



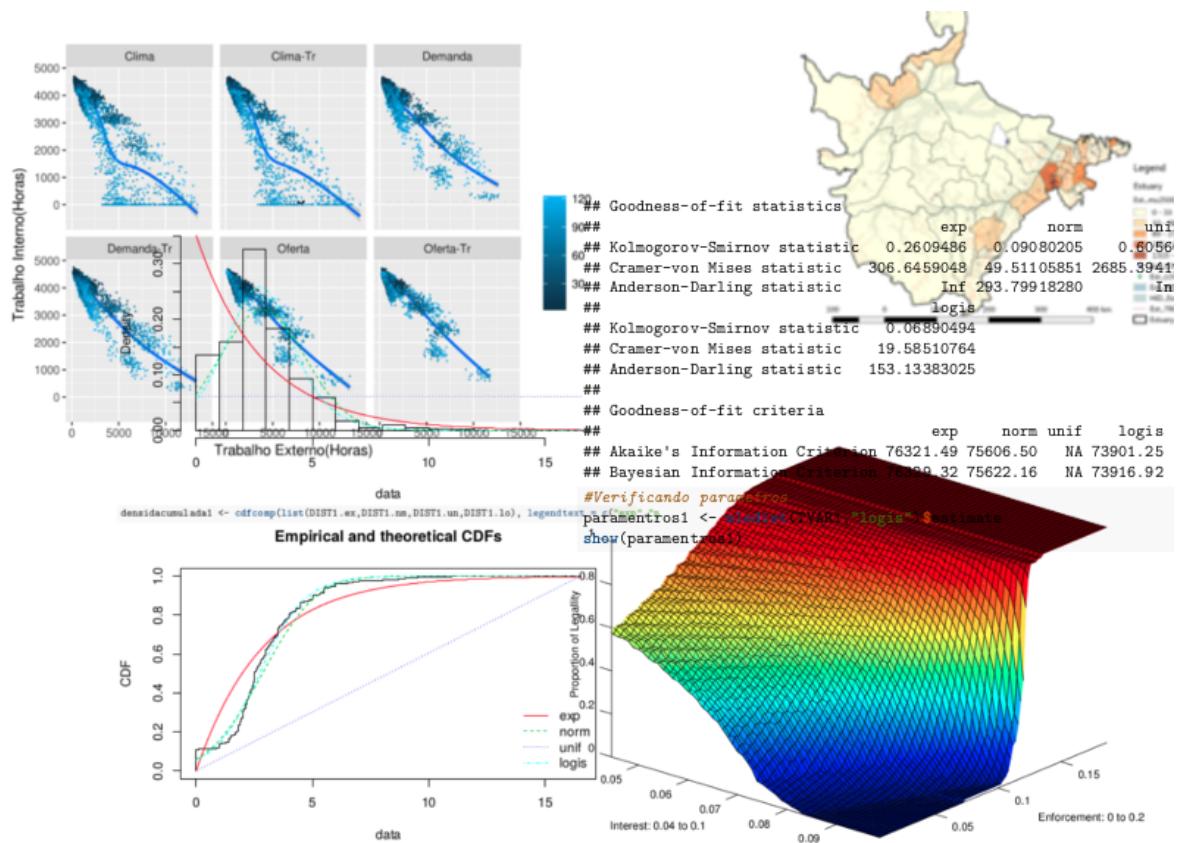
Universidade
Federal do
Pará ®

Graduação e Pós-Graduação em Economia

Introdução ao Tratamento e Análise de Dados em R

Sérgio Rivero, Hilder Farias, Murilo Gomes, Lais Conte
and Erwin Braga

11 de maio de 2019



Introdução ao Tratamento e Análise de Dados em R

Sérgio Rivero and Hilder Farias

Programa de Pós-Graduação em Economia
Instituto de Ciências Sociais Aplicadas
Universidade Federal do Pará
Rua Augusto Correia, 1
Belém, Pará - 66.075-200

Murilo Gomes, Lais Conte and Erwin Braga

Faculdade de Economia
Instituto de Ciências Sociais Aplicadas
Universidade Federal do Pará
Rua Augusto Correia, 1
Belém, Pará - 66.075-200

Approved for public release; distribution is unlimited.

Disclaimer:

Notas de Aula

Curso Ferramentas Computacionais ©2019, All rights reserved

Sumário

1	Introdução	1
2	Instalando e Configurando o R e RStudio	3
2.1	Instalando o R e RStudio	3
2.1.1	Instalando o R	3
2.1.2	Instalando o RStudio.....	6
2.2	Checando a Instalação Existente e os Requisitos.....	9
2.3	Pacotes no R	10
2.3.1	O conceito de pacote e para que serve.....	10
2.3.2	Como sei que pacotes eu preciso?	10
2.3.3	Baixando os pacotes	12
2.3.4	Resolvendo problemas de compilação	14
2.3.5	Utilizando os pacotes no seu programa R	15
3	Produzindo Relatórios Usando R Markdown	16
3.1	O que é o Rmarkdown?	17
3.2	Como o Rmd funciona?	18
3.3	A estrutura de um arquivo .Rmd	19
3.4	Sintaxe de Markdown	19
3.5	Os chunks	22
3.6	Algumas referências sobre o Rmarkdown	24
4	Acessando e Utilizando Bases de Dados	25
4.1	O ciclo de tratamento e análise de dados.....	25
4.2	Tipos de Dados em R.....	29
4.2.1	Escalares.....	29
4.2.2	Vetores	30
4.2.3	Matrizes	31
4.2.4	Array	31
4.3	Dataframes.....	33
4.4	Acessando Arquivos no computador	34
4.4.1	Importando arquivo CSV (Comma separated values) em R	35
4.5	Importando arquivo xlsx em R	35
4.5.1	Mais informações	36
4.6	Acessando Bases de dados via APIs	36
4.6.1	Pacotes Obrigatórios	36
4.7	Trabalhando com bases de dados muito grandes.....	39
5	Limpando e organizando seus dados	40
5.1	O que é uma boa base de dados e que tipos de bases existem?	40
5.1.1	Série Temporal.....	40
5.1.2	Corte Transversal	41

