Curso de Introdução ao R

Tiago Vieira; Paulo Cerqueira Jr.; Fábio Santos; Diogo Mendes

7 de maio de 2009

1 O que é R?

1.1 Introdução a R

R é uma linguagem e ambiente para computação estatística e gráficos. é um projeto GNU que é similar à linguagem e ambiente S que foi desenvolvida no Bell Laboratories (anteriormente AT&T, agora Lucent Technologies) por John Chambers e colegas. R pode ser considerada como uma implementação diferente da S. Há algumas diferenças importantes, mas muito código para S funciona inalterado em R.

R fornece uma ampla variedade de técnicas estatísticas (modelagem linear e não linear, testes estatísticos clássicos, análise de séries temporais, classificação, agrupamento, ...) e gráficos, e é altamente extensível. A linguagem S é muitas vezes o veículo de escolha para pesquisa em metodologia estatística, e R fornece uma rota Open Source para participação naquela atividade.

Um dos pontos fortes de R é a facilidade com que gráficos bem-desenhados com qualidade para publicação podem ser produzidos, incluindo símbolos matemáticos e fórmulas quando necessário. Muitos cuidados tém sido feitos sobre as definições padrão para as menores escolhas em desenho, entretanto o usuário retém controle total.

R é disponível como Software Livre sob os termos da Licença Pública Geral GNU da Free Software Foundation na forma de código fonte. Ela compila e funciona em uma grande variedade de plataformas UNIX e sistemas similares (incluindo FreeBSD e Linux). Ele compila e funciona em Windows 9x/NT/2000 e MacOS.

1.2 O ambiente R

R é um conjunto integrado de facilidades de software para manipulação de dados, cálculo e visualização gráfica. Ele inclui

- uma facilidade efetiva para manipulação e armazenagem de dados,
- um conjunto de operadores para cálculos sobre quadros de dados, em particular as matrizes,
- uma grande e coerente coleção integrada de ferramentas intermediárias para análise de dados,
- facilidades gráficas para análise de dados e visualização na tela ou impressa,

- uma linguagem de programação bem desenvolvida, simples e efetiva que inclui condicionais, alças, funções recursivas definidas pelo usuário, e facilidades para entrada e saída.

O termo "ambiente"pretende caracterizar R como um sistema totalmente planejado e coerente, em vez de uma aglomeração de ferramentas muito específicas e inflexíveis, como é o caso com outros softwares de análise de dados.

R, bem como S, é desenhada ao redor de uma verdadeira linguagem de computador, e permite aos usuários acrescentar funcionalidade adicional por definição de novas funções. Muito do sistema é escrita no dialeto R da S, que faz com que seja fácil para usuários seguir as escolhas algorítimicas feitas. Para tarefas computacionalmente-intensivas, C, C++ e código Fortran podem ser ligados e chamados na hora de calcular. Usuários avançados podem escrever código C para manipular objetos R diretamente.

Muitos usuários pensam em R como um sistema estatístico. Nós preferimos pensar nele como um ambiente dentro do qual técnicas estatísticas são implementadas. R pode ser estendido (facilmente) através de pacotes. Há cerca de oito pacotes fornecidos com a distribuição R e muitos outros são disponíveis através da família CRAN de sítios na Internet cobrindo uma ampla variedade de estatísticas modernas.

R tem seu próprio formato de documentação, parecido com LaTeX, que é usado para fornecer documentação compreensiva, tanto on-line e em uma variedade de formatos, como impressa.

2 Instalando o R

Há várias formas de se instalar o R, que basicamente pode ser reunidas em duas formas: (i) instalação usando arquivos binários ou (ii) instalação compilando os arquivos fonte.

2.1 A partir de arquivos compilados

Para isto é necessário baixar o arquivo de instalação adequado a seu sistema operacional e rodar a instalação. Nas áreas de download do R, como por exemplo em http://cran.br.r-project.org você irá encontrar arquivos de instalação para os sistemas operacionais Linux, Windows e Macintosh.

No caso do Windows siga os links:
`Windows (95 and later) --> base`
e copie o arquivo de instalação `.exe` que deve ser rodado para efetuar a instalação.

Além disto o R está disponível como pacote de diversas distribuições LINUX tais como Ubuntu, Debian, RedHat (Fedora), Suse, entre outras. Por exemplo, para instalar no Debian ou Ubuntu LINUX pode-se fazer (com privilégios de **root**):

No arquivo `/etc/apt/sources.list` adicione uma entrada:

- Ubuntu:

  ```
  deb http://cran.R-project.org/bin/linux/ubuntu intrepid/
  ```

- Debian:

  ```
  deb http://cran.R-project.org/bin/linux/debian
  ```

  atualize a lista de pacotes com:

  ```
  apt-get update
  ```

  A seguir rode na linha de comando do LINUX:

```
apt-get install r-base r-base-core r-recommended apt-get install
r-base-html r-base-latex r-doc-html r-doc-info r-doc-pdf
```

Além destes há diversos outros pacotes Debian para instalação dos pacotes adicionais do R e outros recursos.

2.2 Compilando a partir da fonte

Neste caso pode-se baixar o arquivo fonte do R (`.tar.gz`) que deve ser descompactado e instruções para compilação devem ser seguidas.

Um *script* para facilitar e automatizar a compilação: Eu pessoalmente prefiro no LINUX rodar os comandos disponíveis neste arquivo (http://www.leg.ufpr.br/ paulojus/embrapa/Rembrapa/Rpatch) que baixa o arquivo fonte e compilam o R na máquina do usuário rodando (como **root** ou sudo):
`./Rpatch --install`

Mas note que isto requer que outras dependencias estejam instaladas no sistema. Se quiser que estas dependências sejam automaticamente instaladas use:
`./Rpatch --install --deps`

E para ver todas as opções do script: `./Rpatch --help`

Este *script* funciona integralmente em LINUX ***Ubuntu*** e ***Debian*** e em outras distribuições LINUX sem utilizar a opção `--deps` pois esta utiliza o mecanismo `apt-get` para instalação de pacotes.

Informações completas e detalhadas sobre a instalação do R podem ser obtidas o manual R Instalation and Administration.

2.3 Instalando e usando pacotes (packages) do R

O programa R é composto de 3 partes básicas:

1. **R-base**, o "coração"do R que contém as funções principais disponíveis quando iniciamos o programa,

2. os **pacotes recomendados** (*recommended packages*) que são instalados junto com o R-base mas não são carregados quando iniciamos o programa. Por exemplo os pacotes MASS, lattice, nlme são pacotes recomendados - e há vários outros. Para usar as funções destes pacotes deve-se carregá-los antes com o comando `library()`. Por exemplo o comando `library(MASS)` carrega o pacote MASS.

3. os **pacotes contribuídos** (*contributed packages*) não são instalados junto com o R-base. Estes pacotes disponívies na página do R são pacotes oficiais. Estes pacotes adicionais fornecem funcionalidades específicas e para serem utilizados devem ser copiados, instalados e carregados, conforme explicado abaixo. Para ver a lista deste pacotes com uma descrição de cada um deles acesse a página do R e siga os links para CRAN e Package Sources.

Antes de instalar o pacote você pode ver se ele já está instalado/disponível. Para isto inicie o R e digite o comando:

```
> require(NOME_DO_PACOTE)
```

Se ele retornar `T` é porque o pacote já está instalado/disponível e você não precisa instalar. Se retornar `F` siga os passos a seguir.

A instalação e uso dos pacotes vai depender do seu sistema operacional e os privilégios que você tem no seu sistema. Nas explicações a seguir assume-se que você está em uma máquina conectada à internet. O comando mostrado vai copiar o arquivo para seu computador, instalar o pacote desejado e ao final perguntar se você quer apagar o arquivo de instalação (responda `Y` (*yes*))

1. Instalação em máquinas com Windows98 ou em máquinas NT/XP/LINUX com senha de administrador (instalação no sistema).

   Neste caso basta usar o comando `install.packages()` com o nome do pacote desejado entre aspas. Por exemplo para instalar o pacote CircStats digite:

   ```
   install.packages('CircStats')
   ```

O pacote vai ser instalado no sistema e ficar disponível para tudos os usuários. Para usar o pacote basta digitar `library(CircStats)` ou `require(CircStats)`.

2. Instalação em máquinas NT/XP/LINUX na conta do usuário, sem senha de administrador (instalação na conta do usuário)

   Neste caso o usuário deve abrir um diretório para instalar o pacote e depois rodar o comando de instalação especificando este diretório. Por exemplo, suponha que você queira instalar o pacote CircStats na sua conta no sistema Linux do LABEST. Basta seguir os seguintes passos.

   (a) Na linha de comando do Linux abra um diretório (se já não existir) para instalar os pacotes. Por exemplo, chame este diretório `Rpacks`:

   ```
   % mkdir -p ~/Rpacks
   ```

   (b) Inicie o R e na linha de comando do R digite:

   ```
   > install.packages("CircStats", lib="~/Rpacks")
   ```

   (c) Neste caso o pacote vai ser instalado na área do usuário e para carregar o pacote digite:

   ```
   > library(CircStats, lib="~/Rpacks")
   ```

   NOTA: no Windows você pode, alternativamente, instalar usando o menu do R selecionando a opção `PACKAGES - INSTALL FROM CRAN`.

2.3.1 Pacotes não-oficiais

Além dos pacotes contribuídos existem diversos pacotes não-oficiais disponívies em outros locais na web. Em geral o autor fornece instruções para instalação. As instruções gerais para instalação são as seguintes:

- **Linux**: Os pacotes para Linux em geral vem em um arquivo tipo `PACOTE.tar.gz` e são instalados com um dos comandos abaixo (use o primeiro se for administrador do sistema e o segundo como usuário comum).

  ```
  R INSTALL PACOTE.tar.gz
  ```

  ou

  ```
  R INSTALL -l ~/Rpacks PACOTE.tar.gz
  ```

- **Windows**: No menu do R use a opção `PACKAGES - INSTALL FROM LOCAL .ZIP FILE`

3 Manipulação de dados

3.1 Operações Aritméticas

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

```
> 1+2+3     # somando estes números ...
[1] 6
> 2+3*4     # um pouquinho mais complexo
[1] 14
> 3/2+1
[1] 2.5
> 4*3**3    # potências são indicadas por ** ou ^
[1] 108
```

Nos exemplos acima mostramos uma operação simples de soma. Note no segundo e terceiro comandos a prioridade entre operações. No último vimos que a operação de potência é indicada por `**` . Note que alternativamente pode-se usar o símbolo `^`, por exemplo `4*3^3` produziria o mesmo resultado mostrado acima.

O R também disponibiliza funções usuais como as que são encontradas em uma calculadora:

```
> sqrt(2)
[1] 1.414214
> sin(3.14159)   # seno de (Pi radianos) é zero
[1] 2.65359e-06
```

Note que o ângulo acima é interpretado como sendo em radianos. O valor Pi está disponível como uma constante. Tente isto:

```
> sin(pi)
[1] 1.224606e-16
```

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

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

3.2 Objetos

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

```
> x <- 10
> x
[1] 10
```

O dígito 1 entre colchetes indica que o conteúdo exibido inicia-se com o primeiro elemento do objeto `x`. Você pode armazenar um valor em um objeto com certo nome usando o símbolo `<-`. 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
```

Neste caso lê-se: *x "recebe"a raiz quadrada de 2.* Alternativamente ao símbolo `<-` usualmente utilizado para atribuir valores a objetos, pode-se ainda usar os símbolos `->` ou `=` (este apenas em versões mais recentes do R). O símbolo `_` que podia ser usado em versões mais antigas no R tornou-se inválido para atribuir valores a objetos em versões mais recentes e passou a ser permitido nos nomes dos objetos. As linhas a seguir produzem o mesmo resultado.

