

Software R: Análise estatística de dados utilizando um programa livre

Felipe Micaíl da Silva Smolski
Iara Denise Endruweit Battisti

2018

Sumário

Apresentação	5
1 Introdução	7
1.1 Download e instalação do R e Rstudio	7
1.2 Painéis	7
1.3 Help	8
1.4 Instalação de pacotes	8
1.5 Abrir arquivo de dados	9
1.6 Salvar arquivo de dados	12
1.7 Diretórios de trabalho	13
1.8 Operações	14
1.9 Criação de variáveis	16
1.10 Alguns comandos essenciais	17
1.11 Estrutura de dados	21
1.12 Manipulação de banco de dados	27
1.13 Função <i>edit</i>	27
1.14 Funções	28
1.15 Funções Matemáticas	35
1.16 Conversão de datas	37
2 Estatística Descritiva	39
2.1 Natureza da medida das variáveis	39
2.2 Tabelas e Gráficos	42
2.3 Gráficos	43
2.4 Estatísticas Descritivas	56
3 Estatística Inferencial	61
3.1 Intervalo de Confiança	62
3.2 Teste de hipóteses	66
4 Teste de Qui-Quadrado	77
5 Modelos de Regressão	79
6 RMarkdown	81

Apresentação

A necessidade de flexibilidade e robustez para a análise estatística fez com que fosse criado, na década de 1990, a linguagem de programação R. Capitaneado pelos desenvolvedores Ross Ihaka e Robert Gentleman, dois estatísticos da Universidade de Auckland na Nova Zelândia, o projeto foi uma grande evolução para a análise de dados. A partir de então, a ideia inicial de proporcionar autonomia ao pesquisador, viu na expansão do acesso à internet uma oportunidade para que a pesquisa científica se tornasse cada vez mais colaborativa. Ao mesmo tempo, os códigos e rotinas se tornaram facilmente disponibilizáveis na rede, aumentando a reprodução e replicação dos estudos, práticas estas que podem tornar as análises mais confiáveis.

A linguagem de programação R trouxe consigo inúmeras vantagens aos pesquisadores. Dentre elas, pode-se dizer primeiramente que, basicamente o R trabalha com uma extensa relação de modelos estatísticos, que vão desde a modelagem linear e não-linear, a análise de séries temporais, os testes estatísticos clássicos, análise de agrupamento e classificação, etc. Não bastasse este fato, é possível a apresentação gráfica dos resultados contando com variadas técnicas, passando também pela criação e manipulação de mapas.

Outra questão importante é que o R possui uma comunidade ativa de desenvolvedores, que se expande regularmente. Isto faz com que as técnicas de análise de dados atinjam pesquisadores de variadas disciplinas ao longo do planeta. Inclusive, concebe que o desenvolvimento dos pacotes melhorem constantemente. No ano de 2017, já haviam mais de 6.000 pacotes disponibilizados. Não menos importante, talvez o essencial: o programa é livre, ao passo que entrega o estado da arte da estatística ao usuário.

Outro progresso significativo na utilização do R foi a criação do *software* RStudio, a partir de 2010. Este, por sua vez, se configura em um ambiente integrado com o R e com inúmeras linguagens de marcação de texto (exemplos LaTeX, Markdown, HTML). Possui igualmente versão livre que disponibiliza ao pesquisador a execução, guarda, retomada e manipulação dos códigos de programação diretamente em seu console, bem como a administração de diretórios de trabalhos e projetos.

O material aqui criado é destinado não somente a alunos de graduação, pós-graduação, professores e pesquisadores acadêmicos, mas também para qualquer indivíduo interessado no aprendizado inicial sobre a utilização de técnicas estatísticas com o R. Inclusive, com o objetivo de alcançar um público das mais variadas áreas do conhecimento, esta obra foi elaborada com exemplos gerais, a serem absorvidos em um momento inicial do estudante.

Assim, possui a base para continuar estudos posteriores em estatística e no *software* RStudio. O sistema operacional aqui utilizado é o Windows 10.

Este livro está organizado da seguinte maneira: no capítulo 1 [**Primeiros Passos com o R**], busca-se instruir o pesquisador para a instalação dos programas necessários para acessar o ambiente de programação, bem como orientar sobre a usabilidade do programa em suas funções básicas de carregamento de bases de dados, criação de objetos e princípios de manipulação.

Já no capítulo 2 [**Estatística Descritiva**], leva o leitor ao encontro das técnicas básicas para descrever as variáveis em bancos de dados, como exemplos a média, mínima, máxima, desvio padrão, os quartis e também, apresentar os princípios dos elementos gráficos de apresentação dos dados.

O capítulo 3 [**Estatística Inferencial**] tratará dos métodos de determinação de intervalos de confiança (média e proporção), testes de hipóteses (verificar a normalidade dos dados) e das comparações entre médias de amostras dependentes e independentes.

No capítulo 4 [**Teste de Qui-Quadrado**], serão abordadas as referidas técnicas para verificação de associação entre duas variáveis qualitativas e de aderência a uma distribuição.

No capítulo 5 [**Modelos de Regressão**] serão introduzidos os conhecimentos sobre as técnicas de análise de correlação e regressão linear simples, bem como sobre o diagrama de dispersão, método dos mínimos quadrados, análise de variância, coeficiente e intervalo de predição, da análise dos resíduos e dos princípios de regressão múltipla.

A criação de documentos dinâmicos utilizando o RStudio será tratada no capítulo 6 [**RMarkdown**]. O pesquisador poderá conhecer as formas de integrar a programação no R e a manipulação de bases de dados, criando, compilando e configurando relatórios finais em diversos formatos (HTML, PDF e Word/Libre/Open Office).

Boa leitura!

Capítulo 1

Introdução

O R é um ambiente voltado para análise de dados com o uso de uma linguagem de programação, frente a isso um conhecimento prévio dos princípios de programação facilita a compreensão da condução das análises aplicadas no software. Entretanto, não é pré-requisito. Neste capítulo abordaremos os primeiros passos para o emprego da linguagem de programação R utilizando uma interface “amigável” - o software RStudio. Além disso, serão apresentados os comandos básicos para a manipulação de dados dentro do RStudio.

1.1 Download e instalação do R e Rstudio

R: <http://www.r-project.org>. Clique em Download (CRAN) - escolha o link de um repositório - clique no link do sistema operacional (Linux, Mac ou Windows) - clique em *install R for de first time - Download*.

RStudio: <http://www.rstudio.com/products/rstudio/download>. Em RStudio Desktop, escolha a versão *free*, seguidas da opção do sistema operacional do usuário.

Lembrando que:

- R é o software;
- RStudio é uma ferramenta amigável para o R.

1.2 Painéis

O RStudio é a interface que faz com que seja mais fácil a utilização da programação em R.

- **Fonte/Editor de Scripts:** se constitui do ambiente onde serão abertos os scripts previamente salvos nos mais diversos formatos ou mesmo sendo o local de visualização das bases de dados.

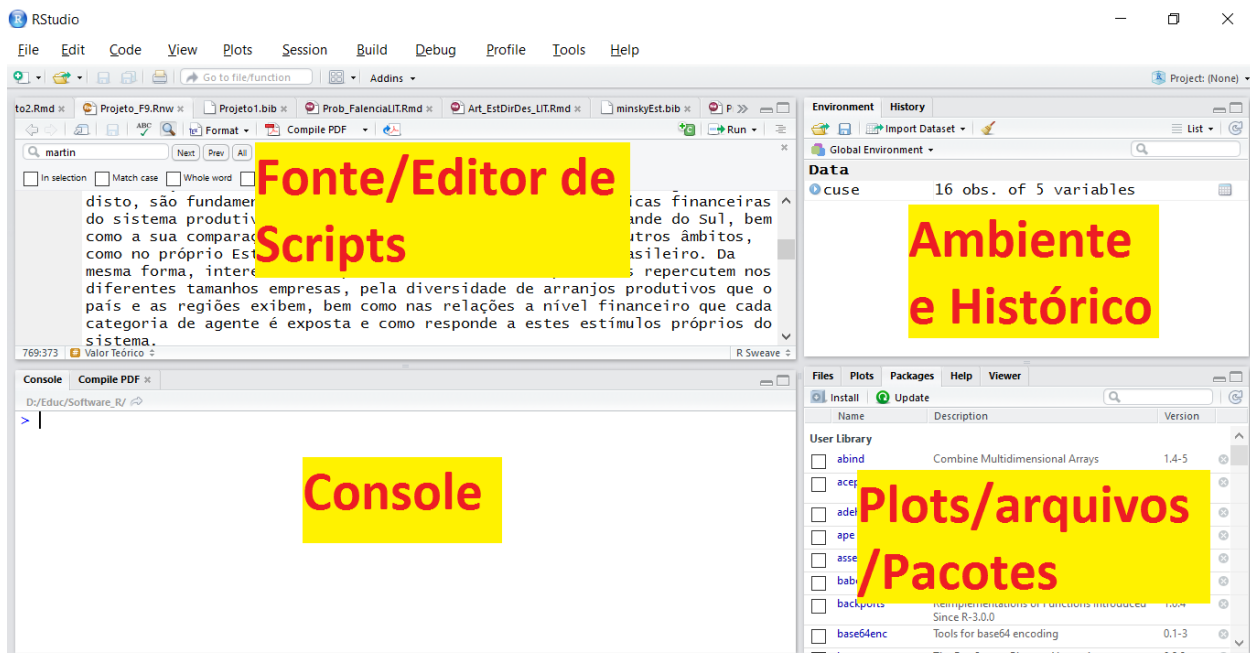


Figura 1.1: Painéis do Rstudio

- **Console:** local onde será efetuada a digitação das linhas de código que serão interpretadas pelo R.
- **Ambiente e Histórico:** o ambiente será visualizado os objetos criados ou carregados durante a sessão e; a aba History retoma os scripts digitados no console.
- **Plots/arquivos/Pacotes:** local onde podem ser acessados os arquivos salvos no computador pela aba *files*; a aba *Plots* carrega os gráficos e plotagens; a aba *Packages* contém os pacotes instalados em seu computador, onde são ativados ou instalados novos; em *Help* constam as ajudas e explicações dos pacotes e; *Viewer* visualiza documentos do tipo html.

1.3 Help

Acessamos a ajuda do RStudio por meio do comando `help()`, através da aba “Help” ou ao clicar no nome do pacote. Pode-se digitar a ajuda que usuário necessita (exemplo `help("summary")`), ou diretamente no colsole digitamos `?` e a função desejada, exemplo: `?mean`.

1.4 Instalação de pacotes

Em alguns situações, o uso de pacotes pode dar ao trabalho mais praticidade, e para isso se faz necessário efetuar a sua instalação. Precisamos ir até o painel dos pacotes em `*packages`,

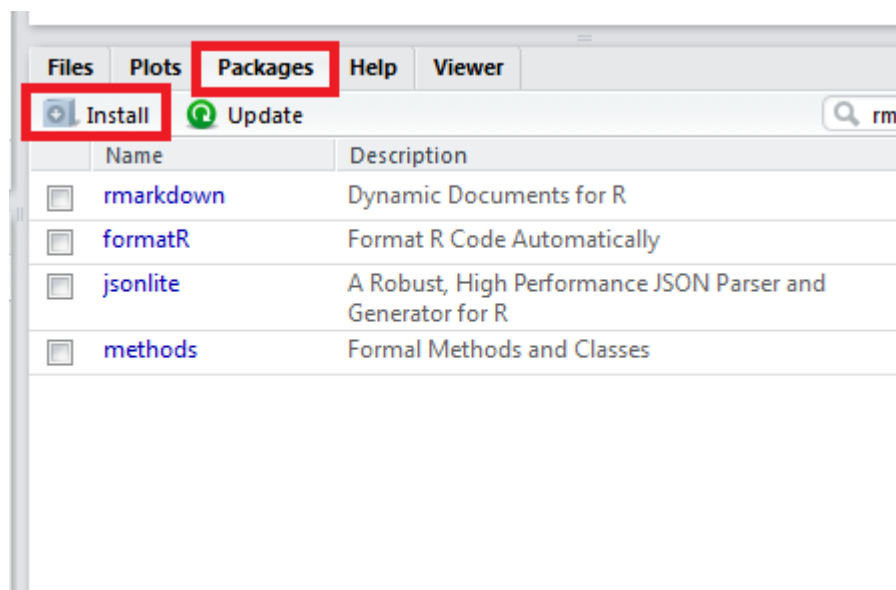


Figura 1.2: Instalação de pacotes

selecionar a opção instalar e inserir o nome do pacote desejado na janela indicada. Ao selecionar a opção instalar, no console receberemos informações do procedimento e do sucesso do mesmo.

A mesma função, para instalação de um pacote, pode ser efetuada diretamente via console: `install.packages("pacote")`. É importante ressaltar a função `library(nomedopacote)` que é utilizada no console para informar ao R e “carregar” o pacote que o usuário irá utilizar. Podem ser instalados mais de um pacote ao mesmo tempo, como no exemplo:

```
install.packages(c("readr", "readxl"))
```

1.5 Abrir arquivo de dados

Dispondo de um banco de dados em uma planilha eletrônica (LibreOffice Calc ou EXCEL), neste caso será utilizado o arquivo `arvores` como exemplo o banco de dados. Os dados derivam de uma pesquisa com espécies de árvores registrando as variáveis diâmetro altura do peito (DAP) e altura. Dados cedidos pela professora Tatiane Chassot.

Pode-se utilizar a linha de comando para carregar os arquivos de dados, da seguinte forma:

```
library(readxl)
```

```
nome.objeto.xls = read_excel("d:/arvores.xls")
```

Outras opções de arquivos podem ser carregados no RStudio, como por exemplo arquivos de texto (.txt ou .csv), arquivos derivados do excel (.xls ou .xlsx), arquivos de dados do SPSS (.sav), do *software* SAS (.sas7bdat) e do STATA (.dta). A instalação de alguns pacotes é

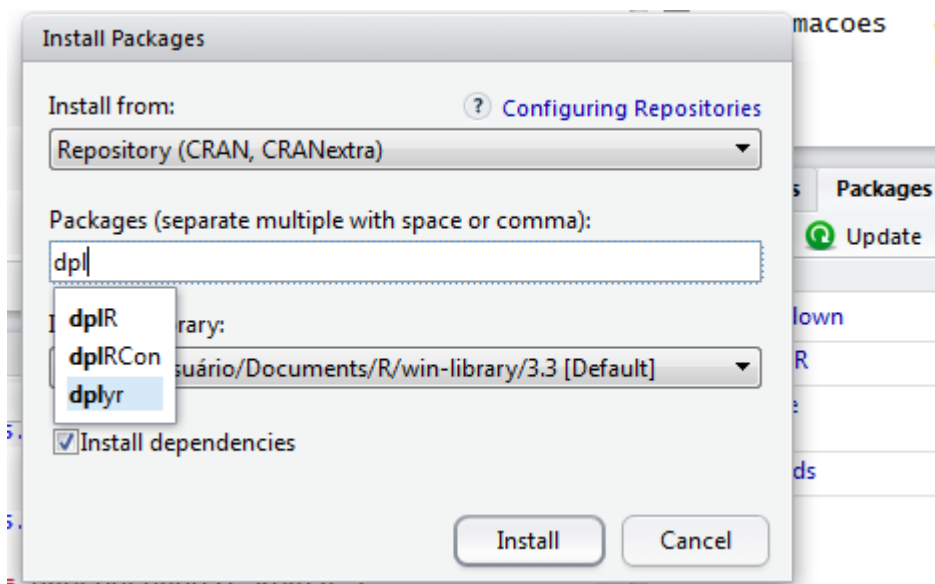


Figura 1.3: Caixa de informação de pacote a ser instalado

requerida, dependendo da origem da base de dados, como por exemplo o `readxl`, `readr` e `haven`, como os exemplos abaixo:

```
library(readr)

nomeobjeto = read.csv("d:/arvores.csv")

library(haven)

nomeobjeto = read_sav("d:/arvores.sav")

nomeobjeto = read_dta("d:/arvores.dta")

nomeobjeto = read_sas("d:/arvores.sas7bdat")
```

Outras opções podem ser comandadas dentro destes comando para abertura de arquivos, como por exemplo, um arquivo csv em que esteja separado por vírgulas pode ser lido como:

```
read.csv("d:/arvores.csv", sep=",")
```

O comando `header=TRUE` diz que a primeira linha do arquivo contém o cabeçalho; `skip=4` faz com que sejam ignoradas as 4 primeiras linhas.

A opção `load()` (exemplo: `load("base.RData")`) pode ser utilizada para carregar as bases de dados salvas com a função `save()`, que será descrita no subcapítulo a seguir.

Outra opção é o carregamento das bases de dados manualmente pelo caminho *Envoirement* > *Import Dataset*, escolhendo o tipo de arquivo:

Na caixa correspondente a File/Url se insere o endereço virtual ou o local onde se encontra o arquivo. Ao importar os dados, carrega-se um objeto criado com as informações contidas

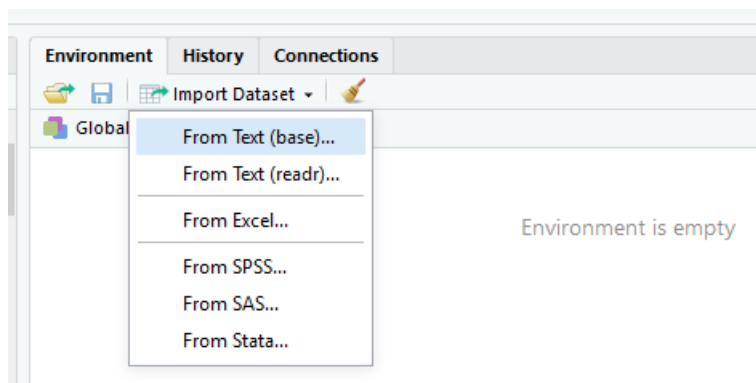


Figura 1.4: Aba *Import Dataset*

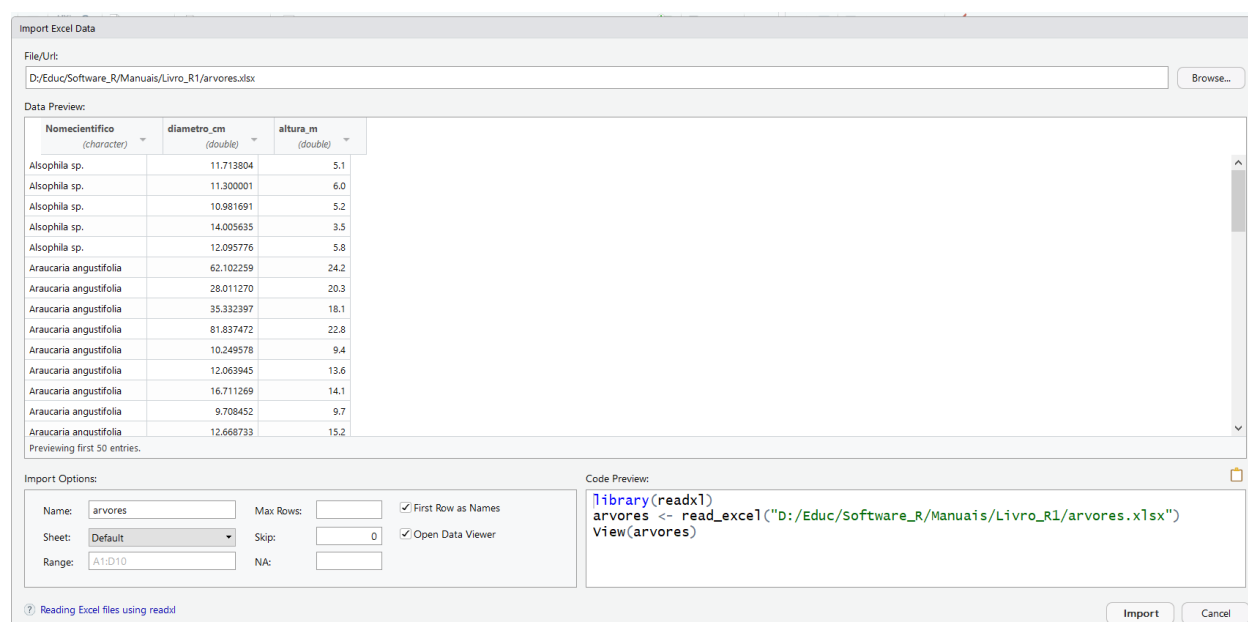


Figura 1.5: Caixa de informações do Import Data

no arquivo. No nosso exemplo, carregamos a planilha arvores (arquivo .xls) como mostra a Figura 1.5, derivado do caminho “Import Dataset > From Excel” do Environment.