5.1.3	Funções de Remodelagem	43
5.1.4	Funções de Filtragem	43
5.2	tidyR	45
5.2.1	Gather	46
5.2.2	Spread	46
5.2.3	Unite	47
5.2.4	Separate	47
5.2.5	Lidando com valores omissos.....	48
6	Gráficos em R.....	50
6.1	ggplot	50
7	Analizando dados com R	58
7.1	Gerando Tabelas com Xtable e Stargazer.....	58
7.2	Gerando Dashboards utilizando o Shiny	63
7.2.1	Estrutura de um aplicativo shiny	63
7.2.2	Layouts do shiny.....	64
7.2.3	Alguns links úteis.....	66
8	Onde Aprender Mais?	69
	Referências Bibliográficas	71

1 Introdução

O R é uma suíte integrada de software que permite a recuperação, o tratamento, e a análise de dados[VS11]. Pode se dizer que o R é um ambiente de tratamento de dados que permite ao usuário, além a análise de dados propriamente dita, escrever extensões e ampliar o seu escopo.

R é uma ferramenta de software livre que atende aos critérios da *Free Software Foundation* e tem uma licença **GNU**¹

Algumas das funcionalidade do R são[VS11]:

- Ferramentas para manuseio e armazenamento de dados
- Um conjunto de operadores que permitem o cálculo numérico e a manipulação de matrizes
- Um enorme conjunto de bibliotecas para análise de dados
- Ferramentas para apresentação gráfica de dados e resultados
- Uma linguagem de programação orientada a objetos e extensível
- A possibilidade de estender a linguagem, suas bibliotecas e funções

O R tem um conjunto extenso de ferramentas que permitem desde uma análise simples de regressão ou de aglomerados (*cluster*) até apresentações de resultados extraídos de bancos de dados em larga escala com ferramentas de busca *SQL*² apresentadas em páginas web (Figura 1).

Neste texto, iremos apresentar um conjunto de ferramentas de uso livre, a sua maioria de código aberto, que permitirão a extração, tratamento, análise e apresentação de dados, tanto para análises estatísticas mais diretas quanto para a tomada de decisões estratégicas de negócios.

¹<https://www.gnu.org/>

²Structured Query Language - <https://pt.wikipedia.org/wiki/SQL>

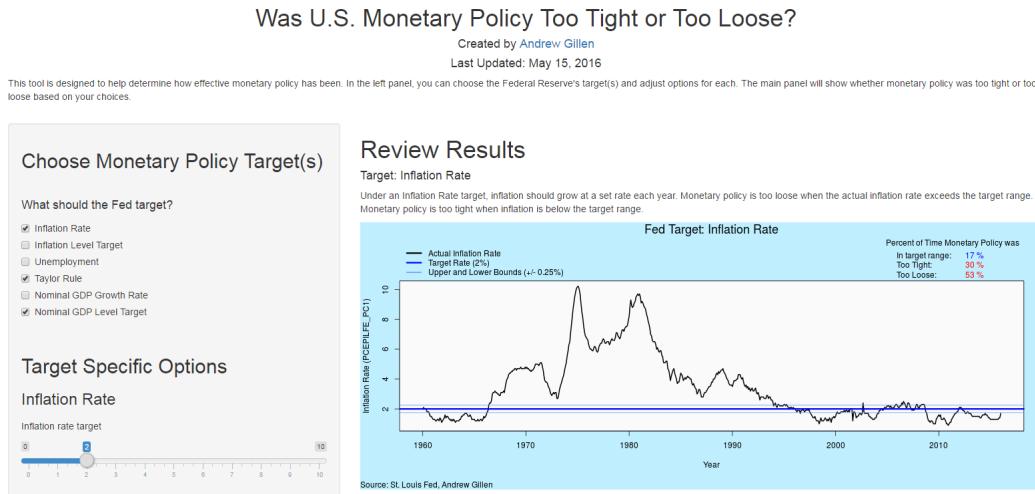


Figura 1. Exemplo de Dashboard de Política monetária usando R [Sho19]

2 Instalando e Configurando o R e RStudio

Nesta aula os alunos aprenderão a baixar o R e RStudio bem como aprenderão a utilizar bibliotecas em R

2.1 Instalando o R e RStudio

2.1.1 Instalando o R

Para instalar o R é preciso acessar o site <https://www.r-project.org/>. Os passos estão enumerados a seguir:

Nesta página clicar em **download R** que acessa a página - <https://cran.r-project.org/mirrors.html> - (Figura 2)

