

Software R: Análise estatística de dados utilizando um programa livre

Iara Denise Endruweit Battisti

Felipe Micaíl da Silva Smolski

2019-04-04

Sumário

Apresentação	3
1 Primeiros Passos com o R	5
1.1 Download e instalação do R e Rstudio	5
1.2 Painéis	5
1.3 Help	6
1.4 Instalação de pacotes	6
1.5 Abrir arquivo de dados	8
1.6 Salvar arquivo de dados	10
1.7 Diretórios de trabalho	11
1.8 Bases de dados nativas do R e de pacotes	12
1.9 Operações	12
1.10 Criação de objetos	14
1.11 Algumas funções e comandos essenciais	15
1.12 Estrutura de dados	20
1.13 Pré tratamento de banco de dados	24
1.14 Manipulação de banco de dados	28
1.15 Funções Matemáticas	44
1.16 Conversão e manipulação de datas	46
1.17 Exercícios	49
2 Estatística Descritiva	51
2.1 Natureza da medida das variáveis	51
2.2 Tabelas	53
2.3 Gráficos	55
2.4 Estatísticas Descritivas	68
2.5 Exercícios	71
3 Estatística Inferencial	72
3.1 Intervalo de Confiança	72
3.2 Teste de hipóteses	77
3.3 Exercícios	85
4 Teste de Qui-Quadrado	87
4.1 Teste de qui-quadrado para verificar associação entre duas variáveis qualitativas	87

4.2	Check list para escolher o teste adequado para verificar a relação entre duas variáveis qualitativas	88
4.3	Exemplo utilizando os recursos do software R	89
4.4	Teste de associação com duas amostras dependentes	91
4.5	Teste de qui-quadrado para verificar aderência a uma distribuição	93
4.6	Exercícios	94
5	Modelos de Regressão Linear Simples	95
5.1	Correlação linear	95
5.2	Diagrama de dispersão	95
5.3	Coefficiente de Correlação Linear de Pearson	97
5.4	Regressão Linear Simples	99
5.5	Modelo de Regressão Linear Simples	100
5.6	Método dos Mínimos Quadrados	101
5.7	Análise de Variância	102
5.8	Coefficiente de Determinação	104
5.9	Intervalo de Predição	106
5.10	Análise dos Resíduos	107
5.11	Exercícios	114
6	RMarkdown	116
6.1	Criando o documento	116
6.2	Compilando os resultados do arquivo	117
6.3	Elementos básicos do RMarkdown	118
6.4	Elementos básicos de formatação	119
6.5	Elementos básicos do YAML	123
6.6	Elementos básicos dos Chunks	125
6.7	Criação de modelos para formatação vinculada	129
6.8	Citações e bibliografias	131
6.9	Exercícios	136
	Sobre os autores	137
	Referências	139

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 melhore constantemente. No ano de 2018, já haviam mais de 12.700 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 elaborado 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, o *software* R ver-

são 3.5.2, RStudio 1.1.463. A solução dos exercícios de cada capítulo está disponibilizada no site <https://smolski.github.io/softwarelivrer/livro.html>. Importante mencionar que este livro originou-se de projeto de extensão aprovado no Edital de Apoio a Programas de Extensão (Número 522/GR/UFFS/2016) da Universidade Federal da Fronteira Sul (UFFS).

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, 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 para verificar a normalidade dos dados e a comparação entre médias de duas amostras dependentes ou 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 Linear Simples**] 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 de determinação 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).

Bons estudos!

Iara Denise Endruweit Battisti, Felipe Micaíl da Silva Smolski (*Organizadores*)

Capítulo 1

Primeiros Passos com o R

Felipe Micaíl da Silva Smolski

Djaina Sibiani Rieger

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 serão abordados 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* (R Core Team, 2019).

RStudio: <http://www.rstudio.com/products/rstudio/download>. Em RStudio Desktop, escolha a versão *free*, seguidas da opção do sistema operacional do usuário (RStudio Team, 2019).

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.

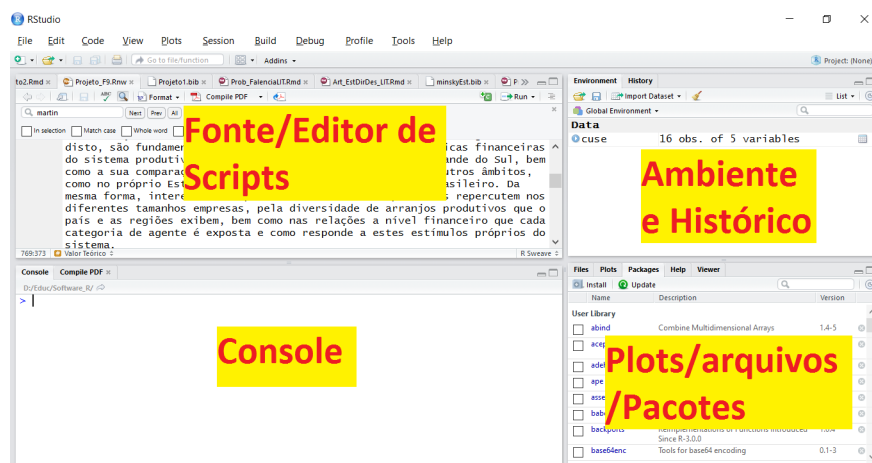


Figura 1.1: Painéis do Rstudio

Fonte: Elaborado pelo(s) autor(es).

- **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.
- **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

A ajuda do RStudio é acessada 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 console digita-se `?` 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. É preciso ir até o painel dos pacotes em *packages*, selecionar a opção instalar e inserir o nome do pacote desejado na janela indicada. Ao selecionar a opção instalar, no console são demonstradas informações do procedimento e do sucesso do mesmo.

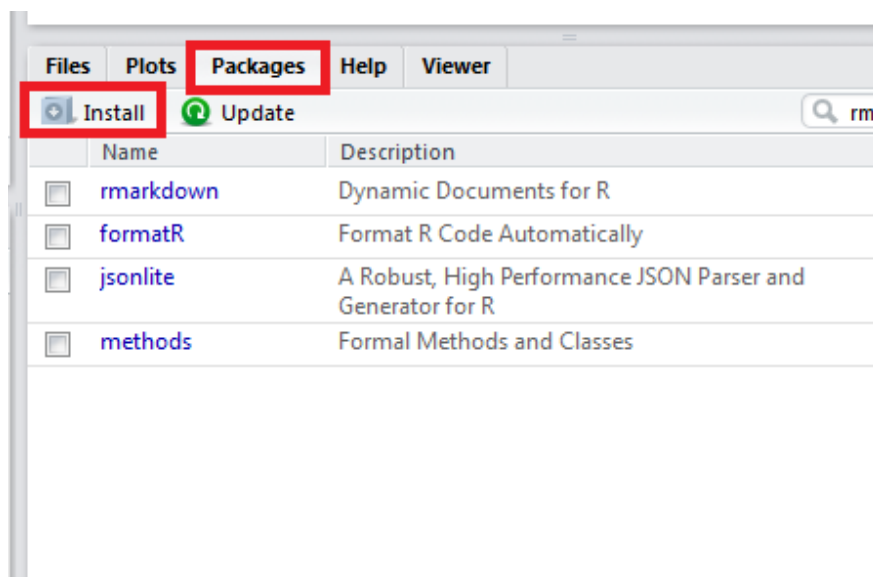


Figura 1.2: Instalação de pacotes

Fonte: Elaborado pelo(s) autor(es).

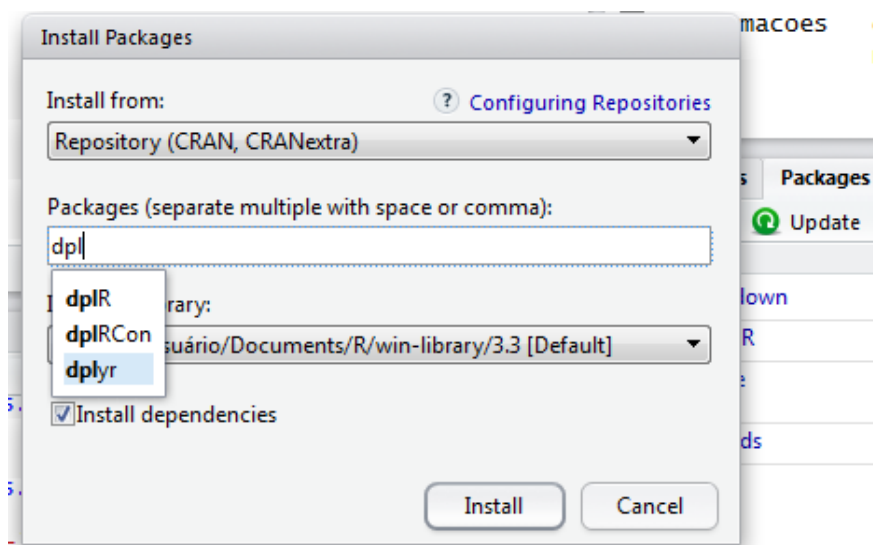


Figura 1.3: Caixa de informação de pacote a ser instalado

Fonte: Elaborado pelo(s) autor(es)

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 [árvores] (<https://github.com/Smolski/livror/raw/master/arvores.xlsx>) como exemplo de 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 requerida, dependendo da origem da base de dados, como por exemplo o **readxl** (Wickham e Bryan, 2018), **readr** (Wickham, Hester e Francois, 2018) e **haven** (Wickham e Miller, 2018), 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 função **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:

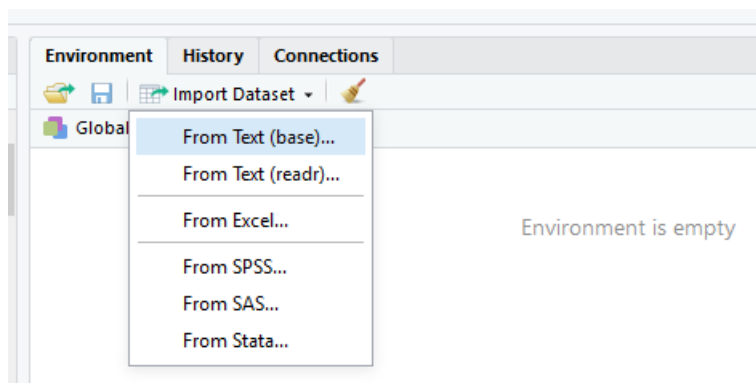


Figura 1.4: Aba Import Dataset

Fonte: Elaborado pelo(s) autor(es).

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 no arquivo. Neste exemplo, é carregada a planilha **arvores** (arquivo .xls) como mostra a Figura 1.5, derivado do caminho “Import Dataset > From Excel” do Environment.

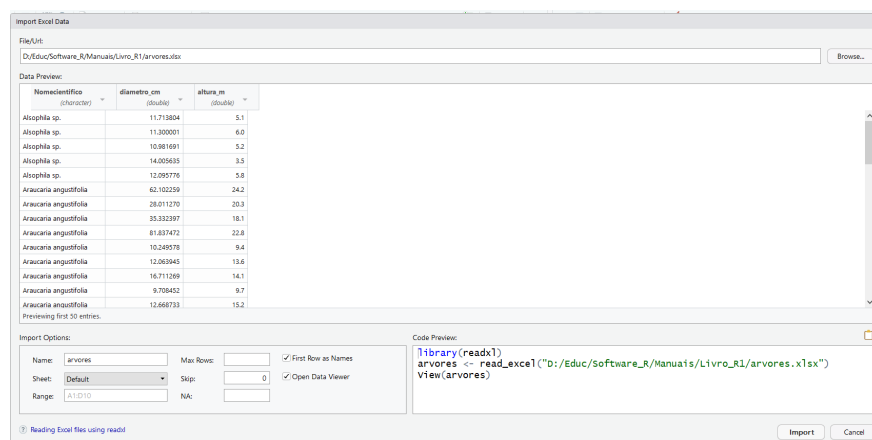


Figura 1.5: Caixa de informações do Import Data

Fonte: Elaborado pelo(s) autor(es).

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

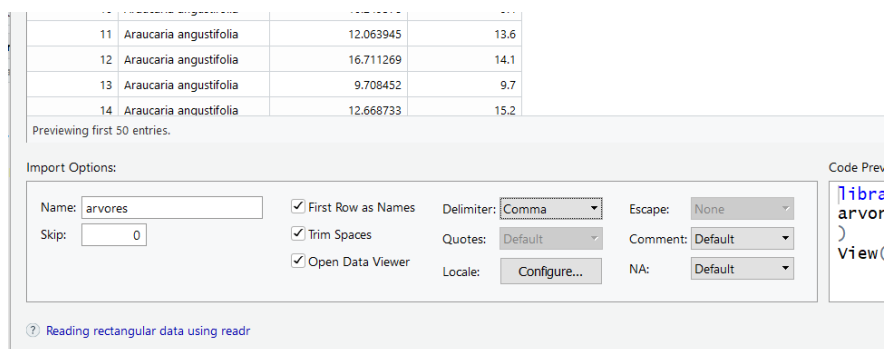


Figura 1.6: Opções da importação de arquivos .csv

Fonte: Elaborado pelo(s) autor(es)

Importante mencionar que em ambos os casos de importação, no campo *Data 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.

Ainda, é possível importar objetos utilizando arquivos hospedados em links da internet, por exemplo o comando abaixo utiliza a função `source()` para carregar um objeto do R denominado `cdc` (“cdc.R”).

```
source("http://www.openintro.org/stat/data/cdc.R")
```

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.

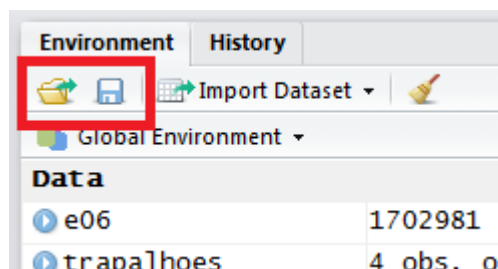


Figura 1.7: Atalho para abrir e salvar arquivo de dados

Fonte: Elaborado pelo(s) autor(es)

Outra opção com mesmo efeito é utilizar o comando a seguir diretamente no console do 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 Bases de dados nativas do R e de pacotes

Para estimular a aprendizagem da linguagem de programação R e o uso do software RStudio, bem como para acompanhar muitos manuais e livros, existem bases de dados pré-estabelecidas que podem ser utilizadas para treino e manipulação pelos usuários. Algumas delas são nativas do RStudio, como por exemplo a famosa base `iris`. Para retomar tal base de dados nativa, basta “chamar” seu nome no console do programa:

```
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

O comando `data()` lista todas as bases nativas do RStudio:

```
data()
```

Outras bases de dados, no entanto, vêm acompanhadas dos pacotes que são instalados no RStudio. Para verificar quais bases estão instaladas e disponíveis ao pesquisador, basta efetuar o comando abaixo:

```
data(package = .packages(all.available = TRUE))
```

É possível carregar uma base de dados de determinado pacote instalado, como exemplo utilizado a partir do pacote `Amelia` (Honaker, King e Blackwell, 2011):

```
data(africa, package="Amelia")
head(africa)
```

	year	country	gdp_pc	infl	trade	civlib	population
1	1972	Burkina Faso	377	-2.92	29.69	0.5000	5848380
2	1973	Burkina Faso	376	7.60	31.31	0.5000	5958700
3	1974	Burkina Faso	393	8.72	35.22	0.3333	6075700
4	1975	Burkina Faso	416	18.76	40.11	0.3333	6202000
5	1976	Burkina Faso	435	-8.40	37.76	0.5000	6341030
6	1977	Burkina Faso	448	29.99	41.11	0.6667	6486870

1.9 Operações

1.9.1 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 último adições e subtrações, de acordo com a ordem que estão dispostas. Utiliza-se o parênteses 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
```

1.9.2 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
```

1.10 Criação de objetos

A linguagem de programação R se configura em uma linguagem orientada a objetos, ou seja, a todo tempo estão sendo criados diversos tipos de objetos e sendo efetuadas 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"      "africa" "objeto"
```

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

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. É preciso 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"
```

1.11 Algumas funções e 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`, é possível 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 é

utilizada 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$Species)
```

```
[1] "factor"
```

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.

1.11.1 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, é alterado 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 selecionar uma coluna do objeto “iris”, por exemplo a coluna “Sepal.Length”, pode-se 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: **novooobjeto=iris\$novacoluna**.

No entanto, para acessar os dados sem o uso do símbolo `$`, é utilizado o seguinte comando: `attach(iris)`. Assim, é possível 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
```

1.11.2 Função *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 filtrar a informação (coluna “Sepal.Length”) média por *Especie*, é possível 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
```

1.11.3 Função *subset*

Utiliza-se o comando `subset()` para formar um subconjunto de dados o qual deseja-se selecionar de um objeto. Por exemplo, se a intenção é criar um novo objeto com somente os dados filtrados da “Especie” denominada “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, é criado 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

1.11.4 Função *table*

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

Para exemplo da utilização da função `table` agora com dados qualitativos (gênero e saúde), é utilizada a base de dados `cdc`:

```
# Carregando a base
source("http://www.openintro.org/stat/data/cdc.R")
#Vizualiza-se as primeiras linhas
head(cdc)

  genhlth exerany hlthplan smoke100 height weight wt desire age gender
1    good      0        1         0    70   175   175  77      m
2    good      0        1         1    64   125   115  33      f
3    good      1        1         1    60   105   105  49      f
4    good      1        1         0    66   132   124  42      f
5 very good      0        1         0    61   150   130  55      f
6 very good      1        1         0    64   114   114  55      f

# Efetua-se a contagem dos dados qualitativos com a função table
table(cdc$genhlth,cdc$gender)
```

	m	f
excellent	2298	2359

```

very good 3382 3590
good      2722 2953
fair      884 1135
poor      283 394

```

```

# Adiciona-se a soma dos valores das linhas e colunas
addmargins(table(cdc$genhlth,cdc$gender))

```

```

          m      f    Sum
excellent 2298 2359 4657
very good 3382 3590 6972
good      2722 2953 5675
fair      884 1135 2019
poor      283 394 677
Sum      9569 10431 20000

```

1.12 Estrutura de dados

1.12.1 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
c(2, 4, 6)

```

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

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

```
c(2:6)
```

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

```

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

```

```
[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:

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

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

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

```
[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
```

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

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

1.12.3 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
```

Comandos para manipulação de listas

Para descobrir de maneira rápida o números de objetos que há na lista, utiliza-se 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 utiliza-se 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
```


1.12.4 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. No exemplo abaixo é possível verificar como inserir os dados para criar a “tabela”. São similares como as matrizes, porém 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$ciudades=c("Nova Hartz","Gramado","Soledade",
                        "Porto Alegre")
informacoes
```

	estudantes	idade	peso	ciudades
1	Camila	21	65	Nova Hartz
2	Pedro	17	79	Gramado
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$ciudades, informacoes$ciudades=="Soledade"))

[1] 1
```

1.13 Pré tratamento de banco de dados

Os bancos de dados da “vida real” muitas vezes carecem de um tratamento inicial antes de serem destinados para a análise estatística. Isto porque, ao serem carregadas ao R estas bases estão permeadas por dados que podem prejudicar a criação de modelos ou mesmo enviesar as apresentações, como por exemplo a presença de dados faltantes (“NAs”), valores extremos (“outliers”) ou também apresentar rótulos das variáveis não adequados.

Longe de apresentar um conjunto de regras rígidas para estas correções, visto que em

muitos casos o tipo de substituição ou correção de variáveis dependerá do problema e da técnica estatística e serem trabalhados, almeja-se mostrar princípios de ações corretivas que podem ser efetuadas no RStudio. Será utilizada a base `starwars` (as 5 primeiras colunas) que consta junto ao pacote `dplyr` (Wickham *et al.*, 2019), como pode ser visto:

```
library(dplyr)
starwars=data.frame(starwars[1:5])
str(starwars)
```

```
'data.frame':  87 obs. of  5 variables:
 $ name      : chr  "Luke Skywalker" "C-3P0" "R2-D2" "Darth Vader" ...
 $ height    : int  172 167 96 202 150 178 165 97 183 182 ...
 $ mass      : num  77 75 32 136 49 120 75 32 84 77 ...
 $ hair_color: chr  "blond" NA NA "none" ...
 $ skin_color: chr  "fair" "gold" "white, blue" "white" ...
```

```
summary(starwars)
```

name	height	mass	hair_color
Length:87	Min. : 66	Min. : 15.0	Length:87
Class :character	1st Qu.:167	1st Qu.: 55.6	Class :character
Mode :character	Median :180	Median : 79.0	Mode :character
	Mean :174	Mean : 97.3	
	3rd Qu.:191	3rd Qu.: 84.5	
	Max. :264	Max. :1358.0	
	NA's :6	NA's :28	

```
skin_color
Length:87
Class :character
Mode :character
```

Nota-se que constam 5 variáveis (“name”, “height”, “mass”, “hair_color”, “skin_color”) que tratam de personagens dos filmes da franquia Star Wars com algumas características dos mesmos. Constam ainda variáveis com valores ausentes e dos mais variados tipos (“chr”, “int”, “num”, “chr”, “chr”). A função `abbreviate()` é utilizada para abreviar observações, sendo que pode ser extremamente útil quando os nomes das variáveis, por exemplo, são muito extensos. Ainda é possível determinar o tamanho dos caracteres, conjuntamente com a função `names()`.

```
names(starwars)=abbreviate(names(starwars), minlength = 3)
names(starwars)
```

```
[1] "nam" "hgh" "mss" "hr_" "sk_"
```

Caso o pesquisador deseje renomear todas as variáveis, a função `names()` pode ser utilizada como é mostrado abaixo, em um primeiro momento somente determinando o nome

da primeira variável, e no segundo exemplo alterando todas as variáveis do objeto:

```
names(starwars)[1]="Nome"
names(starwars)=c("Nome", "Altura", "Peso", "Corcabelo", "Corpele")
names(starwars)
```

```
[1] "Nome"      "Altura"    "Peso"      "Corcabelo" "Corpele"
```

Em sendo pertinente efetuar a alteração de uma variável para fator, utiliza-se a função `as.factor()` como no exemplo abaixo para transformar as variáveis “Corcabelo” e “Corpele”.

```
starwars$Corcabelo=as.factor(starwars$Corcabelo)
starwars$Corpele=as.factor(starwars$Corpele)
summary(starwars$Corpele)
```

blue	blue, grey	brown	brown mottle
2	2	4	1
brown, white	dark	fair	fair, green, yellow
1	6	17	1
gold	green	green-tan, brown	green, grey
1	6	1	1
grey	grey, blue	grey, green, yellow	grey, red
6	1	1	1
light	metal	mottled green	none
11	1	1	1
orange	pale	red	red, blue, white
2	5	1	1
silver, red	tan	unknown	white
1	2	2	2
white, blue	white, red	yellow	
2	1	2	

Como visto anteriormente este objeto apresenta valores faltantes (“NAs”) em diversas variáveis.

```
head(is.na(starwars))
```

	Nome	Altura	Peso	Corcabelo	Corpele
[1,]	FALSE	FALSE	FALSE	FALSE	FALSE
[2,]	FALSE	FALSE	FALSE	TRUE	FALSE
[3,]	FALSE	FALSE	FALSE	TRUE	FALSE
[4,]	FALSE	FALSE	FALSE	FALSE	FALSE
[5,]	FALSE	FALSE	FALSE	FALSE	FALSE
[6,]	FALSE	FALSE	FALSE	FALSE	FALSE

Desta forma, é possível quantificar os valores faltantes do objeto:

```
table(is.na(starwars))
```

```
FALSE TRUE
```

396 39

Em muitos casos é de interesse ao pesquisador substituir os valores faltantes pelo valor da média da variável em questão, é claro considerando somente o restante dos valores excluídos dos NAs. O exemplo abaixo demonstra o sumário das informações após a normalização pelo valor médio da variável “Altura” (também é possível utilizar outro critério, por exemplo o valor da moda ou mediana):

```
# Substituindo NAs por média
```

```
starwars$Altura[is.na(starwars$Altura)]=mean(starwars$Altura, na.rm=TRUE)
summary(starwars$Altura)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
66	168	178	174	190	264

É possível substituir os dados faltantes por zero ou por outro caractere, sendo que no exemplo abaixo utilizou-se a correção por zero sobre a variável numérica “Peso”:

```
# Substituindo NAs por zero
```

```
starwars$Peso[is.na(starwars$Peso)]=0
summary(starwars$Peso)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0	0.0	56.2	66.0	80.0	1358.0

De igual forma, é possível transformar qualquer valor constante em uma variável em valores faltantes. No exemplo abaixo é desfeita operação anterior sobre a variável “Peso”:

```
starwars$Peso[starwars$Peso==0]=NA
```

```
summary(starwars$Peso)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
15.0	55.6	79.0	97.3	84.5	1358.0	28

Ainda, se houver no objeto linhas que tenham pelo menos uma informação faltante (“NA”), estas podem ser excluídas com o comando `na.omit()`. Note que com este procedimento o objeto `starwars` passou a ter a dimensão 54x5 ao invés de 87x5 como apresentado inicialmente com a apresentação de valores faltantes:

```
starwars=na.omit(starwars)
```

```
summary(starwars)
```

Nome	Altura	Peso	Corcabelo
Length:54	Min. : 66	Min. : 15.0	none :27
Class :character	1st Qu.:170	1st Qu.: 56.4	brown :11
Mode :character	Median :182	Median : 79.0	black : 7
	Mean :177	Mean : 77.2	white : 3
	3rd Qu.:193	3rd Qu.: 84.8	blond : 2
	Max. :234	Max. :159.0	auburn, white: 1
			(Other) : 3
Corpele			
fair : 9			