O campo *Code Preview* mostra o comando que está sendo criado para a importação destes dados. Em *Import Options*, delimita-se opções do objeto como o nome (*name*), o número máximo de linhas (*Max Rows*), quantas linhas serão puladas na importação do arquivo (*Skip*), o tratamento das células em branco (*NA*) e se a primeira linha contém os nomes (*First Row as Names*).

Com relação à importação de arquivos de texto separado por caracteres (.csv), ela se dá via “Import Dataset > From Text (readr)” do Environment. Constam algumas solicitações diferentes a serem determinadas pelo usuário no campo *Import Options*, conforme mostra a Figura 1.6. Uma questão importante é a opção *Delimiter*, a qual o pesquisador tem que prestar atenção quando o arquivo está separado por vírgulas (*Comma*), ponto e vírgula

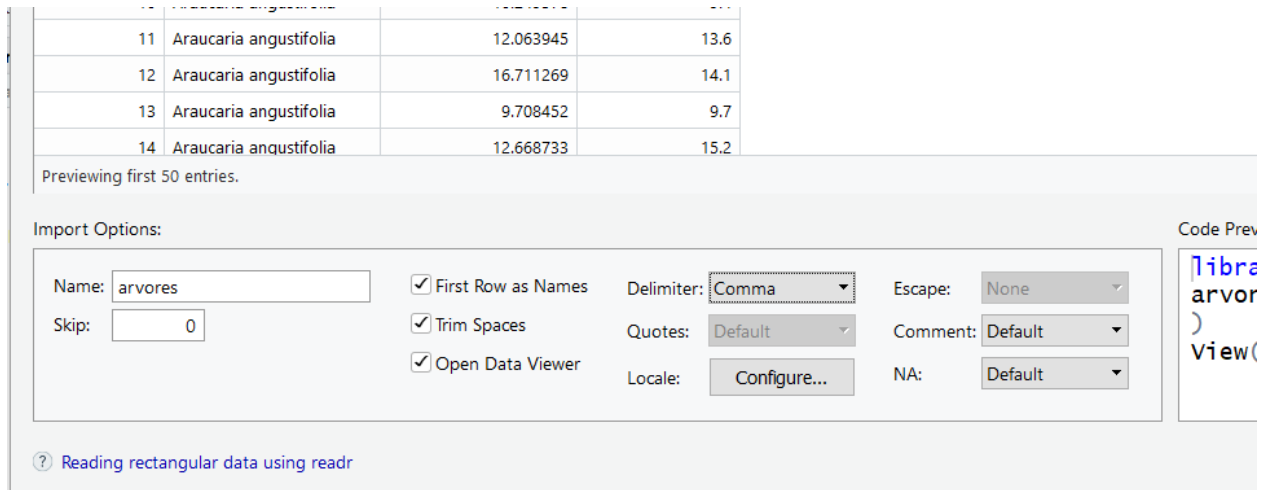


Figura 1.6: Opções da importação de arquivos .csv

(*Semicolon*) ou outro tipo de caractere. A opção *Locale > Configure...* oportuniza determinar os tipos de marca decimal e codificação de textos, por exemplo.

Importante mencionar que em ambos os casos de importação, no campo *Dada Preview* onde constam os dados do arquivo a ser importado, é possível determinar o tipo de dado que cada “coluna” contém. Isto é extremamente importante, pois campos que possuem números, que serão posteriormente utilizados em operações aritméticas, por exemplo, devem ser configurados como tal. No entanto, como será visto adiante, a alteração do tipo do dado também pode ser feita posteriormente sem problema algum.

Alguns tipos de dados:

- **Numeric:** números, valores decimais em geral (5.4).
- **Integer:** números (4).
- **Character:** variável de texto, ou *string* (casa).
- **Double:** cria um vetor de precisão dupla, que abarca os números.
- **Logical:** operadores booleanos (TRUE, FALSE).
- **Date:** opção para datas.
- **Time:** vetor para séries de tempo.
- **Factor:** variável nominal, inclusive como fator ordenado, representam categorias.

1.6 Salvar arquivo de dados

O banco de dados que o R armazena na memória pode ser salvo, junto com todo o ambiente, usando o ícone de disquete na aba “Environment” (salva como arquivo .RData), e depois carregado pelo ícone de pasta (Abrir dados...) na mesma aba. Desta forma, salvará todos os objetos criados no ambiente de trabalho.

Outra opção com mesmo efeito é utilizar o comando a seguir diretamente no console do

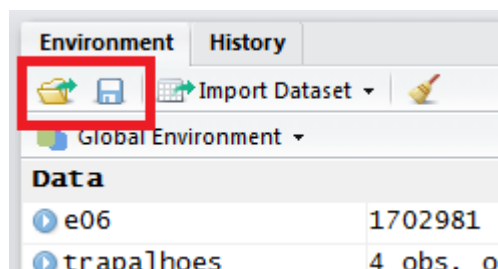


Figura 1.7: Atalho para abrir e salvar arquivo de dados

RStudio:

```
save("nomeDoObjeto",file="nomeDoArquivo.RData")
```

O nome do objeto pode ser uma lista de objetos para salvar mais de um objeto do ambiente, `list("objeto1", "objeto2")`. Para carregar um arquivo RData no ambiente, o comando a ser utilizado pelo usuário é

```
load("arquivo.RData"),
```

desde que o arquivo esteja no diretório de trabalho do R.

É possível exportar as bases trabalhadas para vários formatos de arquivos de dados e de texto, como seguem alguns exemplos:

- `write.csv(nomeobjeto,"file.csv", sep=";")`: salvando em arquivo csv.
- `write.foreign(nomeobjeto,"d:/nome.sps")`: arquivos sps.
- `write.foreign(nomeobjeto,"d:/nome.dta")`: arquivos dta.
- `write.foreign(nomeobjeto,"d:/nome.sas7bdat")`: arquivos sas7bdat.

1.7 Diretórios de trabalho

Os trabalhos efetuados via Rstudio, incluindo as bases de dados, os objetos, os resultados das fórmulas, os cálculos aplicados sobre os vetores e demais arquivos resultantes da utilização do programa podem ser salvos em seu diretório de arquivos. Após instalado o Rstudio destina um diretório padrão salvar estes arquivos, o qual pode ser verificado com o comando `getwd()`.

Este caminho padrão, por sua vez, pode ser alterado via comando

```
setwd("C://file/path")
```

onde o usuário escolhe a pasta desejada que ficará como padrão. O comando `dir()` mostra ao usuário os documentos que constam no diretório padrão ou o escolhido para a consulta.

1.8 Operações

1.8.1 Operações Aritméticas

A realização de uma operação aritmética no R acontece da seguinte forma: onde a resolução das operações segue o padrão, ou seja, primeiro exponenciações, seguido de multiplicações e divisões, deixando por ultimo adições e subtrações, de acordo com a ordem que estão dispostas. Para alterar a prioridade da resolução de operações fazemos o uso do parenteses para destacar a operação que deve ser prioritária na resolução. Seguem alguns exemplos efetuados diretamente no console do RStudio:

```
# soma  
19+26
```

```
[1] 45
```

```
# subtração  
19-26
```

```
[1] -7
```

```
# divisão  
4/2
```

```
[1] 2
```

```
# multiplicação  
4*2
```

```
[1] 8
```

```
# exponenciação  
4^2
```

```
[1] 16
```

```
# prioridade de resolução  
19 + 26 /4 -2 *10
```

```
[1] 5.5
```

```
((19 + 26) /(4 -2))*10
```

```
[1] 225
```

```
# raiz quadrada  
sqrt(16)
```

```
[1] 4
```

```
# Logaritmo  
log(1)
```

```
[1] 0
```

1.8.2 Operações Lógicas

O ambiente de programação Rstudio trabalha com algumas operações lógicas, que serão importantes na manipulação de bases de dados:

- $a == b$ (“a” é igual a “b”)
- $a != b$ (“a” é diferente a “b”)
- $a > b$ (“a” é maior que “b”)
- $a < b$ (“a” é menor que “b”)
- $a >= b$ (“a” é maior ou igual a “b”)
- $a <= b$ (“a” é menor ou igual a “b”)
- `is.na` (“a” é missing - faltante)
- `is.null` (“a” é nulo)

Seguem alguns exemplos da aplicação das operações lógicas:

```
# maior que  
2 > 1
```

```
[1] TRUE
```

```
1 > 2
```

```
[1] FALSE
```

```
# menor que  
1 < 2
```

```
[1] TRUE
```

```
# maior ou igual a  
0 >= (2+(-2))
```

```
[1] TRUE
```

```
# menor ou igual a  
1 <= 3
```

```
[1] TRUE
```

```
# conjunção  
9 > 11 & 0 < 1
```

```
[1] FALSE
```

```
# ou  
6 < 5 | 0 > -1
```

```
[1] TRUE
```

```
# igual a  
1 == 2/2
```

```
[1] TRUE
```

```
# diferente de  
1 != 2
```

```
[1] TRUE
```

1.9 Criação de variáveis

A linguagem de programação R se configura em uma linguagem orientada a objetos, ou seja, a todo tempo estamos criando diversos tipos de objetos e efetuando operações com os mesmos. Por exemplo, a criação de listas, bases de dados, união de bases de dados, data.frames e até mesmo mapas!

```
#Criando um objeto simples  
objeto = "meu primeiro objeto" #enter  
#Agora para retomar o objeto criado:  
objeto #enter
```

```
[1] "meu primeiro objeto"
```

```
#Pode ser efetuada uma operação:  
a= 2+1  
a
```

```
[1] 3
```

O comando `ls()` lista todos os objetos que estão criados no ambiente e `rm(x)` remove o objeto indicado (x). Para remover todos os objetos de uma só vez utiliza-se `rm(list=ls())`.

```
#Lista objetos do ambiente  
ls()
```

```
[1] "a"      "objeto"
```

```
#Remover um banco de dados  
rm(a)
```

1.9.1 Conversão de uma variável

Para a aplicação de algumas funções é importante que cada variável esteja corretamente classificada, o que em alguns casos não ocorre durante o reconhecimento automático do R. Precisamos então reconhecê-la como variável texto, numérica ou fator. Além disso, a classe `ordered` se aplica a variáveis categóricas que podem ser consideradas ordenáveis.


```
idade=c('11', '12', '31')
nomes=c("Elisa", "Priscila", "Carol")
cep=c(98700000,98701000,98702000)
idade= as.numeric(idade)
idade
```

```
[1] 11 12 31
```

```
cep = as.character(cep)
cep
```

```
[1] "98700000" "98701000" "98702000"
```

1.10 Alguns comandos essenciais

A função `head()` mostra as 6 primeiras colunas do arquivo para se ter uma noção do conteúdo. No caso do mesmo ser um `data.frame`, podemos solicitar o número de valores ou linhas a serem mostrados no console através do parâmetro `n` ou na ausência deste, todas as linhas serão impressas, como exemplo `head(x ,n=2)` para ver as duas primeiras linhas.

O comando `summary()` efetua o resumo dos dados, se for qualitativa mostra a frequência absoluta das categorias e se for quantitativa apresenta as categorias. No exemplo abaixo trabalharemos com uma base de dados de treinamento denominada “iris” que está acessível no *software* RStudio através do comando que carrega dados específicos `data()`:

```
#Carregando dados da base do RStudio iris.
```

```
data(iris)
```

```
#Visualizando as primeiras 6 colunas
```

```
head(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

```
#Resumo do objeto
```

```
summary(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min.	:4.30	Min. :2.00	Min. :1.00	Min. :0.1	setosa :50
1st Qu.	:5.10	1st Qu.:2.80	1st Qu.:1.60	1st Qu.:0.3	versicolor:50
Median	:5.80	Median :3.00	Median :4.35	Median :1.3	virginica :50

```

Mean    :5.84    Mean    :3.06    Mean    :3.76    Mean    :1.2
3rd Qu.:6.40    3rd Qu.:3.30    3rd Qu.:5.10    3rd Qu.:1.8
Max.    :7.90    Max.    :4.40    Max.    :6.90    Max.    :2.5

```

O comando `names()` lista os nomes das colunas dos bancos de dados escolhidos, enquanto `tail()` mostra as últimas seis linhas.

```

#Para visualizar os nomes das colunas dos dados:
names(iris)

```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```

#visualizar as ultimas seis linhas do objetos
tail(iris)

```

```

      Sepal.Length Sepal.Width Petal.Length Petal.Width  Species
145          6.7       3.3       5.7       2.5 virginica
146          6.7       3.0       5.2       2.3 virginica
147          6.3       2.5       5.0       1.9 virginica
148          6.5       3.0       5.2       2.0 virginica
149          6.2       3.4       5.4       2.3 virginica
150          5.9       3.0       5.1       1.8 virginica

```

Para que o pesquisador conheça melhor as bases de dados em que está atuando, o comando `class()` serve para identificar o tipo de base ou dados da base. Com o exemplo abaixo constata-se que o objeto “iris” é um *data frame*, a variável “Sepal.Length” é uma variável numérica e que a variável numérica.

```
class(iris)
```

```
[1] "data.frame"
```

```
class(iris$Sepal.Length)
```

```
[1] "numeric"
```

```
class(iris$Especie)
```

```
[1] "NULL"
```

Efeito semelhante possui o comando `ls.str()`:

```
ls.str(iris)
```

```

Petal.Length :  num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
Petal.Width  :  num [1:150] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
Sepal.Length :  num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
Sepal.Width  :  num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
Species      :  Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...

```

Os comandos `ncol()` e `nrow()` mostram o número de colunas e o número de linhas do objeto, respectivamente.

1.10.1 Funções *View* e *dim*

A função `View()` permite visualizar os elementos no script do dataframe requisitado, enquanto a função `dim()` (abreviatura de dimensões) fornece o número de linhas e de colunas, respectivamente.

```
View(iris)
dim(iris)
```

```
[1] 150    5
```

Para alterar um nome de uma variável pode ser utilizado o comando `colnames`. No exemplo acima, vamos alterar o nome da coluna “Species” para “Especie”.

```
#Alterar o nome da coluna, sendo que o '[5]' indica que está na quinta coluna.
colnames(iris)[5]='Especie'
```

Para selecionarmos uma coluna do objeto “iris”, por exemplo a coluna “Sepal.Length”, poderíamos digitar no console o comando `iris$Sepal.Length`. O padrão de carregamento da base de dados nos obriga a dizer ao R qual é a base que quer selecionar (iris), inserindo o símbolo `$` e após o nome da coluna a qual deseja as informações. Para criar um novo objeto com esta informação, basta dizer ao R, como já visto acima, por exemplo: `novoobjeto=iris$novacoluna`.

No entanto, para acessar os dados sem o uso do símbolo `$`, podemos usar o seguinte comando: `attach(iris)`. Assim, podemos efetuar o sumário da coluna “Petal.Width”:

```
#Definindo a função attach para o objeto 'dados'.
attach(iris)
#Efetuando o sumário de 'pop.total'.
summary(Petal.Width)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.1	0.3	1.3	1.2	1.8	2.5

```
#Como a coluna 'distrito' é um fator, o sumário será
#a contagem da quantidade de cada fator na coluna.
summary(Especie)
```

setosa	versicolor	virginica
50	50	50

1.10.2 Comando *tapply*

O comando `tapply()` agrega os dados pelos níveis das variáveis qualitativas. Note que a coluna “Especie” possui dados em forma de fatores. Assim, para filtrarmos a informação (coluna “Sepal.Length”) média por *Especie*, podemos utilizar:

#Função 'tapply', número médio da população total por distrito.

```
tapply(Sepal.Length, Especie, mean)
```

```
setosa versicolor virginica
5.006      5.936      6.588
```

No caso da coluna “Sepal.Length”, se ela possuir um registro NA (faltante), para que se efetue a média por esta coluna neste quesito, há que se adicionar o parâmetro `na.rm=T`, que ignora as células faltantes para calcular-se a média:

#Função 'tapply' considerando NAs:

```
tapply(Sepal.Length, Especie, mean)
```

```
setosa versicolor virginica
5.006      5.936      6.588
```

#Função 'tapply' sem considerar NAs:

```
tapply(Sepal.Length, Especie, mean, na.rm=T)
```

```
setosa versicolor virginica
5.006      5.936      6.588
```

1.10.3 Comando *subset*

Utiliza-se o comando `subset()` para formar um subconjunto de dados o qual desejamos selecionar de um objeto. Por exemplo, se quisermos criar um novo objeto com somente os dados da “Especie” setosa:

```
dadossetosa=subset(iris, Especie=='setosa')
```

```
head(dadossetosa)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Especie
1           5.1          3.5          1.4          0.2  setosa
2           4.9          3.0          1.4          0.2  setosa
3           4.7          3.2          1.3          0.2  setosa
4           4.6          3.1          1.5          0.2  setosa
5           5.0          3.6          1.4          0.2  setosa
6           5.4          3.9          1.7          0.4  setosa
```