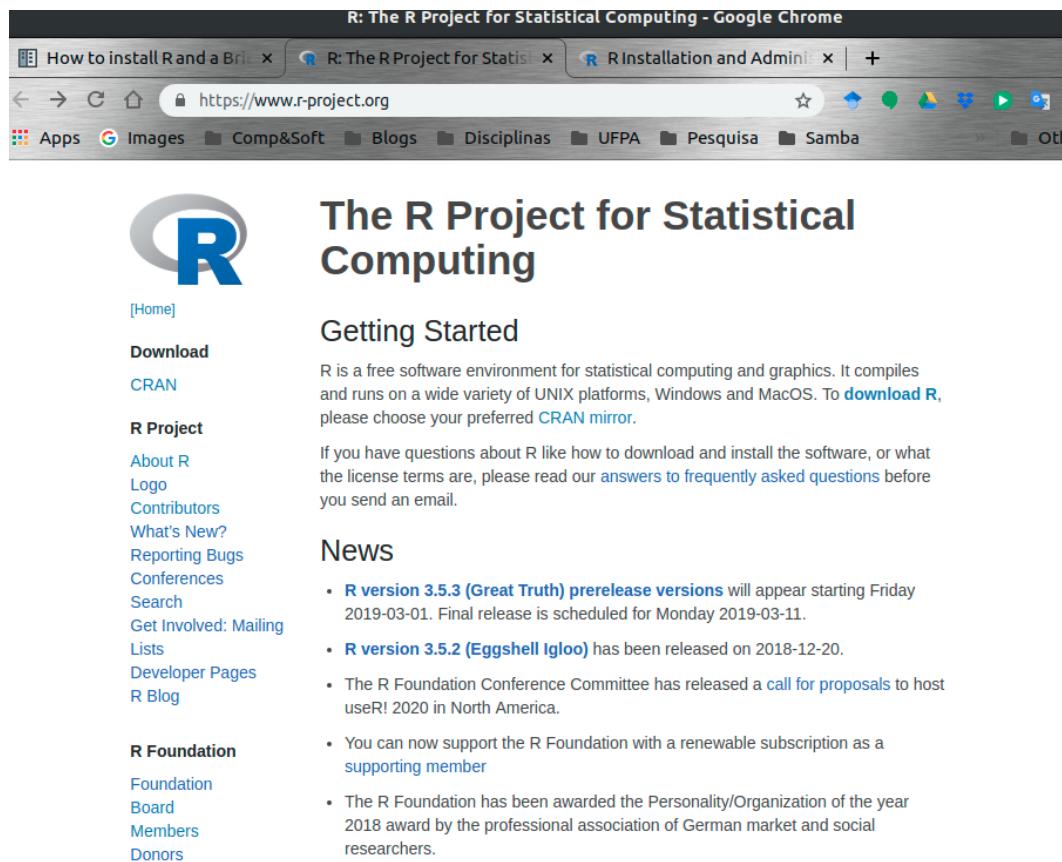


Figura 2. Nesta página clicar em download R

Nesta página você seleciona o espelho que utilizará para baixar o R. Há sites no Brasil ou sítios em nuvem o item 0 - Cloud redirecionará automaticamente para o espelho mais próximo.

mente para sítios apoiados pelo Rstudio - <https://cloud.r-project.org/> (Figura 3) - A diferença entre os prefixos *http* e *https* é que os sites com "s" utilizam encriptação. Muitas vezes é necessário utilizar o *https* em instalações cujos ambientes de TI exigem.

Country	Mirror URL	Sponsor
0-Cloud	https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
0-Cloud	http://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
Algeria	https://cran.usthb.dz/	University of Science and Technology Houari Boumediene
Algeria	http://cran.usthb.dz/	University of Science and Technology Houari Boumediene
Argentina	http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	https://cran.csiro.au/	CSIRO
Australia	http://cran.csiro.au/	CSIRO
Australia	https://mirror.aarnet.edu.au/pub/CRAN/	AARNET
Australia	https://cran.ms.unimelb.edu.au/	School of Mathematics and Statistics, University of Melbourne
Australia	https://cran.curtin.edu.au/	Curtin University of Technology
Austria	https://cran.wu.ac.at/	Wirtschaftsuniversität Wien
Austria	http://cran.wu.ac.at/	Wirtschaftsuniversität Wien
Belgium	https://www.freestatistics.org/cran/	Patrick Wessa
Belgium	http://www.freestatistics.org/cran/	Patrick Wessa
Belgium	https://lib.ugent.be/CRAN/	Ghent University Library
Belgium	http://lib.ugent.be/CRAN/	Ghent University Library
Brazil	http://nbcgil.uesc.br/mirrors/cran/	Computational Biology Center at Universidade Estadual de Santa Cruz
Brazil	https://cran-r.c3sl.ufpr.br/	Universidade Federal do Paraná
Brazil	http://cran-r.c3sl.ufpr.br/	Universidade Federal do Paraná
Brazil	https://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
Brazil	http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
Brazil	https://vps.fmvz.usp.br/CRAN/	University of São Paulo, São Paulo
Brazil	http://vps.fmvz.usp.br/CRAN/	University of São Paulo, São Paulo
Brazil	https://brieger.esalq.usp.br/CRAN/	University of São Paulo, Piracicaba
Brazil	http://brieger.esalq.usp.br/CRAN/	University of São Paulo, Piracicaba

Figura 3. Aqui você seleciona o espelho que vai usar para baixar o R

Você poderá baixar o R (por exemplo em <https://cloud.r-project.org/>) ou em algum dos outros espelhos citados anteriormente, de acordo com seu sistema operacional (Figura 4) Cada sistema (*Linux*, *MacOSX* ou *Windows* tem uma rotina diferente para instalação. Aqui vamos explicar o sistema operacional mais comum (*Windows*). Para outros sistemas, recomenda-se buscar instruções específicas em fóruns adequados.