```
> x <- sin(pi)
> x
[1] 1.224606e-16
> x <- sin(pi)
> x
[1] 1.224606e-16
> x = sin(pi)
> x
[1] 1.224606e-16
```

Neste material será dada preferência ao primeiro símbolo. Usuários pronunciam o comando dizendo que o objeto "recebe"(em inglês "gets") um certo valor. Por exemplo em `x <- sqrt(2)` dizemos que "x recebe a raiz quadrada de 2". Como pode ser esperado você 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 comeaçr com uma letra e podem conter letras, números e pontos. Um fato importante é que o R distingue letras maiúsculas e minúsculas nos nomes dos objetos, por exemplo dados, Dados e DADOS serão interpretados como nomes de três objetos diferentes pela linguagem. *DICA*: tente atribuir nomes que tenham um significado lógico, relacionado ao trabalho e dados em questão. Isto facilita lidar com um grande número de objetos. Ter nomes como `a1` até `a20` pode causar confusão
... A seguir estão alguns exemplos 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
> a1 <- sqrt 10
> a-1 <- 99
> sqrt(x) <- 10
```

No primeiro caso o nome não começa com uma letra, o que é obrigatório, `a99` é um nome válido, mas `99a` não é. No segundo faltou um parentesis na função `sqrt`, o correto seria `sqrt(10)`. No terceiro caso o hífen não é permitido, por ser o mesmo sinal usado em operações de subtração. O último caso é um comando sem sentido.

Também 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)
> a1
[1] 2 5 8
> a2 <- c(23, 56, 34, 23, 12, 56)
> a2
[1] 23 56 34 23 12 56
```

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 podese usar mecanismos do R para facilitar a entrada dos dados como vetores. Examine os seguintes exemplos.

```
> a3 <- 1:10
> a3
[1] 1 2 3 4 5 6 7 8 9 10
> a4 <- (1:10) * 10
> a4
```

```
[1] 10 20 30 40 50 60 70 80 90 100
> a5 <- rep(3, 5)
> a5
[1] 3 3 3 3 3
> a6 <- rep(c(5, 8), 3)
> a6
[1] 5 8 5 8 5 8
> a7 <- rep(c(5, 8), each = 3)
> a7
[1] 5 5 5 8 8 8
```

3.2.1 Tipos de Objetos

Vimos que R trabalha com objetos que são, naturalmente, caracterizada por seus nomes e os seus conteúdos, mas também por atributos que especificam o tipo de Dados representados por um objeto. A fim de compreender a utilidade destes atributos, considerar uma variável que assume o valor 1, 2 ou 3: tal variável poderia ser uma variável inteira (por exemplo, o número de ovos em um ninho), ou a codificação de uma variável categórica (por exemplo, o sexo em algumas populações de crustáceos: masculino, feminino, ou hermafroditas).

É claro que a análise estatística dessa variável não será a mesma em ambos os casos: com R, os atributos do objeto dão as informações necessárias. Mais tecnicamente, a ação de uma função em um objeto depende dos atributos do mesmo.

Todos os objetos têm dois atributos intrínsecos: a *modalidade* e o *comprimento*. O modo é o tipo básico dos elementos do objeto, há quatro principais modos: numéricos, caractere, complexo, e lógico (Verdadeiro ou Falso). Existem outros modos, mas eles não representam dados, por exemplo, função ou expressão. O comprimento é o número de elementos do objeto. Para exibir o modo e o comprimento de um objeto, pode usar as funçãos do `mode` e do `length`, respectivamente:

```
> x <- 1
> mode(x)
[1] "numeric"
> length(x)
[1] 1
> A <- "Gomphotherium"; compar <- TRUE; z <- 1i
> mode(A); mode(compar); mode(z)
[1] "character" [1] "logical" [1] "complex"
```

Seja qual for o modo, a falta de dados são representadas por NA (*not available*). Um grande valor numérico pode ser especificado com uma notação exponencial:

```
> N <- 2.1e23
> N
[1] 2.1e+23
```

O R representa corretamente valores numéricos não-fititos, como $\pm\infty$ com `inf` e `-inf`, ou valores que não são números com `NaN` (*not a number*).

```
> x <- 5/0
> x
[1] Inf
> exp(x)
[1] Inf
> exp(-x)
[1] 0
> x - x
[1] NaN
```

O seguinte quadro dá uma visão geral dos tipos de objetos que representam dados.

Objeto	**Modos**	**vários modos possíveis no mesmo objeto?**
vector	numérico, caractere, complexo ou lógico	não
factor	numérico ou caractere	não
array	numérico, caractere, complexo ou lógico	não
matrix	numérico, caractere, complexo ou lógico	não
data-frame	numérico, caractere, complexo ou lógico	sim
ts	numérico, caractere, complexo ou lógico	não
list	numérico, caractere, complexo, lógico, função, expressão, ...	sim

Um vetor é uma variável no comumente admitidos significado. Um fator é uma variável categórica. Uma matriz é uma tabela com k dimensões, sendo uma matriz um caso particular da matriz com $k = 2$. Note-se que os elementos de um array ou de uma matriz são todos do mesmo modo. Um data-frame é uma tabela composta com um ou vários vectores e/ou factores com o mesmo comprimento, mas possivelmente de difierentes modos. A 'ts' é um conjunto de dados de séries temporais e, portanto, contém mais atributos como freqüências e datas. Finalmente, uma lista pode conter qualquer tipo de objeto, incluídos listas!

Para um vetor, o seu modo e seu comprimento é sufuciente para descrever os dados. Para outros objetos, outra informação são necessárias e são administradas por atributos não-intrínseco. Entre estes atributos, podemos citar, o que corresponde as dimensões de um objeto. Por exemplo, uma matriz com 2 linhas e 2 colunas tem para `dim` o par de valores [2, 2], mas o seu compreimento é 4.

3.3 Importando Dados

3.3.1 Lendo Dados de um Arquivo Texto

Se os dados já estão disponíveis em formato eletrônico, isto é, já foram digitados em outro programa, você 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, você pode na planilha escolher a opção <SALVAR COMO> e gravar os dados em um arquivo em formato texto.

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

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. Consulte o livro para mais detalhes sobre este dados. Esta conjunto de dados será utilizado como exemplo para importação para o R.

Agora copie o arquivo para sua área de trabalho (*working directory do R*). Para importar este arquivo usamos:

```
milsa <- read.table("milsa.txt")
milsa
```

Agora utilize o seguinte comando:

```
milsa <- read.table("milsa.txt", header = TRUE)
milsa
```

Adicionando o comando `header = TRUE`, informamos ao R que a primeira linha corresponde ao nome das variáveis. Para maiores informações consulte a documentação desta função com `help(read.table)` ou `?read.table`.

Embora `read.table()` seja provavelmente a função mais utilizada existem outras que podem ser úteis e determinadas situações.

- `read.fwf()` é conveniente para ler "fixed width formats"
- `read.fortran()` é semelhante à anterior porém usando o estilo Fortran de especificação das colunas
- `scan()` é uma função internamente utilizadas por outras mas que também pode se usada diretamente pelo usuário.
- o mesmo ocorre para `read.csv()`, `read.delim()` e `read.delim2()`

3.3.2 Importando Dados de Outros Programas

É possível ler dados diretamente de outros formatos que não seja texto (ASCII). Isto em geral é mais eficiente e requer menos memória do que converter para formato texto. Há funções para importar dados diretamente de EpiInfo, Minitab, S-PLUS, SAS, SPSS, Stata, Systat e Octave. Além disto é comum surgir a necessidade de importar dados de planilhas eletrônicas. Muitas funções que permitem a importação de dados de outros programas são implementadas no pacote `foreign`.

```
> require(foreign)
```

A seguir listamos (mas não todas!) algumas destas funções

- `read.dbf()` para arquivos DBASE
- `read.epiinfo()` para arquivos `.REC` do Epi-Info
- `read.mtp()` para arquivos "Minitab Portable Worksheet"
- `read.S()` para arquivos do S-PLUS
- `restore.data()` para "dumps"do S-PLUS
- `read.spss()` para dados do SPSS
- `read.systat()`
- `read.dta()` para dados do STATA
- `read.octave()` para dados do OCTAVE (um clone do MATLAB)
- Para dados do SAS há ao menos duas alternativas:
  - O pacote `foreign` disponibiliza `read.xport()` para ler do formato TRANSPORT do SAS e `read.ssd()` pode escrever dados permanentes do SAS (`.ssd` ou `.sas7bdat`) no formato TRANSPORT, se o SAS estiver disponível no seu sistema e depois usa internamente `read.xport()` para ler os dados no R.
  - O pacote Hmisc disponibiliza `sas.get()` que também requer o SAS no sistema.

Para mais detalhes consulte a documentação de cada função e/ou o manual *R Data Import/Export*.

3.3.3 Acesso a Planilhas e Bancos de Dados Relacionais

É comum que dados estajam armazenados em planilhas eletrônicas tais como *MS-Excel* ou *OpenOffice Spreadsheet*. Nestes caso, embora seja possível exportar a partir destes aplicativos os dados para o formato texto para depois serem lidos no R, possivelmente com `read.table()`, pode ser necessário ou conveniente ler os dados diretamente destes formato. Vamos colocar aqui algumas opções para importar dados do MS-Excel para o R.

- O pacote xlsReadWrite implementa tal funcionalidade para arquivos do tipo `.xls` do *MS-Excel*. No momento que este material está sendo escrito esta pacote está implementado apenas para o sistema operacional Windows.
- Um outro pacote capaz de ler dados diretamente de planilhas é o RODBC. No ambiente windows a função `odbcConnectExcel()` está disponível para estabelecer a conexão. Suponha que você possua um arquivo de uma planilha MS-Excel já no seu diretório (pasta) de trabalho do R chamado `milsa.xls`, que esta planilha tenha os dados na aba `Plan1`. Para importar os dados desta parte da planilha pode-se usar os comandos a seguir.

```
> install.packages("RODBC", dep = T)
> require(RODBC)
> milsa <- odbcConnectExcel("milsa.xls")
> milsa <- sqlFetch(milsa, "Plan1")
> head(milsa)
> milsa
```

- Em sistemas onde a linguagem *Perl* está disponível e a estrutura de planilha é simples sem macros ou fórmulas, pode-se usar a função `xls2cvs` combinada com `read.csv()` ou `read.csv2()`, sendo esta última recomendada para dados com vírgula como caractere separados de decimais. O *Perl* é tipicamente instalado em sistemas Linux/Unix e também livremente disponível para outros sistemas operacionais.

```
> dados <- read.csv(pipe("xls2csv milsa.xls"))
> dados <- read.csv2(pipe("xls2csv milsa.xls"))
```

- O pacote gdata possui a função `read.xls()` que encapsula opções mencionadas anteriormente.

3.3.4 Carregando dados já disponíveis no R

Para carregar conjuntos de dados que são já disponibilizados com o R use o comando `data()`. Por exemplo, abaixo mostramos como carregar o conjunto mtcars que está no pacote datasets e depois como localizar e carregar o conjunto de dados topo.

```
> data(mtcars)
> head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
> find("topo")
```

```
character(0)

> require(MASS)
> data(topo)
> head(topo)

    x   y   z
1 0.3 6.1 870
2 1.4 6.2 793
3 2.4 6.1 755
4 3.6 6.2 690
5 5.7 6.2 800
6 1.6 5.2 800
```

O conjunto mtcars está no pacote datasets que é carregado automaticamente quando iniciamos o R, portanto os dados estão prontamente disponíveis. Ao carregar os dados é criado um objeto mtcars no seu "workspace".

Já o conjunto topo está no pacote `MASS` que não é automaticamente carregado ao iniciar o R e portanto deve ser carregado com `require()` para depois podermos acessar os dados.

A função `data()` pode ainda ser usada para listar os conjutos de dados disponíveis. A primeira chamada a seguir lista os conjuntos de dados dos pacotes carregados. A segunda lista os conjuntos de dados de um pacote específico (no exemplo do pacote `nlme`).