Pode ser configurado mais de uma condição para a filtragem dos dados, por exemplo, além de serem filtrados os dados referentes a Especie setosa, aquelas na qual o Sepal.Length é superior a 5. Como no exemplo, criamos um novo objeto com estas condições:

```
dadossetosa2=subset(iris, Especie=='setosa' & Sepal.Length>5)
```

```
head(dadossetosa2)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Especie
1           5.1          3.5          1.4          0.2  setosa
```

6	5.4	3.9	1.7	0.4	setosa
11	5.4	3.7	1.5	0.2	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa

1.11 Estrutura de dados

1.11.1 Vetores

Os fatores são uma classe especial de vetores, que definem variáveis categóricas de classificação, como os tratamentos em um experimento fatorial, ou categorias em uma tabela de contingência.

```
# Criação de um vetor
x= c(2, 4, 6)
x
```

```
[1] 2 4 6
```

Os vetores podem ser criados a partir de uma sequência numérica ou mesmo de um intervalo entre valores:

```
x= c(2:6)
x
```

```
[1] 2 3 4 5 6
```

```
# Criação de um vetor a partir do intervalo entre cada elemento e valores
#mínimo e máximo
x= seq(2, 3, by=0.5)
x
```

```
[1] 2.0 2.5 3.0
```

Criação de um vetor através de uma repetição também é útil em várias situações. No primeiro exemplo repete o intervalo de 1 a 3 4 vezes e no segundo exemplo, a cada 3 vezes:

```
x= rep(1:3, times=4)
x
```

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

```
y= rep(1:3, each=3)
y
```

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

A função factor cria um fator, a partir de um vetor:

```
sexo<-factor(rep(c("F", "M"),each=8))
sexo
```

```
[1] F F F F F F F F M M M M M M M M
Levels: F M
```

```
numeros=rep(1:3,each=3)
numeros
```

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

```
numeros.f<-factor(numeros)
numeros.f
```

```
[1] 1 1 1 2 2 2 3 3 3
Levels: 1 2 3
```

Fatores têm um atributo que especifica seus níveis ou categorias (levels), que seguem ordem alfanumérica crescente, por *default*. Em muitas análises essa ordem é de fundamental importância e dessa forma pode ser alterada através do argumento levels, por exemplo, para que possa ser colocado o controle antes dos tratamentos:

```
tratamentos=factor(rep(c("controle","adubo A","adubo B"), each=4))
tratamentos
```

```
[1] controle controle controle controle adubo A adubo A adubo A adubo A
[9] adubo B adubo B adubo B adubo B
Levels: adubo A adubo B controle
```

```
tratamentos=factor(rep(c("controle","adubo A","adubo B"), each=4),
levels=c("controle", "adubo A", "adubo B"))
tratamentos
```

```
[1] controle controle controle controle adubo A adubo A adubo A adubo A
[9] adubo B adubo B adubo B adubo B
Levels: controle adubo A adubo B
```

Fatores podem conter níveis não usados (vazios):

```
participantes=factor(rep("mulheres",10), levels=c("mulheres","homens"))
participantes
```

```
[1] mulheres mulheres mulheres mulheres mulheres mulheres mulheres mulheres
[9] mulheres mulheres
Levels: mulheres homens
```

Também é possível aplicar uma função aos subconjuntos de um vetor definidos por um fator utilizando a função `tapply()`. Criamos um objeto com o sexo das pessoas, seguido pela dieta e peso (que caracterizamos como numérico). Depois, determinamos a média de peso frente ao sexo e a dieta

```

sexo=factor(rep(c("F","M"),each=9))
dieta=factor(rep(rep(c("normal","light","diet"), each=3),2),
levels=c("normal", "light","diet"))
peso=c(90, 89, 78, 69, 85, 69, 77, 89, 80, 60, 75, 79, 65, 94,
       69, 85, 69, 77)
sexo

```

```

[1] F F F F F F F F F M M M M M M M M
Levels: F M

```

```
dieta
```

```

[1] normal normal normal light light light diet diet diet normal
[11] normal normal light light light diet diet diet
Levels: normal light diet

```

```
peso=as.numeric(peso)
```

```

# média de peso frente ao sexo e dieta
tapply(peso,list(sexo,dieta), mean)

```

```

      normal light diet
F  85.67 74.33   82
M  71.33 76.00   77

```

1.11.1.1 Função *table*

Para contar elementos em cada nível de um fator, usa-se a função `table`:

```
table(participantes)
```

```

participantes
mulheres    homens
      10         0

```

A função pode fazer tabulações cruzadas, gerando uma tabela de contingência, esse tipo de tabela é usado para registrar observações independentes de duas ou mais variáveis aleatórias:

```
table(sexo,dieta)
```

```

      dieta
sexo normal light diet
F         3     3     3
M         3     3     3

```

1.11.2 Matrizes

A função `matrix` tem a finalidade de criar uma matriz com os valores do argumento `data`, argumento este que insere as variáveis desejadas na matriz. O número de linhas é definido pelo argumento `nrow` e o número de colunas é definido pelo argumento `ncol`:

```
nome.da.matriz= matrix(data=1:12,nrow = 3,ncol = 4)
nome.da.matriz
```

```
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Por *default* (ação tomada pelo *software*), os valores são preenchidos por coluna. Para preencher por linha basta instruir o programa de outra forma, alterando o argumento `byrow` para `TRUE`:

```
nome.da.matriz= matrix(data=1:12,nrow = 3,ncol = 4, byrow=T)
nome.da.matriz
```

```
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

Se a matriz inserida tem menos elementos do que a ordem informada para a matriz, os são repetidos até preenchê-la:

```
lista= list(matrix=c(1,2,1), nrow=3, ncol=2))
lista
```

```
$matrix
      [,1] [,2]
[1,]    1    1
[2,]    2    2
[3,]    1    1
```

1.11.3 Listas

As listas podem ser criadas a partir do comando `list()`.

- **nrow**: corresponde ao número de linhas;
- **ncol**: corresponde ao número de colunas.

Para ver quais elementos estão em suas listas é só chamar pelo nome que foi dado para ela, como no exemplo abaixo. Representa uma coleção de objetos.


```
lista= list(matriz=matrix(c(1,2,1,5,7,9), nrow=3, ncol=2),vetor=1:6)
lista
```

```
$matriz
      [,1] [,2]
[1,]    1    5
[2,]    2    7
[3,]    1    9

$vetor
[1] 1 2 3 4 5 6
```

1.11.3.1 Comandos para manipulação de listas

Para descobrirmos de maneira rápida o números de objetos que há na lista, utilizamos o comando `length(nomedalista)`.

```
lista

$matriz
      [,1] [,2]
[1,]    1    5
[2,]    2    7
[3,]    1    9

$vetor
[1] 1 2 3 4 5 6
```

```
length(lista)
```

```
[1] 2
```

O uso do comando `names(nomedalista)` retorna os nomes dos objetos que estão presentes na lista.

```
names(lista)
```

```
[1] "matriz" "vetor"
```

Para chamar várias listas através usamos o comando da seguinte forma:

```
c(nome1, nome2)

lista.1= list(matriz=matrix(c(1,2,1,5,7,9), nrow=3, ncol=2),
              vetor=1:6)
lista.2= list(nomes=c("Marcelo", "Fábio", "Felipe"),
              idade=c(25, 34, 26))
c(lista.1,lista.2)
```

```
$matriz
      [,1] [,2]
[1,]    1    5
[2,]    2    7
[3,]    1    9

$vetor
[1] 1 2 3 4 5 6

$nomes
[1] "Marcelo" "Fábio"  "Felipe"

$idade
[1] 25 34 26
```

1.11.4 Data frames

Com a função `data.frame()` reunimos vetores de mesmo comprimento em um só objeto. Neste caso são criadas tabelas de dados. Cada observação é descrita por um conjunto de propriedades. Abaixo podemos ver como inserir os dados para criar a “tabela”. Similar como matrizes, porem diferentes colunas podem possuir elementos de natureza diferentes .

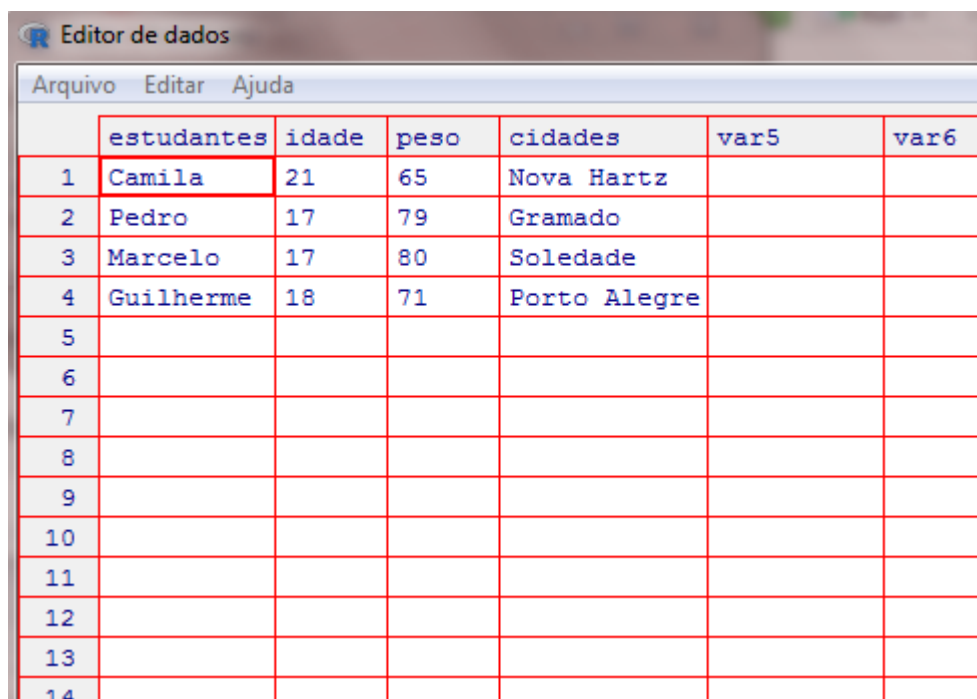
```
estudantes= c("Camila", "Pedro", "Marcelo","Guilherme")
idade=c(21,17,17,18)
peso=c(65,79,80,100)
informacoes=data.frame(estudantes,idade,peso)
informacoes
```

```
  estudantes idade peso
1    Camila    21   65
2    Pedro    17   79
3  Marcelo    17   80
4  Guilherme    18  100
```

Adiciona-se colunas no *data frame* através do comando a seguir, pressupondo que a ordem dos dados esteja correta:

```
nomedodata.frame$variávelaseradicionada
informacoes$cidades=c("Nova Hartz","Gramado","Soledade",
                       "Porto Alegre")
informacoes
```

```
  estudantes idade peso      cidades
1    Camila    21   65  Nova Hartz
2    Pedro    17   79    Gramado
```



	estudantes	idade	peso	cidades	var5	var6
1	Camila	21	65	Nova Hartz		
2	Pedro	17	79	Gramado		
3	Marcelo	17	80	Soledade		
4	Guilherme	18	71	Porto Alegre		
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

Figura 1.8: Editor de dados

```
3   Marcelo      17   80   Soledade
4   Guilherme    18  100 Porto Alegre
```

É possível fazer uma contagem concatenando com a filtragem do pacote `subset`, como no exemplo a contagem dos indivíduos cuja origem é Soledade.

```
length(subset(informacoes$cidades, informacoes$cidades=="Soledade"))
```

```
[1] 1
```

1.12 Manipulação de banco de dados

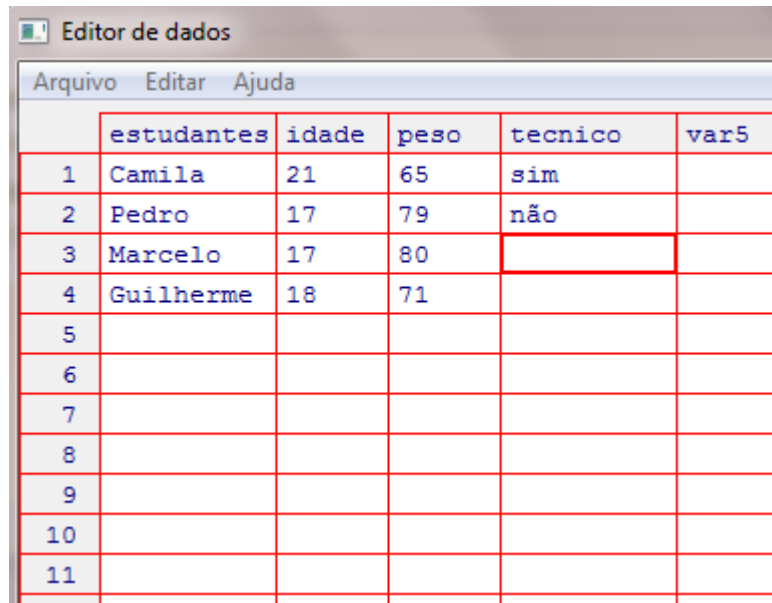
1.13 Função *edit*

Esta função abre uma interface simples de edição de dados em formato planilha, e é útil para pequenas modificações. Mas para salvar as modificações atribua o resultado da função `edit` a um objeto.

Utiliza-se o comando da seguinte forma:

```
novonomedabase = edit(nomeatualdatabase)
```

```
informacoes.2=edit(informacoes)
```



	estudantes	idade	peso	tecnico	var5
1	Camila	21	65	sim	
2	Pedro	17	79	não	
3	Marcelo	17	80		
4	Guilherme	18	71		
5					
6					
7					
8					
9					
10					
11					

Figura 1.9: Acréscimo de uma nova coluna através do editor de dados

Basta clicar no retângulo correspondente a variável que deseja ser modificada, excluir ou adicionar novas colunas.

Logo, chamando o novo banco de dados, teremos:

```
informacoes.2
```

	estudantes	idade	peso	idades
1	Camila	21	65	Nova Hartz
2	Pedro	17	79	Gramado
3	Marcelo	17	80	Soledade
4	Guilherme	18	100	Porto Alegre

1.14 Funções

As funções a seguir são aplicáveis a vetores, `data.frames` e listas, e em muitos casos trazem praticidade a uma análise estatística. Foram criados objetos com informações do nome dos estudantes e altura. Segue o processo de criação do *data frame* com estas informações, lembrando que esta forma de “união” das informações pressupõe que a ordem dos dados esteja correta:

```
# União de um banco de dados (existencia de uma variavel em comum)
```

```
estudantes=c("Guilherme", "Marcelo", "Pedro", "Camila")
altura= c(1.50, 1.9, 1.74, 1.80)
informacoes.3=data.frame(estudantes, altura)
```

Já o comando `merge()` serve para juntar dois *data frames* que possuam uma coluna em comum. Neste caso, unimos o objeto `informacoes.2` com o objeto `informacoes.3` utilizando o nome dos estudantes (informação em comum):

```
informacoes=merge(informacoes.2,informacoes.3, by="estudantes")
```

Adicionar um cálculo entre as colunas é muito simples com o RStudio, neste caso com os dados do peso e altura, pode-se calcular o IMC (Índice de Massa Corporal) em uma nova coluna:

```
informacoes$Imc=c(peso/(altura^2))
informacoes
```

	estudantes	idade	peso	idades	altura	Imc
1	Camila	21	65	Nova Hartz	1.80	28.89
2	Guilherme	18	100	Porto Alegre	1.50	21.88
3	Marcelo	17	80	Soledade	1.90	26.42
4	Pedro	17	79	Gramado	1.74	30.86

Ainda, se houver linhas que tenham pelo menos uma informação faltante (NA), estas podem ser excluídas com o comando `na.omit()`, ou mesmo os NAs serem substituídos por outro caractere (neste caso foi substituído por zero) com o comando `is.na`:

```
# Retirar as linhas que tenham pelo menos um NA:
```

```
informacoes<- na.omit(informacoes)
informacoes
```

	estudantes	idade	peso	idades	altura	Imc
1	Camila	21	65	Nova Hartz	1.80	28.89
2	Guilherme	18	100	Porto Alegre	1.50	21.88
3	Marcelo	17	80	Soledade	1.90	26.42
4	Pedro	17	79	Gramado	1.74	30.86

```
# Substituir NA's por zero no data.frame
```

```
informacoes[is.na(informacoes)] = 0
informacoes
```

	estudantes	idade	peso	idades	altura	Imc
1	Camila	21	65	Nova Hartz	1.80	28.89
2	Guilherme	18	100	Porto Alegre	1.50	21.88
3	Marcelo	17	80	Soledade	1.90	26.42
4	Pedro	17	79	Gramado	1.74	30.86

Outro recurso interessante é a substituição de dados em uma coluna, que pode ser feito de forma automática para uma condição padrão escolhida. No exemplo abaixo, substituímos aquelas informações de idade igual a 17 pelo número 19:

Tabela 1.1: Valores padrão para o IMC

Resultado	Significado
Abaixo de 17	Muito abaixo do peso
Entre 17 e 18,49	Abaixo do peso
Entre 18,5 e 24,99	Peso normal
Entre 25 e 29,99	Acima do peso
Entre 30 e 34,99	Obesidade I
Entre 35 e 39,99	Obesidade II (severa)
Acima de 40	Obesidade III (mórbida)

```
# Substituir números na coluna
```

```
informacoes$idade[informacoes$idade == 17] <- 19
informacoes
```

```
  estudantes idade peso      cidades altura  Imc
1    Camila   21   65   Nova Hartz   1.80 28.89
2  Guilherme   18  100 Porto Alegre   1.50 21.88
3    Marcelo   19   80   Soledade    1.90 26.42
4     Pedro   19   79    Gramado    1.74 30.86
```

A classificação qualitativa das informações, com base em condições definidas pelo usuário podem ser facilmente efetuadas pelo comando `ifelse`. Para quem não tem intimidade com atributos de programação, este comando seleciona “se” (*if*) uma informação desejada é atendida, e cria uma rotina (*else*) que será aplicada “então”.

