formato.

```
> vero.binom <- function(p, dados, n = length(dados), x = sum(dados)){
+   return(dbinom(x, size = n, prob = p, log = TRUE))
+ }
```

Agora vamos obter o gráfico da função de verossimilhança para estes dados. Uma forma de fazer isto é criar uma sequência de valores para o parâmetro p e calcular o valor da verossimilhança para cada um deles. Depois fazemos o gráfico dos valores obtidos contra os valores do parâmetro. No R isto pode ser feito com os comandos abaixo que produzem o gráfico mostrado na Figura 39.

```
> p.vals <- seq(0.01,0.99,l=99)
> logvero <- sapply(p.vals, vero.binom, dados=y)
> plot(p.vals, logvero, type="l")
```

Note que os três comandos acima podem ser substituídos por um único que produz o mesmo resultado:

```
> curve(vero.binom(x, dados=y), from = 0, to = 1)
```

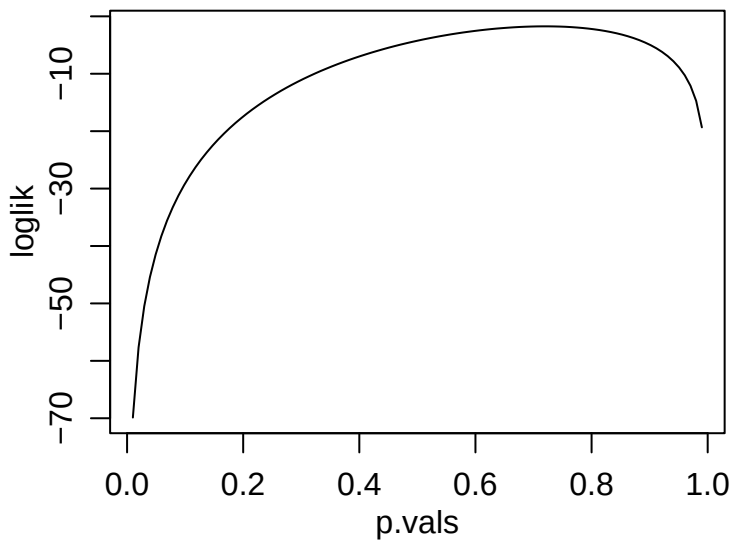

Figura 38: Função de verossimilhança para o parâmetro p da distribuição Bernoulli.

(b)
Dos resultados para distribuição Bernoulli sabemos que o estimador de máxima verossimilhança é dado por

$$\hat{p} = \frac{\sum_{i=1}^{n} y_i}{n}$$

e que a informação esperada coincide com a esperança observada e sendo iguais a:

$$I(\hat{p}) = \frac{n}{\hat{p}(1-\hat{p})}$$

. Para obter os valores numéricos para a amostra dada utilizamos os comandos:

```
> p.est <- mean(y)
> arrows(p.est, vero.binom(p.est,dados=y), p.est, min(logvero))
> io <- ie <- length(y)/(p.est * (1 - p.est))
> io
[1] 124.0079
> ie
[1] 124.0079
```

(c)
O intervalo de confiança baseado na normalidade assintótica do estimador de máxima verossimilhança é dado por:

$$\left(\hat{p} - z_{\alpha/2}\sqrt{I(\hat{p})} \;\;,\;\; \hat{p} + z_{\alpha/2}\sqrt{I(\hat{p})}\right)$$

e para obter o intervalo no R usamos os comandos a seguir.

```
> ic1.p <- p.est + qnorm(c(0.025, 0.975)) * sqrt(1/ie)
> ic1.p
[1] 0.5439957 0.8960043
```

(d)

Vamos agora obter o intervalo baseado na função deviance graficamente. Primeiro vamos escrever uma função para calcular a deviance que vamos chamar de `dev.binom`, lembrando que a deviance é definida pela expressão:

$$D(p) = 2\{l(\hat{p}) - l(p)\}.$$

```
> dev.binom <- function(p, dados, n = length(dados), x =sum(dados)){
+   p.est <- x/n
+   vero.p.est <- vero.binom(p.est, n = n, x = x)
+   dev <- 2 * (vero.p.est - vero.binom(p, n = n, x = x))
+   dev
+ }
```