```
data()
data(package="nlme")
```

3.4 Análise Descritiva

3.4.1 Descritiva Univariada

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

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

- Qualitativas
  - Nominais
  - Ordinais
- Quantitativa
  - Driscreta
  - Contínua

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

3.4.2 Descrevendo o Conjunto de Dados "milsa"de Bussab & Morettin

Como exemplo utilizaremos o conjunto de dados "milsa", visto anteriormente. O que queremos aqui é ver como, no programa R:

- entrar com os dados
- fazer uma análise descritiva

Tabela 1: Dados de Bussab & Morettin

Funcionário	Estado Civil	Instrução	Nº de Filhos	Salário	Ano	Mês	Região
1	solteiro	1º Grau	-	4	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	1º 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	1º Grau	1	7.59	34	10	capital
10	solteiro	1º Grau	-	7.44	23	6	outro
11	casado	1º Grau	2	8.12	33	6	interior
12	solteiro	1º Grau	-	8.46	27	11	capital
13	solteiro	1º Grau	-	8.74	37	5	outro
14	casado	1º Grau	3	8.95	44	2	outro
15	casado	1º Grau	0	9.13	30	5	interior
16	solteiro	1º Grau	-	9.35	38	8	outro
17	casado	1º Grau	1	9.77	31	7	capital
18	casado	1º Grau	2	9.8	39	7	outro
19	solteiro	Superior	-	10.53	25	8	interior
20	solteiro	1º Grau	-	10.76	37	4	interior
21	casado	1º Grau	1	11.06	30	9	outro
22	solteiro	1º Grau	-	11.59	34	2	capital
23	solteiro	1º Grau	-	12	41	0	outro
24	casado	Superior	0	12.79	26	1	outro
25	casado	1º Grau	2	13.23	32	5	interior
26	casado	1º Grau	2	13.6	35	0	outro
27	solteiro	1º Grau	-	13.85	46	7	outro
28	casado	1º Grau	0	14.69	29	8	interior
29	casado	1º Grau	5	14.71	40	6	interior
30	casado	1º Grau	2	15.99	35	10	capital
31	solteiro	Superior	-	16.22	31	5	outro
32	casado	1º 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	1º Grau	2	19.4	48	11	capital
36	casado	Superior	3	23.3	42	2	interior

Estes são dados no "estilo planilha", com variáveis de diferentes tipos: categóricas e numéricas (qualitativas e quantitativas). Portanto o formato ideal de armazenamento destes dados no R é o *data.frame*. Para entrar com estes dados noe diretamente no R podemos 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. Lembre-se que se você precisar corrigir algo na digitação você pode fazê-lo abrindo a planilha novamente com o comando `fix(milsa)`.

```
> milsa <- edit(data.frame())
> milsa
```

```
> fix(milsa)
```

Esta é uma opição para digitalizar conjunto de dados no R, mas como já temos digitalizados em planilha temos que utilizar uma das técnicas já demostradas para importar os dados pra o R.

Após a importação dos dados temos que discriminar, se ouver, as variáveis qualitativas como sendo *factores*. No R para realizar esse procedimento são utilizadas as funções `transform()` e `factor()`, como no exemplo a seguir:

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

Vamos ainda definir uma nova variável única idade a partir das variáveis ano e mes que foram digitadas. Para gerar a variável idade em anos fazemos:

```
> milsa <- transform(milsa, idade = ano + mes/12)
> milsa$idade
 [1] 26.25000 32.83333 36.41667 20.83333 40.58333 28.00000 41.00000 43.33333
 [9] 34.83333 23.50000 33.50000 27.91667 37.41667 44.16667 30.41667 38.66667
[17] 31.58333 39.58333 25.66667 37.33333 30.75000 34.16667 41.00000 26.08333
[25] 32.41667 35.00000 46.58333 29.66667 40.50000 35.83333 31.41667 36.33333
[33] 43.58333 33.58333 48.91667 42.16667
```

3.4.3 Estatísticas Descritivas

O primeiro passo para uma análise preliminar de um conjunto de dados é obter algumas estatísticas descritivas que extraem algumas informações sobre as variáeis em estudo. Com a função `summary()` podemos obter algumas destas estatísticas, como o número de valores nulos, o mínimo, o máximo, a média, a mediana, o 1º quartil e o 3º quartil, para variáveis quantitativas, e para as variáveis qualitativas a função disponibiliza as freqüências de cada categoria.

```
> summary(milsa)
  funcionario         civil       instrucao      filhos          salario
 Min.   : 1.00   solteiro:16   1ºGrau  :12   Min.   : 0.00   Min.   : 4.000
 1st Qu.: 9.75   casado  :20   2ºGrau  :18   1st Qu.: 1.00   1st Qu.: 7.553
 Median :18.50                 Superior: 6   Median : 2.00   Median :10.165
 Mean   :18.50                               Mean   : 1.65   Mean   :11.122
 3rd Qu.:27.25                               3rd Qu.: 2.00   3rd Qu.:14.060
 Max.   :36.00                               Max.   : 5.00   Max.   :23.300
                                             NA's   :16.00
      ano             mes              regiao       idade
 Min.   :20.00   Min.   : 0.000   capital :11   Min.   :20.83
 1st Qu.:30.00   1st Qu.: 3.750   interior:12   1st Qu.:30.67
 Median :34.50   Median : 6.000   outro   :13   Median :34.92
 Mean   :34.58   Mean   : 5.611                 Mean   :35.05
 3rd Qu.:40.00   3rd Qu.: 8.000                 3rd Qu.:40.52
 Max.   :48.00   Max.   :11.000                 Max.   :48.92
```

Se desejar a função `stat.desc`, do pacote `pastecs`, disponibiliza várias estatísticas para variáveis quantitativas.

```
> stat.desc(milsa)
             funcionario civil instrucao     filhos     salario          ano
nbr.val       36.0000000    NA        NA 20.0000000  36.0000000   36.0000000
```

```
nbr.null       0.0000000    NA        NA  4.0000000   0.0000000    0.0000000
nbr.na         0.0000000    NA        NA 16.0000000   0.0000000    0.0000000
min            1.0000000    NA        NA  0.0000000   4.0000000   20.0000000
max           36.0000000    NA        NA  5.0000000  23.3000000   48.0000000
range         35.0000000    NA        NA  5.0000000  19.3000000   28.0000000
sum          666.0000000    NA        NA 33.0000000 400.4000000 1245.0000000
median        18.5000000    NA        NA  2.0000000  10.1650000   34.5000000
mean          18.5000000    NA        NA  1.6500000  11.1222222   34.5833333
SE.mean        1.7559423    NA        NA  0.2835397   0.7645763    1.1229037
CI.mean.0.95   3.5647524    NA        NA  0.5934553   1.5521723    2.2796157
var          111.0000000    NA        NA  1.6078947  21.0447663   45.3928571
std.dev       10.5356538    NA        NA  1.2680279   4.5874575    6.7374221
coef.var       0.5694948    NA        NA  0.7685018   0.4124587    0.1948170
                     mes regiao        idade
nbr.val       36.0000000     NA   36.0000000
nbr.null       4.0000000     NA    0.0000000
nbr.na         0.0000000     NA    0.0000000
min            0.0000000     NA   20.8333333
max           11.0000000     NA   48.9166667
range         11.0000000     NA   28.0833333
sum          202.0000000     NA 1261.8333333
median         6.0000000     NA   34.9166667
mean           5.6111111     NA   35.0509259
SE.mean        0.5481249     NA    1.1175275
CI.mean.0.95   1.1127527     NA    2.2687014
var           10.8158730     NA   44.9592372
std.dev        3.2887495     NA    6.7051650
coef.var       0.5861138     NA    0.1912978
```

Também podem ser obtidas as estatísticas individuais. A seguir segue uma tabela com algumas (não todas) as estatísticas, com suas respectivas funções no R, e com os objetos nais quais elas podem ser utilizadas:

Tabela 2: Estatísticas Descritivas no R

Estatística	Função no R	Objeto
Média	`mean()`	Matriz, vetor e data-frame
Mediana	`median()`	Matriz, vetor e data-frame
Desvio-padrão	`sd()`	Matriz, vetor e data-frame
Variância	`var()`	Vetor
Covariânciac	`cov()`	Matriz e data-frame
Correlação	`cor()`	Matriz e data-frame
Nº de Observações	`lenght()`	Matriz e data-frame
Intervalo Interquartilico	`IQR()`	Vetor

3.5 Análise Descritiva de Tabelas de Contingência

3.5.1 Tabelas Para Dois ou Mais Fatores

Tabelas de contingência podem ser obtidas com as frequências de occorrência dos cruzamentos das variáveis. A seguir mostramos algumas opções da vizualização dos resultados usando a função `table()` e a função `ftable()`. As funções retornam as tabelas de contingência em um objeto que pode ser uma *matrix*, no caso do cruzamento de duas variáveis, ou de forma mais geral, na forma de um *array*, onde o número de dimensões é igual ao número de variáveis. Entretanto a classe do objeto resultante vai depender da função utilizada. Neste caso, para o cruzamento de apenas duas

variáveis, os resultados são exibidos de forma semelhante. No exemplo consideram-se as variáveis civil e instrucao que situadas nas colunas 2 e 3 do *data-frame*.

```
> t1 <- table(milsa[c(2, 3)])
> t1
          instrucao
civil      1°Grau 2°Grau Superior
  solteiro      7      6        3
  casado        5     12        3
> t1f <- ftable(milsa[c(2, 3)])
> t1f
         instrucao 1°Grau 2°Grau Superior
civil
solteiro                7      6        3
casado                  5     12        3
> sapply(list(t1, t1f), class)
[1] "table"  "ftable"
> sapply(list(t1, t1f), is.matrix)
[1] TRUE TRUE
> sapply(list(t1, t1f), is.array)
[1] TRUE TRUE
```

Ambas funções possuem o argumento dnn que pode ser usado para sobrescrever os nomes das dimensões do objeto resultante.

```
> dimnames(t1)
$civil
[1] "solteiro" "casado"

$instrucao
[1] "1°Grau"   "2°Grau"   "Superior"

> t1 <- table(milsa[c(2, 3)], dnn = c("Estado Civil", "Nível de Instrução"))
> dimnames(t1)
$‘Estado Civil‘
[1] "solteiro" "casado"

$‘Nível de Instrução‘
[1] "1°Grau"   "2°Grau"   "Superior"

> t1f <- table(milsa[c(2, 3)], dnn = c("Estado Civil", "Nível de Instrução"))
```

As diferenças na forma de exibir os resultados são mais claras considerando-se o cruzamento de três ou mais variáveis. Enquanto `table()` vai exibir um *array* da forma usual, mostrando as várias camadas separadamente, `ftable()` irá arranjar a tabela de forma plana, em uma visualização mais adequada para a leitura dos dados. Vamos considerar o cruzamento das variáveis *civil*, *instrucao* e *regiao* situadas nas colunas 2, 3 e 8 do *data-frame*.