No nosso exemplo, cria-se um objeto “classificacao” e se a coluna IMC conter dados acima de 25, será marcado como “peso normal”, sendo que do contrário, constará como “excesso de peso”. Após utilizamos o comando `cbind()` para unir os dois objetos pelas colunas. caso não queira utilizar o comando `cbind()`, poderia ser criado uma nova coluna com o nome do objeto sendo “informacoes\$classificacao”.

```
# Classificar qualitativamente informações em um determinado intervalo
```

```
classificacao=ifelse(informacoes$Imc<25, "peso normal",
                     "excesso de peso")
informacoes=cbind(informacoes, classificacao)
informacoes
```

```
  estudantes idade peso      cidades altura  Imc  classificacao
1    Camila   21   65   Nova Hartz   1.80 28.89 excesso de peso
2  Guilherme   18  100 Porto Alegre   1.50 21.88      peso normal
3    Marcelo   19   80   Soledade    1.90 26.42 excesso de peso
4     Pedro   19   79    Gramado    1.74 30.86 excesso de peso
```

No entanto, o IMC possui várias classificações de acordo com o seu resultado (Tabela 1.1), sendo que, por exemplo, resultados abaixo de 17 informam que o indivíduo se encontra

como Muito abaixo do peso, e acima de 40, se encontra em Obesidade III. Para efetuar a classificação desta maneira utilizando o comando `ifelse`, ou seja, com mais de uma condição, pode ser efetuada a estruturação com a aglutinação do comando:

```
informacoes$tipoimc=ifelse(informacoes$Imc<17, "Muito abaixo do peso",
ifelse(informacoes$Imc>=17&informacoes$Imc<=18.49,"Abaixo do peso",
ifelse(informacoes$Imc>=18.5&informacoes$Imc<=24.99,"Peso Normal",
ifelse(informacoes$Imc>=25&informacoes$Imc<=29.99,"Acima do Peso",
ifelse(informacoes$Imc>=30&informacoes$Imc<=34.99,"Obesidade I",
ifelse(informacoes$Imc>=35&informacoes$Imc<=39.99,"Obesidade II",
"Obesidade III"))))))
informacoes
```

	estudantes	idade	peso	idades	altura	Imc	classificacao	tipoimc
1	Camila	21	65	Nova Hartz	1.80	28.89	excesso de peso	Acima do Peso
2	Guilherme	18	100	Porto Alegre	1.50	21.88	peso normal	Peso Normal
3	Marcelo	19	80	Soledade	1.90	26.42	excesso de peso	Acima do Peso
4	Pedro	19	79	Gramado	1.74	30.86	excesso de peso	Obesidade I

A classificação binária dos dados (0,1) também é relevante para o estudo da manipulação dos dados trabalhados pelo pesquisador. Neste exemplo, classificou-se aqueles valores da coluna “classificacao” com o “peso normal” iguais a 1, do contrário classificou-se 0 (zero).

```
# Classificar informações usando o código binário
informacoes$binario= ifelse(informacoes$classificacao
== 'peso normal', 1, 0)
informacoes
```

	estudantes	idade	peso	idades	altura	Imc	classificacao	tipoimc
1	Camila	21	65	Nova Hartz	1.80	28.89	excesso de peso	Acima do Peso
2	Guilherme	18	100	Porto Alegre	1.50	21.88	peso normal	Peso Normal
3	Marcelo	19	80	Soledade	1.90	26.42	excesso de peso	Acima do Peso
4	Pedro	19	79	Gramado	1.74	30.86	excesso de peso	Obesidade I

	binario
1	0
2	1
3	0
4	0

O comando `rbind()` é utilizado para incluir linhas novas abaixo de um objeto já criado pelo pesquisador, sendo que é importante o cuidado de que estas novas informações tenham os mesmos campos (colunas). A exemplo, pede-se para incluir uma nova pessoa no *data frame* `informacoes`: Francisco, 30 anos de idade, peso 59, natural de Ijuí, IMC 21.3387, classificado como peso normal. Lembrando de incluir os campos “tipoimc” e “binario”.

```
nov1=data.frame(estudantes="Francisco", idade=30, peso=59,
                cidades="Ijuí",
                altura="1,59",
```

```

      Imc= 23.30,
      classificacao= "peso normal",
      tipoimc="Peso Normal",
      binario=1)
informacoes=rbind(informacoes, novo1)
informacoes

```

	estudantes	idade	peso	idades	altura	Imc	classificacao	tipoimc
1	Camila	21	65	Nova Hartz	1.8	28.89	excesso de peso	Acima do Peso
2	Guilherme	18	100	Porto Alegre	1.5	21.88	peso normal	Peso Normal
3	Marcelo	19	80	Soledade	1.9	26.42	excesso de peso	Acima do Peso
4	Pedro	19	79	Gramado	1.74	30.86	excesso de peso	Obesidade I
5	Francisco	30	59	Ijuí	1,59	23.30	peso normal	Peso Normal
	binario							
1		0						
2		1						
3		0						
4		0						
5		1						

Outra forma de incluir informações adicionais nos *data frames* através de atributos é utilizando o pacote `dplyr`. Decide-se criar um campo “faixa etária”, sendo que aqueles indivíduos com idade acima de 21 chamaremos de “adulto” e do contrário “não adulto”.

```
require(dplyr)
```

Carregando pacotes exigidos: `dplyr`

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

`filter`, `lag`

The following objects are masked from 'package:base':

`intersect`, `setdiff`, `setequal`, `union`

```

informacoes= mutate(informacoes,
                     "faixa etaria"= ifelse(informacoes$idade<21,
                                             "não adulto", "adulto"))
informacoes

```

	estudantes	idade	peso	idades	altura	Imc	classificacao	tipoimc
1	Camila	21	65	Nova Hartz	1.8	28.89	excesso de peso	Acima do Peso
2	Guilherme	18	100	Porto Alegre	1.5	21.88	peso normal	Peso Normal
3	Marcelo	19	80	Soledade	1.9	26.42	excesso de peso	Acima do Peso

4	Pedro	19	79	Gramado	1.74	30.86	excesso de peso	Obesidade I
5	Francisco	30	59	Ijuí	1,59	23.30	peso normal	Peso Normal

	binario	faixa etaria
1	0	adulto
2	1	não adulto
3	0	não adulto
4	0	não adulto
5	1	adulto

A (re)ordenação das colunas de um *data frame* pode ser muito útil em alguns casos, sendo extremamente fácil efetuá-la, cada número representa o número da respectiva coluna:

```
# Reordenar colunas
informacoes=informacoes[c(8,2,3,4,1,6,5,7,9)]
```

Caso se queira a inversão total da ordem das colunas do objeto estudado, o comando `rev()` pode ser útil:

```
# Inversão do posicionamento dos elementos
rev(informacoes)
```

	binario	classificacao	altura	Imc	estudantes	idades	peso	idade
1	0	excesso de peso	1.8	28.89	Camila	Nova Hartz	65	21
2	1	peso normal	1.5	21.88	Guilherme	Porto Alegre	100	18
3	0	excesso de peso	1.9	26.42	Marcelo	Soledade	80	19
4	0	excesso de peso	1.74	30.86	Pedro	Gramado	79	19
5	1	peso normal	1,59	23.30	Francisco	Ijuí	59	30

	tipoimc
1	Acima do Peso
2	Peso Normal
3	Acima do Peso
4	Obesidade I
5	Peso Normal

A função `table()` faz a contagem os dados; já o comando `sort()` ordena os objetos em ordem crescente (caso queira no formato decrescente, informar `decreasing=TRUE`).

```
# contagem de objetos
table(informacoes$classificacao)
```

excesso de peso	peso normal
3	2

```
# Ordenar os objetos em ordem crescente
sort(informacoes$idade)
```

```
[1] 18 19 19 21 30
```

A ordenação de todo o *data frame* a partir de uma variável, pode ser realizada utilizando o

comando `order`, sendo que pode ser realizada inclusive com variáveis categóricas (no exemplo abaixo o nome das cidades).

```
# Ordem decrescente
```

```
informacoes[order(informacoes$idade, decreasing = TRUE),]
```

	tipoimc	idade	peso	cidades	estudantes	Imc	altura	classificacao
5	Peso Normal	30	59	Ijuí	Francisco	23.30	1,59	peso normal
1	Acima do Peso	21	65	Nova Hartz	Camila	28.89	1.8	excesso de peso
3	Acima do Peso	19	80	Soledade	Marcelo	26.42	1.9	excesso de peso
4	Obesidade I	19	79	Gramado	Pedro	30.86	1.74	excesso de peso
2	Peso Normal	18	100	Porto Alegre	Guilherme	21.88	1.5	peso normal
	binario							
5	1							
1	0							
3	0							
4	0							
2	1							

```
#ordem crescente
```

```
informacoes[order(informacoes$idade, decreasing = FALSE),]
```

	tipoimc	idade	peso	cidades	estudantes	Imc	altura	classificacao
2	Peso Normal	18	100	Porto Alegre	Guilherme	21.88	1.5	peso normal
3	Acima do Peso	19	80	Soledade	Marcelo	26.42	1.9	excesso de peso
4	Obesidade I	19	79	Gramado	Pedro	30.86	1.74	excesso de peso
1	Acima do Peso	21	65	Nova Hartz	Camila	28.89	1.8	excesso de peso
5	Peso Normal	30	59	Ijuí	Francisco	23.30	1,59	peso normal
	binario							
2	1							
3	0							
4	0							
1	0							
5	1							

```
#ordem crescente
```

```
informacoes[order(informacoes$cidades, decreasing = FALSE),]
```

	tipoimc	idade	peso	cidades	estudantes	Imc	altura	classificacao
4	Obesidade I	19	79	Gramado	Pedro	30.86	1.74	excesso de peso
5	Peso Normal	30	59	Ijuí	Francisco	23.30	1,59	peso normal
1	Acima do Peso	21	65	Nova Hartz	Camila	28.89	1.8	excesso de peso
2	Peso Normal	18	100	Porto Alegre	Guilherme	21.88	1.5	peso normal
3	Acima do Peso	19	80	Soledade	Marcelo	26.42	1.9	excesso de peso
	binario							
4	0							
5	1							

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

O comando `rank()` cria uma ranqueamento crescente das informações. Se pretende-se, por exemplo, criar uma coluna com o ranking dos valores do IMC, pode ser utilizado:

```
informacoes$rankingImc=rank(informacoes$Imc)
informacoes
```

	tipoimc	idade	peso	idades	estudantes	Imc	altura	classificacao
1	Acima do Peso	21	65	Nova Hartz	Camila	28.89	1.8	excesso de peso
2	Peso Normal	18	100	Porto Alegre	Guilherme	21.88	1.5	peso normal
3	Acima do Peso	19	80	Soledade	Marcelo	26.42	1.9	excesso de peso
4	Obesidade I	19	79	Gramado	Pedro	30.86	1.74	excesso de peso
5	Peso Normal	30	59	Ijuí	Francisco	23.30	1,59	peso normal

	binario	rankingImc
1	0	4
2	1	1
3	0	3
4	0	5
5	1	2

1.15 Funções Matemáticas

A utilização de funções matemáticas no RStudio contribui para que o pesquisador possa realizar vários experimentos com seus dados. Os cálculos podem ser efetuados diretamente no console do programa ou aplicados aos objetos criados:

```
log(1.5)
```

```
[1] 0.4055
```

```
exp(1)
```

```
[1] 2.718
```

No caso do *data frame* o qual foi criado acima (“informacoes”), pode-se buscar as informações dos valores mínimos (função `min()`), máximos (`max()`) da base:

```
max(informacoes$idade)
```

```
[1] 30
```

```
min(informacoes$idade)
```

```
[1] 18
```

Ainda, se o interesse está em descobrir a posição, no `*data frame`, do peso mínimo e máximo da amostra utiliza-se o comando `which.min` e `which.max`.

```
# Para descobrir em qual posição se encontra o peso mínimo:
which.min(informacoes$peso)
```

```
[1] 5
```

```
which.max(informacoes$peso)
```

```
[1] 2
```

Para descobrir qual é o estudante que possui o peso mínimo, por exemplo, ou o Imc máximo, utiliza-se o seguinte comando (notem que os resultados trazem a lista de todos os estudantes comparados):

```
informacoes$estudantes[which.min(informacoes$peso)]
```

```
[1] Francisco
```

```
Levels: Camila Guilherme Marcelo Pedro Francisco
```

```
informacoes$estudantes[which.max(informacoes$Imc)]
```

```
[1] Pedro
```

```
Levels: Camila Guilherme Marcelo Pedro Francisco
```

O arredondamento de valores numéricos pode ser feito utilizando o comando `round()`, o qual o pesquisador informa o número de casas decimais:

```
# Arredondar para n casas decimais
round(informacoes$Imc, 2)
```

```
[1] 28.89 21.88 26.42 30.86 23.30
```

Já o comando `signif()` determina o número de algarismos significativos da série escolhida, ou seja, ele arredonda para os valores em seu primeiro argumento com os número de dígitos determinados:

```
x2 <- pi * 100^(-1:3)
round(x2, 3)
```

```
[1] 3.100e-02 3.142e+00 3.142e+02 3.142e+04 3.142e+06
```

```
signif(x2, 3)
```

```
[1] 3.14e-02 3.14e+00 3.14e+02 3.14e+04 3.14e+06
```

A soma do total da coluna idade, o desvio padrão, a variância, a média aritmética e mediana podem ser encontrados, respectivamente, pelos comandos `sum()`, `sd()`, `var()`, `mean()`, `median()`:

```
# Realiza a somatória dos valores
sum(informacoes$idade)
```

```
[1] 107
```

```
# Desvio padrão
sd(informacoes$idade)
```

```
[1] 4.93
```

```
# Variância
var(informacoes$idade)
```

```
[1] 24.3
```

```
# Calcula a média aritmética dos valores
mean(informacoes$idade)
```

```
[1] 21.4
```

```
# Informa o valor mediano do conjunto
median(informacoes$idade)
```

```
[1] 19
```

O comando `quantile()` oferece a possibilidade de obter os quartis dos dados de acordo com as probabilidades estabelecidas pelo pesquisador. No exemplo, explora-se a variável `idade`:

```
quantile(informacoes$idade, probs = c(0.5, 1, 2, 5, 10, 50)/100)
```

```
0.5%    1%    2%    5%   10%   50%
18.02 18.04 18.08 18.20 18.40 19.00
```

1.16 Conversão de datas

A configuração e padronização dos formato de datas no RStudio podem ser efetuadas pelo pesquisador, primeiramente ao carregar a base de dados no programa e em um segundo momento durante a manipulação das informações. Assim, seguem alguns dos procedimentos para a correta alteração dos padrões de datas:

```
abertura <- c("03/02/69", "17/08/67")
fechamento <- c("2000-20-01", "1999-14-08")
abertura <- as.Date(abertura, format = "%d/%m/%y")
fechamento <- as.Date(fechamento, format = "%Y-%d-%m")
```

```
# Diferença de dias dos intervalos informados
abertura-fechamento
```

```
Time differences in days
```

```
[1] -11308 24840
```


Capítulo 2

Estatística Descritiva

A Estatística é uma ciência cujo campo de aplicação estende-se a diferentes áreas do conhecimento humano. Tem por objetivo fornecer métodos e técnicas que permitem lidar, racionalmente, com situações sujeitas a incertezas. Apresenta um conjunto de técnicas e métodos de pesquisa que envolvem o planejamento de estudos (experimentais e observacionais), a coleta e organização de dados, a inferência, a análise e a disseminação de informação.

Alguns termos extensamente utilizados em estatística, são definidos a seguir (Triola 2011):

População: é uma coleção completa de todos os elementos (valores, pessoas, medidas etc.) a serem estudados.

Censo: é uma coleção de dados relativos a todos os elementos de uma população.

Amostra: é uma sub-coleção de elementos extraídos de uma população. Parâmetro é a medida numérica que descreve uma característica de uma população.

Estatística: é uma medida numérica que descreve uma característica de uma amostra.

2.1 Natureza da medida das variáveis

Variáveis reporta-se a características ou atributos que podem tomar diferentes valores ou categorias, o que se opõe ao conceito de constante (Almeida e Freire 2000). Assim, variável pode ser definida como sendo a característica dos elementos da amostra ou da população que nos interessa estudar estatisticamente.

Variáveis podem ser classificadas da seguinte forma:

Variáveis quantitativas: consistem em números que representam contagens ou medidas. Dividem-se em:

- a) Variáveis discretas: resultam em um conjunto finito de valores possíveis, ou de um conjunto enumerável desses valores. Ex. número de unidades produzidas.

- b) Variáveis contínuas: resultam de um número infinito de valores possíveis que podem ser associados a pontos em uma escala contínua de tal maneira que não haja lacunas ou interrupções. Ex. Renda das famílias em reais.

Variáveis qualitativas: ou variáveis categóricas, ou atributos que podem ser separados em diferentes categorias que se distinguem por alguma característica não-numérica. Divididas em:

- a) Variável nominal: caracterizada por dados que consistem apenas em nomes, rótulos ou categorias. Os dados não podem ser dispostos segundo um esquema ordenado (como de baixo para cima). Ex. nacionalidade
- b) Variável ordinal: envolve variáveis representadas por nomes que podem ser dispostos em alguma ordem, mas as diferenças entre os valores dos dados não podem ser determinadas, ou não tem sentido. Esse nível dá informações sobre comparações relativas, mas os graus de diferença não servem para cálculos (Triola 2011). Ex. Grau de escolaridade.

Dado: é o valor assumido por uma variável aleatória em um experimento.

A Estatística subdivide-se em descritiva e inferencial. A estatística descritiva se preocupa em descrever os dados. A estatística inferencial, fundamentada na teoria das probabilidades, se preocupa com a análise destes dados e sua interpretação.

Informações estatísticas em jornais, relatórios e outras publicações que consistem de dados reunidos e apresentados de forma clara e resumida, na forma de tabelas, gráficos ou numéricos, são conhecidos como estatísticas descritivas (ANDERSON, 2002).

Exemplo 1

Estaremos utilizando como exemplo os dados de uma pesquisa (dados simulados), cujo banco de dados está intitulado “Dados_pesquisa.ods”. Os dados são referentes aos resultados obtidos por ocasião de uma pesquisa realizada entre os consumidores a fim de analisar características associadas ao mercado consumidor de sucos, sendo que a amostra é composta de 348 entrevistados aleatoriamente selecionados.

- O objetivo primário do estudo foi determinar variáveis que seriam úteis para caracterizar os consumidores que já conhecem o suco e a possibilidade potencial de futuros consumidores. Há também interesse nas relações entre variáveis das características pessoais desses consumidores ou futuros consumidores.
- A pesquisa foi realizada, depois que os participantes realizaram uma visita técnica às instalações da empresa e puderam conhecer seus produtos e processos.

Para cada entrevistado foram registrados dados para as seguintes variáveis:

Sexo – Gênero sexual;

Divulgacao – Forma de acesso ao suco ou publicidade do mesmo;

Renda_h – Renda por hora do entrevistado;

Praticidade – Aspectos quanto a oferta do suco, como por ex. embalagem;

Sabor – Aspectos relacionados ao sabor;

Pessoas_familia – Número de pessoas que compõe o grupo familiar;

Preço – como cada entrevistado classificava o preço do produto;

consumo_anterior – Se já consumia o suco antes da visita técnica;

consumo_pos – Se consumia o suco após a visita técnica;

Idade – Idade dos consumidores;

Altura_(m) – Altura dos consumidores;

Peso_(Kg) – Peso dos consumidores.