E agora vamos fazer o gráfico de forma similar ao que fizemos para função de verossimilhança, definindo uma sequência de valores, calculando as valores da deviance e traçando a curva.

```
> p.vals <- seq(0.3, 0.95, l=101)
> dev.p <- dev.binom(p.vals, dados=y)
> plot(p.vals, dev.p, typ="l")
```

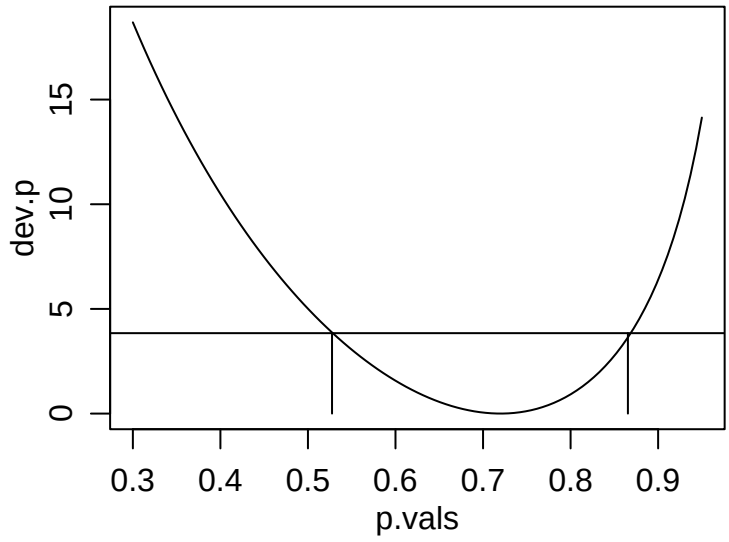

Figura 39: Função deviance para o parâmetro p da distribuição Bernoulli.

Agora usando esta função vamos obter o intervalo gráficamente. Para isto definimos o ponto de corte da função usando o fato que a função deviance $D(p)$ tem distribuição assintótica χ^2. Nos comandos a seguir primeiro encontramos o ponto de corte para o nível de confiança de 95%. Depois traçamos a linha de corte com o comando `abline`. Os comandos seguintes consistem em uma forma simples e aproximada para encontrar os pontos onde a linha conta a função, que definem o intervalo de confiança.

```
> corte <- qchisq(0.95, df=1)
> abline(h=corte)
> dif <- abs(dev.p - corte)
> inf <- ifelse(p.est==0, 0, p.vals[p.vals<p.est][which.min(dif[p.vals<p.est])])
```

```
> sup <- ifelse(p.est==1, 1, p.vals[p.vals>p.est][which.min(dif[p.vals>p.est])])
> ic2.p <- c(inf, sup)
> ic2.p
[1] 0.5275 0.8655
> segments(ic2.p, 0, ic2.p, corte)
```

Agora que já vimos as duas formas de obter o IC passo a passo vamos usar os comandos acima para criar uma função geral para encontrar IC para qualquer conjunto de dados e com opções para os dois métodos.

```
> ic.binom <- function(dados, n=length(dados), x=sum(dados),
+                      nivel = 0.95,
+                      tipo=c("assintotico", "deviance")){
+   tipo <- match.arg(tipo)
+   alfa <- 1 - nivel
+   p.est <- x/n
+   if(tipo == "assintotico"){
+     se.p.est <- sqrt((p.est * (1 - p.est))/n)
+     ic <- p.est + qnorm(c(alfa/2, 1-(alfa/2))) * se.p.est
+   }
+   if(tipo == "deviance"){
+     p.vals <- seq(0,1,l=1001)
+     dev.p <- dev.binom(p.vals, n = n, x = x)
+     corte <- qchisq(nivel, df=1)
+     dif <- abs(dev.p - corte)
+     inf <- ifelse(p.est==0, 0, p.vals[p.vals<p.est][which.min(dif[p.vals<p.est])])
+     sup <- ifelse(p.est==1, 1, p.vals[p.vals>p.est][which.min(dif[p.vals>p.est])])
+     ic <- c(inf, sup)
+   }
+   names(ic) <- c("lim.inf", "lim.sup")
+   ic
+ }
```