```
> t2 <- with(milsa, table(civil, instrucao, regiao))
> t2
, , regiao = capital

          instrucao
civil      1°Grau 2°Grau Superior
  solteiro      2      1        1
```

```
  casado        2      4        1

, , regiao = interior

          instrucao
civil      1°Grau 2°Grau Superior
  solteiro      2      1        1
  casado        1      6        1

, , regiao = outro

          instrucao
civil      1°Grau 2°Grau Superior
  solteiro      3      4        1
  casado        2      2        1

> t2f <- with(milsa, ftable(civil, instrucao, regiao))
> t2f
                   regiao capital interior outro
civil    instrucao
solteiro 1°Grau                 2        2     3
         2°Grau                 1        1     4
         Superior               1        1     1
casado   1°Grau                 2        1     2
         2°Grau                 4        6     2
         Superior               1        1     1
```

Enquanto que o objeto retornado por `table()` não é uma *matrix*, mas sim um *array* de três dimensões, por serem três variáveis. A dimensão do *array* é de $2 \times 3 \times 3$ por haver 2 estados civis, 3 níveis de instrução e 3 regiões. Já o objeto retornado por `ftable()` ainda é uma *matriz*, neste caso de dimensão 6×3 onde $6 = 2 \times 3$ indicando o produto do número de nívies das duas primeiras variáveis.

```
> sapply(list(t2, t2f), is.matrix)
[1] FALSE  TRUE
> sapply(list(t2, t2f), is.array)
[1] TRUE TRUE
> sapply(list(t2, t2f), dim)
[[1]]
[1] 2 3 3

[[2]]
[1] 6 3
```

Com `ftable()` é possível ainda criar outras visualizações da tabela. Os argumentos `row.vars` e `col.vars` podem ser usados para indicar quais variáveis serão colocadas nas linhas e colunas, e em que ordem. No exemplo a seguir colocamos o estado civil e região de procedência (variáveis 1 e 3) nas colunas da tabela e também modificamos o nome das dimensões da tabela com o argumento `dnn`. O objeto resultante é uma *matrix* de dimensão 6×3.

```
> with(milsa, ftable(civil, instrucao, regiao, dnn = c("Estado Civil:",
+ "Nível de Instrução", "Procedência:"), col.vars = c(1, 3)))
                   Estado Civil: solteiro                 casado
                   Procedência:   capital interior outro capital interior outro
```

```
Nível de Instrução
1°Grau                                2       2     3       2       1     2
2°Grau                                1       1     4       4       6     2
Superior                              1       1     1       1       1     1
```

3.5.2 Frequências Relativas

As funções `table()` e `ftable()` retornam objetos das classes `table` e `ftable`, respectivamente. A partir de tais objetos, outras funções podem ser utilizadas tais como `prop.table()` para obtenção de frequências relativas. A distinção entre as classes não é importante no caso de cruzamento entre duas variáveis. Entretanto para três ou mais variáveis os resultados são bem diferentes, devido ao fato já mencionado de que `table()` retorna um *array* de dimensão igual ao número de variáveis, enquanto que `ftable()` retorna sempre uma *matriz*.

Considerando os exemplos da sessão anterior, vejamos primeiro os resultados de frequências relativas para duas variáveis, que não diferem entre as clases. Da mesma forma, no caso de duas variáveis, as margens da tabelas obtidas de uma ou outra forma são as mesmas.

```
> prop.table(t1)
            Nível de Instrução
Estado Civil     1°Grau     2°Grau   Superior
    solteiro 0.19444444 0.16666667 0.08333333
    casado   0.13888889 0.33333333 0.08333333
> prop.table(t1f)
            Nível de Instrução
Estado Civil     1°Grau     2°Grau   Superior
    solteiro 0.19444444 0.16666667 0.08333333
    casado   0.13888889 0.33333333 0.08333333
> prop.table(t1, margin = 1)
            Nível de Instrução
Estado Civil 1°Grau 2°Grau Superior
    solteiro 0.4375 0.3750   0.1875
    casado   0.2500 0.6000   0.1500
> prop.table(t1f, margin = 1)
            Nível de Instrução
Estado Civil 1°Grau 2°Grau Superior
    solteiro 0.4375 0.3750   0.1875
    casado   0.2500 0.6000   0.1500
> margin.table(t1, mar = 1)
Estado Civil
solteiro   casado
      16       20
> margin.table(t1f, mar = 1)
Estado Civil
solteiro   casado
      16       20
> margin.table(t1, mar = 2)
Nível de Instrução
  1°Grau   2°Grau Superior
      12       18        6
> margin.table(t1f, mar = 2)
Nível de Instrução
  1°Grau   2°Grau Superior
      12       18        6
```

Já para três os mais variáveis os resultados são bem diferentes em particular para as frequências

marginais, uma vez que `ftable()` vai sempre retornar uma *matriz* e portanto só possuirá margens 1 e 2.

```
> prop.table(t2)
, , regiao = capital

          instrucao
civil          1°Grau     2°Grau   Superior
  solteiro 0.05555556 0.02777778 0.02777778
  casado   0.05555556 0.11111111 0.02777778

, , regiao = interior

          instrucao
civil          1°Grau     2°Grau   Superior
  solteiro 0.05555556 0.02777778 0.02777778
  casado   0.02777778 0.16666667 0.02777778

, , regiao = outro

          instrucao
civil          1°Grau     2°Grau   Superior
  solteiro 0.08333333 0.11111111 0.02777778
  casado   0.05555556 0.05555556 0.02777778

> prop.table(t2f)
                  regiao    capital   interior      outro
civil    instrucao
solteiro 1°Grau           0.05555556 0.05555556 0.08333333
         2°Grau           0.02777778 0.02777778 0.11111111
         Superior         0.02777778 0.02777778 0.02777778
casado   1°Grau           0.05555556 0.02777778 0.05555556
         2°Grau           0.11111111 0.16666667 0.05555556
         Superior         0.02777778 0.02777778 0.02777778
> prop.table(t2, margin = 1)
, , regiao = capital

          instrucao
civil      1°Grau 2°Grau Superior
  solteiro 0.1250 0.0625   0.0625
  casado   0.1000 0.2000   0.0500

, , regiao = interior

          instrucao
civil      1°Grau 2°Grau Superior
  solteiro 0.1250 0.0625   0.0625
  casado   0.0500 0.3000   0.0500

, , regiao = outro

          instrucao
civil      1°Grau 2°Grau Superior
  solteiro 0.1875 0.2500   0.0625
```

```
  casado   0.1000 0.1000   0.0500

> prop.table(t2f, margin = 1)
                   regiao   capital  interior     outro
civil    instrucao
solteiro 1°Grau           0.2857143 0.2857143 0.4285714
         2°Grau           0.1666667 0.1666667 0.6666667
         Superior         0.3333333 0.3333333 0.3333333
casado   1°Grau           0.4000000 0.2000000 0.4000000
         2°Grau           0.3333333 0.5000000 0.1666667
         Superior         0.3333333 0.3333333 0.3333333
> prop.table(t2, margin = 3)
, , regiao = capital

          instrucao
civil          1°Grau     2°Grau   Superior
  solteiro 0.18181818 0.09090909 0.09090909
  casado   0.18181818 0.36363636 0.09090909

, , regiao = interior

          instrucao
civil          1°Grau     2°Grau   Superior
  solteiro 0.16666667 0.08333333 0.08333333
  casado   0.08333333 0.50000000 0.08333333

, , regiao = outro

          instrucao
civil          1°Grau     2°Grau   Superior
  solteiro 0.23076923 0.30769231 0.07692308
  casado   0.15384615 0.15384615 0.07692308

> prop.table(t2f, margin=3)
Erro em if (d2 == 0L) { : valor ausente onde TRUE/FALSE necessário
```

É possível obter totais marginais com `margin.table()` a partir de um objeto resultante de `table()` mas **não** para um objeto resultante de `ftable()`!

```
> margin.table(t2, mar = 1)
civil
solteiro   casado
      16       20
> margin.table(t2, mar = 2)
instrucao
  1°Grau   2°Grau Superior
      12       18        6
> margin.table(t2, mar = 3)
regiao
 capital interior    outro
      11       12       13
```

3.6 Análise Descritiva Através de Gráficos

O R possue uma infinidade de recursos gráficos, nesta sessão vamos demonstrar alguns (não todos) dos recursos gráficos do R.

A partir das tabelas de contigência e das freqüências relativas é possível implementar técnicas gráficas de análise descritiva. Primeiro demonstraremos funções gráficas adequadas às variáveis qualitativas.

- **Gráfico de setores**:

```
> t3 <- table(milsa[,2])
> t3
solteiro   casado
      16       20
> tp3 <- prop.table(table(milsa[,2]))*100
> tp3
solteiro    casado
 44.44444 55.55556
> pie(t3)
```

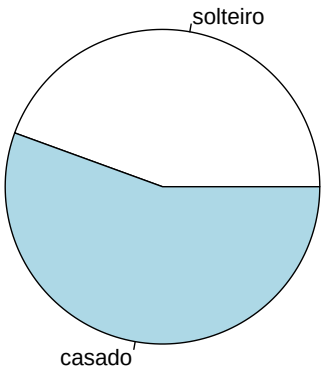

Figura 1: Gráfico de setores para o Estado Civil (sem alteração)

Esta função plota o gráfico de setores *default* do R, mas alguns argumentos e parâmetros gráficos podem ser alterados ou acrescentados para melhor visualização. Alterações como na cor do gráfico, legenda e rótulos.

```
> pie(t3, labels = paste(round(tp3, dig=2), "%"), col = c(2,4))
> legend("topleft", legend=names(t3), fill = c(2,4))
```

- **Gráfico de Barras:**

Utilizando a tabela `t1`, da sessão anterior, como exemplo para criarmos um gráfico de barras para as variáveis Nível de Instrução e Estado Civil.

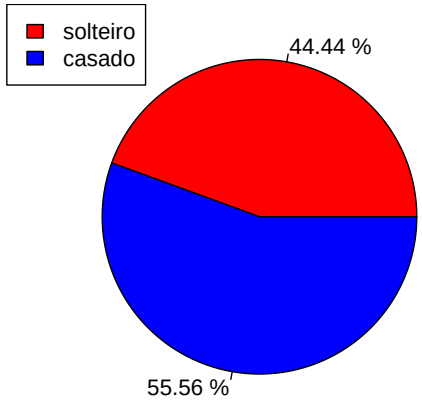

Figura 2: Gráfico de setores para o Estado Civil (com alteração)

```
> t1
          instrucao
civil      1Grau 2Grau Superior
  solteiro      7      6        3
  casado        5     12        3
> barplot(t1)
```

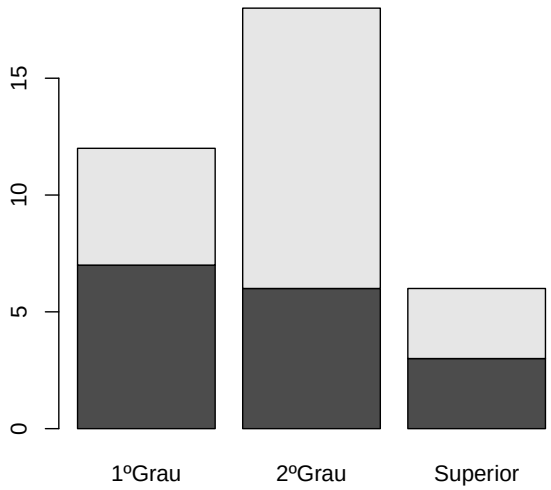

Figura 3: Gráfico de Barras para o Nível de Instrução em relação ao Estado Civil (sem alterao)

```
> barplot(t1, beside = TRUE, legend = rownames(t1),xlab = "Nível de Instrução",
+ ylab = "Freqncia", col = c(2,4))
```

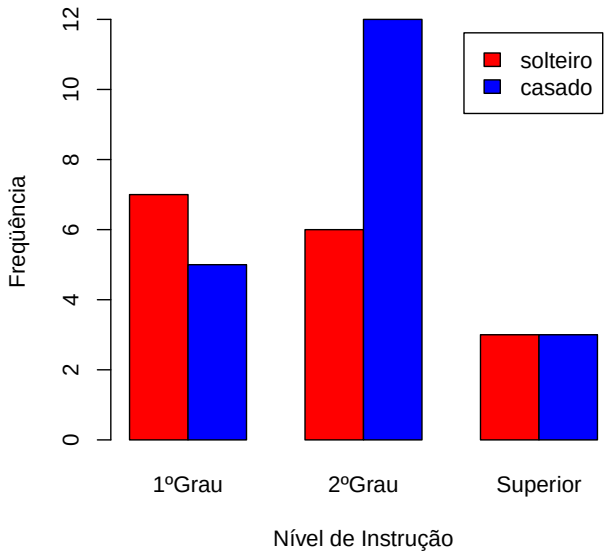

Figura 4: Gráfico de Barras para o Nível de Instrução em relação ao Estado Civil (com alteração)

Para as variáveis quantitativas são usadas outras abordagens gráficas, tais como:

- **Box-plot**:

  O *Box-plot* demonstra graficamente algumas informações sobre a distribuição das variáveis em estudo. Informações como a simetria da distribuição, com o mínimo, o máximo, a mediana, o IQ e os *outliers*.