Pede-se:

1. Salvar inicialmente os dados em formato CSV, xlsx ou outro.
2. Ler os dados no “Environment” pelo “Import Dataset...From CSV” ou outro. No exemplo abaixo foram importados os dados diretamente do arquivo hospedado na internet.
3. Carregar o banco de dados, com a finalidade de usar os objetos (variáveis) diretamente nas funções a serem utilizadas.

`attach(nome_da_planilha)`

```
require(readxl)
```

Carregando pacotes exigidos: readxl

```
url <- "https://goo.gl/37Fdzz"
destfile <- "pesquisa_dados.xlsx"
curl::curl_download(url, destfile)
pesquisa_dados <- read_excel(destfile)
attach(pesquisa_dados)
ls.str(pesquisa_dados)
```

```
Altura_(m) :  num [1:348] 1.82 1.9 1.69 1.89 1.9 1.76 1.83 1.81 1.67 1.55 ...
Caso :      num [1:348] 1 2 3 4 5 6 7 8 9 10 ...
consumo_anterior : chr [1:348] "N" "N" "S" "N" "S" "S" "S" "N" "N" "N" "N" "N" "S" "S"
consumo_pos :     chr [1:348] "N" "S" "N" "S" "N" "S" "N" "S" "N" "S" "S" "S" "S" "S"
Divulgacao :     chr [1:348] "Degustacao" "Radio" "TV" "TV" "Degustacao" "TV" "TV" "Radio"
Idade :          num [1:348] 22 21 20 18 16 28 19 19 22 19 ...
Peso_(Kg) :       num [1:348] 78.5 80 54 78 36 82 75 69 58 49 ...
Pessoas_familia : num [1:348] 4 3 3 7 4 4 3 4 1 4 ...
Praticidade :     chr [1:348] "Pessima" "Otima" "Boa" "Pessima" "Ruim" "Boa" "Regular" ...
Preço :          chr [1:348] "Acima_concorrencia" "Abaixo_concorrencia" ...
Renda_h :        chr [1:348] "1.41" "17.34" "6.86" "2.65" "2.01" "11.32" "6.86" "3.25" ...
```

```
Sabor : chr [1:348] "Otimo" "Pessimo" "Bom" "Otimo" "Otimo" "Regular" "Ruim" "Bom" ...
Sexo : chr [1:348] "Feminino" "Feminino" "Feminino" "Feminino" "Masculino" ...
```

2.2 Tabelas e Gráficos

Segundo Barbetta (2010), dados representados em tabelas e gráficos adequados, permitem observar determinados aspectos relevantes, bem como delinear hipóteses a respeito da estrutura dos dados em estudo, o que conhecemos como análise exploratória de dados. Isto pode ser feito inicialmente com a representação em forma de tabelas.

O comando `table()` é utilizado para elaborarmos tabelas de frequências absolutas. Dependendo da variável a ser representada, podemos usar esse comando de diferentes formas:

2.2.1 Tabela simples para apresentação das frequências absolutas

Uma tabela simples considera quantas vezes ocorre cada categoria (ou nível).

```
table(nome_variável)
```

Ex. Variável **Praticidade**

```
table(Praticidade)
```

```
Praticidade
      Boa   Otima Pessima Regular   Ruim
      82     70     21     80     95
```

2.2.2 Tabela cruzada

A tabela cruzada, também conhecida como tabela de dupla entrada, para apresentação das frequências absolutas.

```
table(nome_variável1,nome_variável2)
```

Ex. Construir uma tabela cruzada apresentando as frequências absolutas das variáveis **Sexo** e **Divulgacao**.

```
table(pesquisa_dados$Sexo,pesquisa_dados$Divulgacao)
```

```
          Degustacao Outro Radio  TV
Feminino          78     6   61 147
Masculino          19     1   15  21
```

2.2.3 Tabela cruzada para apresentação das frequências relativas

Com a introdução do comando `prop.table` é possível gerar, facilmente, tabelas de frequências relativas para as variáveis de interesse. As medidas relativas são importantes para comparar distribuições de frequências (Barbetta 2010).

```
prop.table(table(nome_variável1,nome_variável2))
```

Ex. Construir uma tabela cruzada apresentando as frequências relativas das variáveis **Sexo** e **Divulgacao**.

```
prop.table(table(Divulgacao,Sexo))
```

	Sexo	
Divulgacao	Feminino	Masculino
Degustacao	0.224138	0.054598
Outro	0.017241	0.002874
Radio	0.175287	0.043103
TV	0.422414	0.060345

A função `tapply` serve para calcular um valor usando uma variável categórica como condição, ou seja, aplica uma função qualquer (como média, por exemplo) a uma variável quantitativa para cada classe de uma variável categórica. Assim, permite obter em um só comando, a medida para cada categoria.

```
tapply(var_quantitativa,var_categórica, função_desejada)
```

```
tapply(variavel_quantitativa,variavel_qualitativa, mean)
```

Se um registro possui NA, isto é, dados perdidos: com o parâmetro `na.rm=T`, indicamos para o comando ignorar os NAs nos dados e calcular a média.

```
tapply(variavel_quanti, variavel_quali, mean, na.rm=T)
```

2.3 Gráficos

2.3.1 Gráfico de colunas

As frequências podem ser visualizadas graficamente, usando gráficos de barras elementares, que se aplicam à descrição de qualquer variável qualitativa ou quantitativa discreta, vetor de dados ou tabelas.

No entanto, no caso de dados em banco de dados, quando não utilizamos outros mecanismos de atribuição, precisamos usar o comando `table`.

```
barplot(table(nome_variável))
```

Ex. Construir um gráfico de colunas para a variável **Sexo**.

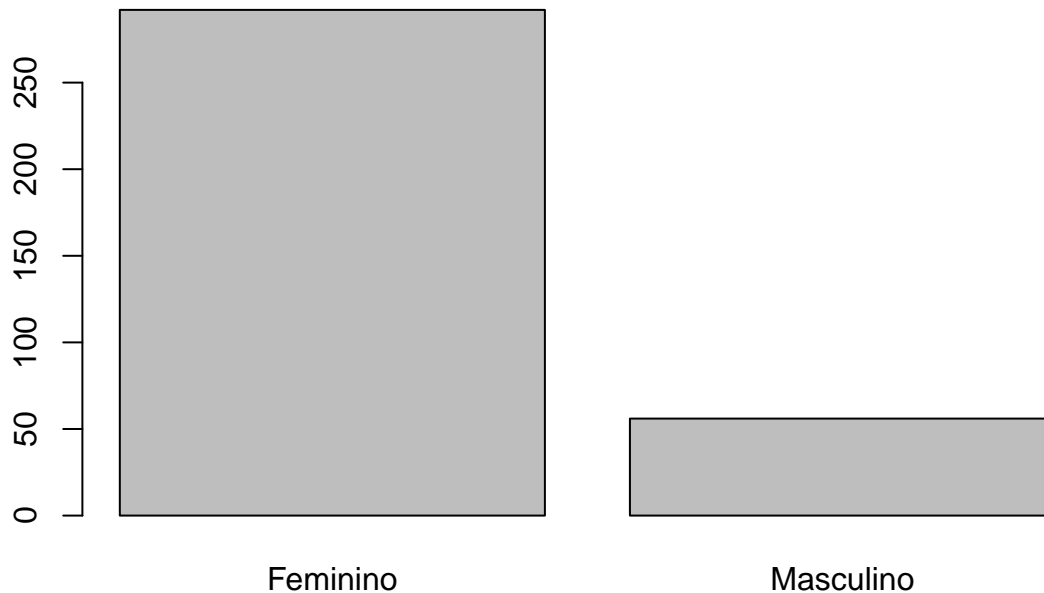


Figura 2.1: Gráfico de colunas com a variável Sexo

```
barplot(table(Sexo))
```

Obs.: É possível personalizar o gráfico, incluindo o título do eixo x (xlab), o título do eixo y (ylab), o título do gráfico (main), a cor da coluna (col) e cor da borda da coluna (border), lembrando que as cores, assim como os comandos devem ser expressas em inglês.

```
barplot(table(nome_variável), col=c("blue","red"), main="Título", xlab="Variável  
do eixo x", ylab = "Informação que consta no eixo y",border="red")
```

Ex.1) Construir um gráfico de colunas para a variável **Pessoas_familia**.

```
barplot(table(`Pessoas_familia`), col=c("blue"), main = "Frequência de pessoas por família")
```

Ex.2) Construir uma tabela de dupla entrada para as variáveis **Sexo** e **Divulgação**.

```
barplot(table(Sexo,Divulgacao), col=c("blue"),  
main = "Frequência de pessoas por Sexo e Divulgacao")
```

Ex.3) Na sequência utiliza o sinal de atribuição <- para atribuir o nome Resultado para esta tabela (tabela de dupla entrada obtida em Ex.2).

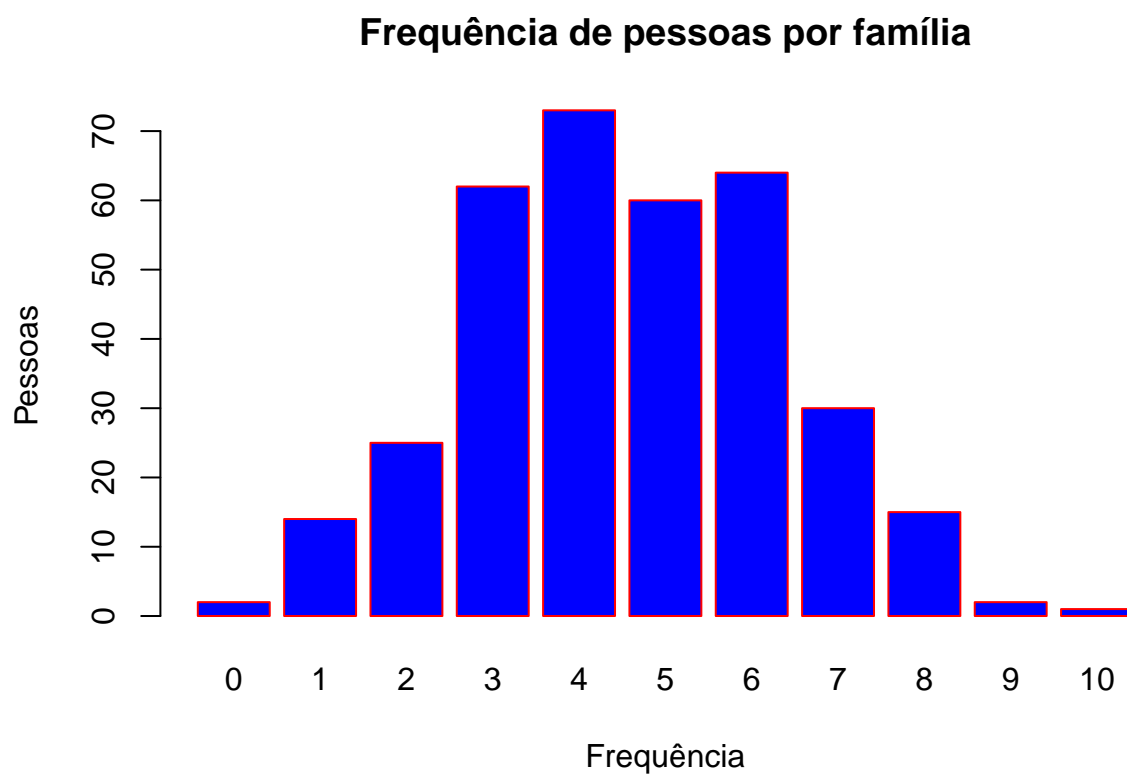


Figura 2.2: Gráfico de colunas com a variável ‘Pessoas família’

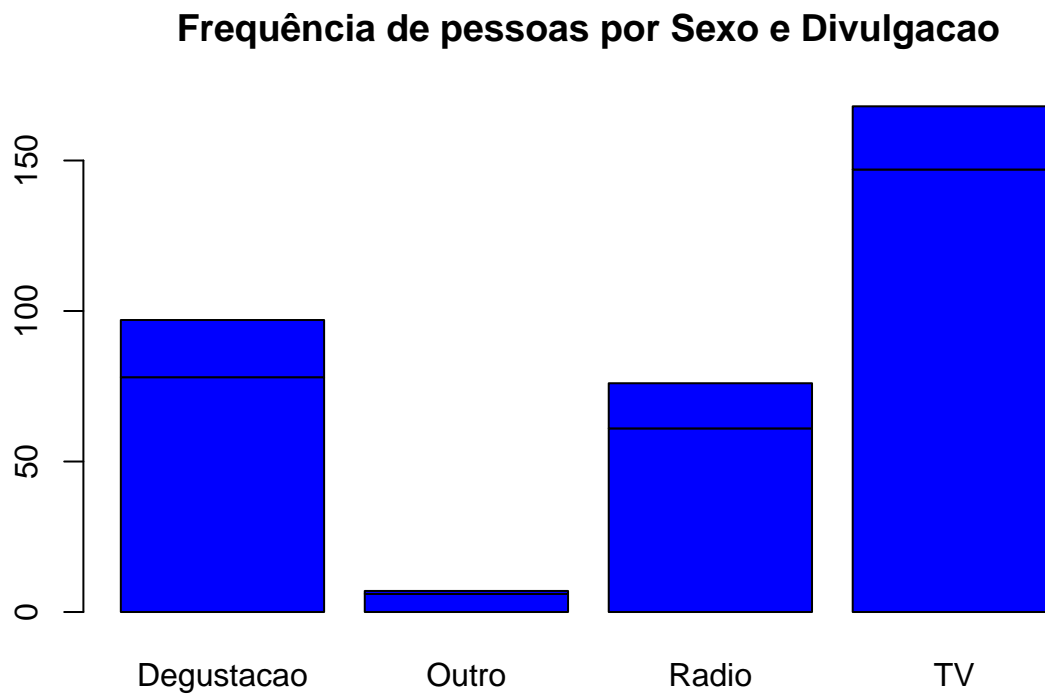


Figura 2.3: Gráfico de colunas com as variáveis Sexo e Divulgacao

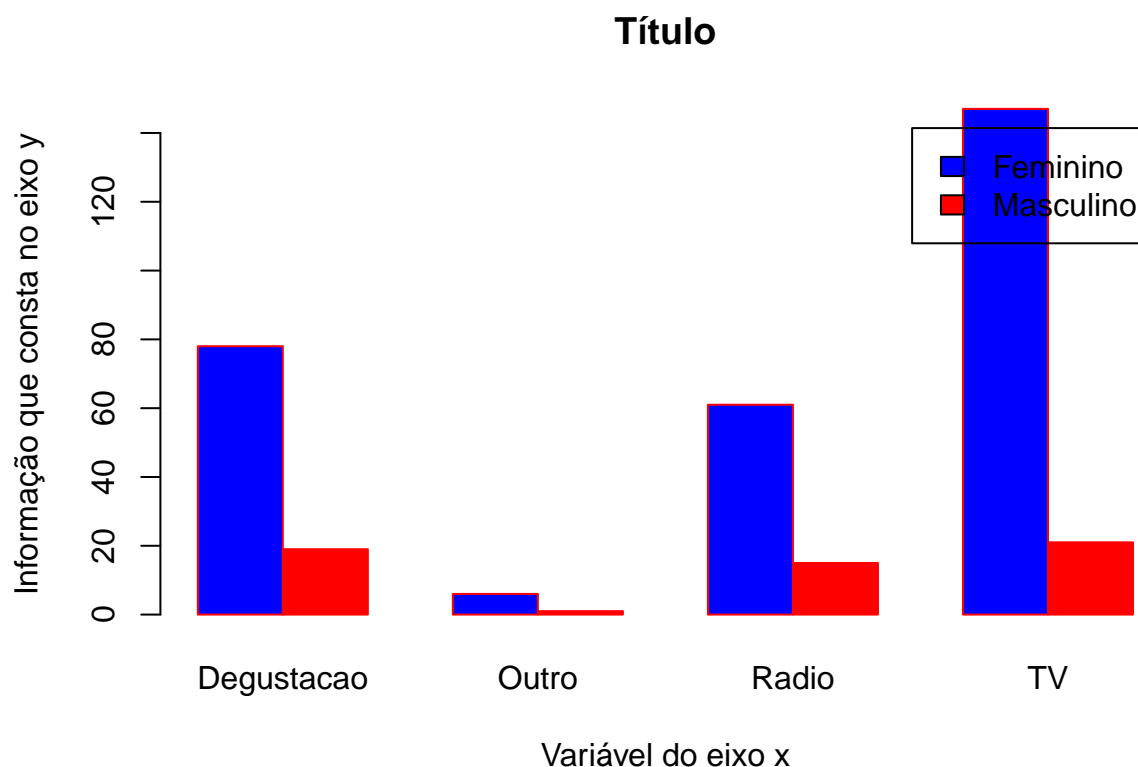


Figura 2.4: Gráfico de colunas com as variáveis Sexo e Divulgacao (2)

```
Resultado<-table(Sexo,Divulgacao)
```

Ex.4) Execute o seguinte comando:

```
barplot(Resultado,col=c("blue","red"),main="Título",xlab="Variável do eixo x",
        ylab="Informação que consta no eixo y", border='red',
        beside=T,legend=rownames(Resultado))
```

Observe que o uso do argumento `beside=T` evita que as barras fiquem empilhadas e o argumento `legend` insere a legenda conforme as cores das colunas.

Ex.5) Repita o exercício a partir do Ex.3, invertendo a ordem entre as variáveis qualitativas.

2.3.2 Setograma ou gráfico de pizza

Os gráficos em setores são utilizados para ilustrar dados qualitativos de modo mais compreensível. Quando a variável é ordinal, gráficos de colunas são mais indicados pelo fato de permitirem manter a ordem das categorias. Isto também vale para os casos em que se tem muitas categorias ou quando se pretende dar mais destaque às categorias mais frequentes.

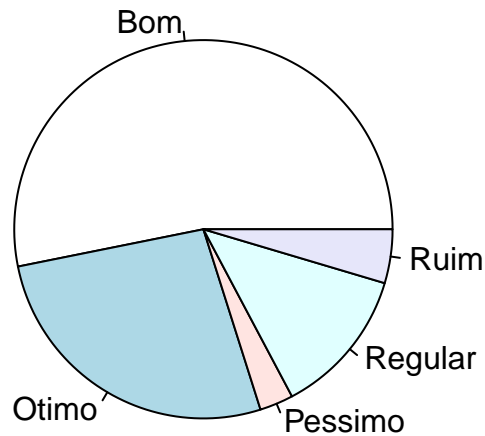


Figura 2.5: Gráfico de pizza com a variável Sabor

(Barbetta 2010).

```
pie(table(nome_variável),main="nome")
```

Ex. Construa um gráfico na forma de Setograma para a variável **Sabor**.

```
pie(table(Sabor))
```

2.3.3 Histograma

No histograma, utilizado em geral quando temos variáveis quantitativas contínuas, a altura dos retângulos representa a frequência de ocorrência de valores no intervalo (deve iniciar sempre em zero), devem ter sempre a mesma largura podendo ser justapostos. O eixo horizontal (dos valores da variável) pode iniciar próximo ao menor valor da variável (Barbetta 2010). Para confecção do histograma devemos usar:

```
hist(nome_variável)
```

Ex. Construa um histograma com a variável **Renda_h**.