```
light : 7
dark  : 4
green : 4
grey  : 4
brown : 3
(Other):23
```

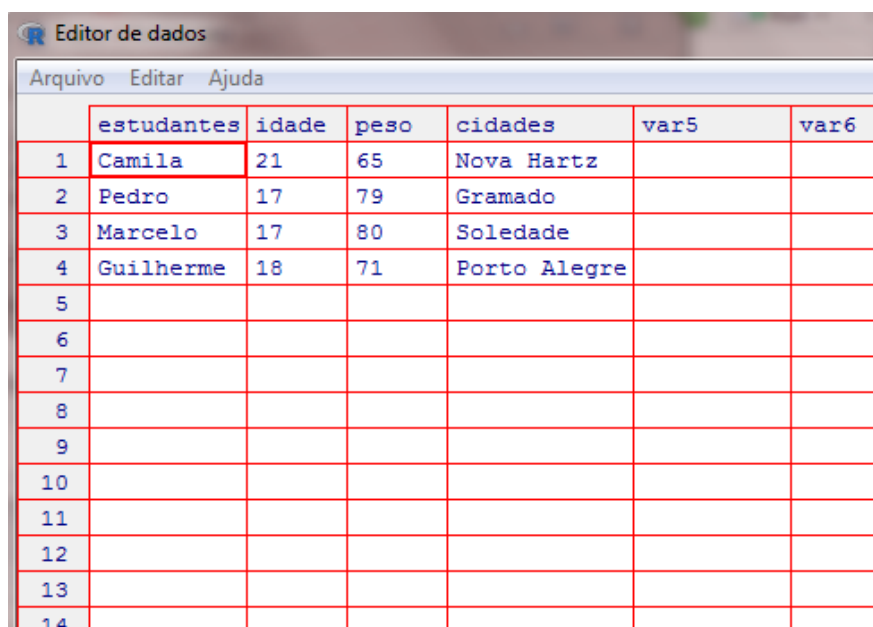
1.14 Manipulação de banco de dados

A função `edit()` 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:

```
novonomedatabase = edit(nomeatualdatabase)
```

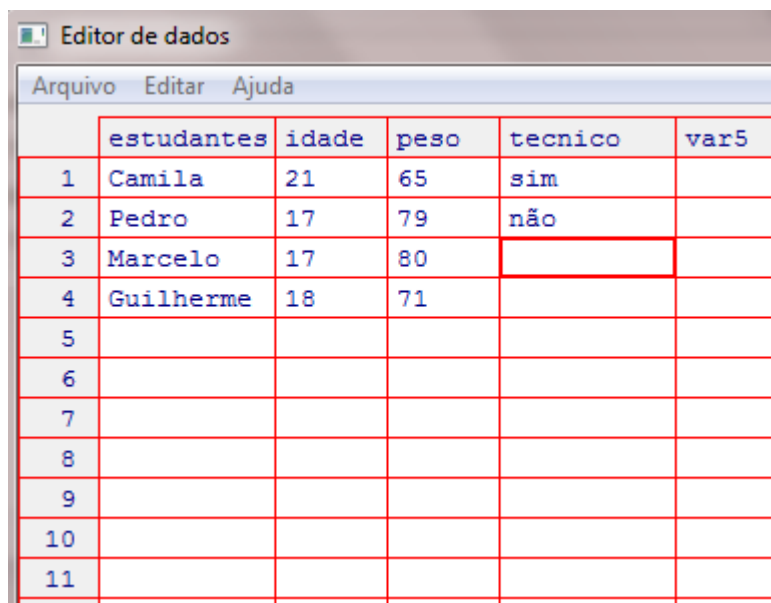
```
informacoes.2=edit(informacoes)
```



	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

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



	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

Logo, chamando o novo banco de dados, é obtido:

```
informacoes.2
```

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

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:

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

```
estudantes altura
1  Guilherme  1.50
2   Marcelo  1.90
3    Pedro  1.74
4   Camila  1.80
```

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):

```
# União de um banco de dados (existência de uma variável em comum)

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

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

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

	estudantes	idade	peso	idades	altura	Imc
1	Camila	21	65	Nova Hartz	1.80	20.06
2	Guilherme	18	100	Porto Alegre	1.50	44.44
3	Marcelo	17	80	Soledade	1.90	22.16
4	Pedro	17	79	Gramado	1.74	26.09

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:

```
# Substituir números na coluna

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

	estudantes	idade	peso	idades	altura	Imc
1	Camila	21	65	Nova Hartz	1.80	20.06
2	Guilherme	18	100	Porto Alegre	1.50	44.44
3	Marcelo	19	80	Soledade	1.90	22.16
4	Pedro	19	79	Gramado	1.74	26.09

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, utilizar o comando `cbind()` para unir os dois objetos pelas colunas. Caso não se deseje utilizar o comando `cbind()`, poderia ser criado uma nova coluna com o nome do objeto sendo “informacoes\$classificacao”.

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)

```
# 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	idades	altura	Imc	classificacao
1	Camila	21	65	Nova Hartz	1.80	20.06	peso normal
2	Guilherme	18	100	Porto Alegre	1.50	44.44	excesso de peso
3	Marcelo	19	80	Soledade	1.90	22.16	peso normal
4	Pedro	19	79	Gramado	1.74	26.09	excesso de peso

Fonte: Adaptado de Brasil (2014) .

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	20.06	peso normal	Peso Normal
2	Guilherme	18	100	Porto Alegre	1.50	44.44	excesso de peso	Obesidade III
3	Marcelo	19	80	Soledade	1.90	22.16	peso normal	Peso Normal
4	Pedro	19	79	Gramado	1.74	26.09	excesso de peso	Acima do Peso

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	20.06	peso normal	Peso Normal
2	Guilherme	18	100	Porto Alegre	1.50	44.44	excesso de peso	Obesidade III
3	Marcelo	19	80	Soledade	1.90	22.16	peso normal	Peso Normal
4	Pedro	19	79	Gramado	1.74	26.09	excesso de peso	Acima do Peso

	binario
1	1
2	0
3	1
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 23,33768, classificado como peso normal. Lembrando de incluir os campos “tipoimc” e “binario”.

```
novol=data.frame(estudantes="Francisco", idade=30, peso=59,
                  cidades="Ijuí",
                  altura="1.59",
                  Imc= 23.33768,
                  classificacao= "peso normal",
                  tipoimc="Peso Normal",
                  binario=1)
informacoes=rbind(informacoes, novol)
informacoes
```

	estudantes	idade	peso	idades	altura	Imc	classificacao	tipoimc
1	Camila	21	65	Nova Hartz	1.8	20.06	peso normal	Peso Normal
2	Guilherme	18	100	Porto Alegre	1.5	44.44	excesso de peso	Obesidade III
3	Marcelo	19	80	Soledade	1.9	22.16	peso normal	Peso Normal
4	Pedro	19	79	Gramado	1.74	26.09	excesso de peso	Acima do Peso
5	Francisco	30	59	Ijuí	1.59	23.34	peso normal	Peso Normal

	binario
1	1
2	0
3	1
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 serão chamados de “adulto” e do contrário “não adulto”.

```
library(dplyr)
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	20.06	peso normal	Peso Normal
2	Guilherme	18	100	Porto Alegre	1.5	44.44	excesso de peso	Obesidade III
3	Marcelo	19	80	Soledade	1.9	22.16	peso normal	Peso Normal
4	Pedro	19	79	Gramado	1.74	26.09	excesso de peso	Acima do Peso
5	Francisco	30	59	Ijuí	1.59	23.34	peso normal	Peso Normal

```

binario faixa etaria
1      1      adulto
2      0    não adulto
3      1    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(8,2,3,4,1,6,5,7,9,10)]
```

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)
```

	faixa etaria	binario	classificacao	altura	Imc	estudantes	idades
1	adulto	1	peso normal	1.8	20.06	Camila	Nova Hartz
2	não adulto	0	excesso de peso	1.5	44.44	Guilherme	Porto Alegre
3	não adulto	1	peso normal	1.9	22.16	Marcelo	Soledade
4	não adulto	0	excesso de peso	1.74	26.09	Pedro	Gramado
5	adulto	1	peso normal	1.59	23.34	Francisco	Ijuí

```

peso idade      tipoimc
1  65    21  Peso Normal
2 100    18 Obesidade III
3  80    19  Peso Normal
4  79    19 Acima do Peso
5  59    30  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
                2                3
```

```
# 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	23.34	1.59	peso normal
1	Peso Normal	21	65	Nova Hartz	Camila	20.06	1.8	peso normal
3	Peso Normal	19	80	Soledade	Marcelo	22.16	1.9	peso normal
4	Acima do Peso	19	79	Gramado	Pedro	26.09	1.74	excesso de peso
2	Obesidade III	18	100	Porto Alegre	Guilherme	44.44	1.5	excesso de peso
	binario	faixa	etaria					
5	1		adulto					
1	1		adulto					
3	1	não	adulto					
4	0	não	adulto					
2	0	não	adulto					

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

	tipoimc	idade	peso	cidades	estudantes	Imc	altura	classificacao
2	Obesidade III	18	100	Porto Alegre	Guilherme	44.44	1.5	excesso de peso
3	Peso Normal	19	80	Soledade	Marcelo	22.16	1.9	peso normal
4	Acima do Peso	19	79	Gramado	Pedro	26.09	1.74	excesso de peso
1	Peso Normal	21	65	Nova Hartz	Camila	20.06	1.8	peso normal
5	Peso Normal	30	59	Ijuí	Francisco	23.34	1.59	peso normal
	binario	faixa	etaria					
2	0	não	adulto					
3	1	não	adulto					
4	0	não	adulto					
1	1		adulto					
5	1		adulto					

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

	tipoimc	idade	peso	cidades	estudantes	Imc	altura	classificacao
4	Acima do Peso	19	79	Gramado	Pedro	26.09	1.74	excesso de peso

```

5  Peso Normal      30   59          Ijuí Francisco 23.34   1.59   peso normal
1  Peso Normal      21   65   Nova Hartz   Camila 20.06   1.8    peso normal
2  Obesidade III    18  100 Porto Alegre  Guilherme 44.44   1.5  excesso de peso
3  Peso Normal      19   80      Soledade   Marcelo 22.16   1.9    peso normal
  binario faixa etaria
4      0   não adulto
5      1      adulto
1      1      adulto
2      0   não adulto
3      1   não adulto

```

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      cidades estudantes   Imc altura  classificacao
1  Peso Normal   21   65   Nova Hartz   Camila 20.06   1.8    peso normal
2  Obesidade III  18  100 Porto Alegre  Guilherme 44.44   1.5  excesso de peso
3  Peso Normal   19   80      Soledade   Marcelo 22.16   1.9    peso normal
4  Acima do Peso  19   79      Gramado     Pedro 26.09   1.74  excesso de peso
5  Peso Normal   30   59          Ijuí Francisco 23.34   1.59    peso normal
  binario faixa etaria rankingImc
1      1      adulto          1
2      0   não adulto          5
3      1   não adulto          2
4      0   não adulto          4
5      1      adulto          3

```

Para utilizar a função `rank` com os maiores valores em primeiro lugar:

```
rank(-informacoes$Imc)
```

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

1.14.1 O pacote *tidyr*

Nesta subseção será utilizado o pacote `tidyr` (Wickham e Henry, 2018) para demonstrar algumas funções que contribuem para a manipulação das bases de dados, tão importante no processo de preparação das informações para posterior análise. Serão utilizadas para demonstração as bases de dados existentes no próprio pacote.

Abaixo segue uma demonstração das convenções a respeito das bases de dados. Desta forma verifica-se que cada variável é apresentada em sua respectiva coluna, bem como as observações são apresentadas em sua própria linha e portanto os valores constam em sua própria célula.

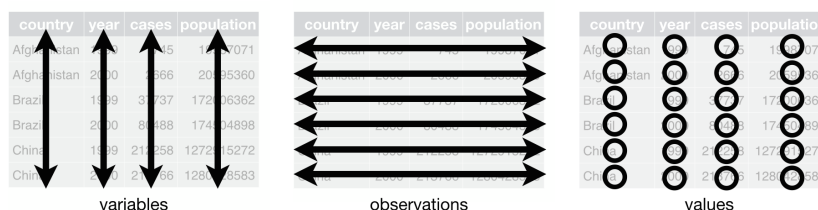


Figura 1.10: Convenção sobre variáveis, observações e valores

Fonte: <http://garrettgman.github.io/tidying/>.

1.14.1.1 Função *spread*

A função `spread()` é utilizada para transformar os valores constantes em uma coluna em nova configuração de colunas. Ainda, é possível determinar a transformação dos valores com o comando `convert = TRUE` informando o tipo de valores (doubles (numerics), integers, logicals, complexes, ou factors) nas colunas a serem criadas (comando `type.convert()`).

```
library(tidyr)
table2
```

```
# A tibble: 12 x 4
  country      year type      count
  <chr>      <int> <chr>      <int>
1 Afghanistan 1999 cases         745
2 Afghanistan 1999 population 19987071
3 Afghanistan 2000 cases         2666
4 Afghanistan 2000 population 20595360
5 Brazil       1999 cases         37737
6 Brazil       1999 population 172006362
7 Brazil       2000 cases         80488
8 Brazil       2000 population 174504898
9 China        1999 cases         212258
10 China       1999 population 1272915272
11 China       2000 cases         213766
12 China       2000 population 1280428583
```

Neste exemplo, a coluna “type” abriga os valores “cases” e “population”, as quais terão suas próprias colunas com seus respectivos valores:

```
spread(table2, type, count)
```

```
# A tibble: 6 x 4
  country      year cases population
  <chr>      <int> <int>      <int>
1 Afghanistan 1999     745 19987071
2 Afghanistan 2000    2666 20595360
```

```
3 Brazil      1999  37737  172006362
4 Brazil      2000  80488  174504898
5 China       1999 212258 1272915272
6 China       2000 213766 1280428583
```

1.14.1.2 Função *gather*

Já a função `gather()` realiza o processo oposto do comando `spread()`, pois agrupa o valor de determinadas variável em uma chave comum.

```
table4a
```

```
# A tibble: 3 x 3
  country `1999` `2000`
* <chr>   <int> <int>
1 Afghanistan  745  2666
2 Brazil      37737 80488
3 China      212258 213766
```

Abaixo a transformação das variáveis “1999” e “2000” em uma única variável “year”, mantendo os valores inseridos na variável “cases”:

```
gather(table4a, "year", "cases", 2:3)
```

```
# A tibble: 6 x 3
  country year cases
  <chr>   <chr> <int>
1 Afghanistan 1999    745
2 Brazil      1999  37737
3 China       1999 212258
4 Afghanistan 2000    2666
5 Brazil      2000  80488
6 China       2000 213766
```

1.14.1.3 Função *separate*

A função `separate()` é utilizada para partir uma determinada variável em novas variáveis da base de dados.

```
table3
```

```
# A tibble: 6 x 3
  country year rate
  <chr>   <int> <chr>
1 Afghanistan 1999 745/19987071
2 Afghanistan 2000 2666/20595360
3 Brazil      1999 37737/172006362
4 Brazil      2000 80488/174504898
5 China       1999 212258/1272915272
6 China       2000 213766/1280428583
```

Neste exemplo, a variável “rate”, que está composta de duas informações separadas pelo caractere “/”, será separada nas novas variáveis “cases” e “population”:

```
separate(table3, rate, into = c("cases", "population"), sep = "/")
```

```
# A tibble: 6 x 4
  country      year cases population
  <chr>      <int> <chr>    <chr>
1 Afghanistan 1999 745    19987071
2 Afghanistan 2000 2666   20595360
3 Brazil      1999 37737  172006362
4 Brazil      2000 80488  174504898
5 China       1999 212258 1272915272
6 China       2000 213766 1280428583
```

Da mesma forma é possível criar duas novas variáveis a partir do segundo caractere do valor que consta nas células utilizando o comando `sep=2`:

```
separate(table3, year, into = c("century", "year"), sep = 2)
```

```
# A tibble: 6 x 4
  country      century year  rate
  <chr>      <chr>    <chr> <chr>
1 Afghanistan 19      99    745/19987071
2 Afghanistan 20      00    2666/20595360
3 Brazil      19      99    37737/172006362
4 Brazil      20      00    80488/174504898
5 China       19      99    212258/1272915272
6 China       20      00    213766/1280428583
```

1.14.1.4 Função *unite*

A função `unite()` é oposta à função `separate()`:

```
table5
```

```
# A tibble: 6 x 4
  country      century year  rate
  * <chr>      <chr>    <chr> <chr>
1 Afghanistan 19      99    745/19987071
2 Afghanistan 20      00    2666/20595360
3 Brazil      19      99    37737/172006362
4 Brazil      20      00    80488/174504898
5 China       19      99    212258/1272915272
6 China       20      00    213766/1280428583
```

Neste exemplo, recria a variável “new” a partir dos dados de “century” e “year”:

```
unite(table5, "new", century, year, sep = "")
```

```
# A tibble: 6 x 3
```

	country	new	rate
	<chr>	<chr>	<chr>
1	Afghanistan	1999	745/19987071
2	Afghanistan	2000	2666/20595360
3	Brazil	1999	37737/172006362
4	Brazil	2000	80488/174504898
5	China	1999	212258/1272915272
6	China	2000	213766/1280428583

1.14.2 O pacote *dplyr*

O pacote `dplyr` é uma poderosa ferramenta para manipulação, criação e transformação de dados no RStudio, agregando agilidade e robustez para o processo de análise e preparação dos dados. Seguem a seguir algumas das principais funções do pacote com a utilização da base de dados nativa do RStudio `mtcars`.

```
head(mtcars)
```

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

1.14.2.1 Função *select*

A função `select()` é utilizada para selecionar as variáveis de interesse do pesquisador, a partir de uma base de dados, neste caso a partir da base `mtcars`:

```
library(dplyr)
```

```
novo=select(mtcars, mpg, cyl)
head(novo)
```

	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4
Hornet 4 Drive	21.4	6
Hornet Sportabout	18.7	8
Valiant	18.1	6

No exemplo são selecionadas todas as variáveis excluindo `mpg`:

```
novo=select(mtcars, -c(mpg))
head(novo)
```


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

É possível selecionar uma sequência de variáveis a partir de seus nomes (utilidade semelhante a `select(mtcars, 2:5)`):

```
novo=select(mtcars, cyl:drat)
head(novo)
```

	cyl	disp	hp	drat
Mazda RX4	6	160	110	3.90
Mazda RX4 Wag	6	160	110	3.90
Datsun 710	4	108	93	3.85
Hornet 4 Drive	6	258	110	3.08
Hornet Sportabout	8	360	175	3.15
Valiant	6	225	105	2.76

1.14.2.2 Função *filter*

A Função `filter()` seleciona as variáveis da base de dados conforme atributos determinados pelo pesquisador:

```
novo=filter(mtcars, hp>146)
head(novo)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	18.7	8	360.0	175	3.15	3.44	17.02	0	0	3	2
2	14.3	8	360.0	245	3.21	3.57	15.84	0	0	3	4
3	16.4	8	275.8	180	3.07	4.07	17.40	0	0	3	3
4	17.3	8	275.8	180	3.07	3.73	17.60	0	0	3	3
5	15.2	8	275.8	180	3.07	3.78	18.00	0	0	3	3
6	10.4	8	472.0	205	2.93	5.25	17.98	0	0	3	4

Abaixo o exemplo da utilização de mais de um critério de filtragem de dados:

```
novo=filter(mtcars, hp>146 & am==1)
head(novo)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	15.8	8	351	264	4.22	3.17	14.5	0	1	5	4
2	19.7	6	145	175	3.62	2.77	15.5	0	1	5	6
3	15.0	8	301	335	3.54	3.57	14.6	0	1	5	8

Utilizando o pacote `stringr` (Wickham, 2018) com sua função `str_detect()`, é possível efetuar a filtragem pelo nome/identificação total ou parcial de um valor contido dentro de uma variável do banco de dados. Utilizou-se o banco de dados “table5” visto no subcapítulo

anterior pra filtrar as informações da variável “country” que contém “Bra”:

```
library(stringr)
table5 %>%
  filter(str_detect(country, "Bra"))
```

A tibble: 2 x 4

	country	century	year	rate
	<chr>	<chr>	<chr>	<chr>
1	Brazil	19	99	37737/172006362
2	Brazil	20	00	80488/174504898

1.14.2.3 Função *mutate*

A função `mutate()` é utilizada para incluir informações ou variáveis na base de dados, como no exemplo abaixo a criação de uma nova variável denominada “novacol”, multiplicando por 100 a variável “mpg” que já consta na base:

```
novo=mutate(mtcars, novacol=(mpg*100))
head(novo)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	novacol
1	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	2100
2	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	2100
3	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	2280
4	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	2140
5	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	1870
6	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	1810

1.14.2.4 Função *summarise*

A função `summarise()` é uma poderosa ferramenta para agregar sumarizações unindo diversos cálculos ao longo de uma base de dados:

```
summarise(mtcars,
  media.hp=mean(hp),
  qtd.hp=length(hp),
  qtdunico.hp=length(unique(hp)))
```

	media.hp	qtd.hp	qtdunico.hp
1	146.7	32	22

Ainda, é possível agrupar as informações com a função `group_by()` ao mesmo tempo em que são efetuados cálculos adjacentes. No exemplo abaixo, agrupa-se o valor médio das variáveis “hp” e “wt”, bem como a quantidade de informações de cada variável (função `n()`), com relação ao agrupamento formado pela variável “cyl”:

```
summarise(group_by(mtcars, cyl.agrup=cyl),
  hp.medio=mean(hp),
  wt.medio=mean(wt),
```

```
qtd=n())
```

```
# A tibble: 3 x 4
  cyl.agrup hp.medio wt.medio   qtd
  <dbl>     <dbl>   <dbl> <int>
1      4      82.6     2.29     11
2      6     122.     3.12      7
3      8     209.     4.00     14
```

1.14.2.5 Função *count*

A função `count()` é utilizada para sumarizar a contagem de determinados objetos dentro de uma variável do banco de dados:

```
count(mtcars, cyl)
```

```
# A tibble: 3 x 2
  cyl     n
  <dbl> <int>
1     4    11
2     6     7
3     8    14
```

1.14.2.6 Função *arrange*

A função `arrange` ordena a base de dados de acordo com o ordenamento da variável escolhida:

```
novo=arrange(mtcars, cyl)
head(novo)
```

```
   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1  22.8   4 108.0  93 3.85 2.320 18.61  1  1   4     1
2  24.4   4 146.7  62 3.69 3.190 20.00  1  0   4     2
3  22.8   4 140.8  95 3.92 3.150 22.90  1  0   4     2
4  32.4   4  78.7  66 4.08 2.200 19.47  1  1   4     1
5  30.4   4  75.7  52 4.93 1.615 18.52  1  1   4     2
6  33.9   4  71.1  65 4.22 1.835 19.90  1  1   4     1
```

Ainda é possível indicar mais de uma variável para este ordenamento, bem como utilizar a função `desc()` para organizar em ordem decrescente:

```
novo=arrange(mtcars, mpg, desc(dis))
head(novo)
```

```
   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1  10.4   8  472 205 2.93 5.250 17.98  0  0   3     4
2  10.4   8  460 215 3.00 5.424 17.82  0  0   3     4
3  13.3   8  350 245 3.73 3.840 15.41  0  0   3     4
4  14.3   8  360 245 3.21 3.570 15.84  0  0   3     4
```

5	14.7	8	440	230	3.23	5.345	17.42	0	0	3	4
6	15.0	8	301	335	3.54	3.570	14.60	0	1	5	8

1.14.2.7 Operador *pipe*

O operador `pipe` (símbolos `%>%`) contribui para que a manipulação de dados com o pacote `dplyr` fiquem mais organizados no código de programação em linguagem R. Abaixo segue um exemplo, onde o objetivo é filtrar os veículos com transmissão manual (`am == 1`), agrupando-os pela quantidade de cilindros (`cyl`) e em seguida retomando a média das variáveis `drat` e `hp` para cada grupamento:

```
novo = mtcars %>%
  filter(am == 1) %>%
  group_by(cyl) %>%
  summarise(displ.drato=mean(displ),
            hp.media=mean(hp))
```

novo

```
# A tibble: 3 x 3
  cyl displ.drato hp.media
<dbl>   <dbl>   <dbl>
1     4     4.18     81.9
2     6     3.81    132.
3     8     3.88    300.
```

A função `starts_with()` seleciona as variáveis com base em um critério determinado pelo pesquisador com relação ao nome da variável, no exemplo abaixo, sendo aquelas que iniciam com a letra “d” (a função inversa é `ends_with()`):

```
mtcars %>%
  select(starts_with("d")) %>%
  head
```

	displ	drato
Mazda RX4	160	3.90
Mazda RX4 Wag	160	3.90
Datsun 710	108	3.85
Hornet 4 Drive	258	3.08
Hornet Sportabout	360	3.15
Valiant	225	2.76

A função `contains()` também filtra aquelas variáveis com algum critério, neste caso de conter:

```
mtcars %>%
  select(contains("a")) %>%
  head
```

drat am gear carb

Mazda RX4	3.90	1	4	4
Mazda RX4 Wag	3.90	1	4	4
Datsun 710	3.85	1	4	1
Hornet 4 Drive	3.08	0	3	1
Hornet Sportabout	3.15	0	3	2
Valiant	2.76	0	3	1

A função `aggregate()` também é utilizada para agregação de resultados pelo pacote `dplyr`:

```
mtcars %>%
  aggregate(. ~ cyl, ., mean)
```

	cyl	mpg	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	4	26.66	105.1	82.64	4.071	2.286	19.14	0.9091	0.7273	4.091	1.545
2	6	19.74	183.3	122.29	3.586	3.117	17.98	0.5714	0.4286	3.857	3.429
3	8	15.10	353.1	209.21	3.229	3.999	16.77	0.0000	0.1429	3.286	3.500

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] Guilherme
```

```
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] 20.06 44.44 22.16 26.09 23.34
```

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 e manipulação 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. Por padrão o RStudio trabalha com o formato ANO-MÊS-DIA, sendo possível ainda ler e incluir dados de alta frequência como horas, minutos e segundos com utilização de outros pacotes. Assim, seguem alguns dos procedimentos para a correta alteração dos padrões de datas:

```
abertura <- c("03/02/69") # Exemplo de criação de data inicial
fechamento <- c("2000-20-01") # Criação de data final
abertura <- as.Date(abertura, format = "%d/%m/%y") # Formatação da data
fechamento <- as.Date(fechamento, format = "%Y-%d-%m")
class(abertura) # Verificando a classe do objeto
```

```
[1] "Date"
```

```
class(fechamento)
```

```
[1] "Date"
```

É possível efetuar cálculos entre datas como segue:

```
# Diferença de dias dos intervalos informados
dif=abertura-fechamento # Efetua o cálculo da diferença entre as datas
dif
```

```
Time difference of -11308 days
```

```
class(dif) # Verifica a classe do objeto
```

```
[1] "difftime"
```

```
as.numeric(dif) # Retoma o valor numérico da diferença
```

```
[1] -11308
```

```
units(dif) # Retoma a unidade da diferença entre as datas
```

```
[1] "days"
```

1.16.1 O pacote *lubridate*

O pacote *lubridate* (Grolemund e Wickham, 2011) é responsável por contribuir de forma eficaz para a manipulação e transformação de variáveis sob o formato de datas e horas. No caso abaixo, os comandos `ymd()` e `mdy()` codificam corretamente para o R as datas que anteriormente não estavam neste padrão.

```
library(lubridate)
```

```
ymd(20190215)
```

```
[1] "2019-02-15"
```

```
mdy("2/15/19")
```

```
[1] "2019-02-15"
```

Para o exemplo posterior para a manipulação de datas, foi criado um objeto denominado “data” com as variáveis `data`, `quant` e `valor`, representando a quantidade e valor de vendas em determinados dias do ano:

```
# Criando uma base de dados
data=data.frame(data=c("01/01/2018","02/02/2019",
                       "02/02/2019","05/02/2019","06/02/2019"),
                 quant=c(100,200,100,150,300),
                 valor=c(550.00,600.00,100.00,150.00,250.00))
head(data)
```

	data	quant	valor
1	01/01/2018	100	550
2	02/02/2019	200	600
3	02/02/2019	100	100
4	05/02/2019	150	150
5	06/02/2019	300	250

Em primeiro lugar é efetuada a normalização da variável `data`:

```
# Configurando a variável data
data$data=dmy(data$data)
```

Depois, são criadas novas variáveis representando o ano, mês, dia e dia da semana de cada venda realizada, incrementando o poder de análise dos objetos:

```
# Criando uma nova variável do ano da venda
data$ano=year(data$data)
data
```

	data	quant	valor	ano
--	------	-------	-------	-----


```
1 2018-01-01    100    550 2018
2 2019-02-02    200    600 2019
3 2019-02-02    100    100 2019
4 2019-02-05    150    150 2019
5 2019-02-06    300    250 2019
```

```
# Criando uma nova variável do mês da venda
```

```
data$mes=month(data$data)
```

```
data
```

```
      data quant valor  ano mes
1 2018-01-01    100   550 2018   1
2 2019-02-02    200   600 2019   2
3 2019-02-02    100   100 2019   2
4 2019-02-05    150   150 2019   2
5 2019-02-06    300   250 2019   2
```

```
# Criando nova variável do dia da venda
```

```
data$dia=day(data$data)
```

```
data
```

```
      data quant valor  ano mes dia
1 2018-01-01    100   550 2018   1   1
2 2019-02-02    200   600 2019   2   2
3 2019-02-02    100   100 2019   2   2
4 2019-02-05    150   150 2019   2   5
5 2019-02-06    300   250 2019   2   6
```

```
# Criando nova variável do dia da semana da venda
```

```
data$diasem=wday(data$data, label=TRUE)
```

```
data
```

```
      data quant valor  ano mes dia diasem
1 2018-01-01    100   550 2018   1   1   seg
2 2019-02-02    200   600 2019   2   2   sáb
3 2019-02-02    100   100 2019   2   2   sáb
4 2019-02-05    150   150 2019   2   5   ter
5 2019-02-06    300   250 2019   2   6   qua
```

Com isso é possível realizar várias análises após a normalização das datas e extração de demais informações, como a sumarização de vendas por exemplo:

```
# Valor das vendas por mês
```

```
aggregate(data$valor, list(Var = data$mes), sum)
```

```
  Var    x
1   1  550
2   2 1100
```

```
# Quantidade de vendas por mês
```

```
aggregate(data$quant, list(Var = data$mes), sum)
```

```

  Var    x
1    1 100
2    2 750

# Valor das vendas por mês e dia da semana
aggregate(valor ~ mes + diasem, data = data, sum)

  mes diasem valor
1    1    seg   550
2    2    ter   150
3    2    qua   250
4    2    sáb   700

# Utilizando o pacote dplyr
library(dplyr)

data %>%
  group_by(diasem) %>%
  summarise(total = sum(valor))

# A tibble: 4 x 2
  diasem total
  <ord>   <dbl>
1 seg      550
2 ter      150
3 qua      250
4 sáb      700

```

1.17 Exercícios

1. Baixe o arquivo “arvores” que se encontra no endereço <https://smolski.github.io/softwarelivrer/livro.html>. Este é um banco de dados com informações cedido pela professora Tatiane Chassot. Abra o arquivo no Rstudio tomando os cuidados necessários (importar no formato correto, prestar atenção nas vírgulas e nomes...). Por meio dos comandos do R, responda as seguintes perguntas, informando o comando utilizado.