```
> boxplot(salario ~ instrucao, data = milsa, col = "lightgray",
+ ylab="Salário dos Funcionários", xlab="Nível de Instrução")
```

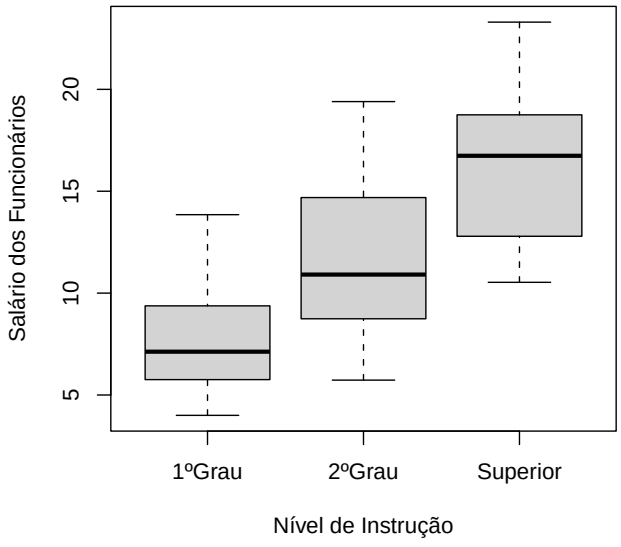

Figura 5: Box-plot para o salário dos funcionários em relação ao seus níveis de instrução

- **Diagrama de Dispersão (Scatterplot):**

  O diagrama de dispersão é um gráfico onde pontos no espaço cartesiano XY são usados para representar simultaneamente os valores de duas variáveis quantitativas medidas em cada elemento do conjunto de dados.

  Através do mesmo é possível identificar o tipo de correlação entre as duas variáveis ou até mesmo se não existe correlação entre elas.

```
> plot(x = milsa[,"salario"], y = milsa[,"idade"], xlab = "Salário",
+ ylab = "Idade", pch = 16, col = "red")
```

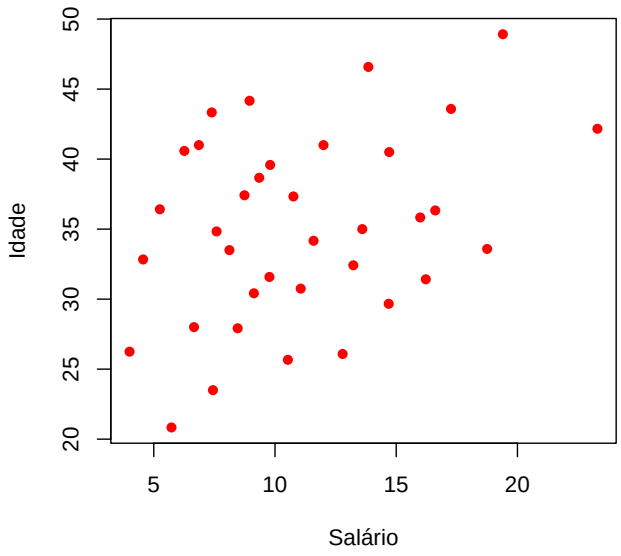

Figura 6: Scatterplot do salário dos funcionários com suas idades

- **Histograma:**

  O histograma *default* para a variável *idade em anos* é o seguinte:

```
> hist(milsa[,"ano"])
```

  Após a alteração na cor, a retirada do título e a adição dos rótulos:

```
> hist(milsa[,"ano"], col = "blue", main = NULL, xlab = "Idade em Anos",
+ ylab = "Freqüência")
```

  e para inserir a curva de densidade

```
> hist(milsa[,"ano"], col = "blue", main = NULL, xlab = "Idade em Anos",
+ ylab = "Densidade", prob = TRUE)
> lines(density(milsa[,"ano"]), lwd = 2)
```

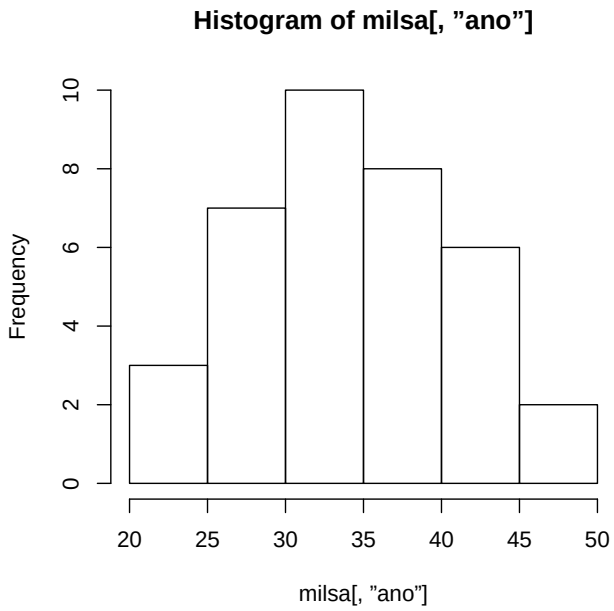

Figura 7: Histograma da idade em anos (sem alteração)

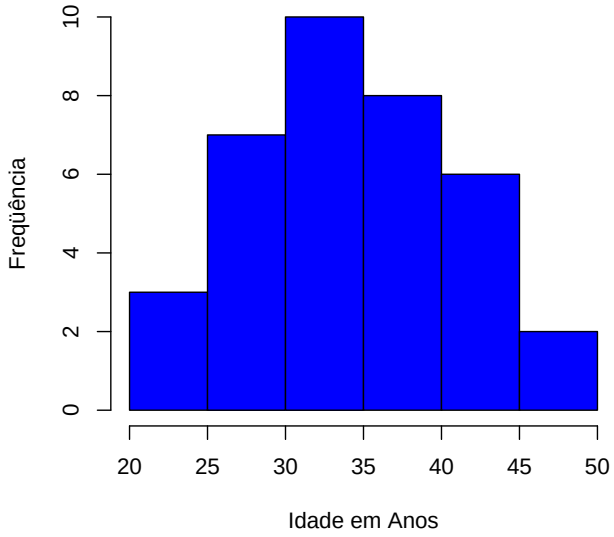

Figura 8: Histograma da idade em anos (com alteração)

4 Distribuições de Probabilidade

O programa `R` inclui funcionalidade para operações com distribuições de probabilidades. Para cada distribuições 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;

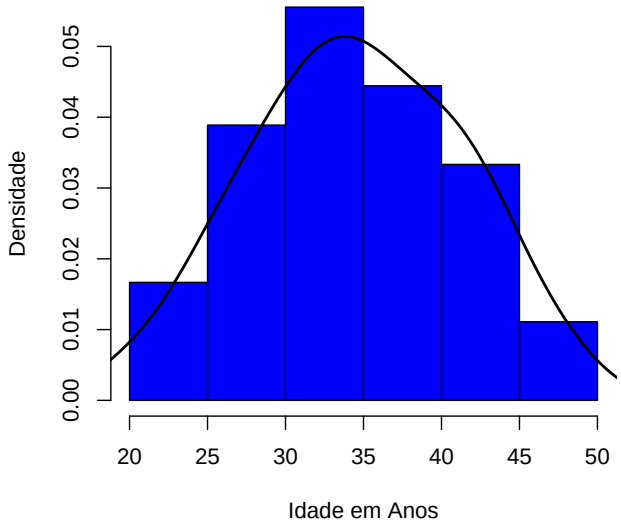

Figura 9: Histograma e Densidade da idade em anos

`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.

Distribuição	**Nome no R**	**Argumentos adicionais**
Beta	`beta`	`forma1, forma2`
Binomial	`binom`	`tamanho, probabilidade`
Exponencial	`exp`	`taxa`
Gama	`gamma`	`forma, escala`
Normal	`norm`	`média, desvio`
Poisson	`pois`	`lambda`
Uniforme	`unif`	`minimo, máximo`
Weibull	`weibull`	`forma, escala`
T Student	`t`	`gl, ncp`

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

```
> dnorm(-1)
[1] 0.2419707
> pnorm(-1)
[1] 0.1586553
> qnorm(0.975)
[1] 1.959964
> rnorm(10)
 [1]  0.65600901  0.15812926  1.13959633  0.74402723  0.04496600  0.25926501
```

```
 [7] -1.76474173  1.17748276  0.57136599  0.68585134
```

O primeiro valor acima corresponde ao valor da densidade da normal,

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

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 $P(X > -1)$. O comando `qnorm(0.975)}` calcula o valor de a tal que $P(X < a) = 0.975$. Finalmente o comando `rnorm(10)` gera uma amostra de 10 elementos da normal padrão. Note que os valores que você 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) NULL
```

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 $N(100, 10)$. 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 10 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 padrão da média, faixa que concentra 99,73% da massa de probabilidade da distribuição normal.

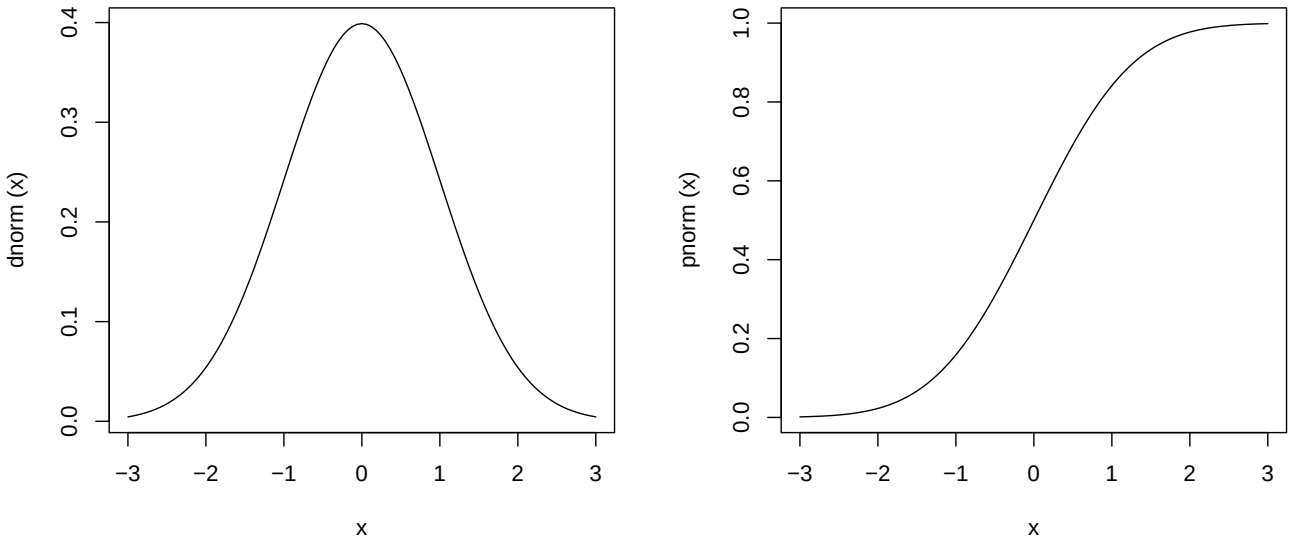

Figura 10: Funções de densidade e probabilidade da distribuição normal padrão.