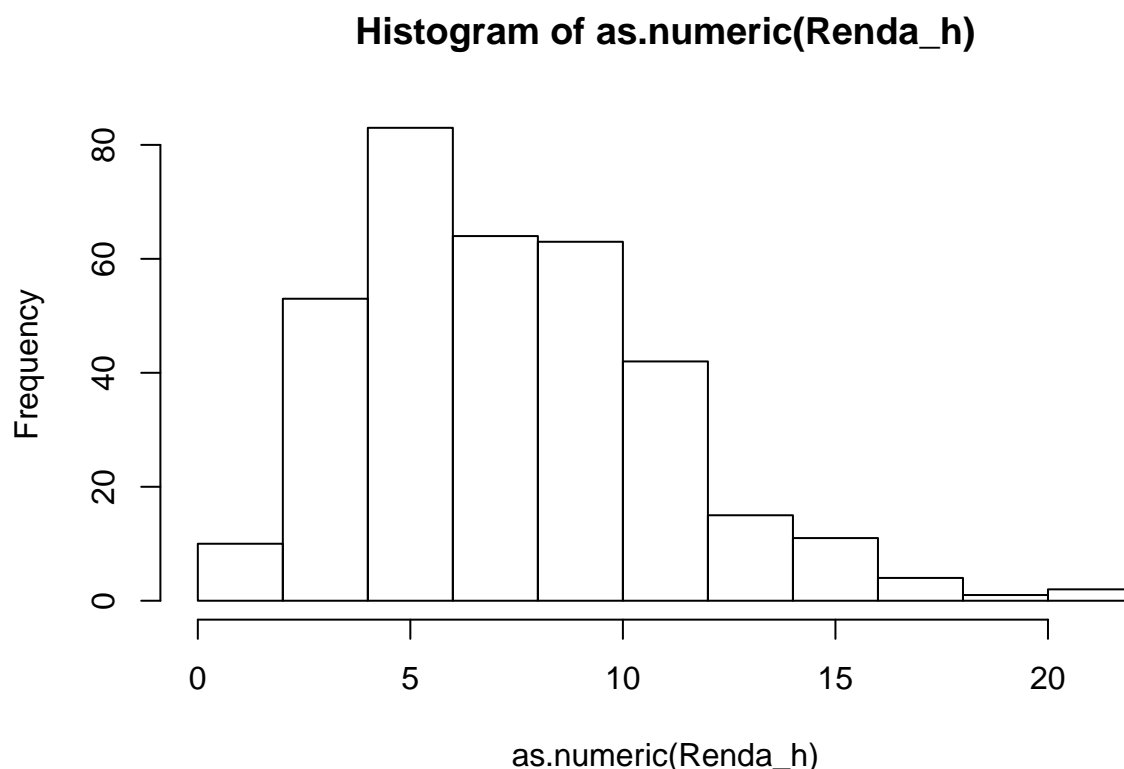


Figura 2.6: Histograma com a variável ‘Renda h’

```
hist(as.numeric(`Renda_h`))
```

Obs. I: Neste caso também é possível personalizar o gráfico, incluindo o título do eixo x (xlab), o título do eixo y (ylab), o título do gráfico (main), a cor da coluna (col) e cor da borda da coluna (border), lembrando que as cores, assim como os comandos devem ser expressas em inglês.

Obs. II: Para definir o número de intervalos no Histograma, usamos:

```
hist(nome_variável, breaks = 5)
```

```
hist(as.numeric(`Renda_h`), breaks=5)
```

Use o argumento `main=NULL` para remover o título.

2.3.4 Boxplot ou diagrama em caixas

Os diagramas em caixa são convenientes para revelar tendências centrais, dispersão, distribuição dos dados e a presença de outliers (valores extremos). Como as medianas revelam uma tendência central, ao passo que os quartis indicam a dispersão dos dados, os diagramas

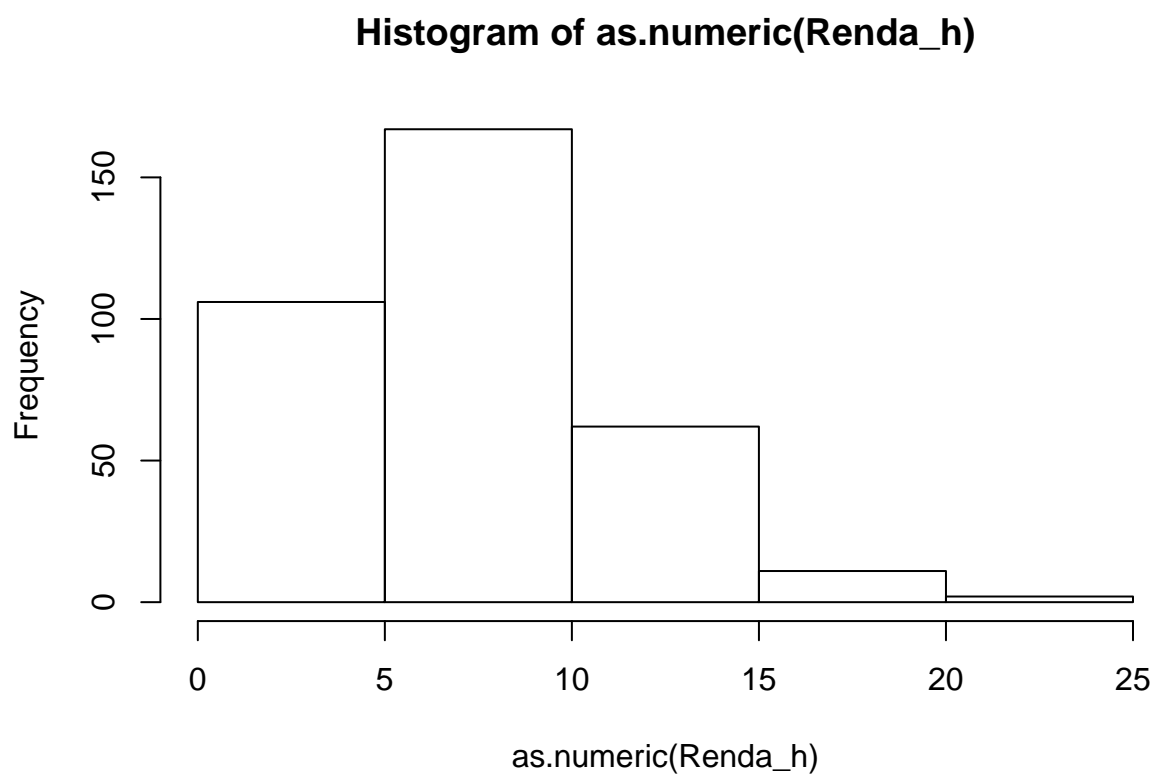


Figura 2.7: Histograma com a variável Renda h com breaks=5

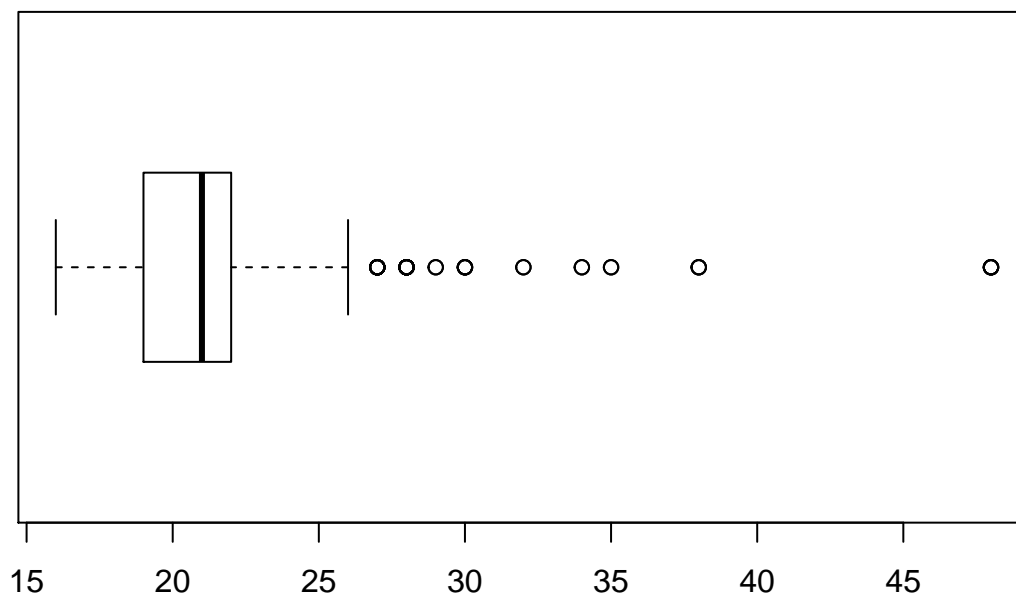


Figura 2.8: Boxplot com a variável Idade

em caixa têm a vantagem de não serem tão sensíveis a valores extremos como outras medidas baseadas na média e no desvio-padrão. Por outro lado, os diagramas em caixa (boxplots) não dão informação tão detalhada quanto os histogramas ou os gráficos ramo-e-folhas, podendo não ser, assim, a melhor escolha quando lidamos com um único conjunto de dados. Os diagramas em caixa são, entretanto, mais convenientes na comparação de dois ou mais conjuntos de dados (Triola 2011).

No diagrama de caixas, torna-se fácil identificar **outliers** (ou valores extremos), que são valores extremamente raros, no sentido de que estão muito afastados da maioria dos dados. Ao explorarmos um conjunto de dados, não podem deixar de considerar os outliers, porque eles podem revelar informações importantes (Triola 2011).

Para obter o boxplot para um conjunto de dados:

```
boxplot(variávelA, variávelB, names=c("A", "B"))
```

Ex.1) Construir um boxplot da variável **Idade**.

```
boxplot(Idade, horizontal = T)
```

Ex.2) Construir um boxplot das variáveis **Peso__(Kg)** e **Altura__(m)**.

2.3.5 Gráfico ramo-e-folhas

Em um gráfico ramo-e-folhas, classificamos os dados segundo um padrão que revela a distribuição subjacente. O padrão consiste em separar um número em duas partes em geral: o ramo consiste nos algarismos mais à esquerda e as folhas consistem nos algarismos mais à direita.

No gráfico Ramo-e-folhas, podemos ver a distribuição desses dados, que é uma vantagem do gráfico ramo-e-folhas e ainda conservar toda a informação da lista original; se necessário, podemos recompor a relação original de valores. Note que as linhas de algarismos em um gráfico ramo-e-folhas são análogas, em natureza, às barras de um histograma (Triola 2011).

`stem(nome_variável)` - comando que permite obter um gráfico Ramo e Folhas.

Ou

`stem(nome_variável,scale=1)`

O “scale=1”, que é o padrão, separa os ramos das folhas a partir das casas decimais.

Caso padrão:

- A ideia do ramo e folhas é separar um número (como 16,0) em duas partes. Assim, a primeira parte inteira (16) chamada de ramo e a segunda, a parte decimal (0) chamada de folha. O padrão do R é separar os números em duas partes (inteira e decimal) e agrupar os números em classes de tamanho 2. Por exemplo, o ramo 16 leva em conta os números 16 e 17.

Obs.: Esse padrão vai se alterando, à medida que o conjunto de dados apresente diferentes casas decimais.

Assim, outras opções podem ser avaliadas:

a) `stem(nome_variável,scale=0.5)`

b) `stem(nome_variável,scale=2)`

Obs.: Quando uma folha relacionada com certo ramo tem uma quantidade tão grande de valores que ele sintetiza essa quantidade usando a denominação +n, e invade a linha seguinte. Isso pode ser melhorado usando **width**.

c) `stem(nome_variável,scale=0.5,width=120)`

Ex. Construa um gráfico Ramo e Folhas com a variável **Idade**.

```
stem(Idade,scale=2)
```

```
The decimal point is at the |
```

```
16 | 000
```

```
17 | 000000000
```

[illegible]

2.3.6 Gráficos de dispersão

Às vezes temos dados emparelhados de forma que associa cada valor de um conjunto a um determinado valor de um segundo conjunto. Um diagrama de dispersão é um gráfico dos dados emparelhados (x, y), com um eixo x horizontal e um eixo y vertical. O diagrama de dispersão, apresenta no eixo horizontal os valores da primeira variável e um eixo vertical para os valores da segunda variável. O padrão dos pontos assim marcados costuma ajudar a determinar se existe algum relacionamento entre as duas variáveis A e B.

```
plot(variável_independente, Variável_dependente)
```

Ou

Tabela 2.1: Evolução dos resultados da fiscalização do trabalho na área rural Brasil 1998-2010

Ano	Empresas.Fiscalizadas
1998	7.042
1999	6.561
2000	8.585
2001	9.641
2002	8.873
2003	9.367
2004	13.856
2005	12.192
2006	13.326
2007	13.390
2008	10.839
2009	13.379
2010	11.978

```
plot(variável_dependente~variável_independente)
```

2.3.7 Gráfico de linhas

Apresenta a evolução de um dado, geralmente ao longo do tempo. Eixos na vertical e na horizontal indicam as informações a que se refere e a linha traçada entre eles, ascendente, descendente constante ou com vários altos e baixos mostra o percurso de um fenômeno específico.

Ex. Considere os dados que descrevem os valores do número de empresas fiscalizadas na fiscalização do trabalho na área rural Brasil 1998-2010.

Para construir um gráfico de linhas, utilizamos o seguinte comando:

```
plot(x,y,type= "Tipo de símbolo")
```

Neste gráfico, podemos utilizar comandos já utilizados anteriormente, para inserir título, nomes dos eixos, etc. Para escolher o formato das linhas, com o uso do argumento “type”, seguem algumas opções:

- "p" para pontos,
- "l" para linhas,
- "b" para pontos e linhas,
- "c" para linhas descontínuas nos pontos,
- "o" para pontos sobre as linhas,
- "n" para nenhum gráfico, apenas a janela.

Para o caso de representação no mesmo gráfico, de duas ou mais variáveis, o processo deverá ser realizado por etapas:

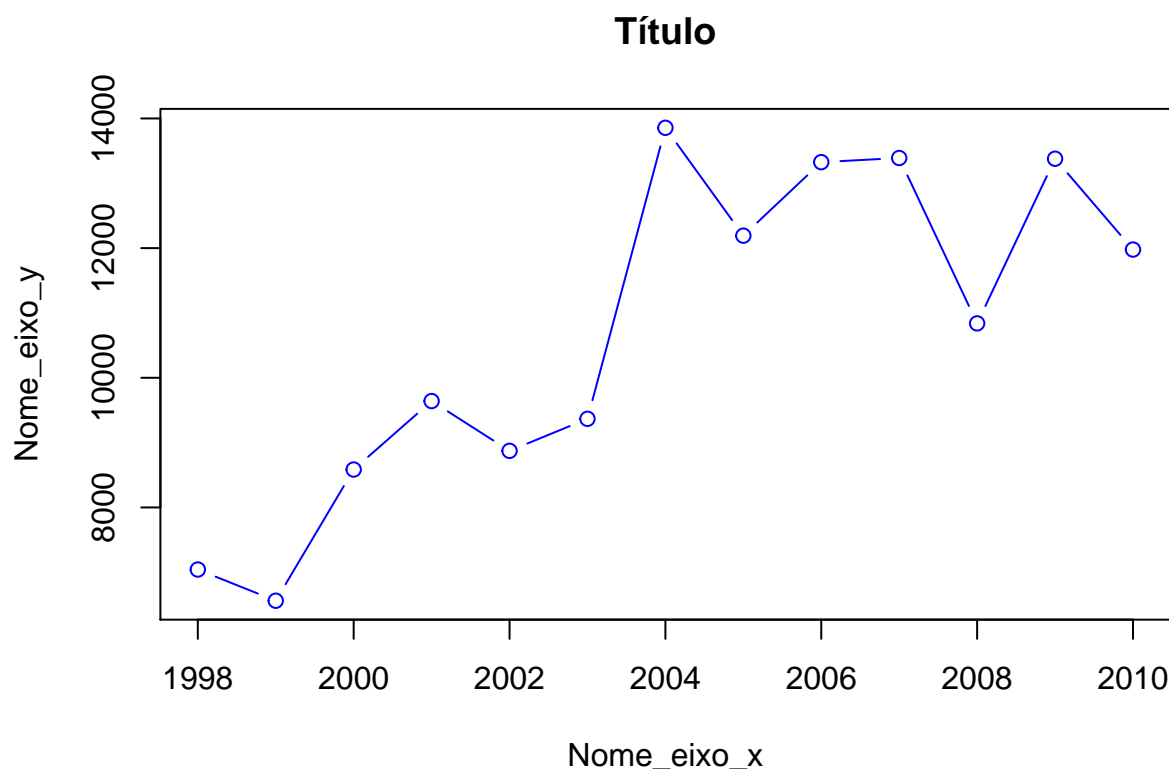


Figura 2.9: Gráfico de linha sobre a fiscalização do trabalho na área rural Brasil 1998-2010

```
plot(x,y1,type="b",main="Título", xlab="Nome_eixo_x",ylab="Nome_eixo_y",
col="cor das linhas",ylim=c(yi,ys))

empfisc=data.frame(ano=c(1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,
2008,2009,2010), qtd=c(7042,6561,8585,9641,8873,9367,
13856,12192,13326,13390,10839,13379,11978))

plot(empfisc$ano,empfisc$qtd,type="b",main="Título",
xlab="Nome_eixo_x",ylab="Nome_eixo_y",
col="blue",xlim=c(1998,2010))
```

onde, no argumento `ylim`, devemos indicar o intervalo de variação dos valores de `y`, ou seja todo o intervalo que será necessário para representar todas as variáveis.

Na sequência adicionamos as instruções para as demais variáveis:

```
lines(x, y2,col="cor_desejada", type="b")
```

Com o argumento `"legend"` instruímos a formatação da legenda:

```
legend(xp,yp,c("representação_variável_1 na legenda", "representação_variável_2
na legenda"),col =c("Cor1","cor2"),pch=Valor entre 0 e 25)‘
```

Obs.: `pch`= número (entre 0 e 25). No Help do R (buscando com `pch`), você encontra a lista completa de símbolos que podem ser utilizados na representação da legenda. Neste caso, pode ser importante também alterar o tamanho da fonte da legenda, com o uso do argumento `"cex"`.