1.1. Qual é a espécie de árvore que possui o maior e menor diâmetro? E quais são estes valores de diâmetro?

1.2. Qual é a altura média, mínima e média das árvores?

1.3. Encontre o diâmetro médio para cada espécie de árvores.

1.4. Com os comandos do R, verifique a quantidade de dados referente as variáveis, bem como o nome referente a cada variável.

1.5. Renomeie a primeira coluna para “espécie”.

1.6. Classifique as árvores quanto ao seu porte, em relação à altura, em que:

Pequeno porte = árvores com altura inferior a 10 metros.

Grande porte = árvores com altura superior a 10 metros.

2. Baixe o arquivo “bancodedados1” que se encontra no endereço <https://smolski.github.io/softwarelivrer/livro.html>. Este é um banco de dados com informações fictícias que serão utilizados a fim de aprendizado. Abra o arquivo no Rstudio tomando os cuidados necessários. Por meio dos comandos do R, responda as seguintes perguntas, informando o comando utilizado.

2.1. Qual é o vendedor com mais sucesso de vendas? E o vendedor com menor número de vendas? Qual foi o número total de vendas?

2.3. Supondo que um vendedor tenha ficado de fora dos dados, insira suas informações no banco de dados que já possuímos.

- Vendedor = Silvia; Idade = 48; Setor = 2; N de vendas = 45.

2.4. Crie uma nova coluna classificando os vendedores como:

- vendas < 25 = “Regular” ; 25 > vendas = “Ótimo”

2.5 Renomeie a coluna “vendas mensais” para “vendas diárias”.

Capítulo 2

Estatística Descritiva

Denize Ivete Reis

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

O termo “variáveis” se reporta à 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 ocupa 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, Sweeney e Williams, 2002).

Exemplo 1

Serão utilizados 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)
```

```
library(readxl)
url <- "https://github.com/Smolski/livror/raw/master/pesquisa_dados.xlsx"
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

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 é conhecido como análise exploratória de dados. Isto pode ser feito inicialmente com a representação em forma de tabelas.

O comando `table()` é utilizado para elaborar tabelas de frequências absolutas. De-

pendendo da variável a ser representada, é possível utilizar esse comando de diferentes formas, como segue nas próximas subseções.

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`, indica-se 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 são utilizados outros mecanismos de atribuição, é preciso usar o comando `table()`.

```
barplot(table(nome_variável))
```

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

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

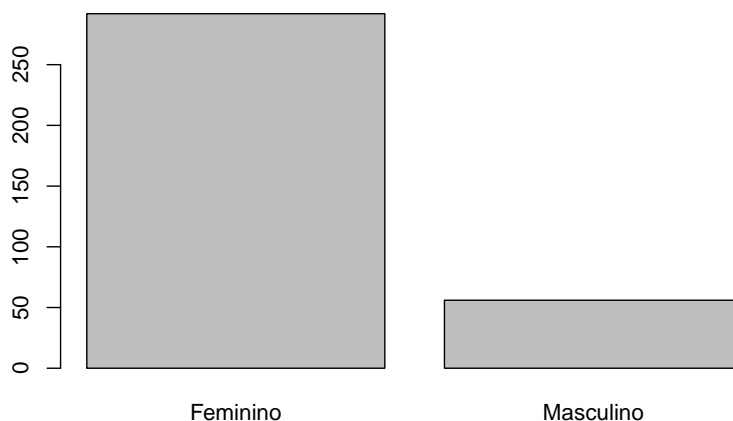


Figura 2.1: Gráfico de colunas com a variável Sexo

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")
```

Para colocar o gráfico na horizontal, pode ser utilizado o comando `horiz=T`:

```
par(las=2) # Altera a direção dos nomes Masculino e Feminino  
par(mar=c(5,8,4,2)) # Aumenta a margem do eixo x  
barplot(table(Sexo), horiz=T)
```

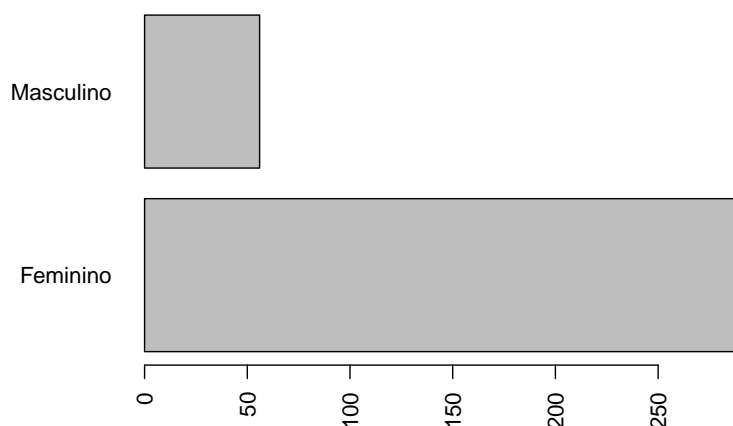


Figura 2.2: Gráfico de colunas com a variável Sexo (Horizontal)

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",  
xlab = "Frequência",  
ylab = "Pessoas",  
border = "red")
```

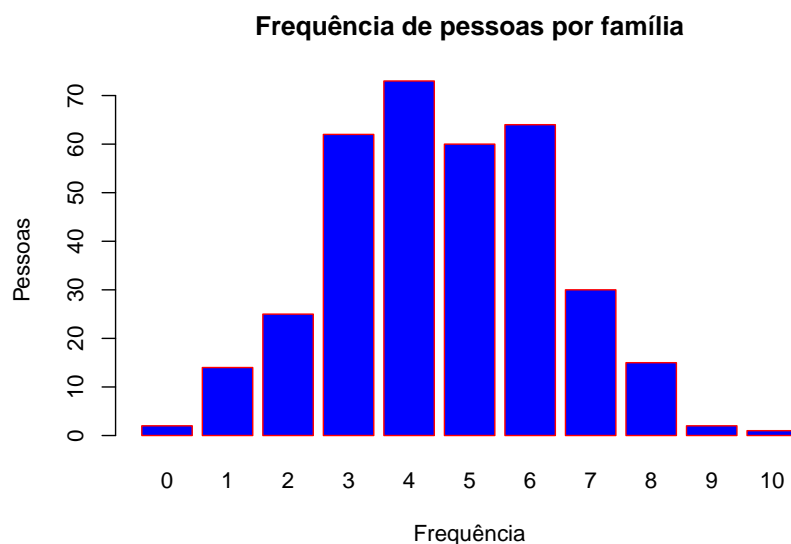


Figura 2.3: Gráfico de colunas com a variável Pessoas familia

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")
```

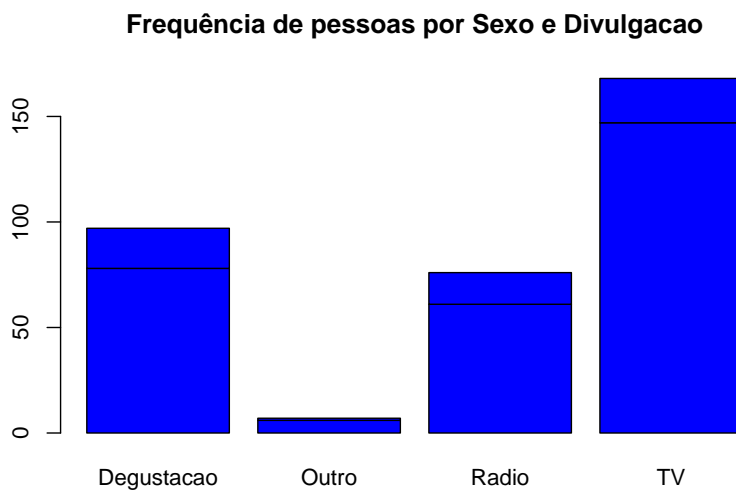


Figura 2.4: Gráfico de colunas com as variáveis Sexo e Divulgacao

Ex.3) Na sequência utiliza-se o sinal de atribuição `<-` para atribuir o nome `Resultado` para esta tabela (tabela de dupla entrada obtida em Ex.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),
        args.legend = list(x = "topleft"))
```

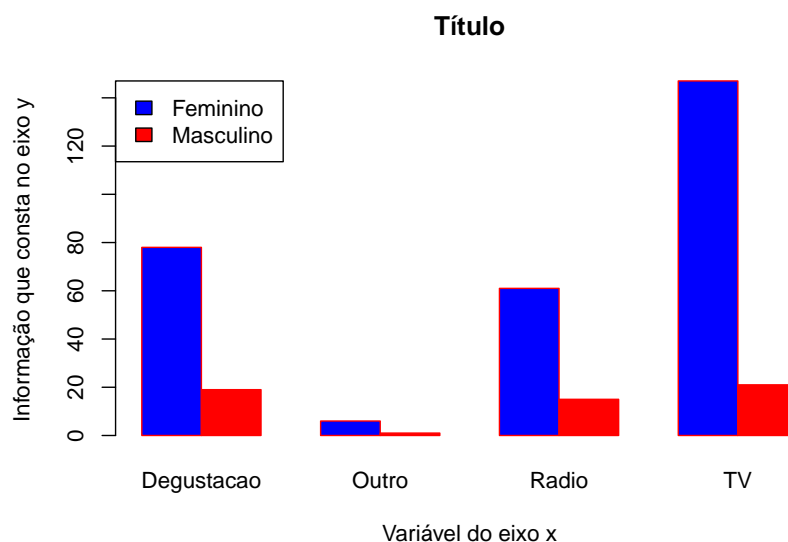


Figura 2.5: Gráfico de colunas com as variáveis Sexo e Divulgacao (2)

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 (Barbetta, 2010).

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

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

```

# Criar objeto com a tabela de Sabor
Sabor1=table(Sabor)

# Calcular o percentual
percent=signif(Sabor1/sum(Sabor1)*100,3)

#Criando os nomes da legenda
nomesleg=c("Bom","Ótimo","Péssimo","Regular","Ruim")

#Plota-se o gráfico de pizza
pie(Sabor1,
    labels = paste(percent, "%", sep=""),
    col = terrain.colors(5), # Determina cores
    radius = 1)
legend(x="topright", # Determina posição da legenda
      legend=nomesleg, # Insere nomes da legenda
      cex = 0.65, # Tamanho do texto
      fill = terrain.colors(5)) # Determina cores

## Alguns exemplos de paletas de cores:
# - rainbow(n)
# - heat.colors(n)
# - terrain.colors(n)
# - topo.colors(n)
# - cm.colors(n)

```

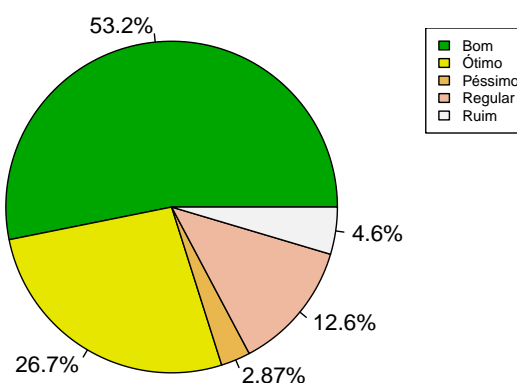


Figura 2.6: Gráfico de pizza com a variável Sabor

2.3.3 Histograma

No histograma, utilizado em geral quando têm-se variáveis quantitativas contínuas, a altura dos retângulos representa a frequência de ocorrência de valores no intervalo (deve iniciar sempre em zero), que 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 pode ser utilizado:

```
hist(nome_variável)
```

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

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

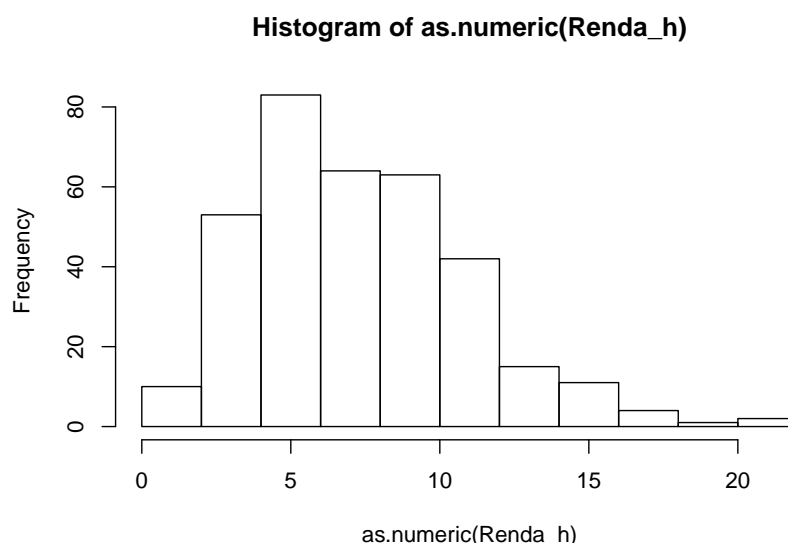


Figura 2.7: Histograma com a variável 'Renda h'

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, utiliza-se:

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

```
hist(as.numeric(`Renda_h`),  
     breaks=5,  
     labels=TRUE,  
     ylim=c(0,200),  
     xlab = 'Renda',  
     ylab = 'Frequência',  
     main = 'Histograma da Renda',  
     col = '#BBDEFB')
```

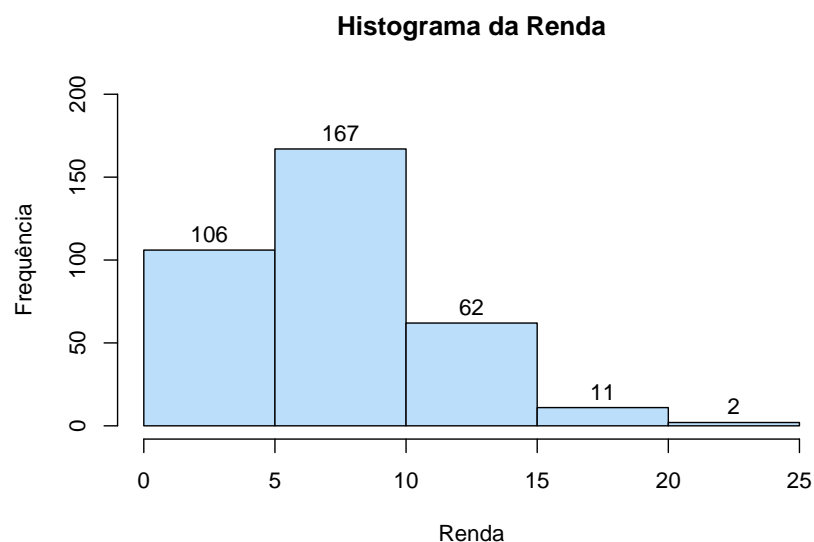


Figura 2.8: Histograma com a variável Renda h com breaks=5

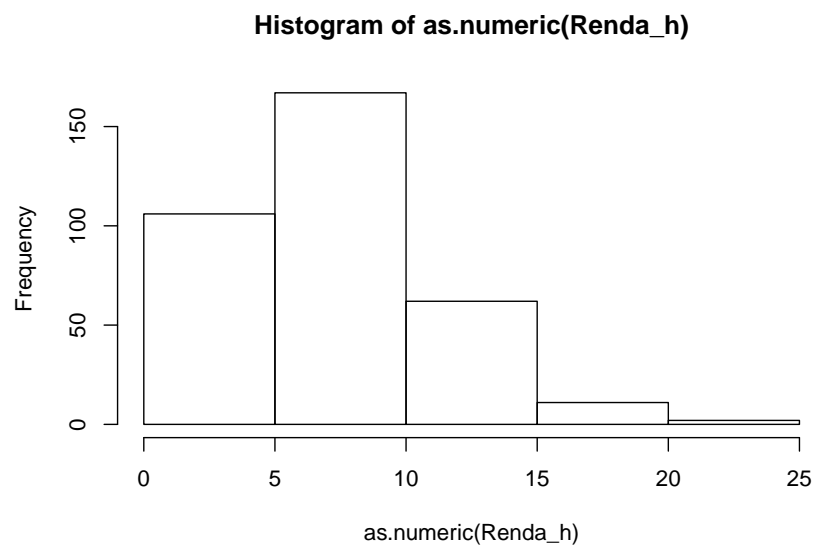
O comando `ylim` determina os limites do eixo y a serem mostrados; `xlab` e `ylab` determinam o nome das variáveis dos eixos x e y; `main` determina o nome do título e `col` determina a cor do gráfico. Use o argumento `main=NULL` para remover o título.

Inserindo as opções `$counts` e `$breaks` retomam-se os valores da contagem dos dados e dos intervalos do histograma:

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

```
[1] 106 167 62 11 2
```

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



```
[1] 0 5 10 15 20 25
```

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 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 fornecem informação tão detalhada quanto os histogramas ou os gráficos ramo-e-folhas, podendo não ser, assim, a melhor escolha quando lida-se 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 explorar um conjunto de dados, é preciso 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)
```

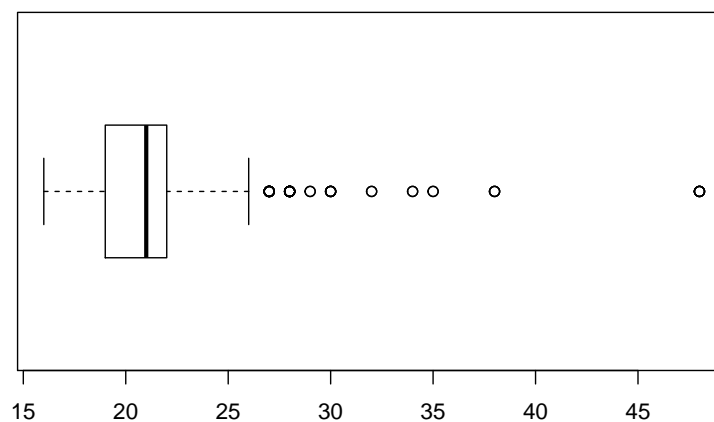


Figura 2.9: Boxplot com a variável Idade

Ainda é possível criar um boxplot analisando a relação da variável contínua de acordo com outras variáveis. Por exemplo, a relação de dispersão dos respondentes relacionando a idade da pessoa com o sabor:

```
boxplot(Idade~Sabor, data=pesquisa_dados)
```

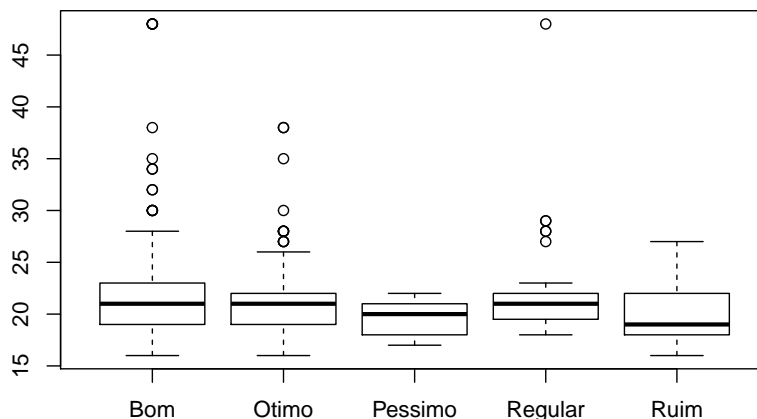


Figura 2.10: Boxplot com as variáveis Idade e Sabor

2.3.5 Gráfico ramo-e-folhas

Em um gráfico ramo-e-folhas, são classificados 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, é possível 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, pode-se 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


```

45 |
46 |
47 |
48 | 00000

```

2.3.6 Gráficos de dispersão

Às vezes dispõe-se de 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

```
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.

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

Fonte: DIEESE (2011).

Para construir um gráfico de linhas, é utilizado o seguinte comando:

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

Neste gráfico, é possível utilizar comandos já inseridos 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:

```
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))
```

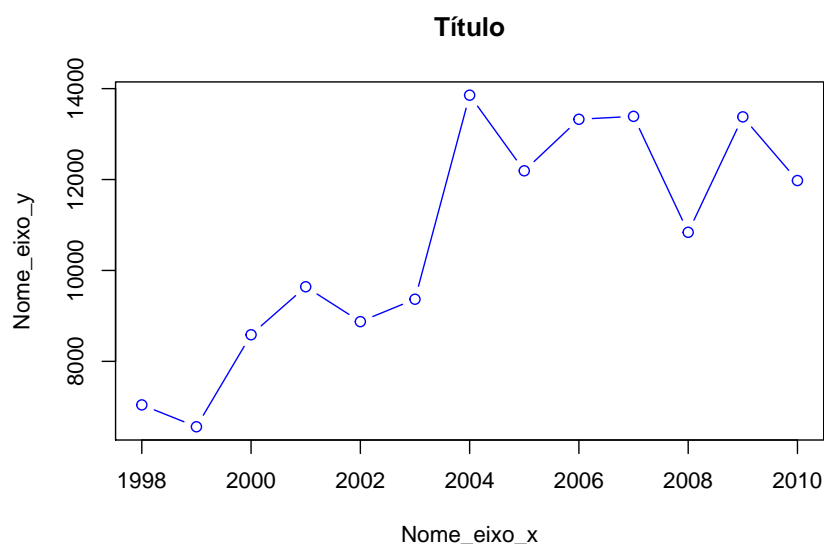


Figura 2.11: Gráfico de linhas sobre a fiscalização do trabalho na área rural Brasil 1998-2010

Fonte: Elaborado pelo(s) autor(es) a partir de DIEESE (2011).

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

Na sequência são adicionadas 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 (2018).

```
library(readr)
inmet <- read_delim("https://goo.gl/2p11WS",
  ";", 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>    <dbl>    <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 metereológica
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)
```

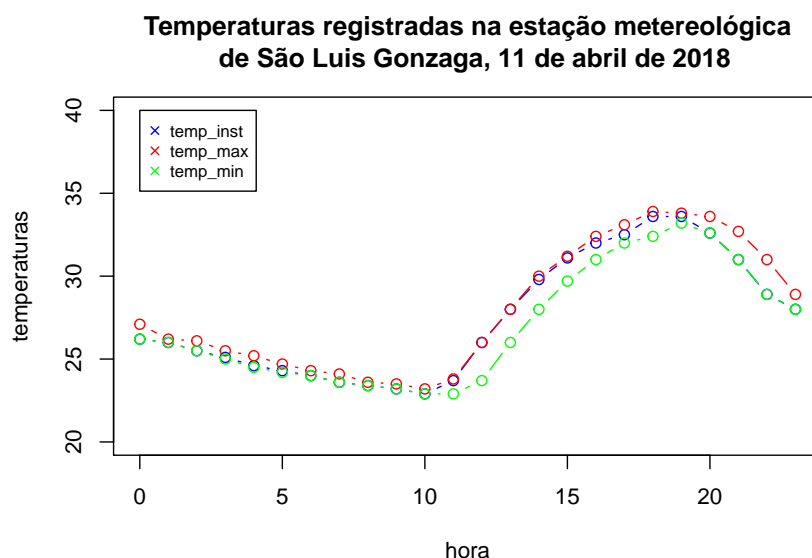


Figura 2.12: Gráfico de linha sobre as temperaturas registradas em São Luiz Gonzaga - RS

Fonte: Elaborado pelo(s) autor(es) a partir de INMET (2018).