Finalmente, para instalar o R, você pode executar o arquivo que está no link apresentado na Figura 5. Você baixará um arquivo **.exe** e, ao executá-lo, o instalador do *Windows* tornará o programa disponível para uso na sua máquina.

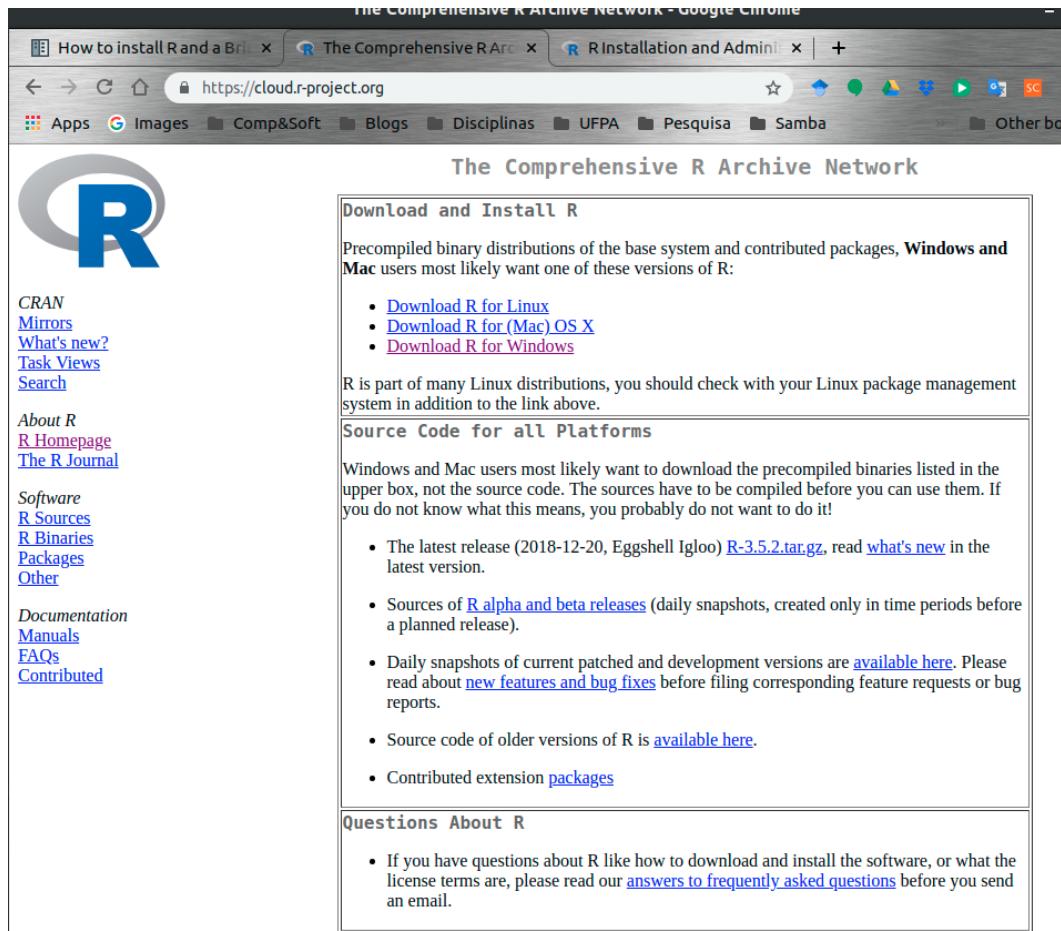


Figura 4. Sítio para baixar o R de acordo com seu sistema operacional

R-3.5.2 for Windows (32/64 bit)

[Download R 3.5.2 for Windows](#) (79 megabytes, 32/64 bit)
[Installation and other instructions](#)
[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Figura 5. Página do Download do R

Você pode conseguir mais informações sobre o R e os detalhes da instalação checando nos *FAQs*³ dos sítios que você estiver acessando <https://cloud.r-project.org/bin/windows/base/rw-FAQ.html>, ou em <https://cran.r-project.org/doc/manuals/R-admin.html>

2.1.2 Instalando o RStudio

O RStudio é um ambiente de desenvolvimento integrado para o R. Incorpora um conjunto de funções que facilita o desenvolvimento de programas em R e a sua execução. O RStudio não é uma interface gráfica para a execução de rotinas do R, é um ambiente que permite implementar e executar programas. É um pacote comercial com partes de uso livre onde é necessário pagar pelos recursos extras.