Exemplo: Segue exemplo de um gráfico de linhas para as temperaturas registradas durante o dia 11/04/2018, pela Estação Meteorológica de São Luiz Gonzaga, RS, conforme dados obtidos no site do Inmet.

```
library(readr)
inmet <- read_delim("https://goo.gl/se71v2",
  ";", escape_double = FALSE,
  col_types = cols(data = col_date(format = "%m/%d/%Y")),
  trim_ws = TRUE)
head(inmet)
```

A tibble: 6 x 6

	codigo_estacao	data	hora	temp_inst	temp_max	temp_min
	<chr>	<date>	<int>	<dbl>	<dbl>	<dbl>
1	A852	2018-04-11	0	26.2	27.1	26.2
2	A852	2018-04-11	1	26	26.2	26
3	A852	2018-04-11	2	25.5	26.1	25.5
4	A852	2018-04-11	3	25.1	25.5	25
5	A852	2018-04-11	4	24.6	25.2	24.5
6	A852	2018-04-11	5	24.3	24.7	24.2

Segue a sequência de comandos, para obtenção do gráfico de linhas:

```
plot(inmet$hora,inmet$temp_inst,type = "b",
  main = "Temperaturas registradas na estação meteorológica
  de São Luis Gonzaga, 11 de abril de 2018",
  xlab = "hora",ylab = "temperaturas",col="blue",
  ylim = c(20,40))

lines(inmet$hora,inmet$temp_max,col="red",type = "b")

lines(inmet$hora,inmet$temp_min,col="green",type = "b")

legend(0,40,c("temp_inst","temp_max","temp_min"),
  col =c("blue","red","green"),pch=4.1,cex = 0.75)
```

2.4 Estatísticas Descritivas

Para determinar o valor máximo de um conjunto de dados, utilizamos:

```
max(nome_da_variável)
```

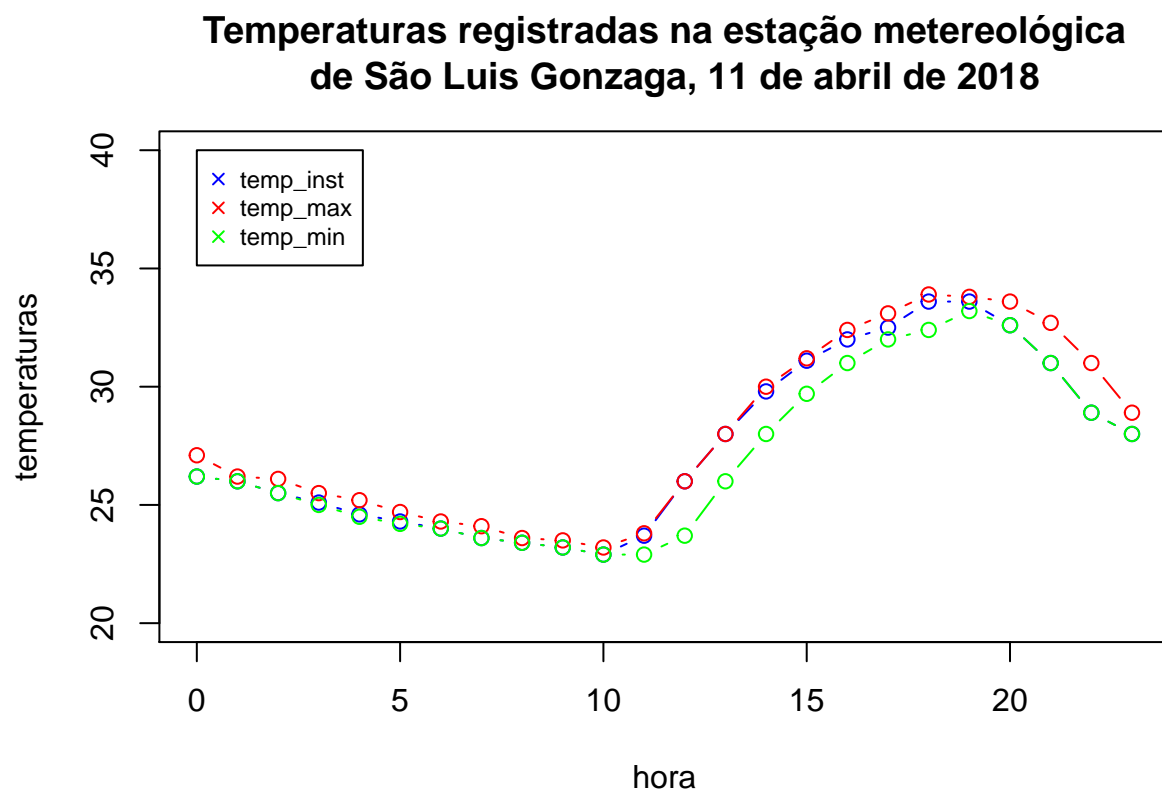



Figura 2.10: Gráfico de linha sobre as temperaturas registradas em São Luiz Gonzaga - RS

Use a variável **Renda_h**

```
#Transforme a variável Renda_h em variável numérica
pesquisa_dados$Renda_h=as.numeric(pesquisa_dados$Renda_h)
#É preciso repetir o comando attach()
attach(pesquisa_dados)
max(Renda_h)
```

```
[1] 21.83
```

De forma análoga, para determinar o valor mínimo de um conjunto de dados, utilizamos:

```
min(nome_da_variável)
```

Use a variável **Renda_h**

```
min(Renda_h)
```

Obs.: Para determinar a amplitude total de um conjunto de dados, utilizamos:

```
max(nome_da_variável)-min(nome_da_variável)
```

Use a variável **Renda_h**

```
max(Renda_h)-min(Renda_h)
```

```
[1] 20.81
```

Para obter as medidas da estatística descritiva, no caso medidas de tendência central (mínimo, quartil 1, mediana, média, quartil 3, máximo):

```
summary(nome_da_variável)
```

Ex. Use a variável **Renda_h**

```
summary(Renda_h)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.02	4.64	6.79	7.31	9.51	21.83

A moda é o valor que tem o maior número de ocorrências em um conjunto de dados.

O R não tem um padrão de função embutida para calcular a moda. Uma sugestão é a criação de uma função pelo usuário, que pode ser obtida, por exemplo por:

```
subset(table(variável), table(variável)==max(table(variável)))
```

Ex. Use a variável **Praticidade**

```
subset(table(Praticidade),
       table(Praticidade)==max(table(Praticidade)))
```

Ruim

Ex. Use a variável quantitativa **Pessoas_familia**

```
table(Pessoas_familia)
```

```
Pessoas_familia
 0  1  2  3  4  5  6  7  8  9 10
2 14 25 62 73 60 64 30 15  2  1
```

Obs.: O primeiro valor encontrado, refere-se ao valor da moda ao passo que o segundo valor representa quantas vezes esse valor foi verificado.

Comando que permite determinar o percentil, no caso o percentil 10:

```
quantile(nome_variável,0.1)
```

Obs.: Experimente usar o comando:

```
quantile(nome_variável)
```

Obs.: Para a obtenção de quartis e decis, basta realizar a conversão para o respectivo percentil e assim calcular normalmente.

```
quantile(Renda_h)
```

```
   0%   25%   50%   75%  100%
1.020 4.638 6.785 9.512 21.830
```

```
quantile(Renda_h,0.1)
```

```
   10%
3.244
```

Para obter as medidas de variabilidade, no caso, variância e desvio-padrão, respectivamente:

```
var(nome_variável)
```

```
sd(nome_variável)
```

Ex. Calcule as medidas de variabilidade com a variável **Pessoas_familia**

```
var(Pessoas_familia)
```

```
[1] 3.245
```

```
sd(Pessoas_familia)
```

```
[1] 1.801
```

A função `subset()`:

Com esta função podemos fazer cálculos utilizando filtros, simultaneamente. A aplicação de filtros é extremamente útil quando queremos explorar os dados de forma rápida e eficiente.

Exemplos:

Ex. 1) Altura das pessoas do sexo masculino: com a função abaixo o R gera um subconjunto com as alturas de todas as pessoas do sexo masculino.

```
subset(`Altura_(m)`, Sexo=="Masculino")
```

```
[1] 1.90 1.76 1.83 1.81 1.67 1.55 1.60 1.84 1.80 1.60 1.75 1.73 1.68 1.81 1.90
[16] 1.80 1.56 1.65 1.60 1.61 1.59 1.75 1.59 1.89 1.62 1.60 1.50 1.65 1.79 1.65
[31] 1.79 1.67 1.59 1.71 1.60 1.72 1.73 1.65 1.65 1.50 1.57 1.86 1.85 1.80 1.77
[46] 1.81 1.73 1.80 1.66 1.71 1.60 1.72 1.81 1.55 1.60 1.80
```

Ex. 2) Média das alturas das pessoas do sexo masculino: inserindo o comando `mean()` ao subconjunto anterior, teremos como resultado a média das alturas das pessoas do sexo masculino.

```
mean(subset(`Altura_(m)`, Sexo=="Masculino"))
```

```
[1] 1.702
```

Ex. 3) Média das alturas das pessoas do sexo masculino com mais de 26 anos:

```
mean(subset(`Altura_(m)`, Sexo=="Masculino"& Idade>25))
```

```
[1] 1.654
```

Ex. 4) Contagem de pessoas do sexo feminino que tenham menos de 60 kg:

```
length(subset(Sexo,Sexo=="Feminino" & `Peso_(Kg)`<60))
```

```
[1] 94
```

Ex. 5) Montando uma tabela para exibir o gênero de pessoas que classificaram o Sabor como “Pessimo”:

```
table(subset(Sexo, Sabor=="Pessimo"))
```

Feminino	Masculino
7	3

Este capítulo não teve a pretensão de esgotar o estudo de todos os comandos a serem aplicados na estatística descritiva (veja `help` do R), nem tampouco os conceitos estatísticos necessários à compreensão. Para mais detalhes sobre os conceitos de estatística descritiva, você pode consultar outras referências ou até mesmo as já citadas neste capítulo.

Capítulo 3

Estatística Inferencial

A inferência estatística, ou estatística inferencial, tem por objetivo concluir e tomar decisões, com base em amostras (Figura 3.1). Usam-se dados extraídos de uma amostra para produzir inferência sobre a população (Lopes et al. 2008).

Em Estatística, o termo **população** é definido como conjunto de indivíduos, ou itens, com pelo menos uma característica em comum, podendo ser finita ou infinita (Lopes et al. 2008). Por exemplo, água de um rio, sangue de uma pessoa, lote de peças produzidas por uma indústria, eleitores de um município.

A **amostra** é um subconjunto, necessariamente finito, de uma população e é selecionada de forma que todos os elementos da população tenham a mesma chance de serem escolhidos.

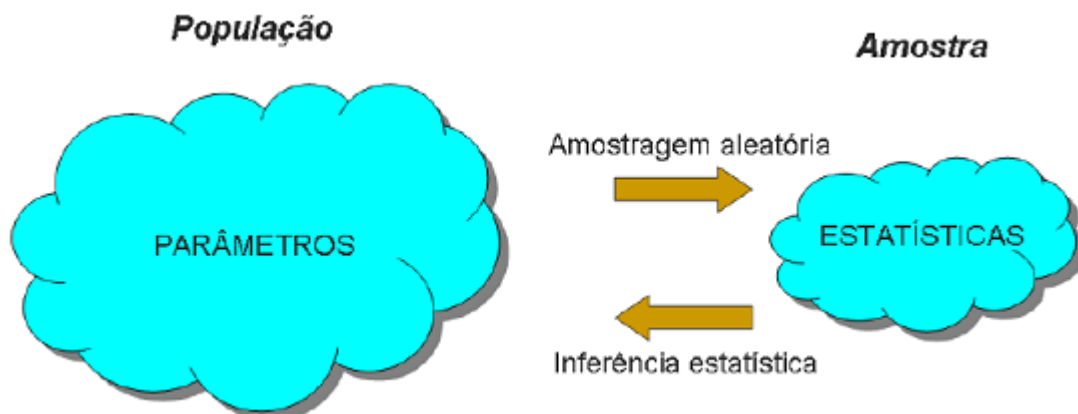


Figura 3.1: Inferência Estatística

3.1 Intervalo de Confiança

Entre as diferentes técnicas de Inferência Estatística, temos a Estimação de Parâmetros, que consiste na determinação de um **Intervalo de Confiança (IC)** para uma média ou proporção populacional, ao um nível $(1 - \alpha)\%$ de confiança.

O nível de confiança $(1 - \alpha)\%$ normalmente varia de 90% a 99%.

3.1.1 Intervalo de confiança para uma média populacional

Um **intervalo de confiança (IC)** é o **intervalo** estimado onde a média de um parâmetro tem uma dada probabilidade de ocorrer. Comumente define-se como o **intervalo** onde há $(1 - \alpha)\%$ de probabilidade da média verdadeira da população inteira ocorrer.

IC (limite inferior $\leq \mu \leq$ limite superior) = $(1 - \alpha)\%$

No software RStudio, o Intervalo de Confiança pode ser obtido usando o teste t.

Exemplo 1: Os dados amostrais a seguir representam o número de horas de estudos semanais para a disciplina de Estatística Básica, de uma amostra de 10 alunos:

19 18 20 16 18 19 19 17 22 21

Qual é o intervalo de confiança para a média populacional de onde essa amostra foi retirada?

```
horasestudo=c(19,18,20,16,18,19,19,17,22,21)
t.test(horasestudo)
```

One Sample t-test

```
data: horasestudo
t = 33, df = 9, p-value = 1e-10
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 17.62 20.18
sample estimates:
mean of x
 18.9
```

IC $(17,6 \leq \mu \leq 20,2) = 95\%$

Com 95% de confiança, a média populacional das horas semanais de estudo para a disciplina de Estatística Básica está entre 17,6 e 20,2 horas. Ou seja, qualquer aluno (de onde essa amostra foi retirada) estuda em média, de 17,6 a 20,2 horas por semana.

Se não informarmos o nível de confiança, o software R considera 95%. No entanto, para mudar o nível de confiança para 90%, acrescentamos a informação `conf.level = 0.90` após o nome da variável:

```
t.test(horasestudo, conf.level = 0.90)
```

One Sample t-test

```
data: horasestudo
t = 33, df = 9, p-value = 1e-10
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 17.86 19.94
sample estimates:
mean of x
 18.9
```

IC ($17,9 \leq \mu \leq 19,9$) = 90%

Com 90% de confiança, a média populacional das horas semanais de estudo para a disciplina de Estatística Básica está entre 17,9 e 19,9 horas. Ou seja, qualquer aluno (de onde essa amostra foi retirada) estuda em média, de 17,9 a 19,9 horas por semana.

Para mudar o nível de confiança para 99%:

```
t.test(horasestudo, conf.level = 0.99)
```

One Sample t-test

```
data: horasestudo
t = 33, df = 9, p-value = 1e-10
alternative hypothesis: true mean is not equal to 0
99 percent confidence interval:
 17.06 20.74
sample estimates:
mean of x
 18.9
```

IC ($17,1 \leq \mu \leq 20,7$) = 99%

Com 99% de confiança, a média populacional das horas semanais de estudo para a disciplina de Estatística Básica está entre 17,1 e 20,7 horas. Ou seja, qualquer aluno (de onde essa amostra foi retirada) estuda em média, de 17,1 a 20,7 horas por semana.

3.1.2 Para verificar normalidade dos dados

Algumas técnicas de inferência estatística têm como requisitos a normalidade dos dados. Para verificar se os dados seguem uma distribuição normal, podemos, inicialmente usar

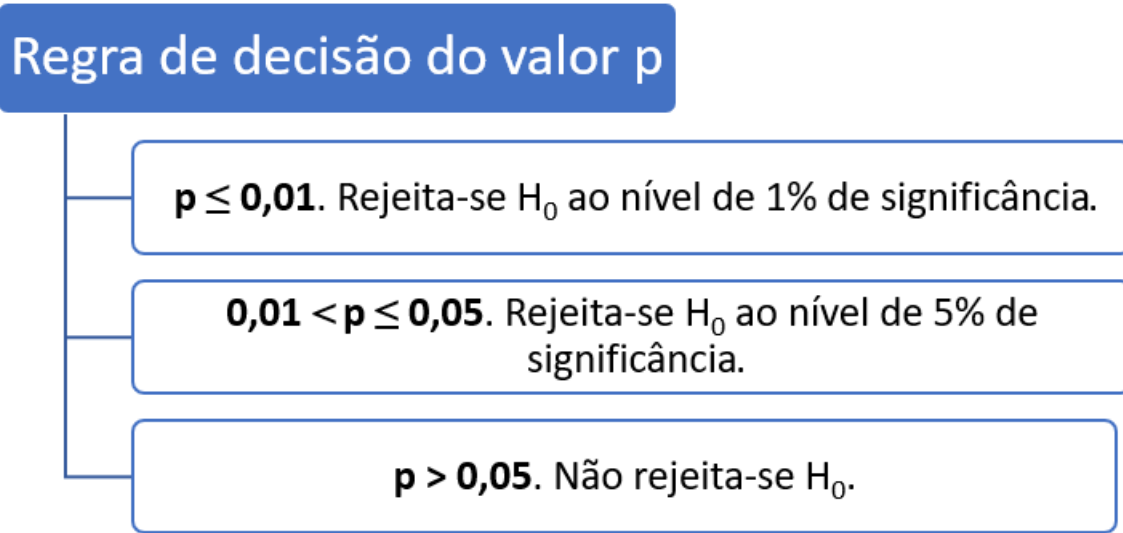


Figura 3.2: Teste de hipóteses

o histograma e depois confirmar com um teste estatístico para testar normalidade como Shapiro-Wilk ou Kolmogorov-Smirnov.

Hipóteses do teste:

- **H0**: os dados seguem uma distribuição normal
- **H1**: os dados não seguem uma distribuição normal

O **valor p** reflete a plausibilidade de se obter tais resultados no caso de H0 ser de fato verdadeira.