2.4 Estatísticas Descritivas

Para determinar o valor máximo de um conjunto de dados, utiliza-se:

```
max(nome_da_variável)
```

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, utiliza-se:

```
min(nome_da_variável)
```

Use a variável **Renda_h**

```
min(Renda_h)
```

```
[1] 1.02
```

Obs.: Para determinar a amplitude total de um conjunto de dados, utiliza-se:

```
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
```

```
95
```

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 é possível fazer cálculos utilizando filtros, simultaneamente. A aplicação de filtros é extremamente útil quando a intenção é de 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, é obtido 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.

2.5 Exercícios

1. Carregue a base de dados denominada “arvores” disponível no site do livro (<https://smolski.github.io/softwarelivrer/livro.html>) e responda as questões abaixo:

1.1 Utilize a função `summary` para identificar os principais indicadores da base de dados. Com a função `table` encontre a contagem das espécies que constam na planilha a partir da variável “Nomecientifico”:

1.2 Utilizando a função `tapply` calcule a média do diâmetro por cada espécie.

1.3 Utilizando as funções `table` e `barplot` construa um gráfico de barras com a quantidade de itens por espécies que constam na base de dados. Não esqueça de utilizar o comando `horiz=TRUE` para melhor visualização.

Capítulo 3

Estatística Inferencial

Tatiane Chassot

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).

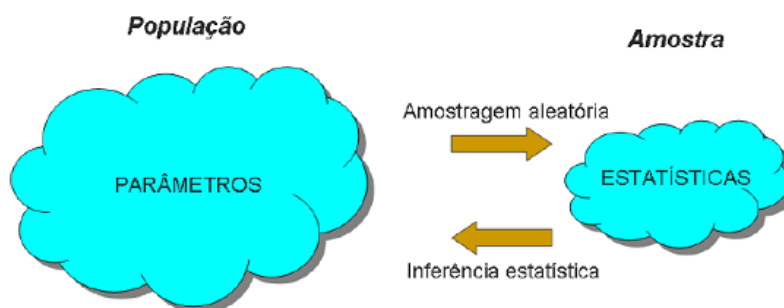


Figura 3.1: Inferência Estatística

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.

3.1 Intervalo de Confiança

Entre as diferentes técnicas de Inferência Estatística, têm-se 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 ≤ μ ≤ 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 for informado o nível de confiança, o software R considera 95%. No entanto, para mudar o nível de confiança para 90%, é acrescentada 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, é possível, inicialmente usar o histograma e depois confirmar com um teste estatístico para testar normalidade como Shapiro-Wilk ou Kolmogorov-Smirnov.

Hipóteses do teste:

- H_0 : os dados seguem uma distribuição normal
- H_1 : os dados não seguem uma distribuição normal

O **valor p** reflete a plausibilidade de se obter tais resultados no caso de H_0 ser de fato verdadeira.

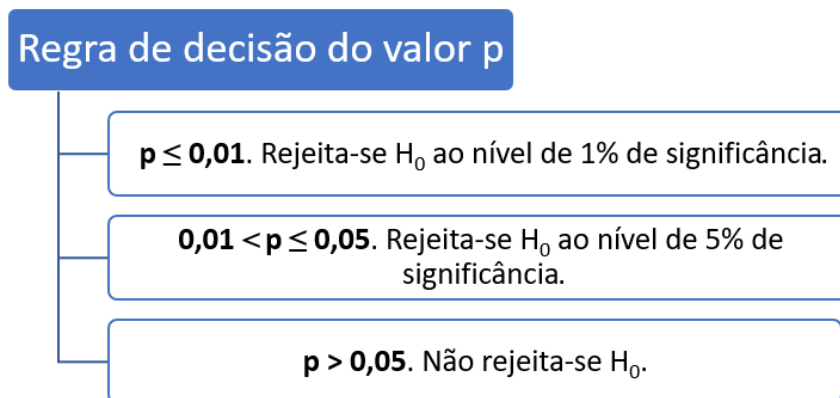


Figura 3.2: Teste de hipóteses

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

Shapiro-Wilk normality test

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

Como $p > 0,05$, não rejeita-se H_0 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-confianca>) 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.

H_0 : hipótese nula (sempre tem a condição de igualdade)

H_1 : hipótese alternativa (tem o sinal de \neq , $>$ ou $<$)

3.2.1 Teste de hipóteses para uma média populacional

$H_0: \mu = \dots\dots$

$H_1: \mu \neq \dots\dots$

$H_0: \mu = \dots\dots$

$H_1: \mu > \dots\dots$

$H_0: \mu = \dots\dots$

$H_1: \mu < \dots\dots$

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 H_0 .

De acordo com as hipóteses, constam 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.

$H_0: \mu = 30 \text{ mm}$

$H_1: \mu \neq 30 \text{ mm}$

```
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 H_0 e conclui-se que a precipitação pluviométrica é igual a 30mm.

Exemplo 4: (adaptado de <https://www.passeidireto.com/arquivo/5533375/lista-esttistica-pronta-p-3-prova-com-respostas/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.

$H_0: \mu = 20$ minutos

$H_1: \mu > 20$ 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 H_0 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 oxi-

gênio 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?

$H_0: \mu = 5$ ppm

$H_1: \mu < 5$ 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

$H_0: \pi = \dots\dots$

$H_1: \pi \neq \dots\dots$

$H_0: \pi = \dots\dots$

$H_1: \pi > \dots\dots$

$H_0: \pi = \dots\dots$

$H_1: \pi < \dots\dots$

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")
```

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?

$$H_0: \pi = 1,5\%$$

$$H_1: \pi > 1,5\%$$

```
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 H_0 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-proporcao>) 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.

$$H_0: \pi = 90\%$$

$$H_1: \pi < 90\%$$

```
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
```

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

Conclusão: Rejeita-se H_0 com nível de significância de 1% e conclui-se que a proporção de médicos que recomendam aspirina é inferior a 90%.

3.2.3 Teste de hipótese para duas médias

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

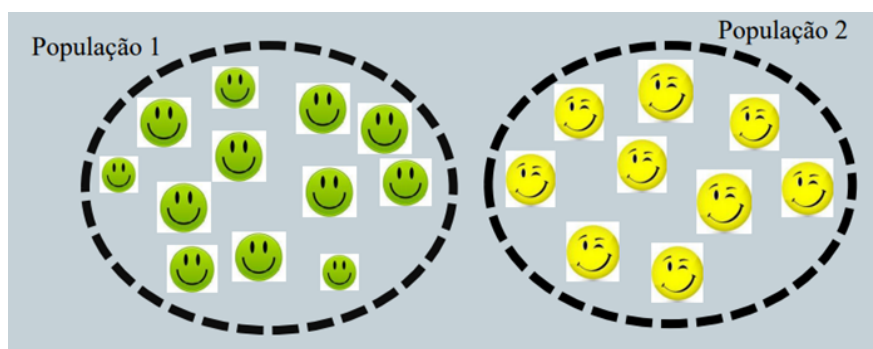


Figura 3.3: Teste de hipótese para dois grupos

É possível 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, é utilizado 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:

H_0 : média antes = média depois

H_1 : 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)
```

Paired t-test

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

```
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. Pode-se concluir que o uso da ração contribuiu para o aumento do peso médio dos animais?

Fonte: <www.inf.ufsc.br/~marcelo/testes2.html>.

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
```

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.

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

3.2.3.2 Teste de hipóteses duas amostras independentes

Primeiramente é preciso 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.

3.2.3.3 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:

Fonte: https://www.ime.unicamp.br/~hildete/Aula_p12.pdf.

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 é possível realizar o teste de comparação de duas amostras independentes.

H_0 : média tecido A = média tecido B

H_1 : 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 H_0 e conclui-se que a média de peso do tecido A é igual à média de peso do tecido B.

3.3 Exercícios

1. A concentração de compostos químicos do solo foi medida em dez amostras aleatórias de um solo de uma área contaminada. A concentração medida, em mg/Kg, foi 1,4 0,6 1,2 1,6 0,5 0,7 0,3 0,8 0,2 e 0,9. Calcule o intervalo de 99% para a média de concentração.

2. Uma pesquisa foi realizada para verificar a satisfação dos alunos de uma escola em relação aos serviços fornecidos pela cantina. Utilizando-se uma amostra aleatória de 25 alunos, as respostas fornecidas foram:

satisfeito	não satisfeito	satisfeito	satisfeito	satisfeito
satisfeito	satisfeito	satisfeito	não satisfeito	satisfeito
não satisfeito	não satisfeito	não satisfeito	satisfeito	satisfeito
não satisfeito	satisfeito	satisfeito	não satisfeito	não satisfeito
não satisfeito	não satisfeito	satisfeito	satisfeito	satisfeito

Construa o intervalo de confiança para a proporção populacional dos alunos satisfeitos considerando um nível de confiança de 90%.

3. (Morettin e Bussab, 2009) Uma companhia de cigarros anuncia que o índice médio de nicotina dos cigarros que fabrica apresenta-se abaixo de 23 mg por cigarro. Um laboratório realiza seis análises desse índice, obtendo: 27, 24, 21, 25, 26, 22. Pode-se aceitar a afirmação do fabricante?

4. (Fonseca e Martins, 2010) As estaturas de 20 recém-nascidos foram tomadas no Departamento de Pediatria da FNRP, cujos resultados em cm são: 41 50 52 49 49 54 50 47 52 49 50 52 50 47 49 51 46 50 49 50. Teste a hipótese de que a média desses recém nascidos é 50 cm.

5. (<https://www.passeidireto.com/arquivo/25297344/aula-19-testes-para-proporcao>) 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.

6. A fim de determinar a eficiência de um medicamento antitérmico, a temperatura corporal (em graus Celsius) de 15 indivíduos foi medida. Em seguida, foi administrado o medicamento e após uma hora a temperatura foi medida novamente. Os resultados podem

ser encontrados na tabela abaixo.

Antes	Depois
37,5	37,8
36,0	36,4
39,0	37,6
38,0	37,2
37,8	36,9
38,5	37,7
36,9	36,8
39,4	38,1
37,2	36,7
38,1	37,3
39,3	38,0
37,5	37,1
38,5	36,6
37,8	35,0
39,0	39,0

Houve ou não diminuição da temperatura dos indivíduos?

7. (http://www.im.ufrj.br/probest/Exercicios/C9_Exercicios.pdf) Foi obtida uma amostra com 20 pilhas elétricas da marca A. Todas elas foram examinadas e sua duração, em horas, foi medida. O mesmo foi feito com uma amostra de 18 pilhas do mesmo tipo, porém da marca B.

Marca A: 176 162 153 137 140 139 165 128 149 148 159 134 173 171 142 142 173 155 157 139

Marca B: 183 196 157 180 188 172 159 184 152 180 169 163 191 151 172 192 121 146

Existe diferença entre as marcas de pilha quanto a sua duração?

Capítulo 4

Teste de Qui-Quadrado

Iara Denise Endruweit Battisti

Quando existem duas variáveis de interesse, a representação tabular das frequências observadas pode ser feita através de uma tabela de contingência, também chamada de tabela cruzada ou tabela de dupla entrada. Cada interseção de uma linha com uma coluna é chamada de casela e o valor que aparece em cada casela é a frequência observada, nomeada como O_{ij} , em que i corresponde a linha e j corresponde a coluna.

4.1 Teste de qui-quadrado para verificar associação entre duas variáveis qualitativas

Exemplo 1: Uma pesquisa sobre “a exposição a agrotóxicos entre trabalhadores rurais no município de Cerro Largo/RS” foi desenvolvida por Letiane Peccin Ristow, no ano de 2017 (dissertação e mestrado no Programa de Pós-Graduação em Desenvolvimento e Políticas Públicas da UFFS, Campus Cerro Largo. Na Tabela 4.1 são apresentados os resultados do “tamanho da propriedade” e “armazenamento seguro do EPI”. Para verificar a existência de associação significativa entre essas duas variáveis é utilizado o teste de qui-quadrado, dado que são duas variáveis qualitativas: variável 1 - tamanho da propriedade (até 25 ha; 26 ha ou mais) e variável 2 – armazenamento seguro (sim; não).

Primeiramente definimos as seguintes hipóteses estatísticas:

H_0 : não existe associação entre tamanho da propriedade e armazenamento seguro (as variáveis são independentes)

H_1 : existe associação entre tamanho da propriedade e armazenamento seguro (as variáveis são dependentes)

Tabela 4.1: Tamanho da propriedade e armazenamento seguro dos agrotóxicos, agricultores de Cerro Largo, RS, 2017.

Tamanho da propriedade	Armazenamento seguro	
	Não	Sim
Até 25 ha	59	8
26 ha ou mais	31	14

Fonte: Ristow (2017).

A estatística de teste para testar as hipóteses apresentadas é o χ^2 (qui-quadrado):

$$\chi_{cal}^2 = \sum_{i=1}^l \sum_{j=1}^c \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

em que:

l : número de linhas

c : número de colunas

O_{ij} : frequência observada na linha i e coluna j

E_{ij} : frequência esperada na linha i e coluna j

com grau de liberdade = $gl = (c - 1)(l - 1)$.

A frequência esperada de uma casela é obtida pela multiplicação do total da linha pelo total da coluna dividido pelo total geral. Por exemplo, a frequência esperada é igual ao total da coluna 1 multiplicada pelo total da linha 1 dividido pelo total geral, ou seja, $(67 \times 90) / 112$.

Porém, é importante conhecer as pressuposições do teste de qui-quadrado de Pearson. Para auxiliar no encaminhamento do teste adequado para verificar a relação de duas variáveis qualitativas, seguimos o seguinte check-list.

4.2 Check list para escolher o teste adequado para verificar a relação entre duas variáveis qualitativas

- O cálculo do teste de qui-quadrado deve ser somente com valores absolutos. Quando se dispõe de uma tabela 2x2, isto é, duas linhas e duas colunas, deve-se utilizar o teste de qui-quadrado com correção de continuidade (correção de Yates). O motivo é que a distribuição de frequências observadas é discreta e está sendo aproximada pela distribuição qui-quadrado, que é contínua (Barbetta, 2010).
- Não se deve aplicar o teste de qui-quadrado quando a frequência esperada em qualquer casela for menor que 5. Neste caso, deve-se usar o teste exato de Fisher, para garantir o grau de certeza do teste.
- Quando se dispõe de duas amostras pareadas (duas amostras dependentes), utiliza-se o teste de McNemar.

- Caso se tenha interesse em avaliar a força da associação entre as duas variáveis, deve-se utilizar algumas medidas de magnitude dessa força, como por exemplo, coeficiente de contingência, razão de prevalência, risco relativo e razão de chances (*odds ratio*). Porém, essas medidas de magnitude são dependentes do tipo de delineamento do estudo.

Para aplicar o teste de qui-quadrado ou um alternativo no software R, primeiramente é preciso informar os dados, é possível efetuar isso de duas formas:

- incluindo os valores no formatado de tabela;
- acessando os valores no banco de dados.

4.3 Exemplo utilizando os recursos do software R

Realizar o teste de associação para os dados da Tabela 4.1, para isso, digitar os dados da tabela cruzada (tabela de contingência) no formato de uma matriz, valor ij , considerando i =linha e j =coluna, em sequência por coluna (por exemplo, digita-se todos os valores da primeira coluna, depois digita-se todos os valores da segunda coluna e assim sucessivamente).

Sintaxe no software R para incluir os valores no formato de tabela:

```
quiquadrado1<-matrix(c(59,31,8,14),nc=2)
quiquadrado1
```

```
      [,1] [,2]
[1,]   59    8
[2,]   31   14
```

O comando `matrix` indica que os dados serão organizados em uma matriz, `nc` indica o número de colunas da tabela, o operador `<-` atribui os valores digitados no nome informado pelo usuário que neste caso é `quiquadrado1`.

O segundo comando `quiquadrado1`, mostra a matriz elaborada, que neste caso representa uma tabela cruzada de duas linhas e duas colunas, conforme a Tabela 4.1.

Primeiramente, deve-se verificar a existência de alguma casela com frequência esperada menor que 5.

```
chisq.test(quiquadrado1)$expected
```

```
      [,1]      [,2]
[1,] 53.84 13.161
[2,] 36.16  8.839
```

Caso não exista, utiliza-se o teste de qui-quadrado com o comando `chisq.test`.

```
chisq.test(quiquadrado1)
```

Pearson's Chi-squared test with Yates' continuity correction

```
data: quiquadrado1
X-squared = 5.1, df = 1, p-value = 0.02
```

Observa-se que o software R identificou a tabela 2x2 e aplicou a correção de continuidade. Porém, é possível informar isso na linha de comando, incluindo opção `correct = TRUE`:

```
chisq.test(quiquadrado1, correct=TRUE)
```

Pearson's Chi-squared test with Yates' continuity correction

```
data: quiquadrado1
```

```
X-squared = 5.1, df = 1, p-value = 0.02
```

Então deve-se concluir pela rejeição ou não da H_0 e interpretar esse resultado.

Caso pelo menos uma casela tenha frequência esperada menor que 5 como por exemplo na tabela abaixo, é utilizado o teste exato de Fisher.

Tabela 4.2: Tamanho da propriedade e devolução das embalagens vazias de agrotóxico, agricultores de Cerro Largo, RS, 2017.

Tamanho da propriedade	Devolução	
	Não	Sim
Até 25 ha	8	59
26 ha ou mais	3	43

Fonte: Ristow (2017).

Definindo as hipóteses estatísticas:

H_0 : não existe associação entre tamanho da propriedade e devolução das embalagens (as variáveis são independentes);

H_1 : existe associação entre tamanho da propriedade e devolução das embalagens (as variáveis são dependentes).

Incluindo os valores:

```
quiquadrado2<-matrix(c(8,3,59,43),nc=2)
quiquadrado2
```

```
 [,1] [,2]
[1,]   8  59
[2,]   3  43
```

Verificando se todas frequências esperadas são maiores ou iguais a 5.

```
chisq.test(quiquadrado2)$expected
```

```
 [,1] [,2]
[1,] 6.522 60.48
[2,] 4.478 41.52
```

Neste caso, o software R apresenta um “aviso” pois observa-se uma frequência espe-

rada menor que 5. Então, se deve optar pelo teste exato de Fisher.

```
fisher.test(quiquadrado2)
```

Fisher's Exact Test for Count Data

```
data: quiquadrado2
p-value = 0.5
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.4316 11.9646
sample estimates:
odds ratio
 1.933
```

É possível concluir, através do valor p , pela rejeição ou não da H_0 e interpretar esse resultados.

4.4 Teste de associação com duas amostras dependentes

No caso de amostras pareadas (dependentes), utiliza-se o teste de McNemar para testar a associação.

```
dados1=matrix(c(5,10,12,8),nc=2)
dados1
```

```
      [,1] [,2]
[1,]    5   12
[2,]   10    8
```

```
mcnemar.test(dados1)
```

McNemar's Chi-squared test with continuity correction

```
data: dados1
McNemar's chi-squared = 0.045, df = 1, p-value = 0.8
```

Importante observar que para executar o teste de McNemar: no software R os dados na matriz (tabela de contingência) devem ser distribuídos da mesma maneira tanto nas linhas quanto nas colunas. Isto é, “a” e “d” devem expressar o mesmo comportamento. Por exemplo: aprovado, desaprovado, aprovado, desaprovado.

Tabela 4.3: Tabela de Contingência.

	Depois	
Antes	Aprovado	Desaprovado

	Depois	
Aprovado	a	b
Desaprovado	c	d

Fonte: Dados simulados.

Exemplo 2: Uma pesquisa foi realizada para verificar o efeito de um medicamento para perda de peso. O estudo foi realizado com 45 cobaias com características semelhantes. Na Tabela abaixo são apresentadas a situação do peso antes e após a intervenção (utilização do medicamento).

Como trata-se de duas amostras dependentes (antes e após) não se deve aplicar o teste de qui-quadrado. O teste adequado é McNemar.

Tabela 4.4: Situação do peso de cobaias do estudo antes e após a intervenção.

	Peso Após	
Peso Antes	Adequado	Sobrepeso
Aprovado	15	5
Desaprovado	18	7

Fonte: Dados simulados.

Hipóteses estatísticas:

H_0 : As frequências das diferentes categorias ocorrem na mesma proporção (Frequências b e c ocorrem na mesma proporção);

H_1 : As frequências b e c ocorrem em proporções diferentes, ou seja, as mudanças são significativas.

```
mcnemar=matrix(c(15,18,5,7),nc=2)
mcnemar
```

```
      [,1] [,2]
[1,]   15    5
[2,]   18    7
```

```
chisq.test(mcnemar)$expected
```

```
      [,1] [,2]
[1,] 14.67  5.333
[2,] 18.33  6.667
```

```
mcnemar.test(mcnemar)
```

McNemar's Chi-squared test with continuity correction

```
data: mcnemar
```

Tabela 4.5: Número de borrachudos nos diferentes pontos

Ponto	Borrachudos
Ponto 1	19
Ponto 2	12
Ponto 3	10
Ponto 4	17
Ponto 5	25
Ponto 6	22
Ponto 7	15

McNemar's chi-squared = 6.3, df = 1, p-value = 0.01

4.5 Teste de qui-quadrado para verificar aderência a uma distribuição

Neste caso é utilizado o teste de qui-quadrado para verificar se o conjunto de dados segue uma distribuição teórica especificada.

Exemplo 3: Deseja-se verificar se o número de borrachudos é o mesmo em diferentes pontos da margem de um rio. O número de borrachudos observados para cada ponto (local) é apresentado na Tabela 4.5.

Fonte: Dados simulados.

Para um nível de 5% de significância, as hipóteses a serem testadas:

H_0 : O número de borrachudos não muda conforme o ponto;

H_1 : Pelo menos um dos pontos tem número de borrachudos diferente dos demais.

```
borrach<-c(20,12,10,17,30,22,35)
chisq.test(borrach)$expected
```

```
[1] 20.86 20.86 20.86 20.86 20.86 20.86 20.86
```

```
chisq.test(borrach)
```

Chi-squared test for given probabilities

```
data: borrach
```

```
X-squared = 24, df = 6, p-value = 6e-04
```

Exemplo 4: Suponha que deseja-se verificar se o número de borrachudos segue uma distribuição específica, informado em “dist”. Lembrando que os valores no vetor “dist” devem estar no formato de proporção (por exemplo, 0,35).

H_0 : O número de borrachudos segue a distribuição teórica informada;

H_1 : O número de borrachudos não segue a distribuição teórica informada.

```

borrachudos<-c(20,12,10,17,30,22,35)
dist<-c(0.10,0.10,0.10,0.15,0.15,0.15,0.25)
chisq.test(borrachudos)$expected

[1] 20.86 20.86 20.86 20.86 20.86 20.86 20.86

chisq.test(borrachudos, p=dist)

```

Chi-squared test for given probabilities

```

data:  borrachudos
X-squared = 8.1, df = 6, p-value = 0.2

```

4.6 Exercícios

1. Uma pesquisa com consumidores foi realizada para verificar a aceitação de um novo produto. Utilize o teste apropriado para avaliar a aceitação do produto em relação ao sexo dos consumidores que participaram da pesquisa.

Tabela 4.6: Aceitação do produto em relação ao sexo dos clientes.

Aceitação	Sexo	
	F	M
Comprariam	43	20
Não Comprariam	37	40

Fonte: Dados simulados.

- 1.1 Apresente as hipóteses estatísticas para esta pesquisa.
- 1.2 Analise o resultado do teste adequado a partir do resultado do software R.
- 1.3 Justifique a escolha do teste utilizado em 1.2.
- 1.4 Apresente o valor p e interprete os resultados do teste considerando nível de 5% de significância.

Capítulo 5

Modelos de Regressão Linear Simples

Iara Denise Endruweit Battisti

Erikson Kaszubowski

Felipe Micaíl da Silva Smolski

Muitas vezes, há a necessidade de estudar duas ou mais variáveis ao mesmo tempo com o objetivo de prever uma variável em função da(s) outra(s). Por exemplo, verificar se sólidos removidos de um material se relaciona com o tempo de secagem e qual é a forma dessa relação. Outros exemplos: relação entre tempo de estudo e desempenho a uma avaliação; relação entre investimento em comunicação e vendas; entre outros.

A análise de correlação permite quantificar a relação linear entre duas variáveis quantitativas. Os modelos de regressão permitem demonstrar a forma da relação entre duas ou mais variáveis. Neste capítulo, serão estudados os modelos de regressão linear simples na qual a variável resposta (Y) é quantitativa e a variável preditora (X_i) é quantitativa ou qualitativa.

5.1 Correlação linear

A correlação linear é a técnica mais simples para estudar a relação entre duas variáveis quantitativas. Os dados compõem uma única amostra de pares de valores (x_i, y_i) , correspondendo aos valores das variáveis X e Y , respectivamente, mensurados em cada elemento de uma amostra ou uma população. Para analisar a existência de relação entre as duas variáveis de forma exploratória, primeiramente pode-se fazer o Diagrama de Dispersão.

5.2 Diagrama de dispersão

O diagrama de dispersão é um gráfico para verificar a existência de relação entre os pares de variáveis X e Y . É composto por pontos, os quais correspondem aos pares de valores (x_i, y_i) , sendo a variável X representada no eixo horizontal e a variável Y representada no eixo vertical.

O diagrama de dispersão fornece uma visualização gráfica do comportamento conjunto das duas variáveis em estudo. Na Figura 5.1a, percebe-se uma correlação (relação) linear positiva entre as variáveis X e Y, ou seja, os valores das duas variáveis crescem conjuntamente. Na Figura 5.1b, percebe-se uma correlação linear negativa entre as variáveis X e Y, neste caso, os valores de uma variável crescem enquanto os valores da outra variável decrescem. A Figura 5.1c informa a ausência de relação entre as duas variáveis e, a Figura 5.1d mostra uma correlação não linear, que não será abordada neste capítulo.