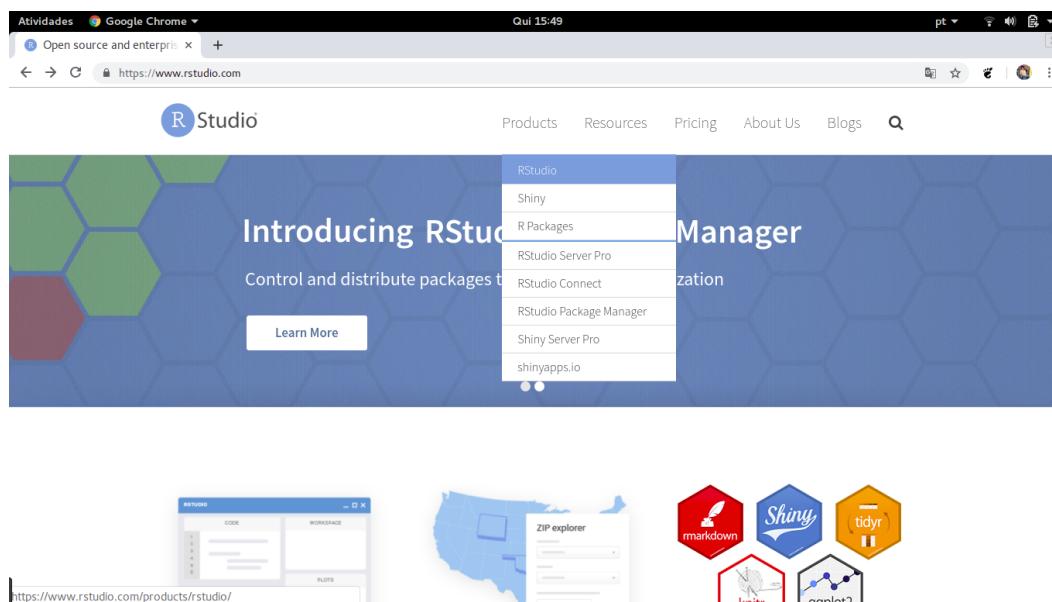


Figura 6. Página inicial do Rstudio

Para iniciar o download da plataforma Rstudio é preciso acessar o site (<https://www.rstudio.com/>). Na parte superior há a aba **Products** que dará acesso ao item **Rstudio** como pode ser visto na figura 6.

Após clicar no item **Rstudio** na aba **Products**, você será redirecionado para a página da figura 7. Nesta etapa é preciso clicar na opção **Rstudio Desktop**.

³Frequently Asked Questions

The screenshot shows the RStudio website at <https://www.rstudio.com/products/rstudio/>. At the top, there's a navigation bar with links for Products, Resources, Pricing, About Us, and Blogs. Below the navigation, the word "RStudio" is prominently displayed. Underneath, there's a heading "Take control of your R code" followed by a brief description of what RStudio is: "RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. Click here to see more RStudio features." Below this, there are two main options: "Desktop" and "Server". Each option has a circular icon, a title, a brief description, and a "View Details" link.

Figura 7. Aba de produtos do Rstudio

The screenshot shows the RStudio Desktop product page. At the top, there's a navigation bar with links for Products, Resources, Pricing, About Us, and Blogs. Below the navigation, the word "RStudio Desktop" is displayed. The main content area compares the "Open Source Edition" and the "Commercial License". The "Overview" section lists features for both. The "Support" section shows "Community forums only" for open source and "Priority Email Support" with an 8-hour response time for commercial. The "License" section indicates "AGPL v3" for open source and "RStudio License Agreement" for commercial. The "Pricing" section shows "Free" for open source and "\$995/year" for commercial. At the bottom, there are two prominent blue buttons: "DOWNLOAD RSTUDIO DESKTOP" and "BUY NOW".

Figura 8. Diferença entre planos

Após clicar em **Rstudio Desktop** na figura 7, será possível ter acesso a figura 8 que mostra algumas das principais diferenças entre a plataforma de código aberto e a plataforma de licença comercial. Ainda na figura 8, para

continuar o processo de download, é preciso clicar na aba **DOWNLOAD RSTUDIO DESKTOP**.

Após os procedimentos da figura 8, você terá acesso a interface da figura 9 que mostrará cada plano oferecido pelo **Rstudio**, bem como os benefícios e preços. O plano utilizado nesse curso será o primeiro da esquerda para direita, intulado **RStudio Desktop Open Source License**. Então para ter acesso ao download desse plano, basta clicar na respectiva aba de **download** do plano.



Figura 9. Planos, preços e benefícios

Por fim, após os passos da figura 9, basta escolher o sistema operacional conforme a figura 10

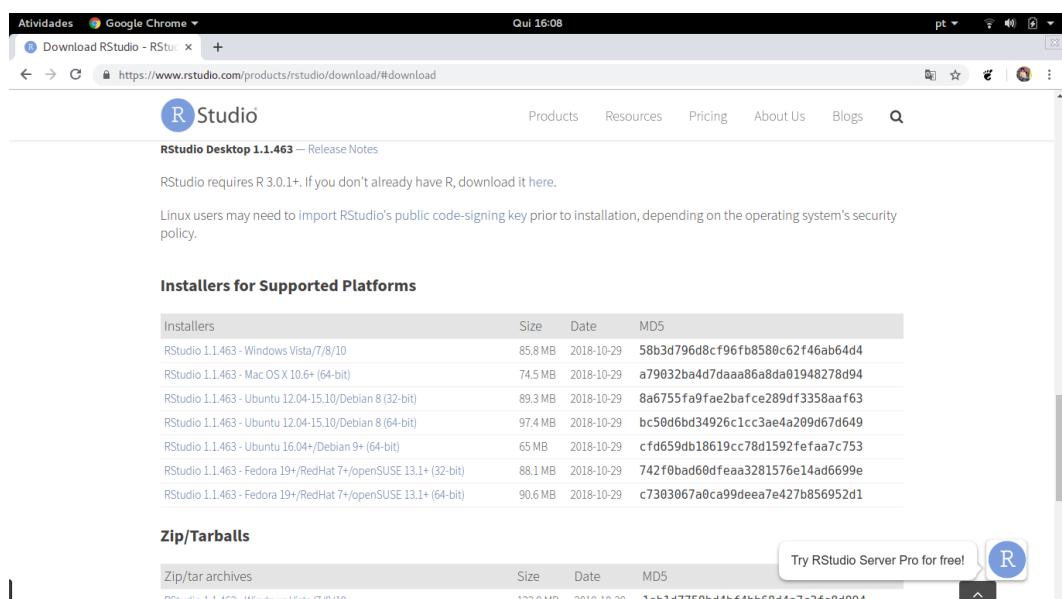


Figura 10. Fim! Escolha o sistema operacional

Após o download basta executar o arquivo **RStudio-1.1.463.exe** e fazer a instalação padrão.