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

A Figura 11 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)
```

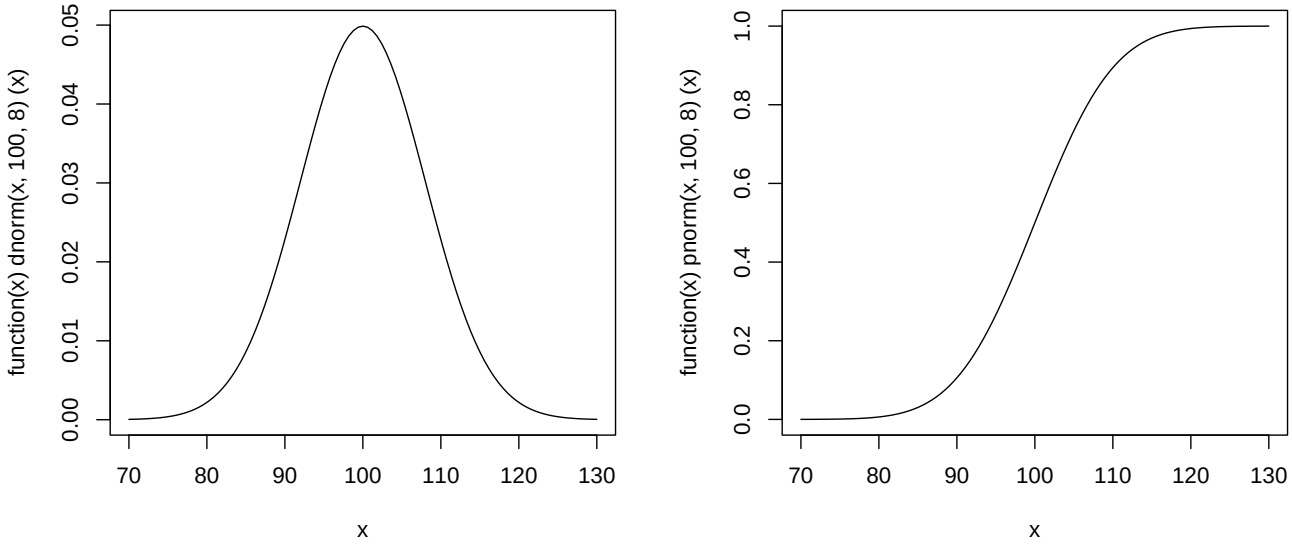

Figura 11: Funções de densidade de probabilidade (esquerda) e função de distribuição acumulada (direita) da $N(100, 64)$.

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 12 e nos dois primeiros comandos abaixo. Os demais comandos mostram como colocar diferentes densidades em um mesmo gráfico como ilustrado à direita da mesma Figura.

```
> plot(dnorm, -3, 3, xlab = "valores de X",
+ ylab = "densidade de probabilidade")
> title("Distribuição 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(110, 0.05, c("N(100,64)", "N(90,64)", "N(100,225)"),
+ fill = 1:3)
```

4.2 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) NULL
> 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 da função de densidade;
2. Idem para a função de probabilidade;
3. Calcular $P[X = 7]$;

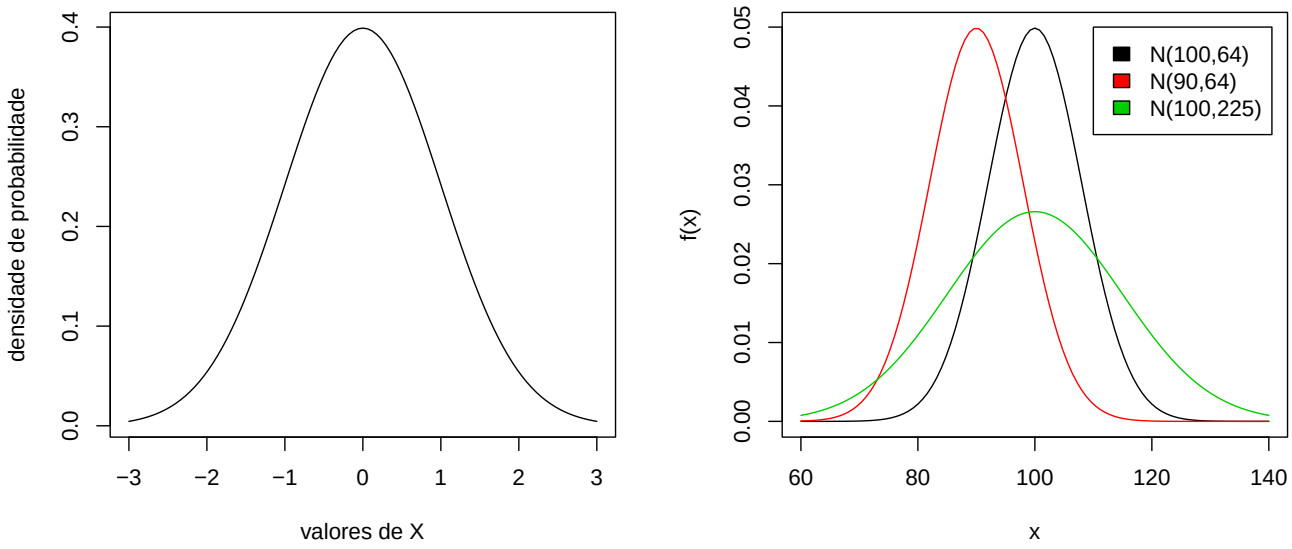

Figura 12: Gráfico com texto nos eixos e título (esquerda) e várias distribuições em um mesmo gráfico (direita).

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 13.

```
> 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")
```

As probabilidades pedidas são obtidas com os comandos a seguir.

```
> 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
```

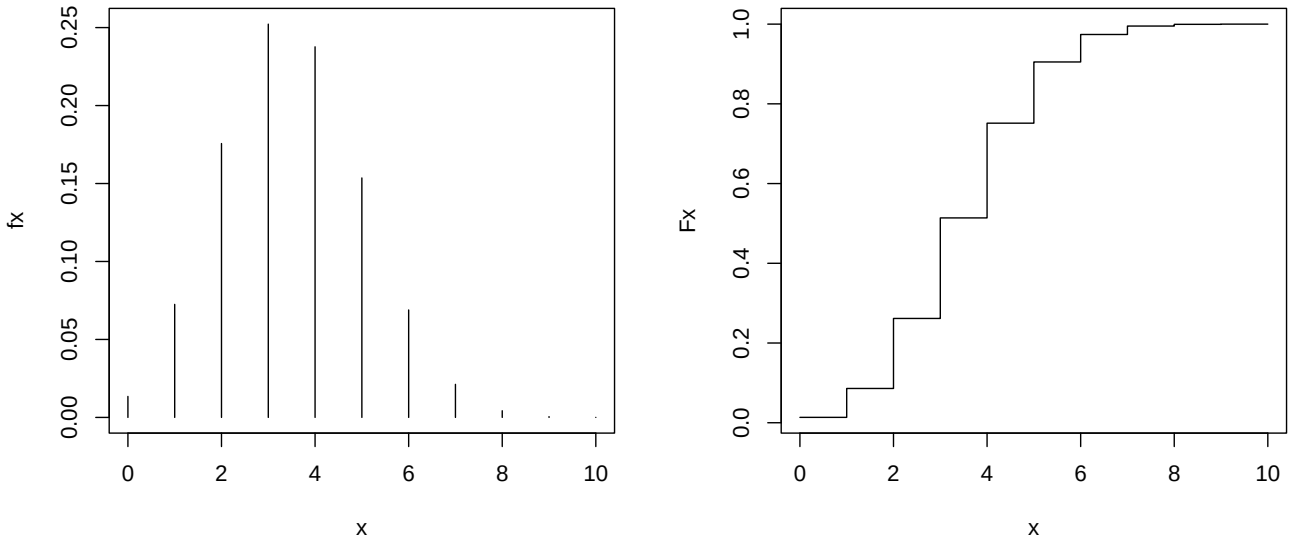

Figura 13: Funções de probabilidade (esquerda) e de distribuição acumulada (direita) da Binomial(10, 0.35).

4.3 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. e Morettin, P.A. *Estatística Básica*. 4a edição. Atual Editora. 1987.

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

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

(a) mostre que esta 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 14.

```
> f1 <- function(x){
+ fx <- ifelse(x < 0, 0, 2*exp(-2*x)) + return(fx) + }
> plot(f1)
> plot(f1,0,10)
```

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

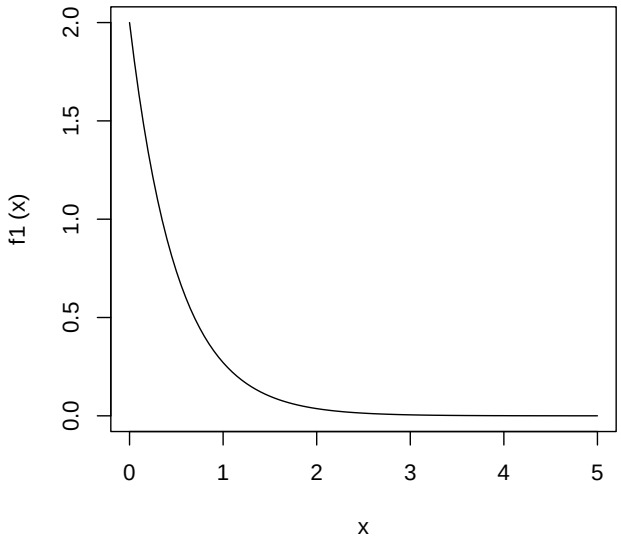

Figura 14: Gráfico da função de probabilidade do Exemplo 1.

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

$$p_b = P(X > 1) = \int_1^\infty f(x)dx = \int_1^\infty 2e^{-2x}dx$$

$$p_c = P(2 < X < 8) = \int_2^8 f(x)dx = \int_2^8 2e^{-2x}dx$$

cuja representação gráfica é mostrada na Figura 15. 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. Assim os comando são os seguintes,

```
> 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])))
```

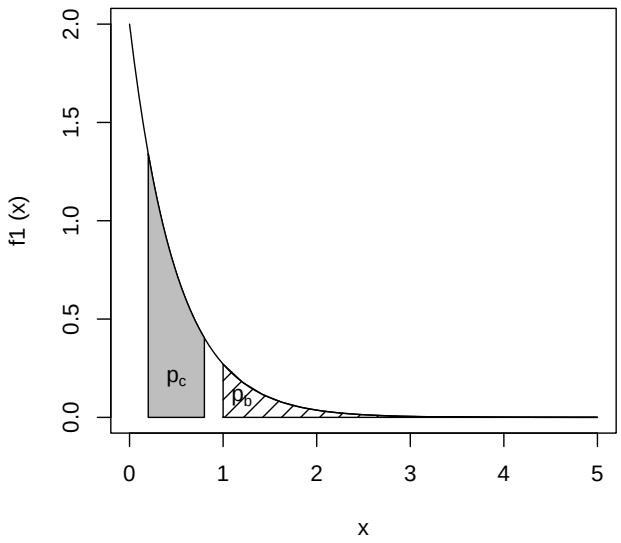

Figura 15: Probabilidades pedidas nos itens (b) e (c) do Exemplo 1.

e como obter as probabilidades pedidas usamos `integrate()`.