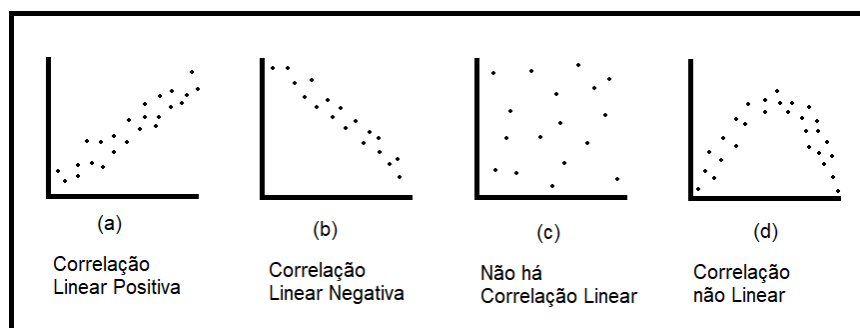


Figura 5.1: Diagramas de Dispersão

Fonte: Elaborado pelo(s) autor(es).

Exemplo: Suponha que 15 alunos foram selecionados aleatoriamente na turma de Estatística, sendo registrado o tempo de estudo e nota da atividade avaliativa. O objetivo da pesquisa é verificar se existe relação entre o tempo de estudo e a nota.

Tabela 5.1: Relação entre o tempo (horas) de estudo e a nota.

Tempo	4,0	6,0	5,5	5,0	6,8	6,5	3,5	4,5	7,5	8,0	5,4	6,5	7,7	7,5	5,8
Nota	5,5	7,5	8,0	7,0	8,1	8,6	4,7	7,5	9,5	9,5	7,8	8,0	9,1	9,5	8,0

Fonte: Dados simulados.

Primeiramente, deve-se digitar os dados para cada variável diretamente na linha de comando ou em um arquivo de dados que será importado. No caso de linha de comando:

```
tempo=c(4,6,5.5,5,6.8,6.5,3.5,4.5,7,8,5.4,6.5,7.7,7.5,5.8)
nota=c(5.5,7.5,8,7,8.1,8.6,4.7,7.5,9.5,9.5,7.8,8,9.1,9.5,8)
```

Para elaborar o diagrama de dispersão o comando utilizado é o `plot` em que y corresponde ao vetor de dados da variável resposta e x corresponde ao vetor de dados da variável preditora.

```
plot(variável_preditora, variável_resposta)
```

Para o exemplo o comando é o seguinte:

```
plot(tempo,nota)
```

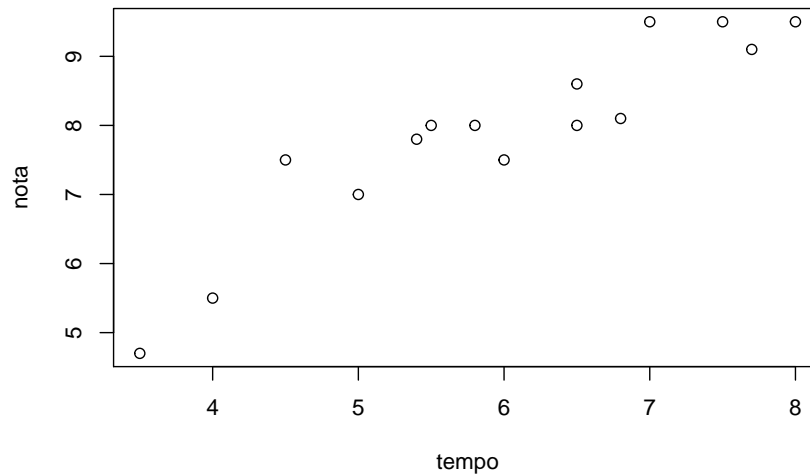


Figura 5.2: Diagrama de dispersão da nota em relação ao tempo de estudo dos participantes do estudo

Fonte: Elaborado pelo(s) autor(es).

Resultando no diagrama de dispersão apresentado na Figura 5.2.

5.3 Coeficiente de Correlação Linear de Pearson

O coeficiente de correlação linear de Pearson (Karl Pearson 1857-1936) mede o grau de relacionamento linear entre os valores pareados x_i e y_i em uma amostra. O coeficiente linear de Pearson é obtido da seguinte forma:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

em que:

- n : número de pares na amostra;
- x : valores da variável x ;
- y : valores da variável y ;
- \bar{x} : média dos valores de x ;
- \bar{y} : média dos valores de y ;

O coeficiente de correlação linear (r) é uma estatística amostral, representando a magnitude da relação entre duas variáveis na amostra. O parâmetro populacional é representado por ρ , que é calculado da mesma forma. O coeficiente de correlação linear assume valores entre -1 e +1, inclusive. Se o valor de r está próximo de 0, conclui-se que não há correlação

linear entre as variáveis X e Y. Se o valor de r está próximo de -1 ou +1, conclui-se pela existência de correlação linear entre as variáveis X e Y, sendo que o sinal indica uma relação linear positiva (direta) ou negativa (inversa).

Sintaxe no software R:

```
cor(variável_preditora, variável_resposta)
```

Lembrando que os valores de x e y são numéricos. No caso do exemplo, segue o comando e a resposta.

```
cor(tempo,nota)
```

```
[1] 0.9224
```

No resultado é apresentado o valor do coeficiente de correlação linear, que neste caso, está próximo de +1, expressando uma correlação forte e direta entre as duas variáveis em análise.

Caso o usuário deseja testar a significância do coeficiente de correlação, isto é, se o resultado do coeficiente de correlação amostral pode ser inferido para a população, utiliza-se o comando:

```
cor.test(variável_preditora, variável_resposta)
```

Para o exemplo:

```
cor.test(tempo, nota)
```

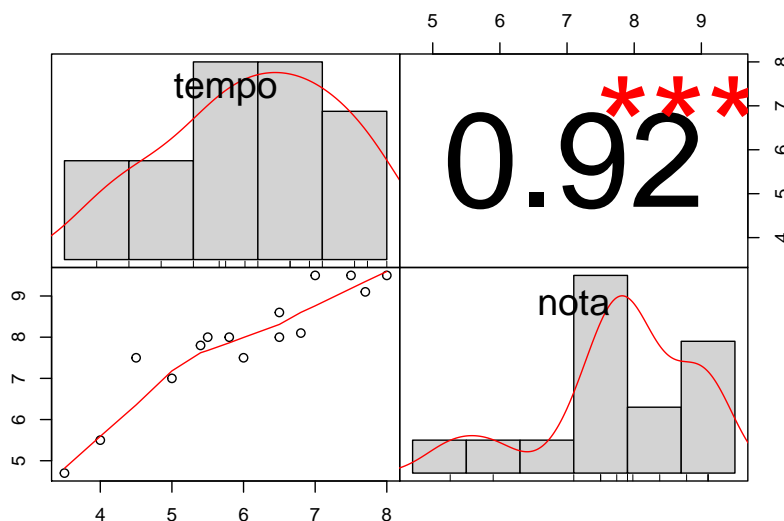
Pearson's product-moment correlation

```
data:  tempo and nota
t = 8.6, df = 13, p-value = 1e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7776 0.9743
sample estimates:
      cor
0.9224
```

Como o valor de p é menor que 0,01 então pode-se afirmar que existe correlação linear positiva significativa, isto é, o resultado pode ser projetado para a população de onde a amostra foi extraída.

Ainda, com o pacote `PerformanceAnalytics` (Peterson e Carl, 2018) é possível aprimorar a exibição dos resultados sobre o coeficiente de correlação com as variáveis observadas, que além dos coeficientes de correlação, mostra a distribuição das variáveis e o gráfico de dispersão:

```
library("PerformanceAnalytics")
chart.Correlation(cbind(tempo,nota), histogram=TRUE, pch=19)
```



5.4 Regressão Linear Simples

O estudo de regressão estabelece-se aos casos em que se pretende estabelecer uma relação entre uma variável Y considerada dependente (variável resposta ou desfecho) e uma ou mais variáveis X_1, X_2, \dots, X_k (variáveis explicativas ou preditoras) consideradas independentes.

O objetivo da análise de regressão é ajustar uma equação que permita explicar o comportamento da variável resposta de maneira que o valor previsto possa estar próximo do que seria observado, dado um conjunto de valores observados para as variáveis preditoras. A forma do modelo de regressão depende da relação entre as variáveis, expressa visualmente pelo diagrama de dispersão, conforme exemplificado na Figura 5.1.

A análise de regressão é uma técnica muito utilizada em variáveis quantitativas, como por exemplo:

- vendas em função do investimento em comunicação;
- altura de crianças em função da idade;
- nota obtida em função de horas de estudo;
- produtividade de uma cultura em relação a quantidade de adubação.

Na Figura 5.3 é apresentada a variação explicada e não explicada na análise por modelo regressão.

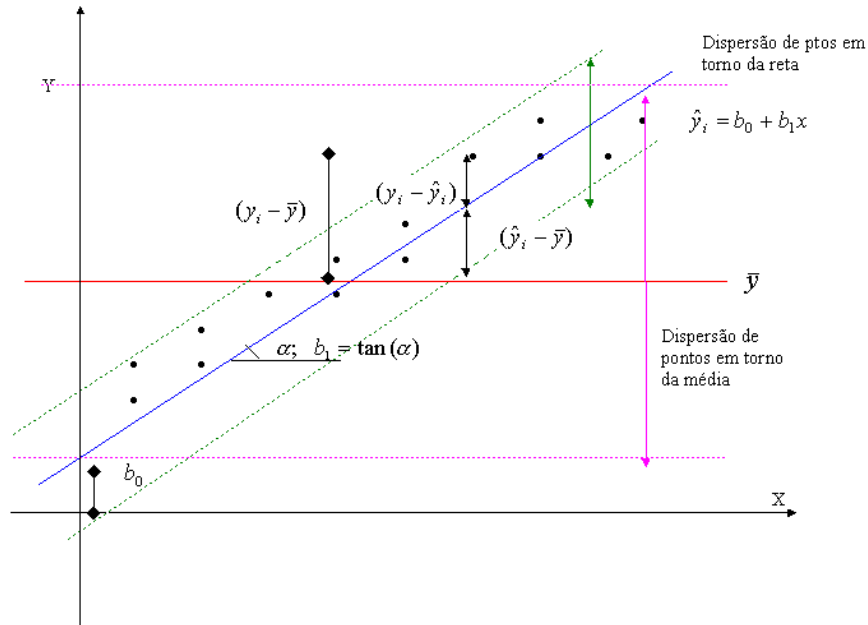


Figura 5.3: Variação explicada e não explicada na análise de regressão

Fonte: Elaborado pelo(s) autor(es).

Observa-se na Figura 5.3, uma identidade na regressão, conforme a seguinte expressão:

$$\sum (y_i - \bar{y}_i)^2 = \sum (\hat{y}_i - \bar{y}_i)^2 + \sum (y_i - \hat{y}_i)^2$$

Soma de Quadrado Total = Soma de Quadrado de Regressão + Soma de Quadrado de Resíduo

Assim, a partir da expressão apresentada e buscando maximizar a variância explicada, o modelo de regressão mais adequado será aquele com maior proporção de “Soma de Quadrado de Regressão” em relação à “Soma de Quadrado Total”, minimizando a “Soma de Quadrado do Resíduo.”

5.5 Modelo de Regressão Linear Simples

O modelo de regressão linear simples é usado quando a resposta da variável dependente se expressa de forma linear (Figura 5.3 e neste caso com apenas uma variável explicativa, expresso da seguinte maneira (Hoffmann e Vieira, 1998):

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

Em que:

y_i : valores da variável resposta (dependente, desfecho), $i = 1, 2, \dots, n$ observações;

x_i : valores da variável explicativa (independente, preditora), $i = 1, 2, \dots, n$ observações;

β_0 : coeficiente linear (intercepto). Interpretado como o valor da variável dependente quando a variável independente é igual a 0;

β_1 : coeficiente angular (inclinação). Interpretado como acréscimo ou decréscimo na variável dependente para a variação de uma unidade na variável independente;

ε_i : erros aleatórios costumeiramente assumidos como provenientes de uma população normal, com média 0 e variância constante $[\varepsilon_i \sim N(0, \sigma^2)]$.

5.6 Método dos Mínimos Quadrados

O método dos mínimos quadrados (MMQ) é utilizado para a obtenção dos coeficientes linear e angular. Consiste em minimizar a Soma de Quadrados de Resíduos, ou seja, minimizar:

$$\sum (y_i - \hat{y}_i)^2 = \sum (y_i - b_0 - b_1 x_i^2)$$

As expressões para os coeficientes, que minimizam a Soma de Quadrado dos Resíduos são obtidas pela derivada desta soma de quadrados em relação a b_0 e em relação a b_1 e podem ser descritas por (Hoffmann e Vieira, 1998):

$$b_1 = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \frac{(\sum x)^2}{n}}$$

em que:

n : número de pares na amostra;

x : valores da variável x ;

y : valores da variável y .

e

$$b_0 = \bar{y} - b_1 \bar{x}$$

em que:

\bar{x} : média dos valores de x ;

\bar{y} : média dos valores de y ;

b_1 : valor calculado do coeficiente angular.

Obtendo-se a seguinte equação de regressão linear simples estimada:

$$\hat{y} = b_0 + b_1 x$$

em que:

b_0 : coeficiente linear estimado;

b_1 : coeficiente angular estimado;

x : valores da variável explicativa.

Esta equação refere-se a reta de regressão, sendo que se b_1 é um valor positivo a reta é crescente, demonstrando uma relação positiva entre as variáveis; mas se b_1 é um valor negativo, a reta é decrescente, demonstrando uma relação inversa entre as variáveis.

No software R utiliza-se o comando `lm` para executar a análise de regressão linear, em que `y` corresponde aos valores numéricos da variável resposta e `x` são valores numéricos da variável preditora. No caso do segundo comando, “base” corresponde ao nome da base de dados em que estão armazenadas as variáveis. Lembrando que “regressao” é o nome fornecido pelo usuário.

```
nome_para_regressao=lm(variável_dependente ~ variável_preditora)
```

Por exemplo:

```
regressaolinear=lm(nota~tempo)
regressaolinear
```

Call:

```
lm(formula = nota ~ tempo)
```

Coefficients:

(Intercept)	tempo
2.221	0.947

No resultado observa-se o valor do coeficiente linear (intercept) igual a 2,2214 e o valor do coeficiente angular (tempo) igual a 0,947 interpretando que a cada aumento de uma unidade de tempo (hora), a nota do aluno aumenta, em média, 0,947 pontos.

5.7 Análise de Variância

A Análise de Variância, técnica introduzida por Fisher, na década de 20, testa o ajuste da equação como um todo, ou seja, um teste para verificar se a equação de regressão obtida pode ser exclusivamente fruto do erro amostral em uma situação em que a variável preditiva não possui nenhuma relação linear com o desfecho. Isto é, testar a significância da equação ajustada.

As hipóteses testadas na Análise de Variância da Regressão são:

$$H_0 : \beta_1 = 0 \text{ (a regressão não é significativa)}$$

$$H_1 : \beta_1 \neq 0 \text{ (a regressão é significativa)}$$

No caso de regressão linear simples, a análise de variância é definida como apresentada na Tabela 5.2.

Tabela 5.2: Análise de variância para a regressão linear simples.

FV	GL	SQ	QM	F
Regressão	1	SQRegressão	QMRegressão	Fc
Desvios	n-2	SQResíduos	QMResíduos	-
Total	n-1	SQTotal	-	-

Fonte: Elaborado pelo(s) autor(es).

No qual:

FV: Fonte de Variação;

GL: Grau de Liberdade;

SQ: Soma de Quadrados;

QM: Quadrados Médios;

F: F calculado.

Em que:

$$SQRegressão = \frac{(\sum xy - \frac{(\sum x \sum y)^2}{n})}{\sum x^2 - \frac{(\sum x)^2}{n}}$$

$$SQTotal = \sum y^2 - \frac{(\sum y)^2}{n}$$

$$SQResíduo = SQTotal - SQRegressão$$

$$QMRegressão = SQRegressão / GLRegressão$$

$$QMResíduo = SQResíduo / GLResíduo$$

$$Fc = QMRegressão / QMResíduo$$

Espera-se que o QMResíduo seja mínimo, assim o modelo de regressão estará bem ajustado.

A distribuição de probabilidade para a razão de duas variâncias é conhecida como a distribuição F. Se a hipótese nula for rejeitada ao nível de significância α (rejeita-se H_0) e, portanto a regressão é significativa ao nível α de significância.

No software R, utiliza-se a função `anova()` para obter a análise de variância informando o nome dado ao modelo de regressão previamente.

```
anova(nome_para_regressão)
```

Por exemplo:

```
anova(regressaolinear)
```

Analysis of Variance Table

Response: nota

```
Df Sum Sq Mean Sq F value Pr(>F)
```



```
tempo      1    22.8    22.82    74.2 9.9e-07 ***
Residuals 13     4.0     0.31
```

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

No resultado observam-se as fontes de variação: tempo (variável preditora) e residuals (resíduos); graus de liberdade (Df), soma de quadrado (Sum Sq), quadrado médio (Mean Sq), valor do F calculado (F value) e o valor p (Pr). Neste caso, como $p < 0,01$ rejeita-se H_0 ao nível de 1% de significância e, portanto a equação é significativa ($p < 0,01$).

5.8 Coeficiente de Determinação

O coeficiente de determinação representa o percentual de variação total que é explicada pela equação de regressão, sendo obtido da seguinte forma:

$$R^2 = \frac{SQ_{Regressão}}{SQ_{Total}}$$

Quanto mais próximo de 1 (ou 100%) for o R^2 , melhor será o ajuste da equação de regressão. Também, é possível utilizar o coeficiente de determinação ajustado (R^2 ajustado), o qual considera o número de variáveis e o tamanho da amostra, sendo que este é o mais indicado para regressão múltipla. No software R, o valor do coeficiente de determinação é obtido pelo comando `summary()`, informando o nome dado ao modelo de regressão previamente, conforme segue:

```
summary(nome_para_regressao)
```

Por exemplo:

```
summary(regressaolinear)
```

Call:

```
lm(formula = nota ~ tempo)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-0.8372 -0.4109  0.0418  0.3733  1.0154
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.221      0.673     3.30  0.0057 **
tempo           0.947      0.110     8.61  9.9e-07 ***
```

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 0.555 on 13 degrees of freedom

Multiple R-squared: 0.851, Adjusted R-squared: 0.839

F-statistic: 74.2 on 1 and 13 DF, p-value: 9.88e-07

No resultado, observa-se o valor de coeficiente de determinação (multiple R-squared) igual a 0,85, indicando que 85% da variação da nota (variável resposta) é devido a variação do tempo de estudo (variável preditora).

A reta de regressão pode ser visualizada no diagrama de dispersão com o comando `abline()`, como segue:

```
abline(nome_para_regressao)
```

Para o exemplo:

```
plot(nota~tempo)
abline(regressaolinear)
```

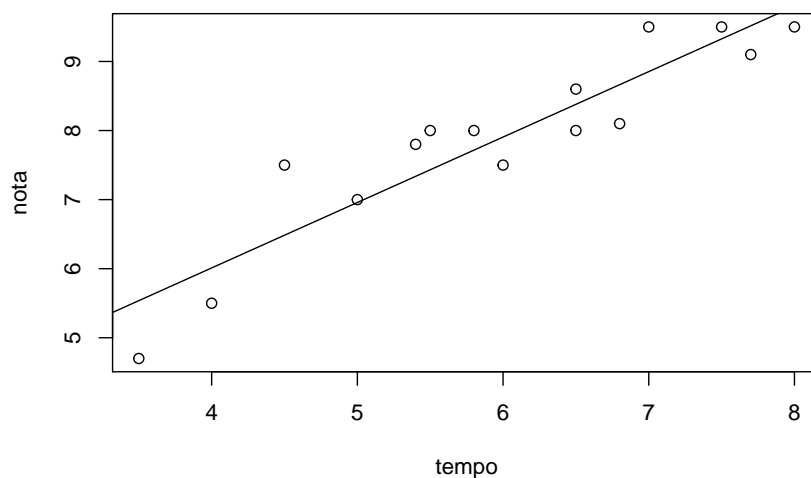


Figura 5.4: Reta de regressão ajustada da nota em relação ao tempo de estudo dos participantes da pesquisa

Fonte: Elaborado pelo(s) autor(es).

O intervalo de 95% de confiança para os coeficientes de regressão são obtidos, no software R, pelo comando `confint()`, da seguinte forma:

```
confint(nome_para_regressao)
```

Para o exemplo:

```
confint(regressaolinear)
```

	2.5 %	97.5 %
(Intercept)	0.7671	3.676
tempo	0.7097	1.185

5.9 Intervalo de Predição

Após o ajuste da equação de regressão linear simples, verificada a significância da equação ($p < 0,05$). No caso da equação estimada se ajusta bem aos dados pelo valor do coeficiente de determinação então é possível prever valores da variável Y (resposta) a partir de valores da variável X (explicativa). Caso a regressão não seja significativa a melhor predição para a variável Y é média dos valores de y , ou seja, \hat{y} .

A predição de valores só tem sentido nos seguintes casos:

- regressão significativa;
- os valores de X devem estar dentro dos limites inferior e superior dos dados amostrais;
- as inferências referem-se somente a população de onde a amostra aleatória foi extraída;
- as suposições sobre os resíduos devem ser satisfeitas.

Quando tem-se uma equação estimada do tipo $\hat{y} = b_0 + b_1x$, \hat{y} representa o valor predito da variável Y para um dado valor da variável X, ou seja, é uma predição pontual, porém esta não informa a sua precisão, a qual é contemplada no intervalo de predição. O intervalo de predição para um determinado Y é dado por:

$$\hat{y} \pm \varepsilon$$

em que:

$$\varepsilon = t_{(n-2; \frac{\alpha}{2})} \cdot S_e \cdot \sqrt{1 + \frac{1}{n} + \frac{n(x_p - \bar{x})^2}{n(\sum x^2) - (\sum x)^2}}$$

onde:

x_p : o valor dado para x

S_e : o erro padrão da estimativa, definido por:

$$S_e = \sqrt{\text{QMResíduo}} = \sqrt{\frac{\sum (y - \hat{y})^2}{n - 2}}$$

Assim, obtêm-se o intervalo de predição para um determinado Y, que também pode ser expresso da seguinte forma:

$$(\hat{y} - \varepsilon; \hat{y} + \varepsilon)$$

A sintaxe do comando é:

```
x0=data.frame(x=valor_numérico)
predict(regressao,x0,interval="prediction")
```

No comando apresentado o x0 recebe o valor de x e predict calcula o intervalo de predição para o valor de Y dado x0.

Para o exemplo:

```
x0=data.frame(tempo=5.5)
predict(regressaolinear, x0, interval="prediction")
```

```
      fit    lwr    upr
1 7.432 6.189 8.675
```

5.10 Análise dos Resíduos

O resíduo da análise de regressão é a diferença do valor do Y observado e Y estimado referente a cada par de valores do conjunto de dados, isto é, $E_i = Y_i - \hat{Y}_i$.

No software R, pode-se utilizar o comando “residuals”, para visualizar os resíduos, lembrando que “regressaolinear” é o nome dado ao modelo executado, como segue:

```
residuals(regressaolinear)
```

A análise dos resíduos é importante para a validade dos intervalos de confiança e testes de hipóteses, uma vez que as suposições das observações de Y independentes e o erro adere a distribuição aproximadamente normal com média 0 e variância constante devem ser satisfeitas.

O método gráfico pode ser utilizado para testar estas suposições dispondo os valores da variável preditora no eixo x e os respectivos valores dos resíduos no eixo y. Ainda, pode-se dispor os valores ajustados no eixo x e os respectivos valores dos resíduos do eixo y.

Se o modelo ajustado for apropriado para os dados, os pontos devem estar distribuídos de forma aleatória no gráfico dos resíduos, conforme Figura 5.5a. Caso a suposição não seja satisfeita, métodos alternativos podem ser utilizados como: método dos mínimos quadrados ponderados para o caso de não homocedasticidade; o método dos mínimos quadrados generalizados para o caso de erros correlacionados; e, métodos não-paramétricos para o caso de não normalidade.

Além da análise gráfica, existem testes para avaliar a homocedasticidade como o Teste de Bartlett e para avaliar a normalidade aplicam-se os testes de Shapiro Wilks ou Kolmogorov-Smirnov.

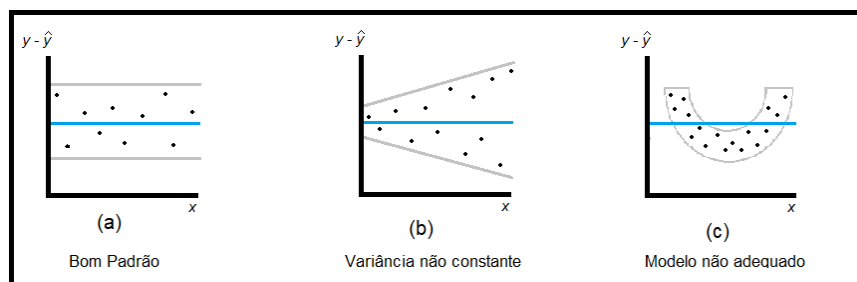


Figura 5.5: Gráficos para análise de resíduos em regressão

Fonte: Elaborado pelo(s) autor(es).

A seguir é apresentado o comando do gráfico de resíduos apresentando os valores ajustados pela equação de regressão e os resíduos:

```
plot(fitted(nome_para_regressao), residuals(nome_para_regressao),
     xlab="Valores ajustados", ylab="Resíduos")
abline(h=0)
```

Nesta sintaxe, o termo “nome_para_regressao” é o nome dado ao modelo de regressão, “fitted” define os valores ajustados no eixo horizontal, “residuals” define os resíduos no eixo vertical, “xlab” indica o rótulo do eixo horizontal e “ylab” indica o rótulo do eixo vertical. e “abline(h=0)” apresenta uma linha constante em $y = 0$ para facilitar a visualização dos desvios dos resíduos.