E agora vamos utilizar a função, primeiro com a aproximação assintótica e depois pela deviance. Note que os intervalos são diferentes!

```
> ic.binom(dados=y)
  lim.inf   lim.sup
0.5439957 0.8960043
> ic.binom(dados=y, tipo = "dev")
lim.inf lim.sup
  0.528   0.869
```

(e)

O cálculo do intervalo de cobertura consiste em:

1. simular dados com o valor especificado do parâmetro;
2. obter o intervalo de confiança;
3. verificar se o valor está dentro do intervalo
4. repetir (1) a (3) e verificar a proporção de simulações onde o valor está no intervalo.

Espera-se que a proporção obtida seja o mais próximo possível do nível de confiança definido para o intervalo.

Para isto vamos escrever uma função implementando estes passos e que utiliza internamente `ic.binom` definida acima.

```
> cobertura.binom <- function(n, p, nsim, ...){
+   conta <- 0
+   for(i in 1:nsim){
+     ysim <- rbinom(1, size = n, prob = p)
+     ic <- ic.binom(n = n, x = ysim, ...)
+     if(p > ic[1] & p < ic[2]) conta <- conta+1
+   }
+   return(conta/nsim)
+ }
```

E agora vamos utilizar esta função para cada um dos métodos de obtenção dos intervalos.

```
> set.seed(123)
> cobertura.binom(n=length(y), p=0.8, nsim=1000)
[1] 0.885
> set.seed(123)
> cobertura.binom(n=length(y), p=0.8, nsim=1000, tipo = "dev")
[1] 0.954
```

Note que a cobertura do método baseado na deviance é muito mais próxima do nível de 95% o que pode ser explicado pelo tamanho da amostra. O IC assintótico tende a se aproximar do nível nominal de confiança na medida que a amostra cresce.

25.2 Exercícios

1. Re-faça o ítem (e) do exemplo acima com $n = 10$, $n = 50$ e $n = 200$. Discuta os resultados.

2. Seja $X_1, X_2, \cdots, X_n$ uma amostra aleatória da distribuição $U(0, \theta)$. Encontre uma quantidade pivotal e:

   (a) construa um intervalo de confiança de 90% para θ

   (b) construa um intervalo de confiança de 90% para $\log \theta$

   (c) gere uma amostra de tamanho $n = 10$ da distribuição $U(0, \theta)$ com $\theta = 1$ e obtenha o intervalo de confiança de 90% para θ. Verifique se o intervalo cobre o verdadeiro valor de θ.

   (d) verifique se a probabilidade de cobertura do intervalo é consistente com o valor declarado de 90%. Para isto gere 1000 amostras de tamanho $n = 10$. Calcule intervalos de confiança de 90% para cada uma das amostras geradas e finalmente, obtenha a proporção dos intervalos que cobrem o verdadeiro valor de θ. Espera-se que este valor seja próximo do nível de confiança fixado de 90%.

   (e) repita o item (d) para amostras de tamanho $n = 100$. Houve alguma mudança na probabilidade de cobertura?

   Note que se $-\sum_i^n \log F(x_i; \theta) \sim \Gamma(n, 1)$ então $-2\sum_i^n \log F(x_i; \theta) \sim \chi^2_{2n}$.