```
> 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,

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

(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 16.

```
> 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) + }
```

A seguir verificamos que a integral da função é 1 e fazemos o seu gráfico mostrado na Figura 16.

```
> integrate(f2, 0, 3)
1 with absolute error < 1.1e-15
> plot(f2, -1, 4)
```

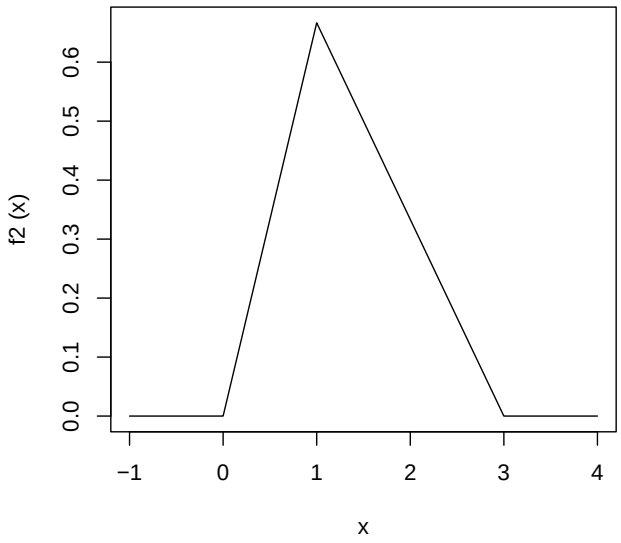

Figura 16: Gráfico da função densidade de probabilidade do Exemplo 2.

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
[1] 40
```

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

```
> 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)
```

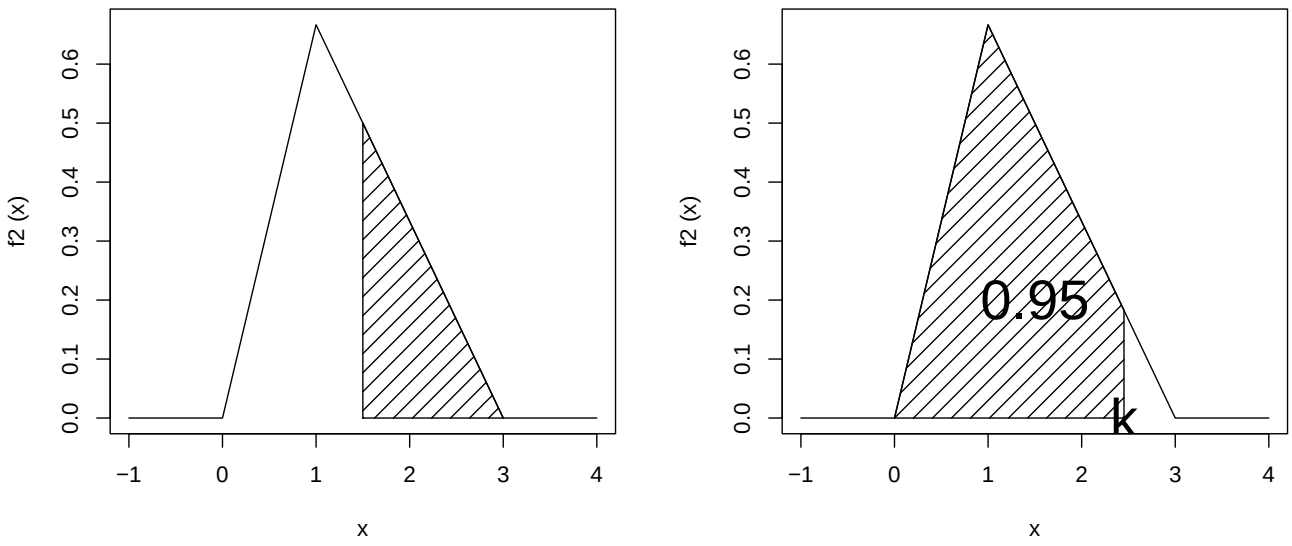

Figura 17: Gráficos indicando as soluções dos itens (a) e (c) do Exemplo 2.

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.

4.4 Distribuição Exponencial

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

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

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

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=1/50, lower=F)`, onde o argumento corresponde a $1/\lambda$ na equação da exponencial.

A Figura 18 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)
```

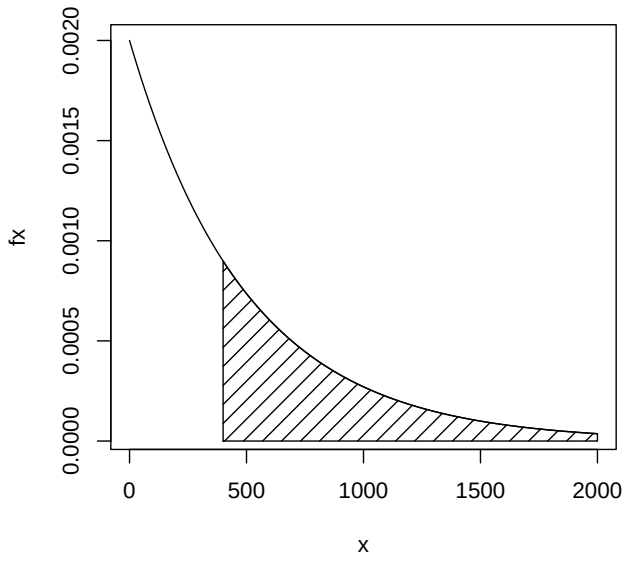

Figura 18: Função de densidade da $Exp(500)$ com a área correspondente à $P[X > 400]$.

4.5 Esperança e Variância

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

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

```
#calculando a esperança da exponencial:

> 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
```

4.6 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 [5] 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 [5] 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)`—

4.7 Aproximação pela Normal

Nos livros texto de estatística você 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 $X \sim N(5, 2.5)$. A Figura 19 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)
```

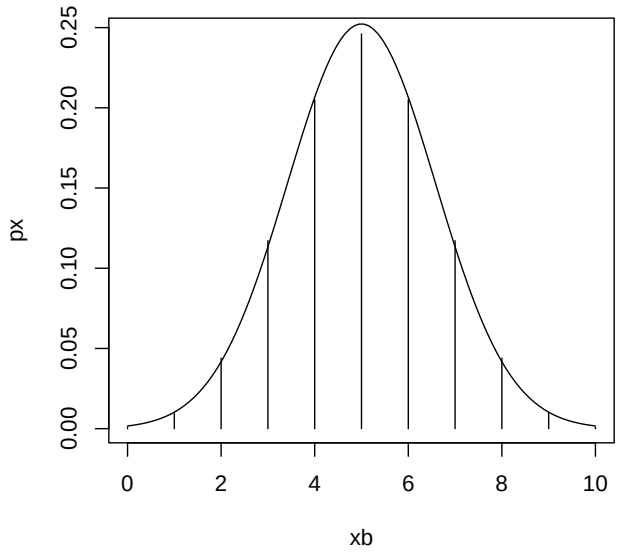

Figura 19: Função de probabilidade da $B(10, 1/2)$ e a aproximação pela $N(5, 2.5)$.

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 $P[X_B \leq 6.5] \approx P[X_N \leq 6.5]$

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

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

```
> 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 $P[X_B \geq 2] = 1 - P[X_B \leq 1] \approx 1 - P[X_N \geq 1.5]$

```
> 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
```

4.8 Exercícios

1. 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. Sendo X uma variável seguindo o modelo Binomial com parâmetro $n = 15$ e $p = 0.4$, pergunta-se:

(a) $P(X \geq 14)$
(b) $P(8 < X \leq 10)$
(c) $P(X < 2 ou X \geq 11)$
(d) $P(X \geq 11 ou X > 13)$
(e) $P(X > 3 e X < 6)$
(f) $P(X \leq 13 | X \geq 11)$

3. Para $X \sim N(90, 100)$, obtenha:

(a) $P(X \leq 115)$
(b) $P(X \geq 80)$
(c) $P(X \leq 75)$
(d) $P(85 \leq X \leq 110)$

4. Faça os seguintes gráficos:

(a) da função de densidade de uma variável com distribuição de *Poisson* com parâmetro $\lambda = 5$
(b) da densidade de uma variável $X \sim N(90, 100)$
(c) sobreponha ao gráfico anterior a densidade de uma variável $Y \sim N(90, 80)$ e outra $Z \sim N(85, 100)$

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 façaa um gráfico de sua função de distribuição.

6. Em uma determinada localidade a distribuição de renda, em u.m. (unidade monetária) é uma variável aleatória X com função de distribuição de probabilidade:

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

(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?

5 Modelos de Regressão Linear

5.1 Lendo os Dados no R

O arquivo de dados `skull.dat` contém medidas tomadas em diversos escalpos encontrados em sítios arqueológicos do Egito. Este arquivo pode ser lido no R como um `data-frame`. Como a primeira linha do arquivo tem as descrições das colunas devemos adicionar o argumento `head=T` à função `read.table()` e o R vai usar o conteúdo desta linha para atribuir nomes às colunas:

```
> skull <- read.table("skull.dat",head=T)
> dim(skull)
[1] 150   5
> head(skull)
  MaxBreadth BaseHeight BaseLength NasalHeight  Year
1        131        138         89          49 -4000
2        125        131         92          48 -4000
3        131        132         99          50 -4000
4        119        132         96          44 -4000
5        136        143        100          54 -4000
6        138        137         89          56 -4000
```

Aqui nós queremos investigar a relação entre BaseHeight (largura da base) do escalpo e BaseLength (comprimento da base).

5.2 Fazendo um gráfico dos dados

Podemos fazer um gráfico de pontos ("scatterplot") das duas quantidades:

```
> plot(skull$BaseH,skull$BaseL)
```

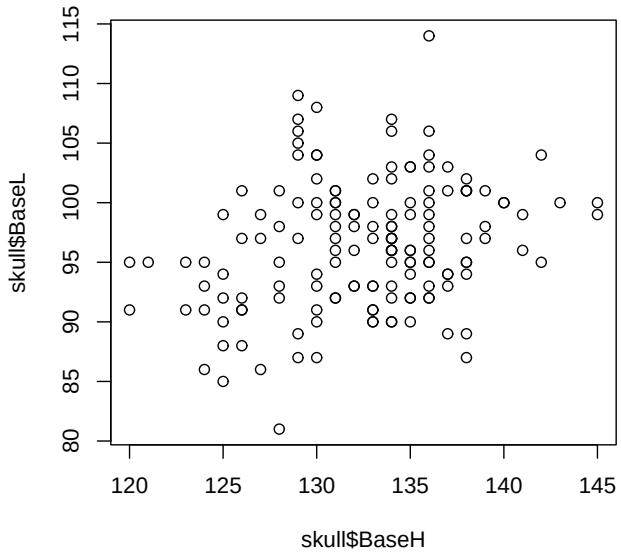

Figura 20: Diagrama de dispersão entre BaseHeight (largura da base) do escalpo e BaseLength (comprimento da base).

e podemos calcular o coeficiente de correlação linear de Pearson entre as variáveis:

```
> cor(skull$BaseH,skull$BaseL)
[1] 0.2643529
```

Queremos também ajustar um modelo que relacione as variáveis "Base Length"e "Base Height".