2.2 Checando a Instalação Existente e os Requisitos

Muitas vezes é possível que você já tenha uma versão do R e do RStudio instaladas. Um problema que pode ocorrer é você não ter a última versão do programa. Caso você já tenha o R instalado, é uma boa estratégia atualizar os pacotes para a nova versão (falaremos de pacotes na seção 2.3).

Para checar a instalação do R você pode executar o seguinte comando:

```
> sessionInfo()

R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.2 LTS

Matrix products: default
BLAS:    /usr/lib/x86_64-linux-gnu/openblas/libblas.so.3
LAPACK:  /usr/lib/x86_64-linux-gnu/libopenblas-r0.2.20.so

locale:
[1] LC_CTYPE=pt_BR.UTF-8        LC_NUMERIC=C
[3] LC_TIME=pt_BR.UTF-8        LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=pt_BR.UTF-8     LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=pt_BR.UTF-8       LC_NAME=C
[9] LC_ADDRESS=C                LC_TELEPHONE=C
[11] LC_MEASUREMENT=pt_BR.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats      graphics   grDevices utils      datasets  methods   base

other attached packages:
[1] knitr_1.22

loaded via a namespace (and not attached):
[1] compiler_3.6.0 tools_3.6.0    xfun_0.4
```

>

O comando dará as informações da versão, plataforma, Bibliotecas, e Locales⁴

2.3 Pacotes no R

A maior parte da funcionalidade existente no R é fornecida por programas, funções e bibliotecas agrupadas em pacotes. Estes pacotes são, em boa parte, resultado do esforço colaborativo de milhares de pesquisadores e entusiastas no mundo. Estes pesquisadores desenvolvem e compartilham estes pacotes na CRAN⁵. Lá você pode encontrar, entre os mais de 13 mil pacotes existentes, aquilo que provavelmente vai resolver seu problema.

2.3.1 O conceito de pacote e para que serve

Um pacote em R é um conjunto de funções e arquivos de dados (*Datasets*), desenvolvido pela comunidade do R e que amplia o conjunto de funcionalidades do software. Pacotes executam funções específicas (como determinados tipos de funções estatísticas) ou mesmo todo um conjunto de funcionalidades para tarefas mais gerais, como produção de gráficos (ggplot2) ou tratamento de dados (dplyr e tidyr).⁶

uma boa introdução sobre desenvolvimento, estrutura, e implementação de pacotes em R pode ser encontrada em <http://r-pkgs.had.co.nz/>.

2.3.2 Como sei que pacotes eu preciso?

Há uma profusão enorme de pacotes no R. Em geral há um padrão de documentação nos pacotes que permitem ao usuário compreender a finalidade e a forma de uso do pacote. Alguns pacotes mais difundidos, como o *ggplot2* e o *xtable*. Coisas mais específicas, porém, como algoritmos genéticos (*ga*, *genalg*) ou escores de propensão (*MatchIt*) precisam ser encontradas a partir do domínio específico que se precisa.

Uma boa pedida é usar numa ferramenta de busca específica com termos como *a função que eu quero* em R. (A busca em inglês é bem mais eficiente e dera mais resultados).

⁴Os *Locales* dão detalhes sobre a codificação de caracteres, data, moeda, etc...

⁵The Comprehensive R Archive Network

<https://cran.r-project.org/web/packages/index.html>

⁶ <https://www.datacamp.com/community/tutorials/r-packages-guide>

<https://www.rstudio.com/products/rpackages/>

The screenshot shows a web browser window with the URL <https://cran.r-project.org/web/packages/index.html> in the address bar. The title bar of the browser says "Contributed Packages". The page content includes sections for "Available Packages", "Installation of Packages", "Package Check Results", "Writing Your Own Packages", and "Repository Policies". It also lists "Related Directories" such as "Archive", "Orphaned", "bin/windows/contrib", "bin/macos/el-capitan/contrib", and "bin/linux/contrib". The browser's toolbar at the top includes icons for back, forward, search, and other functions, and the address bar shows the current URL.

Contributed Packages

Available Packages

Currently, the CRAN package repository features 13853 available packages.

[Table of available packages, sorted by date of publication](#)

[Table of available packages, sorted by name](#)

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual [R Installation and Administration](#) (also contained in the R base sources) explains the process in detail.

[CRAN Task Views](#) allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 39 views are available.

Package Check Results

All packages are tested regularly on machines running [Debian GNU/Linux](#), [Fedora](#), OS X, Solaris and Windows.

The results are summarized in the [check summary](#) (some [timings](#) are also available). Additional details for Windows checking and building can be found in the [Windows check summary](#).

Writing Your Own Packages

The manual [Writing R Extensions](#) (also contained in the R base sources) explains how to write new packages and how to contribute them to CRAN.

Repository Policies

The manual [CRAN Repository Policy \[PDF\]](#) describes the policies in place for the CRAN package repository.

Related Directories

[Archive](#)

Previous versions of the packages listed above, and other packages formerly available.

[Orphaned](#)

Packages with no active maintainer, see the corresponding [README](#).