3. Acredita-se que o número de trens atrasados para uma certa estação de trem por dia segue uma distribuição Poisson(θ), além disso acredita-se que o número de trens atrasados em cada dia seja independente do valor de todos os outros dias. Em 10 dias sucessivos, o número de trens atrasados foi registrado em:

```
5 0 3 2 1 2 1 1 2 1
```

Obtenha:

(a) o gráfico da função de verossimilhança para θ com base nestes dados

(b) o estimador de máxima verossimilhança de θ, a informação observada e a informação de Fisher

(c) um intervalo de confiança de 95% para o número médio de trens atrasados por dia baseando-se na normalidade assintótica de $\hat{\theta}$

(d) compare o intervalo obtido em (c) com um intervalo de confiança obtido com base na distribuição limite da função deviance

(e) o estimador de máxima verossimilhança de ϕ, onde ϕ é a probabilidade de que não hajam trens atrasados num particular dia. Construa intervalos de confiança de 95% para ϕ como nos itens (c) e (d).

4. Encontre intervalos de confiança de 95% para a média de uma distribuição Normal com variância 1 dada a amostra

```
9.5 10.8 9.3 10.7 10.9 10.5 10.7 9.0 11.0 8.4
10.9 9.8 11.4 10.6 9.2 9.7 8.3 10.8 9.8 9.0
```

baseando-se:

(a) na distribuição assintótica de $\hat{\mu}$

(b) na distribuição limite da função deviance

5. Acredita-se que a produção de trigo, X_i, da área i é normalmente distribuída com média θz_i, onde z_i é quantidade (conhecida) de fertilizante utilizado na área. Assumindo que as produções em diferentes áreas são independentes, e que a variância é conhecida e igual a 1, ou seja, $X_i \sim N(\theta z_i, 1)$, para $i = 1, \cdots, n$:

(a) simule dados sob esta distribuição assumindo que $\theta = 1.5$, e $z = (1, 2, 3, 4, 5)$. Visualize os dados simulados através de um gráfico de $(z \times x)$

(b) encontre o EMV de θ, $\hat{\theta}$

(c) mostre que $\hat{\theta}$ é um estimador não viciado para θ (lembre-se que os valores de z_i são constantes)

(d) obtenha um intervalo de aproximadamente 95% de confiança para θ baseado na distribuição assintótica de $\hat{\theta}$

26 Rodando o R dentro do Xemacs

Esta página contém instruções sobre como rodar o programa estatístico R dentro do editor Xemacs.

Este procedimento permite um uso ágil do programa R com facilidades para gravar o arquivo texto com os comandos de uma sessão e uso das facilidades programadas no pacote `ESS (Emacs Speaks Statistics)` que é um complemento do editor Xemacs.

Para utilizar esta funcionalidade deve-se seguir os seguintes passos:

1. Instalar o programa R. *(clique para baixar programa de instalação)*
   Assume-se aqui que o R esteja instalado em:
   `C:\ARQUIVOS DE PROGRAMAS\rw`

   Note que na instalação do R é sugerido um nome do diretório de instalação do tipo `rw1071`. Sugiro que voce mude para `rw` apanes para não ter que alterar a configuração abaixo toda vez que atualizar a sua versão do R.

2. Instalar o programa Xemacs. As versões mais recentes já veem com o pacote *ESS* incluído. *(clique para baixar programa de instalação)*

3. Modifique a variável `PATH` do seu computador adicionando a ela o caminho para o diretório `bin` do R. No **windows 98** isto é feito modificando o arquivo `C:\AUTOEXEC.BAT` inserindo a seguinte linha no final do arquivo
   `SET PATH=%PATH%;C:\ARQUIVOS DE PROGRAMA\rw\bin`
   No **Windows XP** isto é feito adicionado este diretório à esta variável de ambiente.

4. Inicie o programa Xemacs e clique na barra de ferramentas em:
   `Options ---> Edit init file`

5. Adicionar a seguinte linha:
   `(require 'ess-site)`

6. Gravar o arquivo e sair do Xemacs.

7. Se usar o **Windows 98**: reinicialize o seu computador.

8. Tudo pronto! Para começar a utilizar basta iniciar o programa Xemacs. Para iniciar o R dentro do Xemacs use a combinação de teclas:
   `ESC SHIFT-X SHIFT-R`

9. Use sempre a extensão `.R` para os seus arquivos de comandos do R.

10. Lembre-se que voce pode usar `ESC CTRL-X-2` para dividir a tela em 2.