5.3 A Função lm()

Queremos ajustar um modelo linear da forma $y = a.x + b$ aos nossos dados. A função `lm()` faz isto:

```
> lm(skull$BaseL ~ skull$BaseH)       # podemos abreviar nomes...
Call:
lm(formula = skull$BaseL ~ skull$BaseH)

Coefficients:
 (Intercept) skull$BaseH
     58.3103      0.2878
```

Note com a função `lm()` é chamada com o formato `lm(y ~ x)` - lemos isto como y depende de x. Esta é a forma básica na qual diversos modelos são construídos, com a variável que está sendo medida (resposta) como a valor de y e as quantidades controladas (preditores) como as variáveis x. No nosso caso temos apenas dois conjuntos de medidas ... mas vamos em frente como exemplo de uso da função.

5.4 Armazenando o ajuste

Assim como a maioria das funções do R, você pode armazenar os resultados retornados pela função `lm()` em um objeto. O valor retornado por `lm()` é uma lista, assim como em `t.test()`, porém com elementos diferentes:

```
> sfit <- lm(skull$BaseL~skull$BaseH)
> is.list(sfit)
[1] TRUE
> names(sfit)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

Todos estes são importantes resultados do modelo ajustado. O primeiro e talvez mais importante a ser examinado é o elemento `coefficients`. Você pode usá-lo para adicionar uma linha ao gráfico de pontos:

```
> plot(skull$BaseH,skull$BaseL)
> sfit$coef
 (Intercept) skull$BaseH
    58.31026   0.2878212
> abline(sfit$coeff)
```

A função `abline()` recebe um vetor com dois elementos e adiciona uma reta ao gráfico com estes valores de *intercepto* e *inclinação*, respectivamente.

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

```
> plot(sfit$fitted,sfit$resid)
> lines(sfit$fitted,rep(0,length(sfit$fitted)), col = "red")
```

que pode mostrar desvios dos dados em relação ao modelo linear, caso hajam curvaturas claras no gráfico.

Pode-se também fazer um histograma dos resíduos para verificar a presença de dados discrepantes ("outliers"). Se os dados são realmente originários de um modelo linear com ruído Gaussiano então os resíduos devem mostar uma distribuição aproximadamente normal:

```
> hist(sfit$resid)
```

O R produz automaticamente uma série de gráficos para o ajuste de um modelo linear. A digitar o comando abaixo serão mostrados diversos gráficos (pressione ENTER para obter o gráfico seguinte) relacionados ao ajuste do modelo linear:

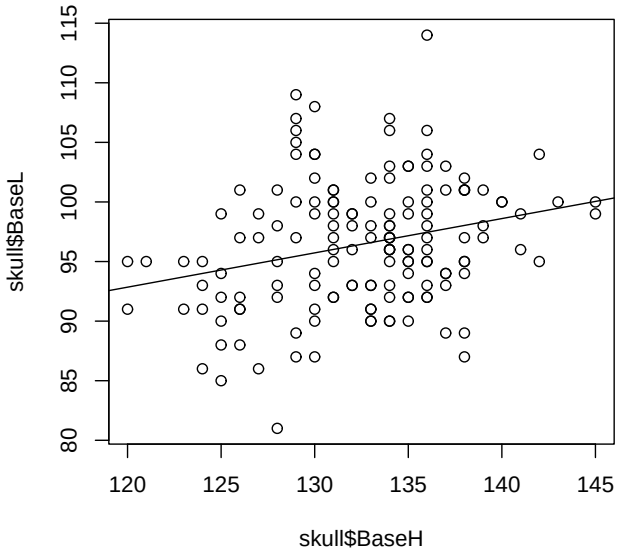

Figura 21: Diagrama de dispersão entre BaseHeight (largura da base) do escalpo e BaseLength (comprimento da base), com a reta ajustada.

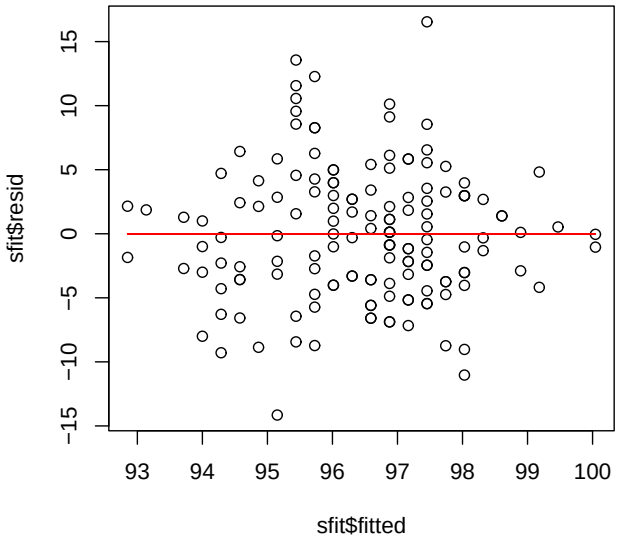

Figura 22: Valor ajustado pelo resíduo do modelo.

```
> par(mfrow=c(2,2))
> plot(sfit)
```

5.5 Classe dos Objetos

Diferentes classes podem ser atribuídas a diferentes objetos do R. Este mecanismo facilita a automação de diversas tarefas no R. Por exemplo podemos descobrir qual a classe atribuída a um objeto que armazana resultados da função `lm()` digitando:

```
> class(sfit)
```

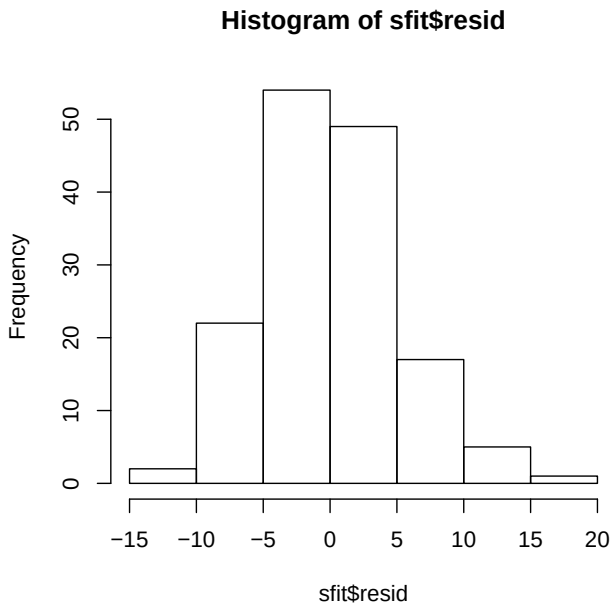

Figura 23: Histograma dos resíduos do modelo.

```
[1] "lm"
```

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

```
> par(mfrow=c(2,2))
> plot(sfit)
```

Isto é possível porque `plot()` é uma função genérica: o tipo de gráfico produzido depende da classe do objeto no argumento da função.

Outra função genérica é `summary()`:

```
> summary(sfit)

Call:
lm(formula = skull$BaseL ~ skull$BaseH)

Residuals:
     Min       1Q   Median       3Q      Max
-14.1514  -3.5868  -0.2196   2.9815  16.5461

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 58.31026   11.44841   5.093 1.06e-06 ***
skull$BaseH  0.28782    0.08631   3.335  0.00108 **
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 5.204 on 148 degrees of freedom
Multiple R-squared: 0.06988,    Adjusted R-squared: 0.0636
F-statistic: 11.12 on 1 and 148 DF,  p-value: 0.001080
```

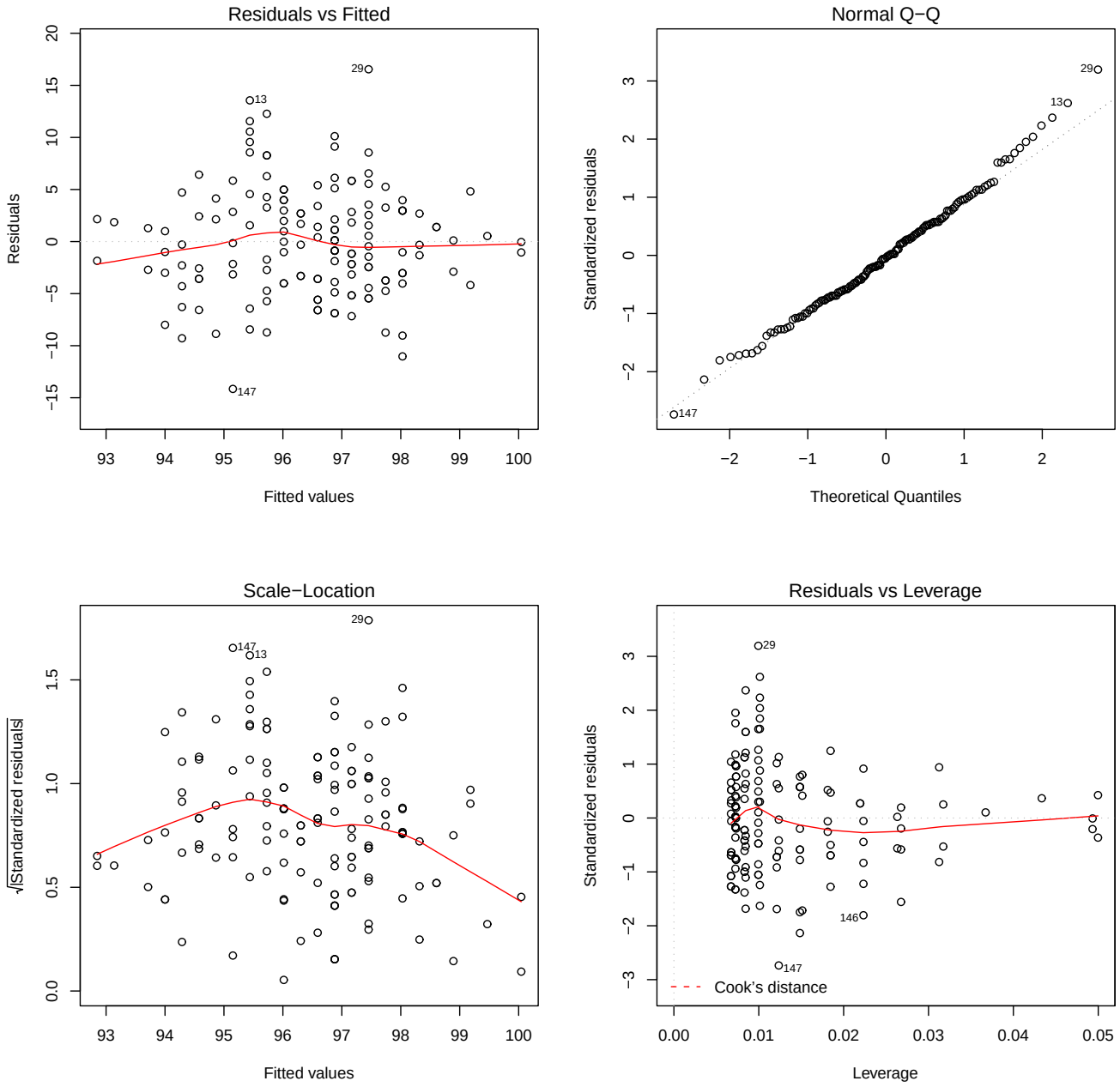

Figura 24: Gráficos de análise de resíduos.

Referências

[1] Ribeiro Jr, P. J., *Introdução ao Sistema Estatístico R*, notas do mini-curso EMBRAPA, Brasília-DF, 2005.

[2] Ribeiro Jr, P. J., *Tutorial de Introdução ao R*, site: http://leg.ufpr.br/Rtutorial/contents.html.

[3] Beasley, C. R., *Bioestatística Usando R: Apostila de Exemplo Para o Biólogo*, Bragança-PA, 2004.

[4] Paradis, E., *R for Beginners*, Montpellier, France, 2005.

[5] Hoff, K. J., *R-Manual for Biometry: An Introduction for Students of Horticulture and Plant Biotechnology*, Hannover, 2005.