[bin/windows/contrib](#)

Windows binaries of contributed packages

[bin/macos/el-capitan/contrib](#)

OS X El Capitan binaries of contributed packages

Figura 11. Página dos Pacotes Mantidos por pessoas que contribuem com o R (CRAN)

Mais informações podem ser encontradas abaixo:

- <https://blog.revolutionanalytics.com/2017/01/cran-10000.html>
- https://cran.r-project.org/web/packages/available_packages_by_name.html
- <https://cran.r-project.org/web/packages/>

Além do *CRAN*⁷, uma outra fonte importante para programas no R é o *GitHub*⁸. Para utilizar o *GitHub*, porém, é necessário baixar o pacote *devtools*. Na seção 2.3.3, falaremos mais sobre baixar e instalar pacotes no R.

2.3.3 Baixando os pacotes

O R provê uma maneira de baixar os pacotes diretamente do terminal de linha de comando. É o comando *install.packages*. Abaixo, um exemplo:

```
> install.packages("ggplot2")
> library(ggplot2)
```

Acima está o exemplo de instalação do pacote *ggplot2* utilizando o R na linha de comando. O mesmo pacote, instalado via RStudio está no exemplo abaixo (figura 12)

Para utilizar o pacote é necessário disponibilizá-lo na seção do R que você está executando. Isto é feito com o comando *library(nomeDoPacote)*.

Para o *GitHub*, algum pacote específico para acessar o site deve ser instalado, o mais comum é o *devtools()*. Utilizando o *devtools* é possível acessar pacotes que não estão no CRAN.

Você pode encontrar mais informações sobre o assunto em:

- <https://www.r-bloggers.com/installing-r-packages/>
- <https://www.r-bloggers.com/how-to-install-and-include-an-r-package/>

⁷<https://cran.r-project.org/>

⁸<https://github.com/>

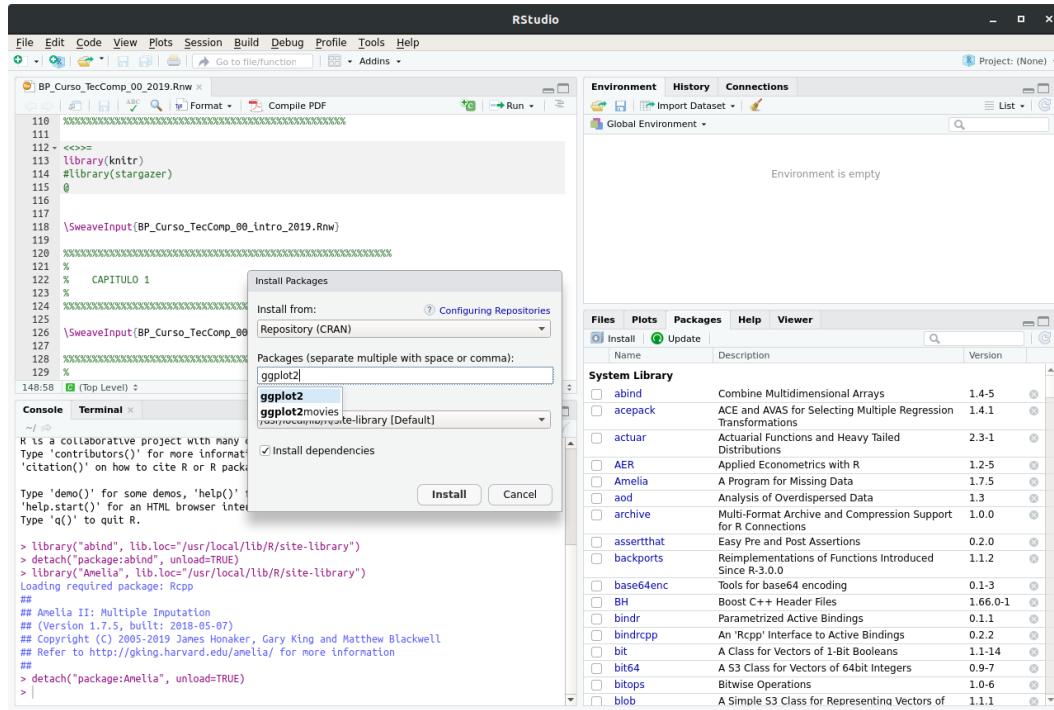


Figura 12. Instalando o pacote ggplot2 via Rstudio

- http://kbroman.org/pkg_primer/pages/build.html

Para a instalação de pacotes a partir do *GitHub* você pode utilizar o pacote *devtools*. Há outros pacotes disponíveis, mas o *devtools* é o mais utilizado no momento.

```
> install.packages("devtools")
> library(devtools)
```

Depois de instalar a biblioteca é possível fazer uso do pacotes R disponíveis no *GitHub*. Depois do pacote instalado e a biblioteca ativada no R, você pode baixar os pacotes do github com os seguintes comandos.

```
> install_github("DeveloperName/PackageName")
> githubinstall("PackageName")
```

O primeiro *install_github()* instala o pacote a partir do nome do usuário github, mais o nome do pacote, o segundo *githubinstall()* o faz a partir do nome do pacote.

Há alguns pacotes que estão no *GitHub* e no **CRAN**, a diferença é que, em geral, no *GitHub* a verão acessível é a mais recente ou de desenvolvimento. A desvantagem é que os pacotes do *GitHub*, como são versões de desenvolvimento, podem, eventualmente ter problemas.