```
shapiro.test(horasestudo)
```

```
Shapiro-Wilk normality test
```

```
data: horasestudo
W = 0.98, p-value = 0.9
```

Como $p > 0,05$, não rejeita-se H0 e conclui-se que os dados seguem uma distribuição normal.

3.1.3 Intervalo de confiança para uma proporção populacional

IC (limite inferior $\leq \pi \leq$ limite superior) = $(1 - \alpha)\%$

Exemplo 2: (adaptado de <https://www.passeidireto.com/arquivo/3802950/capitulo7---intervalos-de-conf>)
Entre 500 pessoas entrevistadas a respeito de suas preferências eleitorais, 260 mostraram-se

favoráveis ao candidato B. Qual é a proporção amostral dos favoráveis ao candidato B? E a proporção populacional dos favoráveis?

Sintaxe no software RStudio:

```
prop.test(x,n,conf.level=nível de confiança)
```

Em que:

x = número de sucessos

n= tamanho da amostra

nível de confiança = 0,90 a 0,99

```
prop.test(260,500)
```

1-sample proportions test with continuity correction

```
data: 260 out of 500, null probability 0.5
X-squared = 0.72, df = 1, p-value = 0.4
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4752 0.5645
sample estimates:
      p
0.52
```

A proporção amostral dos eleitores favoráveis ao candidato B é de 0,52.

IC ($0,48 \leq \pi \leq 0,56$) = 95%

Com 95% de confiança, a proporção populacional dos eleitores favoráveis ao candidato B está entre 0,48 e 0,56.

Para mudar o nível de confiança para 90%:

```
prop.test(260,500,conf.level = 0.90)
```

1-sample proportions test with continuity correction

```
data: 260 out of 500, null probability 0.5
X-squared = 0.72, df = 1, p-value = 0.4
alternative hypothesis: true p is not equal to 0.5
90 percent confidence interval:
 0.4822 0.5575
sample estimates:
      p
0.52
```

IC $(0,48 \leq \pi \leq 0,56) = 90\%$

Com 90% de confiança, a proporção populacional dos eleitores favoráveis ao candidato B está entre 0,48 e 0,56.

Para mudar o nível de confiança para 99%:

```
prop.test(260,500,conf.level = 0.99)
```

```
1-sample proportions test with continuity correction
```

```
data: 260 out of 500, null probability 0.5
X-squared = 0.72, df = 1, p-value = 0.4
alternative hypothesis: true p is not equal to 0.5
99 percent confidence interval:
 0.4616 0.5779
sample estimates:
      p
0.52
```

IC $(0,46 \leq \pi \leq 0,58) = 99\%$

Com 99% de confiança, a proporção populacional dos eleitores favoráveis ao candidato B está entre 0,46 e 0,58.

3.2 Teste de hipóteses

O teste de hipóteses é uma outra forma de fazer inferência estatística. Formula-se uma hipótese (H_0) para um parâmetro populacional e, partir de uma amostra dessa população, aceita-se ou rejeita-se esta hipótese.

H₀: hipótese nula (sempre tem a condição de igualdade)

H₁: hipótese alternativa (tem o sinal de \neq , $>$ ou $<$)

3.2.1 Teste de hipóteses para uma média populacional

H₀: $\mu = \dots$

H₁: $\mu \neq \dots$

H₀: $\mu = \dots$

H₁: $\mu > \dots$

H₀: $\mu = \dots$

H1: $\mu < \dots$

No software RStudio, usa-se o `t.test` para a realização do teste de hipóteses para uma média populacional, levando-se em conta o valor de p-value para aceitar ou rejeitar H0.

De acordo com as hipóteses, temos variações do `t.test`, conforme segue:

sintaxe: `t.test(amostra, opções)`

- **amostra:** Vetor contendo a amostra da qual se quer testar a média populacional.
- **opções:** `alternative`: string indicando a hipótese alternativa desejada. Valores possíveis: `"two-sided"`, `"less"` ou `"greater"`.
- μ : valor indicando o verdadeiro valor da média populacional.

Exemplo 3: (adaptado de www.leg.ufpr.br/~paulojus/CE002/pratica/praticase8.xml)
A precipitação pluviométrica mensal numa certa região nos últimos 9 meses foi a seguinte:

30,5 34,1 27,9 35,0 26,9 30,2 28,3 31,7 25,8

Construa um teste de hipóteses para saber se a média da precipitação pluviométrica mensal é igual a 30,0 mm.

H0: $\mu = 30$ mm

H1: $\mu \neq 30$ mm

```
chuva=c(30.5,34.1,27.9,35,26.9,30.2,28.3,31.7,25.8)
chuva
```

```
[1] 30.5 34.1 27.9 35.0 26.9 30.2 28.3 31.7 25.8
```

```
t.test(chuva,alt="two.sided",mu=30)
```

One Sample t-test

```
data: chuva
t = 0.042, df = 8, p-value = 1
alternative hypothesis: true mean is not equal to 30
95 percent confidence interval:
 27.62 32.47
sample estimates:
mean of x
 30.04
```

Conclusão: Aceita-se H0 e conclui-se que a precipitação pluviométrica é igual a 30mm.

Exemplo 4: (adaptado de <https://www.passeidireto.com/arquivo/5533375/lista-eststistica-pronta-p-3-pr>)
3) Um empresário desconfia que o tempo médio de espera para atendimento de seus clientes é superior a 20 minutos. Para testar essa hipótese ele entrevistou 20 pessoas e questionou quanto tempo demorou para ser atendido. O resultado dessa pesquisa foi o seguinte:

22 20 21 23 22 20 23 22 20 24 21 20 21 24 22 22 23 22 20 24

Teste a hipótese de que o tempo de espera é superior a 20 minutos.

H0: $\mu = 20$ minutos

H1: $\mu > 20$ minutos

```
tempo=c(22,20,21,23,22,20,23,22,20,24,21,20,21,24,22,22,23,22,20,24)
tempo
```

```
[1] 22 20 21 23 22 20 23 22 20 24 21 20 21 24 22 22 23 22 20 24
```

```
t.test(tempo,alt="greater",mu=20)
```

One Sample t-test

```
data:  tempo
t = 5.8, df = 19, p-value = 8e-06
alternative hypothesis: true mean is greater than 20
95 percent confidence interval:
 21.26      Inf
sample estimates:
mean of x
      21.8
```

Conclusão: Rejeita-se H0 com nível de significância de 1% e conclui-se que o tempo de espera é superior a 20 minutos.

Exemplo 5: (adaptado de https://docs.ufpr.br/~vayego/pdf_11_2/pratica_04_zoo.pdf) Os resíduos industriais jogados nos rios, muitas vezes, absorvem oxigênio, reduzindo assim o conteúdo do oxigênio necessário à respiração dos peixes e outras formas de vida aquática. Uma lei estadual exige um mínimo de 5 p.p.m. (Partes por milhão) de oxigênio dissolvido, a fim de que o conteúdo de oxigênio seja suficiente para manter a vida aquática. Seis amostras de água retiradas de um rio, durante a maré baixa, revelaram os índices (em partes por milhão) de oxigênio dissolvido:

4,9 5,1 4,9 5,5 5,0 4,7

Estes dados são evidência para afirmar que o conteúdo de oxigênio é menor que 5 partes por milhão?

H0: $\mu = 5$ ppm

H1: $\mu < 5$ ppm

```
amostras=c(4.9,5.1,4.9,5.5,5.0,4.7)
t.test(amostras,alt="less",mu=5)
```

One Sample t-test

```
data:  amostras
t = 0.15, df = 5, p-value = 0.6
alternative hypothesis: true mean is less than 5
95 percent confidence interval:
  -Inf 5.24
sample estimates:
mean of x
    5.017
```

Conclusão: Aceita-se H_0 e conclui-se que o conteúdo de oxigênio é igual a 5 ppm.

3.2.2 Teste de hipóteses para uma proporção populacional

$H_0: \pi = \dots$

$H_1: \pi \neq \dots$

$H_0: \pi = \dots$

$H_1: \pi > \dots$

$H_0: \pi = \dots$

$H_1: \pi < \dots$

No software RStudio, usa-se o `prop.test` para a realização do teste de hipóteses para uma proporção populacional, levando-se em conta o valor de p-value para aceitar ou rejeitar H_0 .

Sintaxe:

```
prop.test(x,n,p=...,alt="...")
```

em que:

x = número de sucessos;

n = tamanho da amostra;

p = proporção a ser testada;

alt = "two.sided", "greater" ou "less".

Exemplo 6: (adaptado de <https://docs.ufpr.br/~soniaisoldi/TP707/Aula8.pdf>) Uma máquina está regulada quanto produz 3% de peças defeituosas. Uma amostra aleatória de 80 peças selecionadas ao acaso apresentou 3 peças defeituosas. Teste a hipótese de que a máquina está regulada.

$H_0: \pi = 3\%$

$H_1: \pi \neq 3\%$

```
prop.test(3,80,p=0.03,alt="two.sided")
```

Warning in prop.test(3, 80, p = 0.03, alt = "two.sided"): Chi-squared approximation may be incorrect

1-sample proportions test with continuity correction

```
data: 3 out of 80, null probability 0.03
X-squared = 0.0043, df = 1, p-value = 0.9
alternative hypothesis: true p is not equal to 0.03
95 percent confidence interval:
 0.009735 0.113171
sample estimates:
      p
0.0375
```

Conclusão: Aceita-se H_0 e conclui-se que a máquina produz 3% de peças defeituosas, ou seja, a máquina está regulada.

Exemplo 7: (adaptado de <www.ebah.com.br/content/ABAAAAdLkAI/metodos-estatistico-und-v-lista-resolvida>) As condições de mortalidade de uma região são tais que a proporção de nascidos que sobrevivem até 60 anos é de 0,6. Testar essa hipótese se em 1.000 nascimentos amostrados aleatoriamente, verificou-se 530 sobreviventes até 60 anos.

$H_0: \pi = 0,6$

$H_1: \pi \neq 0,6$

```
prop.test(530,1000,p=0.6,alt="two.sided")
```

1-sample proportions test with continuity correction

```
data: 530 out of 1000, null probability 0.6
X-squared = 20, df = 1, p-value = 7e-06
alternative hypothesis: true p is not equal to 0.6
95 percent confidence interval:
 0.4985 0.5613
sample estimates:
      p
0.53
```

Conclusão: Rejeita-se H_0 com nível de significância de 1% e conclui-se que a proporção de nascidos que sobrevivem até os 60 anos é diferente de 0,6.

Exemplo 8: (adaptado de <https://docs.ufpr.br/~jomarc/intervaloeteste.pdf>) Uma empresa retira periodicamente amostras aleatórias de 500 peças de sua linha de produção para análise da qualidade. As peças da amostra são classificadas como defeituosas ou não, sendo que a

política da empresa exige que o processo produtivo seja revisto se houver evidência de mais de 1,5% de peças defeituosas. Na última amostra, foram encontradas nove peças defeituosas. O processo precisa ser revisto?

H0: $\pi = 1,5\%$

H1: $\pi > 1,5\%$

```
prop.test(9,500,p=0.015,alt="greater")
```

1-sample proportions test with continuity correction

```
data: 9 out of 500, null probability 0.015
X-squared = 0.14, df = 1, p-value = 0.4
alternative hypothesis: true p is greater than 0.015
95 percent confidence interval:
 0.009766 1.000000
sample estimates:
      p
0.018
```

Conclusão: Não rejeita H0 e conclui-se que a proporção de peças defeituosas é igual a 1,5%, ou seja, o processo não precisa ser revisto.

Exemplo 9: (adaptado de <https://www.passeidireto.com/arquivo/25297344/aula-19---testes-para-propor>)
Uma pesquisa conclui que 90% dos médicos recomendam aspirina a pacientes que têm filhos. Teste a afirmação contra a alternativa de que a percentagem é inferior a 90%, se numa amostra aleatória de 100 médicos, 80 recomendam aspirina.

H0: $\pi = 90\%$

H1: $\pi < 90\%$

```
prop.test(80,100,p=0.90,alt="less")
```

1-sample proportions test with continuity correction

```
data: 80 out of 100, null probability 0.9
X-squared = 10, df = 1, p-value = 8e-04
alternative hypothesis: true p is less than 0.9
95 percent confidence interval:
 0.0000 0.8618
sample estimates:
      p
0.8
```

Conclusão: Rejeita-se H0 com nível de significância de 1% e conclui-se que a proporção de médicos que recomendam aspirina é inferior a 90%.

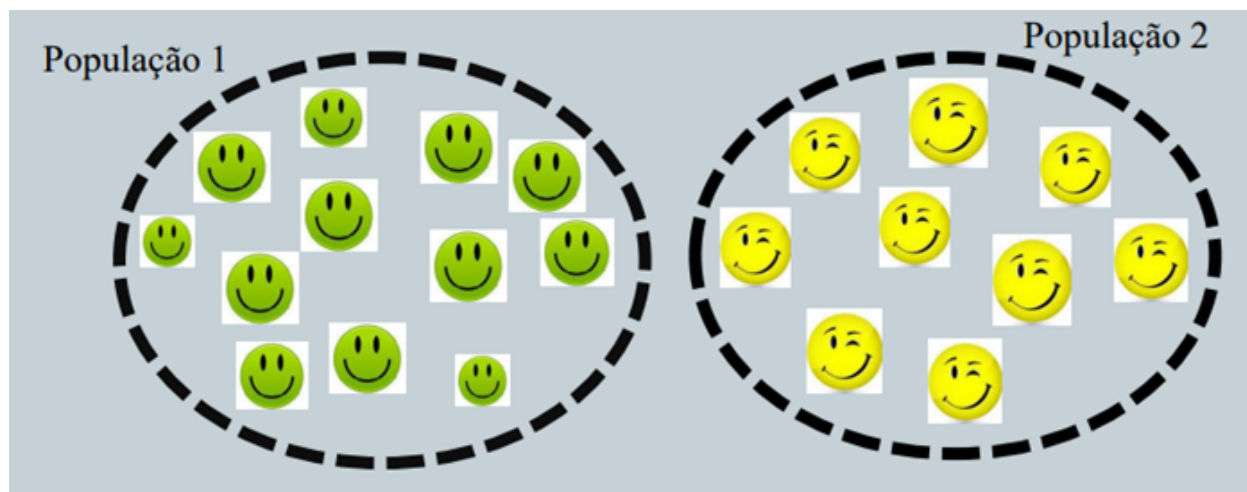


Figura 3.3: Teste de hipótese para dois grupos

Tabela 3.1: Amostras dependentes

Indivíduo	A	B	C	D	E	F
Peso antes do treinamento	99	62	74	59	70	73
Peso depois do treinamento	94	62	66	58	70	76

3.2.3 Teste de hipótese para duas médias

O teste de hipótese para duas médias aplica-se quando se deseja comparar dois grupos:

Podemos comparar duas médias de duas amostras dependentes, também chamadas de pareadas, ou médias de duas amostras independentes.

3.2.3.1 Teste de hipóteses duas amostras dependentes

Exemplo 10: Foi obtido o peso de seis indivíduos antes e após um treinamento de exercício físico. Teste a hipótese de que a média antes do treinamento é diferente da média após o treinamento.

No software RStudio, usa-se o `t.test` para a realização do teste de hipóteses para uma média populacional, levando-se em conta o valor de p-value para aceitar ou rejeitar H_0 .

Hipóteses:

H0: média antes = média depois

H1: média antes \neq média depois

```
antes=c(99,62,74,59,70,73)
depois=c(94,62,66,58,70,76)
t.test(antes,depois,paired=TRUE)
```


Tabela 3.2: Amostras dependentes - caso 2

Cobaia	1	2	3	4	5	6	7	8	9	10
Antes	635	704	662	560	603	745	698	575	633	669
Depois	640	712	681	558	610	740	707	585	635	682

Paired t-test

```
data: antes and depois
t = 1.1, df = 5, p-value = 0.3
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.334  6.000
sample estimates:
mean of the differences
      1.833
```

Conclusão: Não rejeita-se H_0 e conclui-se que a média de peso antes do treinamento é igual à média de peso depois do treinamento.

Exemplo 11: (adaptado de <www.inf.ufsc.br/~marcelo/testes2.html>) Dez cobaias foram submetidas ao tratamento de engorda com certa ração. Os pesos em gramas, antes e após o teste são dados a seguir. Podemos concluir que o uso da ração contribuiu para o aumento do peso médio dos animais?

H_0 : média antes = média depois

H_1 : média antes \neq média depois

```
cobaiaantes=c(635,704,662,560,603,745,698,575,633,669)
cobaiadepois=c(640,712,681,558,610,740,707,585,635,682)
t.test(cobaiaantes,cobaiadepois,paired=TRUE)
```

Paired t-test

```
data: cobaiaantes and cobaiadepois
t = -3, df = 9, p-value = 0.02
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -11.638  -1.562
sample estimates:
mean of the differences
      -6.6
```

Tabela 3.3: Comparação de dois tipos diferentes de tecidos

Tecido A	36	26	31	38	28	20	37
Tecido B	39	27	35	42	31	39	22

Conclusão: Rejeita-se H_0 com nível de significância de 5% e conclui-se que a média antes da engorda é diferente da média depois da engorda.

3.2.3.2 Teste de hipóteses duas amostras independentes

Primeiramente precisamos saber se existe homogeneidade de variâncias populacionais, a qual poderá ser verificada por meio de um teste de homogeneidade de variâncias utilizando os dados das duas amostras.

3.2.3.2.1 Teste para verificar homogeneidade de variâncias

Exemplo 12: (adaptado de https://www.ime.unicamp.br/~hildete/Aula_p12.pdf) Dois tipos diferentes de tecido devem ser comparados. Uma máquina de testes pode comparar duas amostras ao mesmo tempo. O peso (em miligramas) para sete experimentos foram:

Teste se um tecido é mais pesado que o outro.

H_0 : as variâncias são homogêneas

H_1 : as variâncias são heterogêneas

```
tecidoa=c(36,26,31,38,28,20,37)
tecidob=c(39,27,35,42,31,39,22)
var.test(tecidoa,tecidob)
```

F test to compare two variances

```
data:  tecidoa and tecidob
F = 0.84, num df = 6, denom df = 6, p-value = 0.8
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1441 4.8823
sample estimates:
ratio of variances
      0.8389
```

Conclusão: Não rejeita-se H_0 e conclui-se que as variâncias são homogêneas.

Agora podemos realizar o teste de comparação de duas amostras independentes.

H_0 : média tecido A = média tecido B

H1: média tecido A \neq média tecido B

```
t.test(tecidoa, tecidob, var.equal = TRUE, paired=FALSE)
```

Two Sample t-test

```
data: tecidoa and tecidob
t = -0.73, df = 12, p-value = 0.5
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -10.815    5.386
sample estimates:
mean of x mean of y
   30.86    33.57
```

Conclusão: Não rejeita-se H0 e conclui-se que a média de peso do tecido A é igual à média de peso do tecido B.

Capítulo 4

Teste de Qui-Quadrado

Capítulo 5

Modelos de Regressão

Capítulo 6

RMarkdown

Referências

Bibliografia

- Almeida, L e T Freire (2000). *Metodologia da investigação em psicologia e educação*. 2^a ed. Braga: Psiquilíbrios.
- Barbetta, Pedro Alberto (2010). *Estatística aplicada às ciências sociais*. 7^a ed. Florianópolis: Ed. UFSC.
- Lopes, L. F. D. et al. (2008). *Caderno didático: estatística Geral*. 3^a ed. Santa Maria: UFSM, p. 209.
- Triola, M. F. (2011). *Introdução à estatística*. 10^a ed. Rio de Janeiro: LTC.