Na Figura 5.6 é apresentado o gráfico de resíduo do exemplo, no qual os resíduos são apresentados no eixo y e os valores ajustados são apresentados no eixo x. Lembrando que “regressaolinear” é o nome definido para a regressão do exemplo.

```
plot(fitted(regressaolinear), residuals(regressaolinear),
     xlab="Valores ajustados", ylab="Resíduos")
abline(h=0)
```

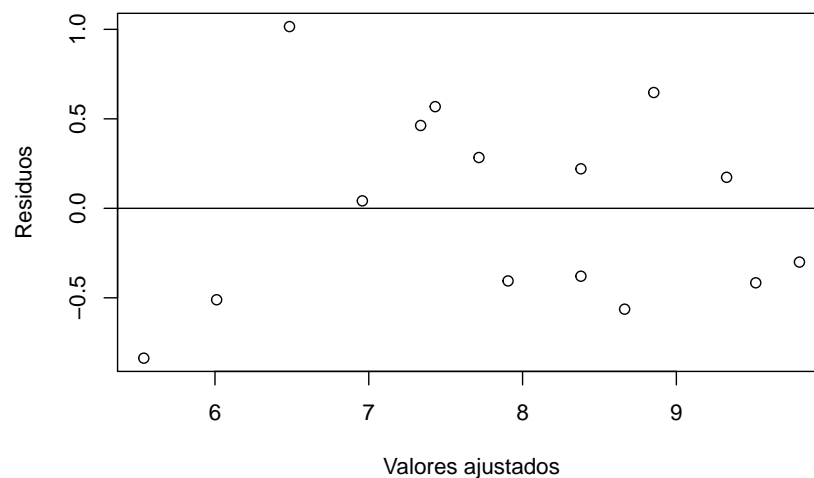


Figura 5.6: Gráfico dos resíduos em relação aos valores ajustados para os dados do exemplo

Fonte: Elaborado pelo(s) autor(es).

Outro gráfico de resíduos que é possível elaborar na análise de resíduos representa a variável preditora (X) no eixo X e o resíduos no eixo Y, com a seguinte sintaxe:

```
plot(tempo, residuals(nome_para_regressao),
     xlab="Variável Preditora", ylab="Resíduos")
```

Para os dados do exemplo:

```
plot(tempo, residuals(regressaolinear),
     xlab = "Tempo",
```

```
ylab="Resíduos")
abline(h=0)
```

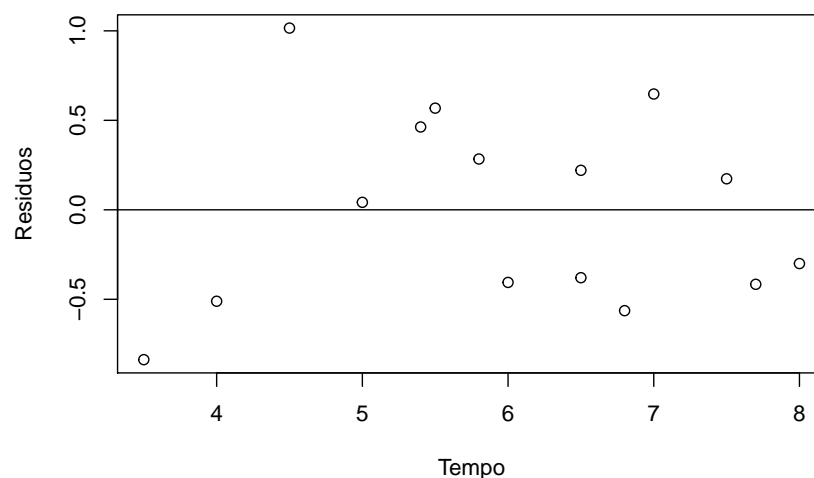


Figura 5.7: Gráfico gerado para análise dos resíduos com os valores da variável preditora

Fonte: Elaborado pelo(s) autor(es).

O resultado desse comando é apresentado na Figura 5.7 em que no eixo y constam os valores dos resíduos e no eixo x constam os valores da variável independente.

Considerando os dados do exemplo, suponha que um aluno estudou 6,5 horas ($x=6,5$), então o valor ajustado da nota (y) é dado por $2,2214+0,9474*6,5$, resultando em 8,38. Para esse caso, o resíduo é:

$$Y_{\text{observado}} - Y_{\text{estimado}} = 8 - 8,38 = -0,38$$

Para exibir os resíduos e os valores ajustados da equação de regressão utilizam-se os seguintes comandos, respectivamente:

```
residuals(regressaolinear)
fitted(regressaolinear)
```

Para testar a suposição que os erros aleatórios têm distribuição normal, pode-se elaborar o gráfico de probabilidade normal, conforme segue:

```
qqnorm(residuals(nome_para_regressao))
```

Para o exemplo, o comando é o seguinte:

```
qqnorm(residuals(regressaolinear))
```

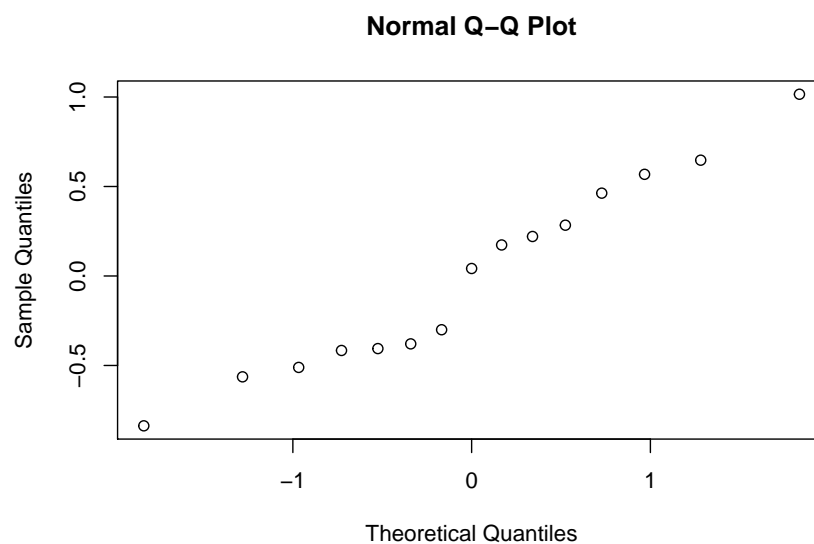


Figura 5.8: Gráfico de probabilidade normal para verificar normalidade dos resíduos

Fonte: Elaborado pelo(s) autor(es).

E o resultado é apresentado na Figura 5.8.

Ainda, pode-se construir o gráfico com a distribuição da probabilidade dos resíduos, através de um histograma, verificando assim se a cauda é simétrica ou não:

```
hist(x = regressaolinear$residuals,  
     xlab = "Resíduos",  
     ylab = "Densidade",  
     main = "",  
     col = "lightgreen",  
     probability = TRUE)  
lines(density(regressaolinear$residuals))
```

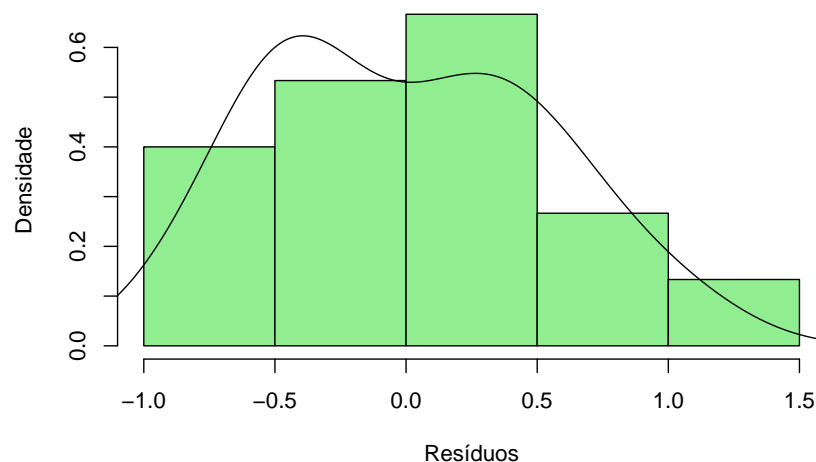


Figura 5.9: Histograma de distribuição da probabilidade para os resíduos

Fonte: Elaborado pelo(s) autor(es).

O resultado desse comando é apresentado na Figura 5.9.

Também, pode-se aplicar o teste de normalidade de Shapiro-Wilk para verificar a normalidade dos resíduos. O comando utilizado é o seguinte:

```
shapiro.test(residuals(nome_para_regressao))
```

Para o exemplo:

```
shapiro.test(residuals(regressaolinear))
```

Shapiro-Wilk normality test

```
data: residuals(regressaolinear)
```

```
W = 0.96, p-value = 0.6
```

Conclui-se que os resíduos são normais se o valor de $p \geq 0,05$.

5.10.1 Valores outliers na regressão

Para análise dos valores outliers nos resíduos (*residuals standard* e *residuals studentized*), utilizam-se os seguintes comandos:

```
rstudent(nome_para_regressao)
```

```
rstandard(nome_para_regressao)
```

Para o exemplo:

```
rstudent(regressaolinear)
```

1 2 3 4 5 6 7 8


```
-1.04742 -0.74389  1.07142  0.07646 -1.07311  0.40066 -2.01860  2.29138
      9      10      11      12      13      14      15
1.26283 -0.60069  0.86125 -0.69777 -0.81958  0.32859  0.51493
```

```
rstandard(regressaolinear)
```

```
      1      2      3      4      5      6      7      8
-1.04353 -0.75701  1.06538  0.07956 -1.06691  0.41426 -1.81531  1.98916
      9      10      11      12      13      14      15
1.23490 -0.61602  0.86993 -0.71196 -0.83013  0.34048  0.53013
```

E o gráfico para verificar valores outliers nos resíduos:

```
plot(rstudent(nome_para_regressao))
```

```
plot(rstandard(nome_para_regressao))
```

Os gráficos dos resíduos padronizados (standard) e studentizados (student) para o exemplo estão apresentados nas Figuras 5.10 e 5.11, respectivamente, utilizando os comandos que segue:

```
plot(rstandard(regressaolinear))
abline(h=2,col="red")
abline(h=-2,col="red")
```

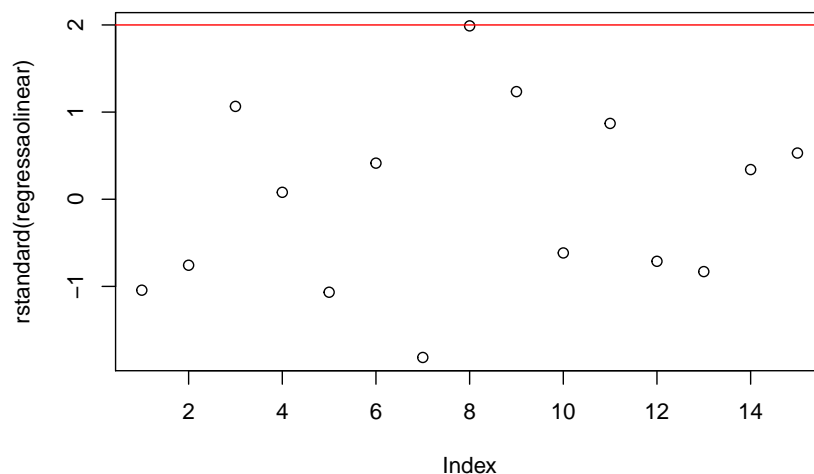


Figura 5.10: Resíduos padronizados para o exemplo

Fonte: Elaborado pelo(s) autor(es).

Aqueles valores fora do intervalo $(-2, +2)$ são possíveis outliers.

```
plot(rstudent(regressaolinear))
abline(h=2,col="red")
abline(h=-2,col="red")
```

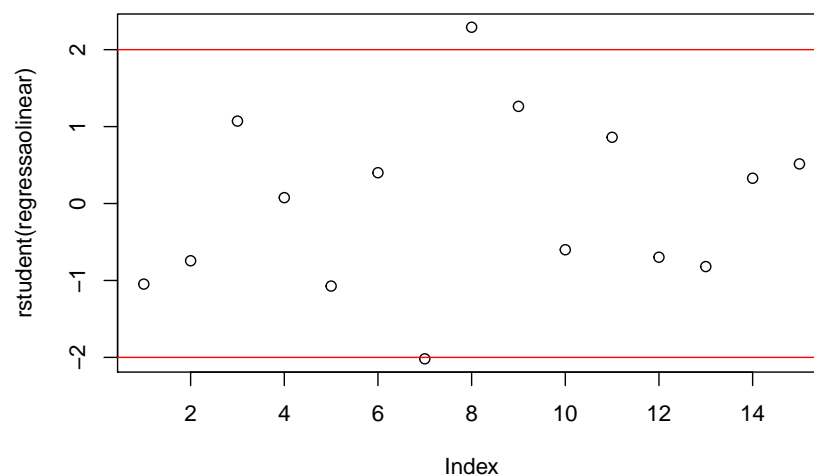


Figura 5.11: Resíduos studentizados para o exemplo

Fonte: Elaborado pelo(s) autor(es).

5.10.2 Valores influentes na regressão

Para análise dos valores influentes, utiliza-se:

```
dffits(nome_para_regressao)
```

Para os dados do exemplo:

```
dffits(regressao linear)
```

1	2	3	4	5	6	7	8
-0.55767	-0.19884	0.30669	0.02611	-0.34386	0.11597	-1.34854	0.97320
9	10	11	12	13	14	15	
0.43848	-0.32566	0.25379	-0.20196	-0.38792	0.14210	0.13902	

Aqueles valores maiores que $2 * (p/n)^{(1/2)}$ são possíveis pontos influentes. Em que, p = número de parâmetros do modelo e n = tamanho da amostra.

Para esse exemplo:

```
2*(2/15)^(1/2)
```

```
[1] 0.7303
```

O gráfico para detectar pontos influentes para os dados do exemplo:

```
plot(dffits(regressao linear))
abline(h=-0.73,col="red")
abline(h=0.73,col="red")
```

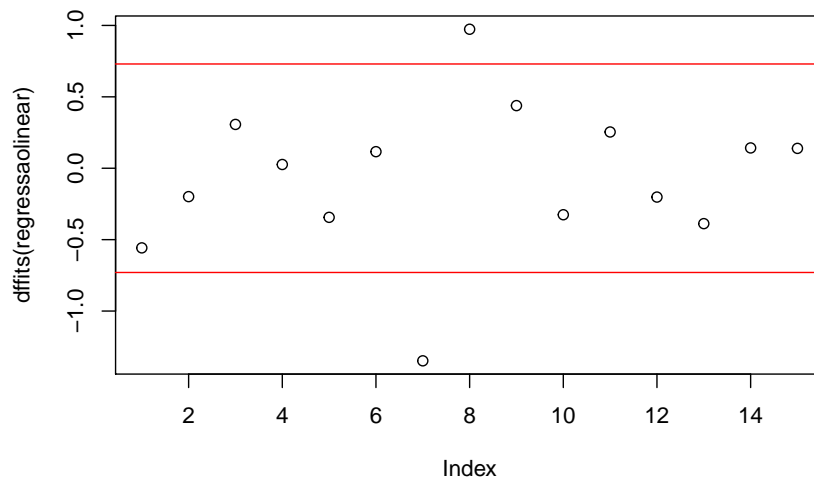


Figura 5.12: Pontos influentes para o exemplo

Fonte: Elaborado pelo(s) autor(es).

O comando `plot(nome_para_regressao)` elabora diferentes gráficos para o diagnóstico do modelo.

5.11 Exercícios

1. No site do livro (<https://smolski.github.io/softwarelivrer/livro.html>) está disponível uma planilha de dados com o nome “peixes1”, na primeira coluna consta a quantidade de ovos (m^3) e na segunda coluna consta a quantidade de oxigênio no rio. O objetivo da pesquisa é comparar alguns ambientes do rio sobre a desova e o crescimento das larvas de peixes. Analisar a relação da quantidade de ovos com a quantidade de oxigênio no rio. Para isso utilize coeficiente de correlação linear e regressão linear simples.

2. Baixe no site https://www.openintro.org/stat/data/?data=gun_violence_us a base de dados `gun_violence_us.csv` (GUN VIOLENCE IN THE UNITED STATES), apresentando o relacionamento entre propriedade de armas e violência nos Estados Unidos. A variável dependente é a taxa de propriedade de armas (`ownership_rate`: percentual de adultos em cada estado que é proprietário de uma arma em 2013) e a variável independente é a taxa de mortalidade (`mortality_rate`: número de mortes por 100.000 em cada estado em 2014).

2.1 Faça um diagrama de dispersão para visualizar a relação da taxa e ano. O que você pode concluir?

2.2 Calcule o coeficiente de correlação linear. Conclua sobre ele.

2.3 Encontre e interprete a equação ajustada.

2.4 Apresente e interprete o coeficiente de determinação (R^2).

- 2.5** Teste a significância da equação de regressão através da ANOVA.
- 2.6** Faça o intervalo de predição para $x=0.50$.
- 2.7** Trace a reta de regressão ajustada no diagrama de dispersão.
- 2.8** Faça análise de resíduos.

Capítulo 6

RMarkdown

Felipe Micaíl da Silva Smolski

Markdown é uma linguagem de marcação de textos utilizada para a criação de diversos documentos, incluindo artigos, livros e apresentações. A grande inovação do **RMarkdown** no RStudio neste sentido é a utilização desta linguagem por meio do pacote **rmarkdown** (arquivos .Rmd) para integrar a criação de documentos com a análise e manipulação de dados em um único documento (Figura 6.1). Desta forma, é possível efetuar pesquisas científicas que podem ser reproduzidas de forma muito mais fácil.



Figura 6.1: Processo de criação de documentos no RMarkdown

Fonte: Adaptado de Allaire *et al.* (2017).

Para criação dos documentos é preciso a instalação dos pacotes denominados **rmarkdown** (Allaire *et al.*, 2017) e **knitr** (Xie, 2018) dentro do RStudio, bem como sugere-se a instalação, no Windows, do programa MiKTeX (<https://miktex.org/download>), que se encarrega de suporte à configurações da linguagem de marcação de textos LaTeX no caso de criação dos arquivos PDF.

6.1 Criando o documento

Para criação do documento RMarkdown, no RStudio clique em “File > New File > R Markdown”, ou mesmo através do atalho para criação de documentos conforme mostra a Figura 6.2. Haverá a escolha entre a criação de documentos (HTML, PDF e Word/Libre/Open Office), a criação de uma apresentação (*Presentation*), a criação de um documento Shiny (do-

cumento dinâmico para criação de *dashboards*) e o carregamento de um modelo de documento pré-estabelecido (*From Template*).

Neste exemplo será criado um documento em Word, onde são preenchidos os campos com o título do documento, o nome do autor e escolha o tipo de documento.

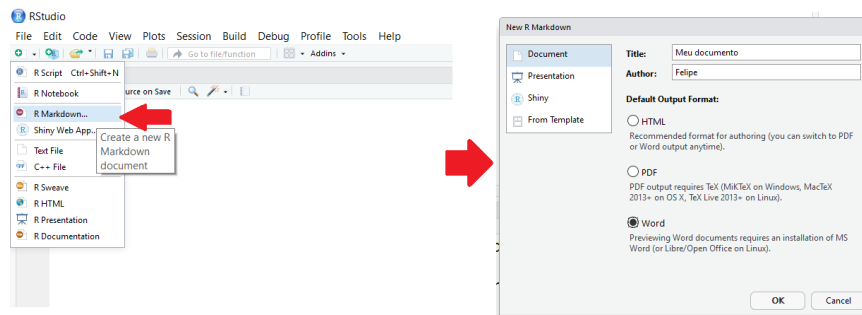


Figura 6.2: Criar documento RMarkdown

Fonte: Elaborado pelo(s) autor(es).

6.2 Compilando os resultados do arquivo

O **RMarkdown** cria um documento inicial padrão, contendo alguns exemplos básicos de inserção de textos e de formatação, que serão vistos adiante. Para compilação do documento para o formato desejado (neste caso Word), o usuário deve clicar na aba “Knit > Knit to Word”, ou pelo atalho no teclado CTRL+SHIFT+K.

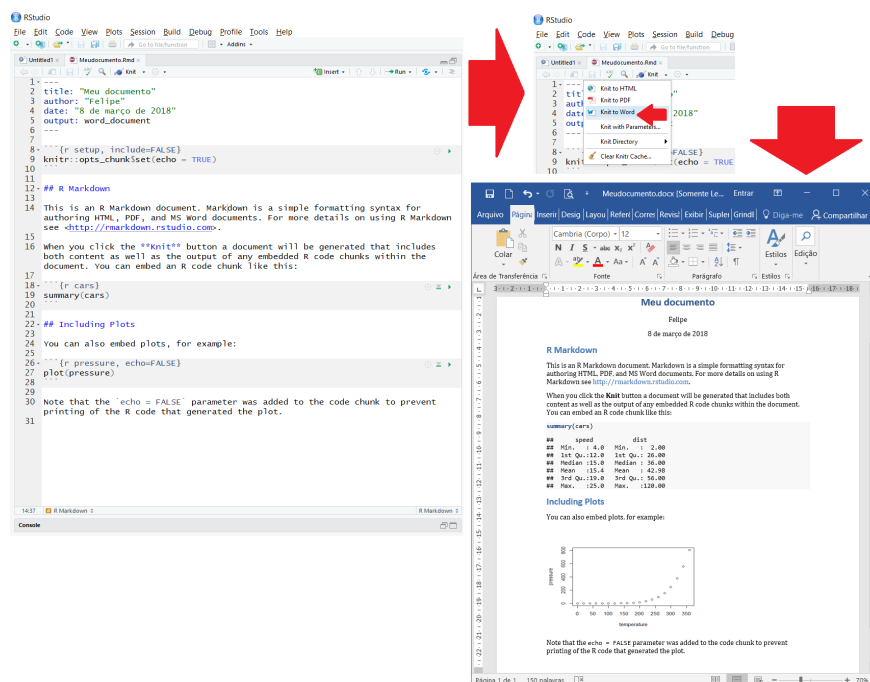


Figura 6.3: Compilando o documento RMarkdown

Fonte: Elaborado pelo(s) autor(es).

Caso ocorram erros com relação à codificação do documento, no que diz respeito aos caracteres de acentuação da língua portuguesa, este pode ser resolvido salvando o documento criado com a codificação UTF-8. Para isto, clique em “File > Save with Encoding > UTF-8”. Deve ser feito este procedimento para cada tipo de arquivo: Word, HTML e PDF.

```
1 ---
2 title: "Meu documento"
3 author: "Felipe"
4 date: "8 de março de 2018"
```

Figura 6.4: Erro de codificação do documento RMarkdown

Fonte: Elaborado pelo(s) autor(es).

6.3 Elementos básicos do RMarkdown

A configuração básica de um arquivo RMarkdown divide-se entre a YAML Header e o corpo do documento. A YAML (Yet Another Markup Language) Header, ou metadados, é um cabeçalho onde são inseridas as informações sobre o arquivo e das opções de compilação. Sempre devem iniciar o documento, sendo inseridas dentro de dois campos de sinais — — —.

Já abaixo do YAML, situa-se o local onde o pesquisador digitará o texto, bem como integrará a inserção de códigos do R e também efetuará as análises posteriores (análises descritivas, regressões, tabelas, fórmulas, etc.). Por sua vez, os códigos do R (para manipulação

de dados, como visto até o capítulo anterior deste livro) são “embutidos” no texto por meio das **Code Chunks**. Já o texto é inserido normalmente em forma de parágrafos (“fora” dos Chunks), sendo que o novo parágrafo é iniciado após pressionar a tecla “Enter” entre os textos informados.

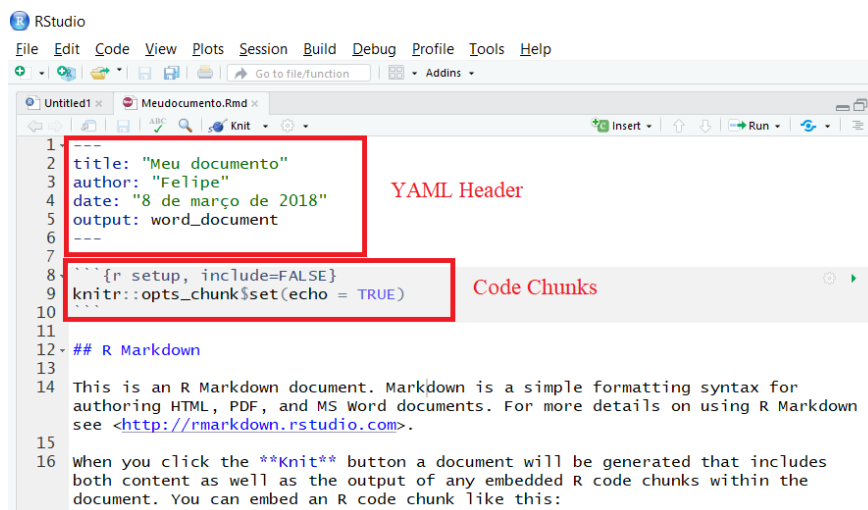


Figura 6.5: Tela inicial do arquivo RMarkdown

Elaborado pelo(s) autor(es).

Desta forma, ao efetuar a compilação do documento, o RStudio “lê” todas as informações inseridas no arquivo e cria como resultado um arquivo escolhido com todas as análises feitas pelo usuário.

No exemplo acima (Figura 6.5), a compilação irá gerar um arquivo em Word, de acordo com o `output` escolhido, no caso `word_document`. Se o usuário desejar gerar como arquivo de texto final um documento que pode ser aberto inclusive em software livre, pode utilizar o formato OpenDocument (`.otd`). Para isto, basta substituir o `output` para `odt_document`.

6.4 Elementos básicos de formatação

Dentro do documento **RMarkdown**, depois dos metadados, começa o espaço destinado ao texto do documento. Nesta etapa seguem algumas condições para a formatação do texto, bem como da configuração dos títulos e fórmulas matemáticas. A linguagem *markdown* preza pela simplicidade na formatação do texto, a qual posteriormente pode ser exportada para diversos tipos de documentos de uma só vez. Desta forma, como visto anteriormente, cria documentos totalmente dinâmicos entre si.

Os níveis de títulos dos documentos RMarkdown são definidos pelo símbolo #:

# Título nível um {#ancora}	Título nível um
## Título nível dois	Título nível dois
### Título nível três	Título nível três
#### Título nível quatro	Título nível quatro
##### Título nível cinco	Título nível cinco
##### Título nível seis	Título nível seis

Figura 6.6: Títulos no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

A acentuação das palavras, dentro do texto, é feita normalmente pelo teclado do usuário. Os caracteres `##/()` `[]<>` podem ser escritos normalmente dentro do texto, no entanto os demais (exemplo do cifrão `$`) devem ser escritos precedidos de uma barra: `\$`. Por outro lado, a formatação em itálico, negrito, subscrito, sobrescrito, links e demais formatações são feitas no documento (Figura 6.7).

Formatação itálico	<i>Formatação itálico</i>
Formatação negrito	Formatação negrito
Negrito itálico	<i>Negrito itálico</i>
~Tachado~	<u>Tachado</u>
Palavra ^Sobrescrita^	Palavra ^{Sobrescrita}
Palavra ~Subscrita~	Palavra _{Subscrita}
`Linha de código`	Linha de código
<!-- Comentário de texto, não aparece no documento final-->	
> Bloco de citações	Bloco de citações
Bloco de Equações: $E=mc^2$	Bloco de Equações: $E = mc^2$
Equações em linha de texto: $E=mc^2$	Equações em linha de texto: $E = mc^2$

Figura 6.7: Formatação no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

Como visto, é possível escrever as fórmulas em notação matemática, o que facilita e muito a vida do pesquisador. No ambiente matemático do **RMarkdown**, elas são escritas por meio da linguagem de marcação de textos LaTeX. Existem muitos manuais sobre esta

linguagem, e para facilitar a escrita, sites como <https://www.codecogs.com/latex/eqneditor.php?lang=pt-br> ajudam o pesquisador nesta empreitada.

É possível efetuar a inserção de links nos documentos, para páginas externas ou mesmo internas ao documento (Figura 6.8).

<code><http://www.rstudio.com></code>	http://www.rstudio.com
<code>[link] (www.rstudio.com)</code>	link
<code>Pule para o [Título 1] (#ancora)</code>	Pule para o Título 1

Figura 6.8: Links no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

A inserção de imagens externas no documento, em diversos formatos (aqui no exemplo .png) é feita a partir do direcionamento do nome da imagem salva na mesma pasta do arquivo .Rmd criado, ou mesmo pelo link na internet (Figura 6.9).

`! [Título da imagem] (RStudio.png)`



Título da imagem

`! [] (http://www.google.com/images/logo.gif)`



Figura 6.9: Imagens no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

A Figura 6.10 demonstra algumas formas de criar listas e itens no decorrer do corpo de texto no **RMarkdown**.

* Lista desordenada	• Lista desordenada
+ Sub-item 1	◦ Sub-item 1
+ Sub-item 2	◦ Sub-item 2
* Item 2	• Item 2
1. Lista ordenada	1. Lista ordenada
2. Item 2	2. Item 2
i) sub-item 1	i. sub-item 1
A. sub-sub-item 1	A. sub-sub-item 1
(@) Lista automática que continua	1. Lista automática que continua
depois de uma	depois de uma
(@) Interrupção	2. Interrupção
- Outra forma de intenizar	• Outra forma de intenizar
- Item 2	• Item 2

Figura 6.10: Listas no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

A criação de tabelas simples segue a disposição dos elementos pré-definidos, sendo que o alinhamento da coluna se dá pelo caractere “:” (dois pontos) conforme a Figura 6.11:

Direita *Esquerda* *Default* *Centro*	<i>Título da tabela</i>
----- :----- ----- :-----	<i>Direita Esquerda Default Centro</i>
10 27 52 70 100	10 27 52 70
200 13 52 14	200 13 52 14
2 20 200 400	2 20 200 400
Table: Título da tabela	

Figura 6.11: Tabelas simples no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

As notas de rodapé são inseridas no texto dentro das chaves precedidas do acento circunflexo `^[]`. O pesquisador adiciona-os durante o texto, e o programa enumera automaticamente no documento final em Word (Figura 6.12).

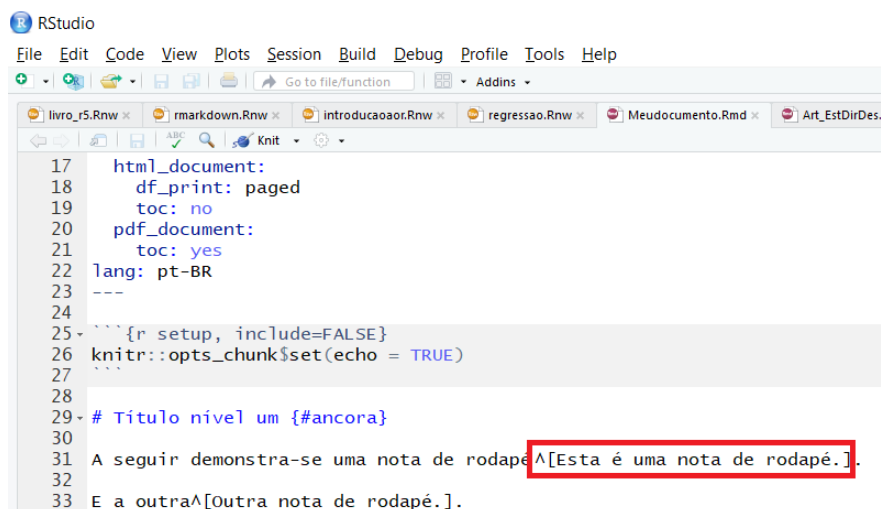


Figura 6.12: Notas de rodapé no RMarkdown

Fonte: Elaborado pelo(s) autor(es).

6.5 Elementos básicos do YAML

O YAML, ou os metadados do documento, são informações básicas do documento que podem ser alteradas (Figura 6.13). Dentre elas *title* define o título do documento; em *author* é inserido o autor ou autores e as informações do currículo do pesquisador são inseridas via nota de rodapé dentro do símbolo `^[]`; o campo *date* é opcional.

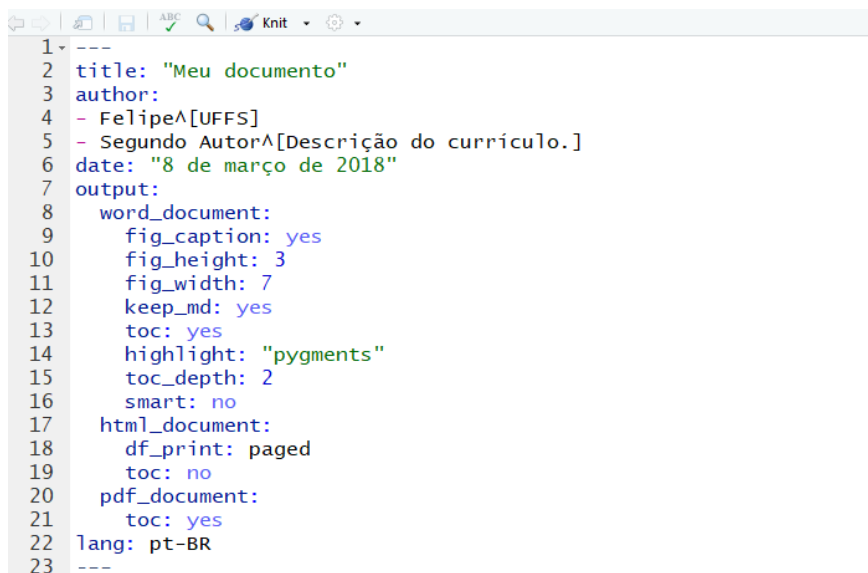


Figura 6.13: Configuração do YAML

Fonte: Elaborado pelo(s) autor(es).

Já o campo *output* define a opção de salvamento do arquivo final. Pode ser informado todos os tipos de arquivos previamente, sendo que no momento da compilação será utilizado o primeiro tipo de arquivo, no exemplo, em Word. Para salvar em PDF, é só colocar o campo `pdf_document` em primeiro lugar juntamente com a configuração dentro deste tipo de arquivo.

Abaixo do tipo de arquivo a ser salvo, constam as opções de salvamento. No caso do exemplo, abaixo de Word está constando a opção `fig_caption`, que dita se as figuras do documento em Word serão inseridas com títulos.

Os campos `fig_height` e `fig_width` determinam a altura e largura padrão de todas as imagens do documento Word. Abaixo seguem algumas opções do YAML relacionando-se com a saída do documento em Word:

- **fig_caption** - As figuras devem ter título?
- **fig_height**, **fig_width** - Altura e largura padrão das imagens.
- **highlight** - Estilo de saída pré-definido, inclui “default”, “tango”, “pygments”, “kate”, “monochrome”, “espresso”, “zenburn”, e “haddock”.
- **keep_md** - Salva uma cópia em arquivo .md juntamente com os outros arquivos.
- **md_extensions** - Extensões Markdown a serem incluídas como definições padrão no RMarkdown.
- **pandoc_args** - Argumentos adicionais para utilizar com o pandoc.
- **reference_docx** - Arquivo docx com as configurações de estilos de texto padrão. Deve ser salvo na mesma pasta do documento .rmd criado.
- **toc** - Adiciona o sumário no início do texto.
- **toc_depth** - Determina o menor nível de títulos que será exibido no sumário. Exemplo, 1 mostra somente o primeiro nível.

Também é possível incluir um campo `abstract` para o resumo, no caso de artigo e suas respectivas palavras-chave:

```
3 author:  
4 - Felipe  
5 abstract: |  
6 **Resumo**  
7  
8 Lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
9 ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
10 lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
11 ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
12 lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
13 ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
14 ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
15 lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
16 lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
17  
18 **Palavras-Chave:** xxxxxx; xxxxx; xxxxx; xxxxx; xxxxx
```

Figura 6.14: Abstract no YAML

Fonte: Elaborado pelo(s) autor(es).

6.6 Elementos básicos dos Chunks

Os **Code Chunks**, como já visto, são espaços destinados à inclusão de códigos diretamente do RStudio, como se a informação fosse inserida em seu Console. Desta forma, por exemplo, ao efetuar uma operação matemática ou ao ser carregada uma base de dados para ser trabalhada, as rotinas serão efetuadas no momento em que for compilado o arquivo .Rmd trabalhado.

A criação das Chunks é feita manualmente no corpo do documento .Rmd pela inclusão do código demonstrado abaixo, ou via plataforma RStudio, no menu “Insert > Insert a new R chunk”, conforme demonstra a Figura 6.15:

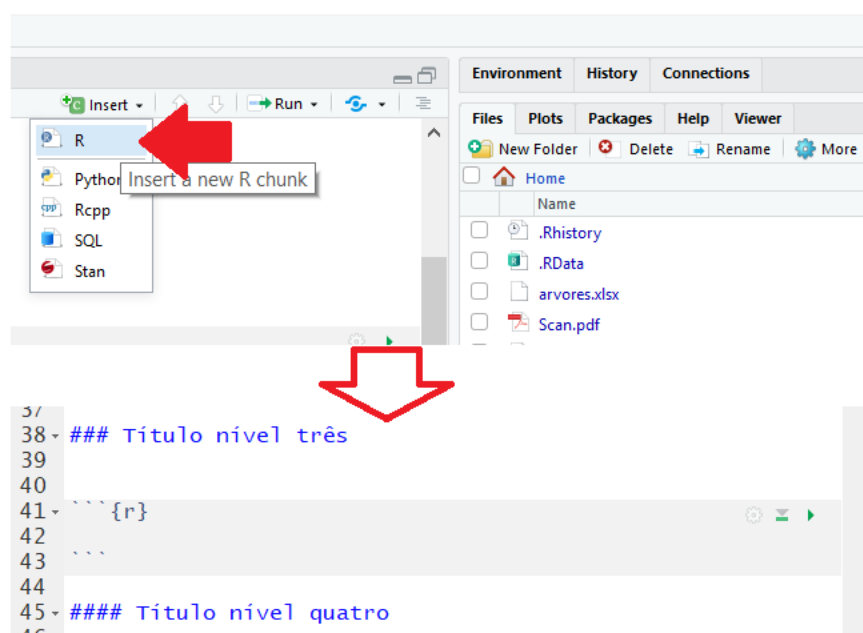


Figura 6.15: Criação de Chunks

Fonte: Elaborado pelo(s) autor(es).

Nota-se que o corpo do documento .Rmd ficou de outra cor, indicando que está inserida uma Chunk naquele local. Dentro das chaves, a Chunk divide-se entre uma identificação/nome para aquele campo (é opcional, no entanto se constar não pode ser repetido no documento) e; as opções da Chunk.

No exemplo abaixo, o nome da Chunk criada foi “r nomedochunk”. E no campo das opções, constaram `echo=FALSE`, `fig.height=10` e `fig.width=5`. Lembrando que estes campos determinam as opções somente para este chunk.

A primeira opção, `echo=FALSE`, informa que no arquivo compilado, somente será mostrado o resultado da rotina inserida na Chunk (1+1), portanto será mostrado somente o valor 2. Caso o usuário almejasse inserir, no arquivo final, o código do R escrito (1+1) juntamente com o resultado da operação, marcaria `echo=TRUE`.

```

47 {nome do chunk, opções }
48 Códigos do R
49
41
42 {r nomedachunk, echo=FALSE, fig.height=10, fig.width=5}
43 #Texto de apoio, não será compilado
44 1+1
45

```




Figura 6.16: Criação de Chunks (2)

Fonte: Elaborado pelo(s) autor(es).

As opções `fig.height` e `fig.width` referem-se à altura e largura caso o resultado final da Chunk fosse uma figura ou gráfico derivado de dados inseridos na mesma. Vale lembrar que somente seriam determinadas as medidas para esta Chunk.

Para padronizar todas as Chunks para que tenham as mesmas opções, uma maneira utilizada usualmente é a inserção de uma Chunk `global`. Ela é incluída no início do texto, sendo que a sua inclusão é facultativa. No entanto, contribui para padronizar o texto, ao mesmo tempo que se existir uma Chunk durante o texto que deva ser configurada de forma diferente (por exemplo, o tamanho da imagem), pode ser efetuado em cada Chunk individual.

```

24
25 {r setup, include=FALSE}
26 knitr::opts_chunk$set(echo = TRUE, message = TRUE,
27                       warning = TRUE, fig.height=5, fig.width=5)
28

```

Figura 6.17: Chunk global

Fonte: Elaborado pelo(s) autor(es).

Seguem algumas importantes opções das Chunks dos arquivos RMarkdown (Allaire *et al.*, 2017):

- **echo** - O código da Chunk deve ser incluído no resultado final? Padrão = FALSE.
- **error** - Mostra mensagens de erro no documento (TRUE) ou para quando os erros ocorrem.
- **fig.align** - Alinhamento da figura: “left”, “right” ou “center” (padrão = “default”).
- **fig.height**, **fig.width** - Tamanho das figuras em polegadas.
- **include** - Para incluir a Chunk depois de compilar (padrão = TRUE).
- **message** - Mostra as mensagens por ventura existentes no documento (padrão = TRUE).
- **results** - (default=“markup”) “asis” - processa os resultados na saída do documento; “hide” - não mostra os resultados; “hold” - coloca os resultados abaixo do código.
- **warning** - Mostra avisos de advertência no documento (padrão = TRUE).

Como mencionado no início deste capítulo, a grande vantagem do **RMarkdown** é a sua versatilidade na criação de documentos concatenados com as análises estatísticas no RStudio. Desta forma, dentro das Chunks, podem ser criadas bases de dados, bem como

importados de sites ou mesmo carregados de arquivos trabalhados previamente no RStudio.

No exemplo abaixo, foi criado um *data frame* nomeado “amost” diretamente no console dentro da Chunk. Em um segundo momento, para utilizar um determinado pacote instalado no RStudio, se insere, dentro da Chunk, o comando `require()` juntamente com o pacote necessário. Podem ser inseridos tantos pacotes quanto forem utilizados no documento, conforme a Figura 6.18.

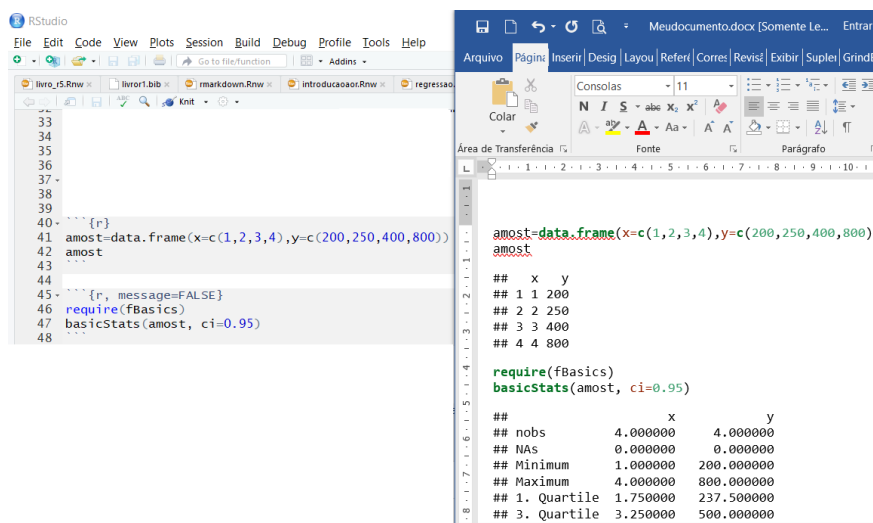


Figura 6.18: Exemplo de criação de Chunk e carregamento de pacote

Fonte: Elaborado pelo(s) autor(es).

6.6.1 Inserindo tabelas com as Chunks

Como visto, algumas ações extremamente úteis podem ser efetuadas por meio das Chunks. Dentre elas, inclui-se a plotagem de tabelas no texto final, derivadas de objetos criados pelo pesquisador no RStudio. Os exemplos trazidos abaixo incluem a utilização dos pacotes `kable`, `xtable` e `flextable` para a criação das tabelas.

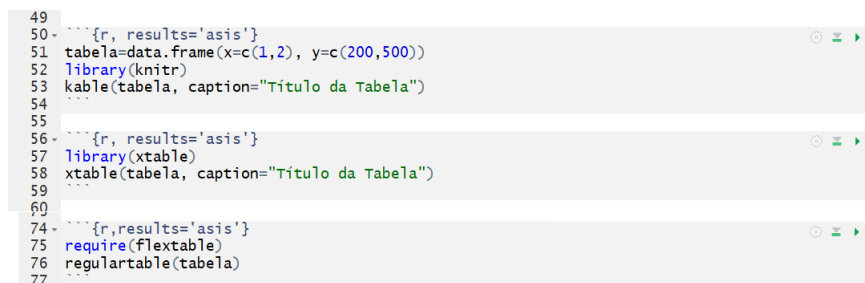
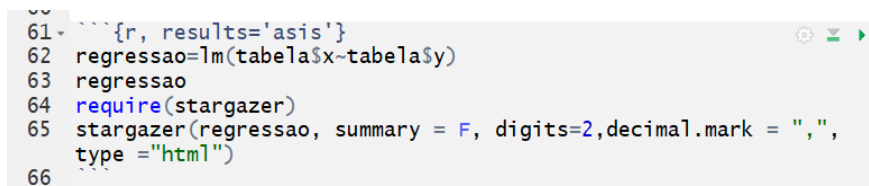


Figura 6.19: Exemplo de criação de tabelas com os pacotes `kable`, `xtable` e `flextable`

Fonte: Elaborado pelo(s) autor(es).

Além disso, o pacote **stargazer** é extremamente útil para geração de tabelas com resultados de regressões com a saída dos documentos em PDF.



```

61- {r, results='asis'}
62 regressao=lm(tabela$x~tabela$y)
63 regressao
64 require(stargazer)
65 stargazer(regressao, summary = F, digits=2, decimal.mark = ",",
66 type ="html")

```

Figura 6.20: Exemplo de criação de tabelas com stargazer

Fonte: Elaborado pelo(s) autor(es).

Existem diversos pacotes dentro do RStudio com a função de gerarem tabelas a partir dos dados analisados (neste caso sendo gerados dentro de chunks). Por sua vez, abaixo segue um resumo testado por este autor, em que os pacotes criam corretamente as tabelas no RMarkdown para cada saída de arquivo (HTML, PDF, Word). Por exemplo, o pacote **knitr** (com a função **kable**) consegue gerar as tabelas para as três saídas de arquivos; já o pacote **xtable** não está configurado para criar a referida tabela com a saída do documento em Word.

Tabela 6.1: Pacotes para elaboração de tabelas no RMarkdown

Pacote	HTML	PDF	Word
knitr(função kable) ¹	ok	ok	ok
pander ²	ok	ok	ok
stargazer ³	ok	ok	-
xtable ⁴	ok	ok	-

Fonte: Elaborado pelo(s) autor(es).

Outra forma de passar as tabelas para o Word é criando-a no formato HTML e copiando para o arquivo em Word (veja em https://cran.r-project.org/web/packages/kableExtra/vignettes/kableExtra_and_word.html).

6.6.2 Inserindo imagens com as Chunks

Da mesma forma que as tabelas, as imagens também podem ser inseridas com o auxílio de Chunks. Lembrando que a imagem deve estar na mesma pasta do arquivo ou na pasta indicada:

¹Mais informações em <https://www.rdocumentation.org/packages/knitr/versions/1.21/topics/kable>.

²Daróczy e Tsegelskyi (2018).

³Hlavac (2018).

⁴Dahl *et al.* (2018).

```
33  
34 {r, fig.align='center', fig.width=0.5, fig.height=0.5}  
35 knitr::include_graphics('Rstudio.png')  
36  
37
```

Figura 6.21: Exemplo de inserção de imagens pelos Chunks

Fonte: Elaborado pelo(s) autor(es).

6.7 Criação de modelos para formatação vinculada

Para os pesquisadores que trabalham intensamente com o Word ou Libre/Open Office, a formatação dos resultados decorrentes das análises compiladas no RMarkdown podem ser incrementadas. Isto porque existe um recurso de criação de modelos vinculados aos editores de texto, estes que serão responsáveis pela definição da formatação de todos os itens (títulos, subtítulos, parágrafos, fontes, etc.), como será visto a seguir.

6.7.1 Primeiro passo: criação de um documento modelo

Primeiramente deve-se criar um documento mínimo padrão que será utilizado como modelo. Crie um novo documento (Rmd), aqui será denominado de “modelo” (o usuário pode escolher o nome), que será salvo em .Rmd e gerado o respectivo arquivo Word (ou no formato .odt), na mesma pasta que o pesquisador salvar arquivos a serem formatados.

Como já visto, para criação de documentos .Rmd clique em “File > New File > R Markdown”. Escolha o nome e salve na pasta escolhida. Gere o documento em Word (.docx) ou em outro arquivo de texto (exemplo .odt) em “File > Knit Document”.

6.7.2 Segundo passo: formatação do modelo

Abra o arquivo em Word (denominado “modelo.docx”). Atente para a caixa de seleção de estilos do Word, que será trabalhado nesta etapa (Figura 6.22).

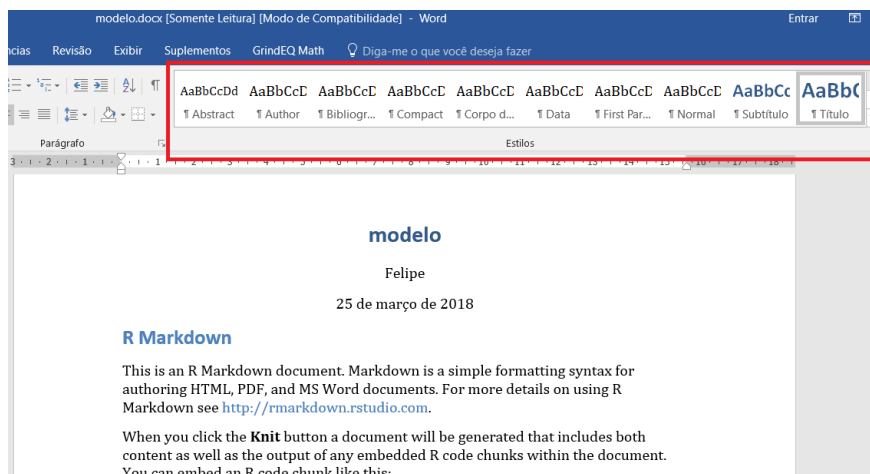


Figura 6.22: Caixa estilos no Word

Fonte: Elaborado pelo(s) autor(es).

Note que para o resultado desta compilação, o menu estilos traz várias formatações das diferentes partes do texto, entre elas “Abstract”, “Author”, “Normal”, “Título”, “Título 1”, etc. Estes estilos serão alterados pelo usuário, para adequar às necessidades do pesquisador na criação do documento padrão. Clique com o botão direito nos estilos e em “Modificar” para definir a formatação padrão para cada parte do texto.

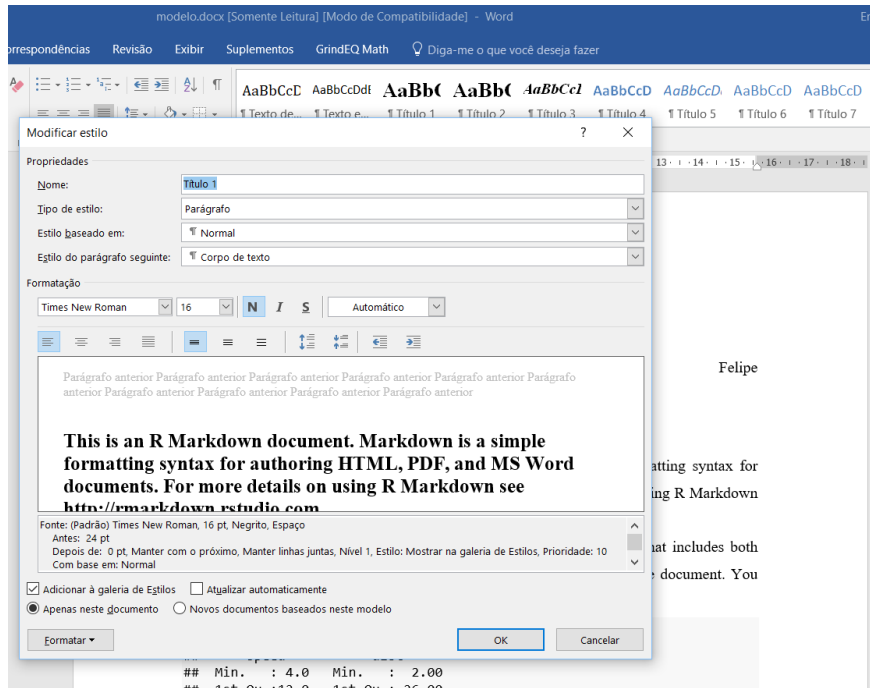
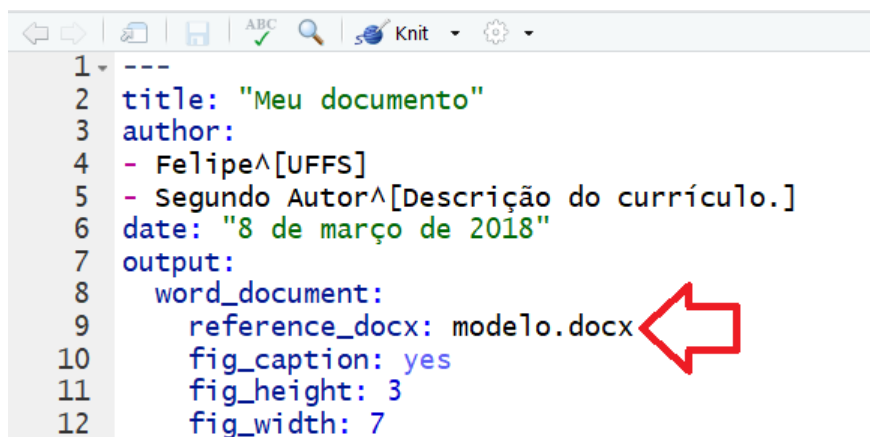


Figura 6.23: Modificação de estilos no Word

Fonte: Elaborado pelo(s) autor(es).

6.7.3 Terceiro passo: vinculação do modelo ao arquivo em RMarkdown

Após determinar as alterações em todos os campos de estilos do documento modelo no Word, o pesquisador deve vincular este modelo ao documento .Rmd principal. Além de deixar salvo o modelo em Word na mesma pasta, deve-se incluir a seguinte informação no YAML mostrada na Figura 6.24 (`reference_docx`). Lembrando que para arquivos em Open/Libre Office, deve ser inserida a opção `reference_odt` seguida do arquivo (.odt) do modelo.



```
1 ---
2 title: "Meu documento"
3 author:
4 - Felipe^[UFFS]
5 - Segundo Autor^[Descrição do currículo.]
6 date: "8 de março de 2018"
7 output:
8   word_document:
9     reference_docx: modelo.docx
10   fig_caption: yes
11   fig_height: 3
12   fig_width: 7
```

Figura 6.24: Vinculação do modelo

Fonte: Elaborado pelo(s) autor(es).

A partir de então, as compilações do arquivo .Rmd criado pelo pesquisador seguirão as formatações de estilo que estão determinadas no arquivo “modelo.docx”.

6.8 Citações e bibliografias

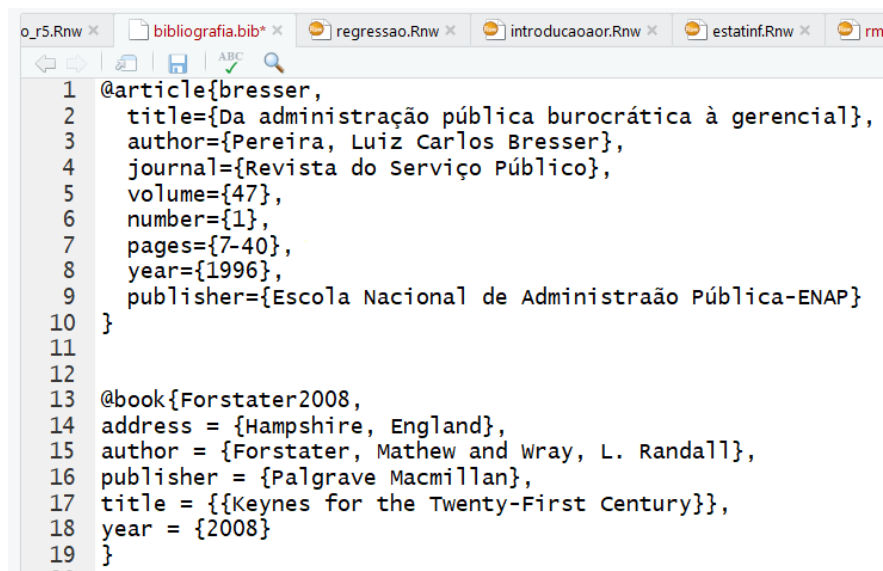
Na escrita de trabalhos acadêmicos com o RMarkdown é possível efetuar um gerenciamento de citações e bibliografias de maneira extremamente satisfatória e automática. Para isto, o *software* MiKTeX contribuirá nesta empreitada.% juntamente com o programa Pandoc.

O exemplo abaixo será utilizado com o formato BibLaTeX (extensão .bib). Primeiramente crie um documento .bib, que será o local onde o pesquisador armazenará as bibliografias, que serão posteriormente utilizadas. Crie um novo arquivo de texto (“File > New File > Text file”) e depois salve-o na mesma pasta do arquivo .Rmd em que serão inseridas as citações (salve com a extensão “.bib” - exemplo: “bibliografia.bib”).

Dentro deste arquivo serão armazenadas as referências bibliográficas, não deve-se preocupar neste momento com a ordem das referências. Como mostra a Figura 6.25, inserimos duas bibliografias a serem citadas posteriormente.

A primeira (@article), demonstra que é um artigo de uma revista enquanto a segunda

(`@book`) se trata de um livro. Dentro das chaves estão os dados das referências, como o título (`title`), autores (`author`) e o ano (`year`) por exemplo.



```

1 @article{bresser,
2   title={Da administração pública burocrática à gerencial},
3   author={Pereira, Luiz Carlos Bresser},
4   journal={Revista do Serviço Público},
5   volume={47},
6   number={1},
7   pages={7-40},
8   year={1996},
9   publisher={Escola Nacional de Administração Pública-ENAP}
10 }
11
12
13 @book{Forstater2008,
14   address = {Hampshire, England},
15   author = {Forstater, Mathew and Wray, L. Randall},
16   publisher = {Palgrave Macmillan},
17   title = {{Keynes for the Twenty-First Century}},
18   year = {2008}
19 }
20

```

Figura 6.25: Arquivo .bib

Fonte: Elaborado pelo(s) autor(es).

O BibLaTeX gerencia todos os tipos de bibliografias sendo que, como visto acima, as bibliografias possuem campos padrão a serem informados no arquivo “.bib”. Por exemplo, a categoria de artigos, possui como campos obrigatórios `author`, `title`, `journal` e `year`. Abaixo seguem algumas especificações dos tipos de bibliografias (adaptado de Lehman e Kime (2006)), explicitando os itens obrigatórios e opcionais que devem ou podem constar em cada registro:

@article - Artigo de revista. **OBRIGATÓRIOS:** `author`, `title`, `journal`, `year`. **OPCIONAIS:** `volume`, `number`, `pages`, `month`, `note`, `key`.

@book - Livro. **OBRIGATÓRIOS:** `author/editor`, `title`, `publisher`, `year`. **OPCIONAIS:** `volume/number`, `series`, `address`, `edition`, `month`, `note`, `key`.

@inbook - Parte de livro. **OBRIGATÓRIOS:** `author/editor`, `title`, `chapter/pages`, `publisher`, `year`. **OPCIONAIS:** `volume/number`, `series`, `type`, `address`, `edition`, `month`, `note`, `key`.

@incollection - Parte de livro com título próprio. **OBRIGATÓRIOS:** `author`, `title`, `booktitle`, `publisher`, `year`. **OPCIONAIS:** `editor`, `volume/number`, `series`, `type`, `chapter`, `pages`, `address`, `edition`, `month`, `note`, `key`.

@inproceedings - Trabalho de anais de conferência. **OBRIGATÓRIOS:** `author`, `title`, `booktitle`, `year`. **OPCIONAIS:** `editor`, `volume/number`, `series`, `pages`, `address`, `month`, `organization`, `publisher`, `note`, `key`.

@mastersthesis - Dissertação mestrado. **OBRIGATÓRIOS:** `author`, `title`, `school`, `year`. **OPCIONAIS:** `type`, `address`, `month`, `note`, `key`.

@phdthesis - Tese de doutorado. **OBRIGATÓRIOS:** author, title, school, year. **OPCIONAIS:** type, address, month, note, key.

Estas configurações do BibLateX são comuns nos programas de gerenciamento de bibliografias, como por exemplo no *software* Mendeley. Os usuários deste programa tem uma facilidade na exportação para o formato do BibLateX, pois podem copiar as entradas com as informações de um trabalho e as inserir dentro do arquivo .bib (Figura 6.26).

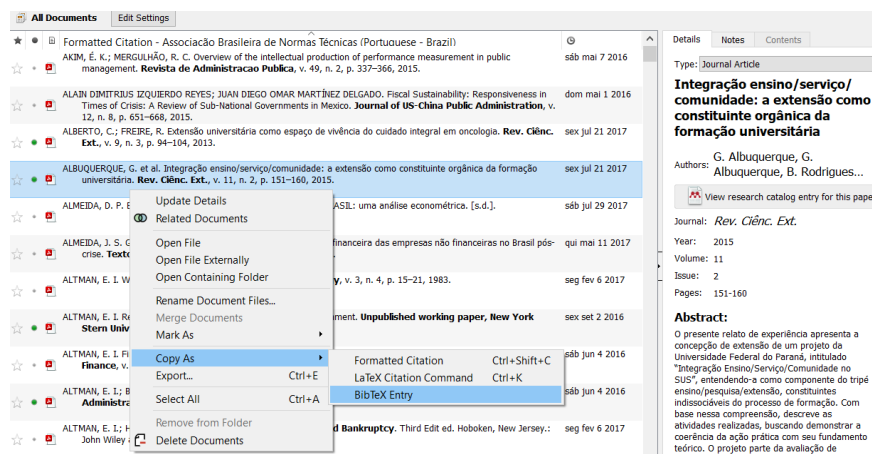


Figura 6.26: Utilização do Mendeley para exportação de dados de bibliografias

Fonte: Elaborado pelo(s) autor(es).

Após escolhidas as bibliografias a serem utilizadas no trabalho, o pesquisador deve inserir estas entradas como referências dentro do texto. Para isto, utiliza o nome da bibliografia inserida no arquivo .bib, no nosso exemplo **bresser** e **Forstater2008**, como mostra a Figura 6.27.

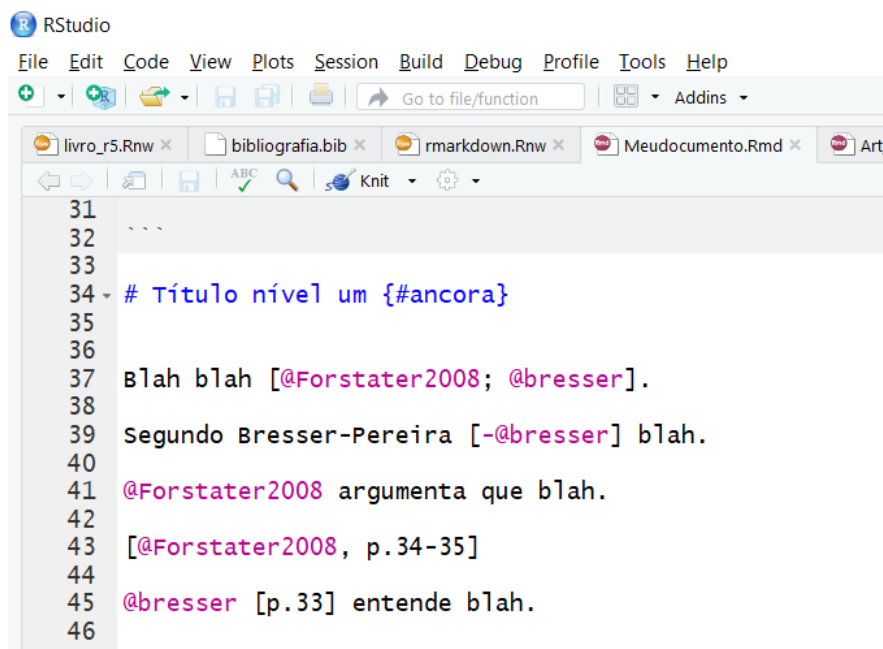


Figura 6.27: Inserção de citações no arquivo .Rmd

Fonte: Elaborado pelo(s) autor(es).

Para que o arquivo que foi criado com as referências bibliográficas (bibliografia.bib) seja utilizado, o pesquisador deve informar o seu nome dentro do YAML no campo **bibliography**. Mas qual norma será utilizada para as citações e a criação de referências bibliográficas neste trabalho, já que existem diversas delas? Uma solução é a utilização de arquivos “.csl” (Citation Style Language), que nada mais são do que arquivos com as descrições de cada estilo das diversas normas existentes, para ajudar o pesquisador a citar e gerenciar suas referências.

Estes arquivos podem ser encontrados em diversos locais, como por exemplo em <https://github.com/citation-style-language/styles> (copie este⁵ conteúdo para um arquivo “.txt” e o renomeie para “.csl”). Lembrando que o arquivo “.csl” deve ser salvo na mesma pasta do arquivo “.Rmd”. O arquivo csl aqui utilizado refere-se às normas da ABNT (Associação Brasileira de Normas Técnicas) utilizados pelo IPEA (Instituto de Pesquisa Econômica Aplicada). Verifica-se na Figura 6.28 a configuração final do YAML. Neste site <http://editor.citationstyles.org/searchByName/> também são encontrados arquivos para várias normas bibliográficas.

⁵<https://raw.githubusercontent.com/citation-style-language/styles/44808db510152943c5d9dc471a9c8982a3edfbea/associacao-brasileira-de-normas-tecnicas-ipea.csl>

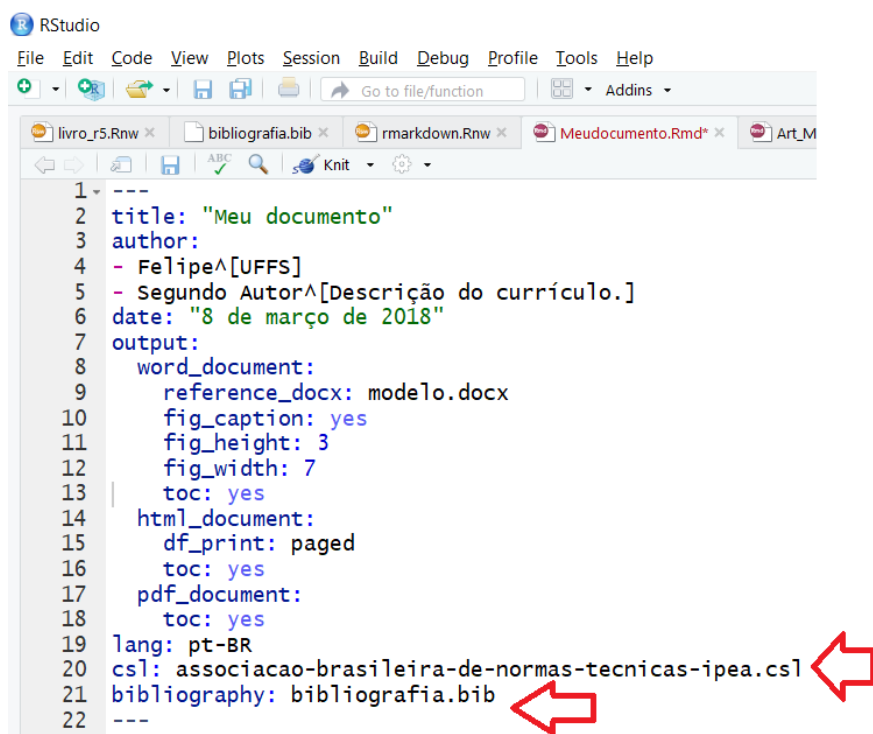


Figura 6.28: Configurando YAML para citações e referências

Fonte: Elaborado pelo(s) autor(es).

Por fim, após inserir a citação no texto e informar ao RMarkdown os arquivos “.bib” e “.csl” no YAML, basta compilar o arquivo no formato desejado (atalho no teclado CTRL+SHIFT+K), neste caso Word. Lembre-se de inserir um título # **Referências** ou # **Referências Bibliográficas** ou # **Bibliografia** (como preferir), no final do texto, pois serão inseridas as referências no final do trabalho.

A partir de então fica muito mais fácil alterar a norma necessária para a produção do trabalho acadêmico, utilizando os mesmos dados de um artigo ou outro material a ser citado. Isto agiliza a produção acadêmica e proporciona, como visto neste livro, uma interação muito proveitosa com a geração das análises por meio do RStudio.

Segue o resultado do arquivo final:

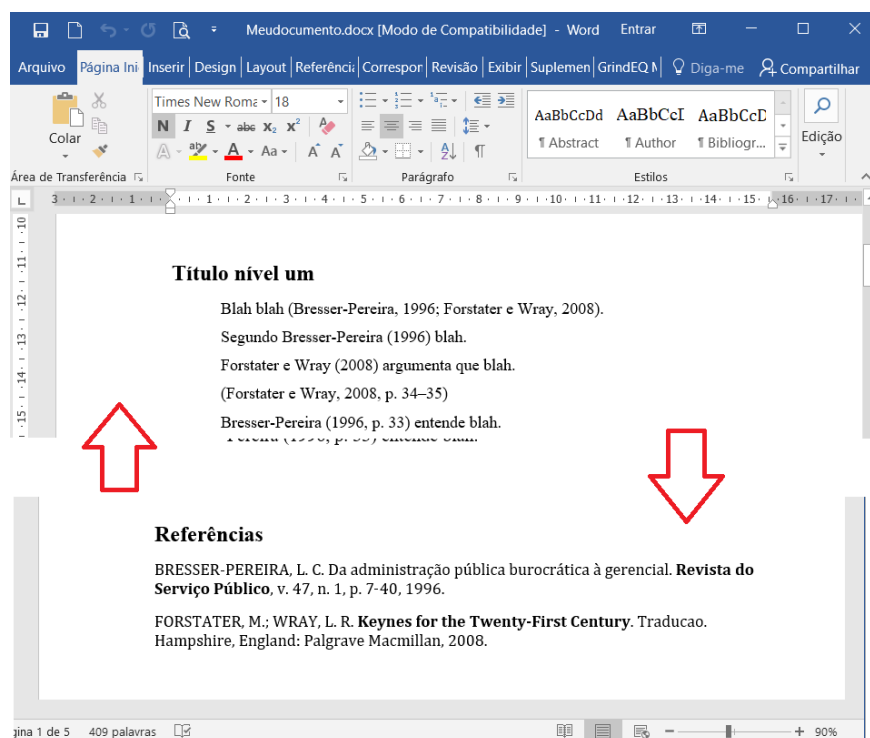


Figura 6.29: Resultado final das citações e referências com RMarkdown

Fonte: Elaborado pelo(s) autor(es).

6.9 Exercícios

1. Ao final deste livro, você aprendeu a linguagem de programação em R no console RStudio, passando posteriormente para a aprendizagem dos comandos para efetuar estatísticas descritivas, análises com estatística inferencial, teste de qui-quadrado e chegando ao modelo de regressão simples. Desta forma, escolha algum dos exercícios que constam nos capítulos anteriores e com seu código de programação crie um arquivo em RMarkdown (.rmd, como aprendido no presente capítulo) com todos os passos efetuados na análise e seus resultados. Crie um documento dinâmico testando os formatos de saída HTML, Word e PDF.

Sobre os autores

Denize Ivete Reis: Possui Licenciatura Plena em Matemática pela Universidade Regional do Noroeste do Estado do Rio Grande do Sul (1994), especialização em Estatística Aplicada pela Universidade de Santa Cruz do Sul (2003), mestrado em Modelagem Matemática pela Universidade Regional do Noroeste do Estado do Rio Grande do Sul (1997) e doutorado em Qualidade Ambiental pela Universidade Feevale (2015). Atualmente é professora adjunta da Universidade Federal da Fronteira Sul, onde atua na área de Probabilidade e Estatística, Estatística Descritiva e Inferência Estatística. E-mail: denizeir@uffs.edu.br.

Djaina Sibiani Rieger: Acadêmica do curso de Matemática da Universidade Federal da Fronteira Sul (UFFS) Campus Chapecó, foi aluna bolsista de extensão, integrante e conteudista dos cursos ofertados no Campus Cerro Largo sobre o *software* livre R.

Erikson Kaszubowski: Doutor em Psicologia pela Universidade Federal de Santa Catarina, sob orientação do Prof. Dr. Fernando Aguiar, com a tese “Modelos de tópicos para associações livres”. Formado em Psicologia pela Universidade Federal de Santa Catarina, nas graduações Bacharelado e Formação de Psicólogo (2006), e Licenciatura (2008). Foi professor de Psicologia da Educação na Universidade Federal da Fronteira Sul, ministrando as disciplinas da área de Psicologia nos cursos de Licenciatura e Pós-Graduação. Trabalha atualmente como psicólogo clínico no Serviço de Atenção Psicológica da UFSC. E-mail: erikson84@yahoo.com.br.

Felipe Micaíl da Silva Smolski: Possui graduação em Ciências Econômicas pela Universidade Regional do Noroeste do Estado do Rio Grande do Sul - UNIJUÍ (2009), pós-graduação em Gestão de Investimentos pela Faculdade Integrada Grande Fortaleza - FGF (2012), mestrado em Desenvolvimento e Políticas Públicas pela Universidade Federal da Fronteira Sul - UFFS, Campus Cerro Largo (2017). E-mail: felipesmolski@hotmail.com.

Iara Denise Endruweit Battisti: Possui graduação em Informática pela Universidade Regional do Noroeste do Estado do Rio Grande do Sul (1996), mestrado em Estatística e Experimentação Agropecuária pela Universidade Federal de Lavras (2001) e doutorado em Epidemiologia pela Universidade Federal do Rio Grande do Sul (2008). Atualmente é professora associada na Universidade Federal da Fronteira Sul, campus Cerro Largo (RS). É docente dos Programas de mestrado em Ambiente e Tecnologias Sustentáveis e mestrado em Desenvolvimento e Políticas Públicas. Atua principalmente nos seguintes temas: amostragem complexa, relação ambiente e saúde utilizando modelagem estatística. E-mail: iara.battisti@us.edu.br.

Tatiane Chassot: Possui graduação em Engenharia Florestal pela Universidade Federal de Santa Maria (2008), mestrado (2009) e doutorado em Engenharia Florestal também

pela Universidade Federal de Santa Maria (2013). Atualmente é professora adjunta da Universidade Federal da Fronteira Sul - Campus Cerro Largo onde ministra as disciplinas de Introdução à Informática, Estatística Básica, Experimentação Agrícola, Sistemas Agroflorestais, Silvicultura e Práticas Integradoras de Campo. E-mail: tatianechassot@uffs.edu.br.

Referências

- ALLAIRE, J. J. *et al.* **rmarkdown: Dynamic Documents for R**, 2017. Disponível em: <<https://CRAN.R-project.org/package=rmarkdown>>. Acesso em: 1 mar. 2018
- ALMEIDA, L.; FREIRE, T. **Metodologia da investigação em psicologia e educação**. 2. ed. Braga: Psiquilíbrios, 2000.
- ANDERSON, D. R.; SWEENEY, D. J.; WILLIAMS, T. A. **Estatística aplicada à administração e economia**. 2. ed. São Paulo: Pioneira Thomson Learning, 2002.
- BARBETTA, P. A. **Estatística aplicada às ciências sociais**. 7. ed. Florianópolis: Editora UFSC, 2010.
- BRASIL, M. **Estratégias para o cuidado da pessoa com doença crônica: obesidade**. Cadernos de Atenção Básica, n 38 ed. Brasília: Ministério da Saúde, 2014.
- DAHL, D. B. *et al.* **xtable: Export Tables to LaTeX or HTML**, 2018. Disponível em: <<https://CRAN.R-project.org/package=xtable>>
- DARÓCZI, G.; TSEGELSKYI, R. **pander: An R 'Pandoc' Writer**, 2018. Disponível em: <<https://CRAN.R-project.org/package=pander>>
- DIEESE, D. **Estatísticas do meio rural 2010-2011**. 6. ed. São Paulo: DIEESE, 2011.
- FONSECA, J. S. DA; MARTINS, G. DE A. **Curso de Estatística**. 6. ed. São Paulo: Atlas, 2010.
- GROLEMUND, G.; WICKHAM, H. Dates and Times Made Easy with lubridate. **Journal of Statistical Software**, v. 40, n. 3, p. 1–25, 2011.
- HLAVAC, M. **stargazer: Well-Formatted Regression and Summary Statistics Tables**. Bratislava, Slovakia: Central European Labour Studies Institute (CELSI), 2018.
- HOFFMANN, R.; VIEIRA, S. **Análise de regressão: uma introdução à econometria**. 2. ed. São Paulo: Hucitec, 1998.
- HONAKER, J.; KING, G.; BLACKWELL, M. Amelia II: A Program for Missing Data. **Journal of Statistical Software**, v. 45, n. 7, p. 1–47, 2011.
- INMET. **Instituto Nacional de Meteorologia**, 2018. Disponível em: <<http://www.inmet.gov.br/portal/>>. Acesso em: 1 dez. 2018
- LEHMAN, P.; KIME, P. **BibLATEX – Sophisticated Bibliographies in LATEX**, 2006. Disponível em: <<https://ctan.org/pkg/biblatex>>. Acesso em: 1 mar. 2018
- LOPES, L. F. D. *et al.* **Caderno didático: estatística Geral**. 3. ed. Santa Maria: UFSM, 2008. p. 209
- MORETTIN, P. A.; BUSSAB, W. DE O. **Estatística Básica**. 6. ed. São Paulo: Saraiva,

2009.

PETERSON, B. G.; CARL, P. **PerformanceAnalytics: Econometric Tools for Performance and Risk Analysis**, 2018. Disponível em: <<https://CRAN.R-project.org/package=PerformanceAnalytics>>

R CORE TEAM. **R: A Language and Environment for Statistical Computing** Vienna, Austria R Foundation for Statistical Computing,, 2019. Disponível em: <<https://www.R-project.org>>. Acesso em: 20 fev. 2019

RISTOW, L. P. **Exposição ocupacional de trabalhadores rurais a agrotóxicos e relação com políticas públicas**.139 f.Dissertação (Mestrado em Desenvolvimento e Políticas Públicas) - Universidade Federal da Fronteira Sul; Cerro Largo,, 2017.

RSTUDIO TEAM. **RStudio: Integrated Development Environment for R** Boston, MA.RStudio, Inc., 2019. Disponível em: <<http://www.rstudio.com/>>. Acesso em: 20 fev. 2019

TRIOLA, M. F. **Introdução à estatística**.. 10. ed. Rio de Janeiro: LTC, 2011.

WICKHAM, H. **stringr: Simple, Consistent Wrappers for Common String Operations**, 2018. Disponível em: <<https://CRAN.R-project.org/package=stringr>>

WICKHAM, H.; BRYAN, J. **readxl: Read Excel Files**, 2018. Disponível em: <<https://CRAN.R-project.org/package=readxl>>

WICKHAM, H. *et al.* **dplyr: A Grammar of Data Manipulation**, 2019. Disponível em: <<https://CRAN.R-project.org/package=dplyr>>

WICKHAM, H.; HENRY, L. **tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions**, 2018. Disponível em: <<https://CRAN.R-project.org/package=tidyr>>

WICKHAM, H.; HESTER, J.; FRANCOIS, R. **readr: Read Rectangular Text Data**, 2018. Disponível em: <<https://CRAN.R-project.org/package=readr>>

WICKHAM, H.; MILLER, E. **haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files**, 2018. Disponível em: <<https://CRAN.R-project.org/package=haven>>

XIE, Y. **knitr: A General-Purpose Package for Dynamic Report Generation in R**, 2018. Disponível em: <<https://CRAN.R-project.org/package=knitr>>