Algumas boas fontes de pacotes do *GitHub* são:

- <https://github.com/trending/r>
- <https://github.com/hadley>

2.3.4 Resolvendo problemas de compilação

A instalação de pacotes em R é feita usualmente com a função *install.packages()*⁹. Nesta função, informamos uma *cadeia de caracteres* com o nome do pacote que queremos instalar (por exemplo - *install.packages("yaml")*). Há diversos tipos e pacotes em R, muitos destes pacotes podem ter sido escritos em R ou C, C++ ou Fortran. No caso de pacotes escritos em C, C++ e Fortran instalados com o tipo *fonte*, é necessária a compilação deste pacote. O R faz isso automaticamente, não é necessária nenhuma intervenção do usuário.

Certos pacotes, como *ggplot2* são escritos em R e não necessitam de uma compilação. Pode haver, porém, pacotes que necessitam de bibliotecas do sistema operacional ou programas (como o Java, por exemplo) que não estejam instaladas no seu computador. Neste caso, é necessário instalar a biblioteca utilizando esta função no seu sistema operacional, para depois instalar o pacote no R.

Abaixo um caso típico.

Neste caso acima, o pacote *fftw*¹⁰, para utilização de transformadas de Fourier necessita de uma biblioteca (*fftw3*) que deve ser instalada no sistema operacional, antes de se instalar o pacote R. Você pode descobrir isto olhando as mensagens da instalação do pacote. Nestas mensagens há a informação de que **"checking for FFTW... configure: error: Package requirements (fftw3) were not met."** e a instalação terminou com *status* "não-zero", o que significa erro. Para corrigir o erro será necessário instalar a biblioteca que o sistema informa que está faltando.

⁹<https://www.rdocumentation.org/packages/utils/versions/3.5.3/topics/install.packages>

¹⁰<https://cran.r-project.org/web/packages/fftw/index.html>

```

> install.packages('fftw')
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
trying URL 'https://cloud.r-project.org/src/contrib/fftw_1.0-5.tar.gz'
Content type 'application/x-gzip' length 38684 bytes (37 KB)
=====
downloaded 37 KB

* installing *source* package 'fftw' ...
** package 'fftw' successfully unpacked and MD5 sums checked
  checking for gcc... gcc -std=gnu99
  checking whether the C compiler works... yes
...
etc
...
  checking for pkg-config... /usr/bin/pkg-config
  checking pkg-config is at least version 0.9.0... yes
  checking for FFTW... configure: error: Package requirements (fftw3) were not met:
    No package 'fftw3' found
...
etc
...
ERROR: configuration failed for package 'fftw'
* removing '/usr/local/lib/R/site-library/fftw'

The downloaded source packages are in
'/tmp/RtmpT62d7h/downloaded_packages'
Warning message:
In install.packages("fftw") :
  Installation of package 'fftw' had non-zero exit status

```

Figura 13

Mais informação pode ser encontrada abaixo:

- <https://stackoverflow.com/questions/23135703/package-install-error-compilation->
- [https://support.rstudio.com/hc/en-us/community/posts/200522573-Can't-install-p](https://support.rstudio.com/hc/en-us/community/posts/200522573-Can-t-install-p)
- <http://mazamascience.com/WorkingWithData/?p=1185>

2.3.5 Utilizando os pacotes no seu programa R

Para utilizar um pacote em R você necessita simplesmente executar o comando *library(nomeDoPacote)*.

Eventualmente, se você precisar retirar o pacote da memória do seu computador, se não quiser mais utilizá-lo durante a execução daquele programa é só utilizar o comando *detach(package:nomeDoPacote)*

3 Produzindo Relatórios Usando R Markdown

Neste capítulo discutiremos dois aspectos importantes do trabalho com dados em R. Primeiramente veremos as funcionalidades implementadas no R para a produção de relatórios dinâmicos, execução de comandos R e apresentação de resultados utilizando uma linguagem simples chamada **Markdown**¹¹. Depois veremos as ferramentas para apresentação de resultados gerados no R em tabelas, com a utilização dos pacotes xtable e stargazer.

R Markdown é um formato de arquivo que permite a produção documentos dinâmicos utilizando R. Um documento R Markdown é escrito na linguagem **Markdown** e contém partes de código R incorporado em seções específicas do arquivo, chamadas *chunks*:

Markdown é duas coisas: (1) uma sintaxe de formatação de texto simples; e (2) uma ferramenta de software, escrita em Perl, que converte a formatação de texto simples em HTML. Consulte a página Sintaxe para obter detalhes relacionados à sintaxe de formatação do Markdown. Você pode experimentá-lo agora mesmo, usando o Dingus online .

A meta de design predominante da sintaxe de formatação do Markdown é torná-la o mais legível possível. A ideia é que um documento formatado com Markdown seja publicável como está, como texto simples, sem parecer ter sido marcado com tags ou instruções de formatação. Embora a sintaxe do Markdown tenha sido influenciada por vários filtros de texto para HTML existentes, a maior fonte de inspiração para a sintaxe do Markdown é o formato do email em texto simples (Gruber, 2004).¹²

O uso da linguagem **Markdown** para a produção de documentos quando se analisa dados utilizando o R, permite uma transição suave e contínua entre o processo de análise e a elaboração de um Relatório de resultados. Neste sentido um arquivo *Rmarkdown* (Rmd) funciona como uma espécie de *bloco de notas* que pode evoluir para o formato de um relatório final em um arquivo Html, PDF, docx, etc. Isto economiza tempo e esforço.

¹¹Uma definição do Markdown pode ser achada em : <https://tools.ietf.org/html/rfc7763>

¹² <https://daringfireball.net/projects/markdown/>

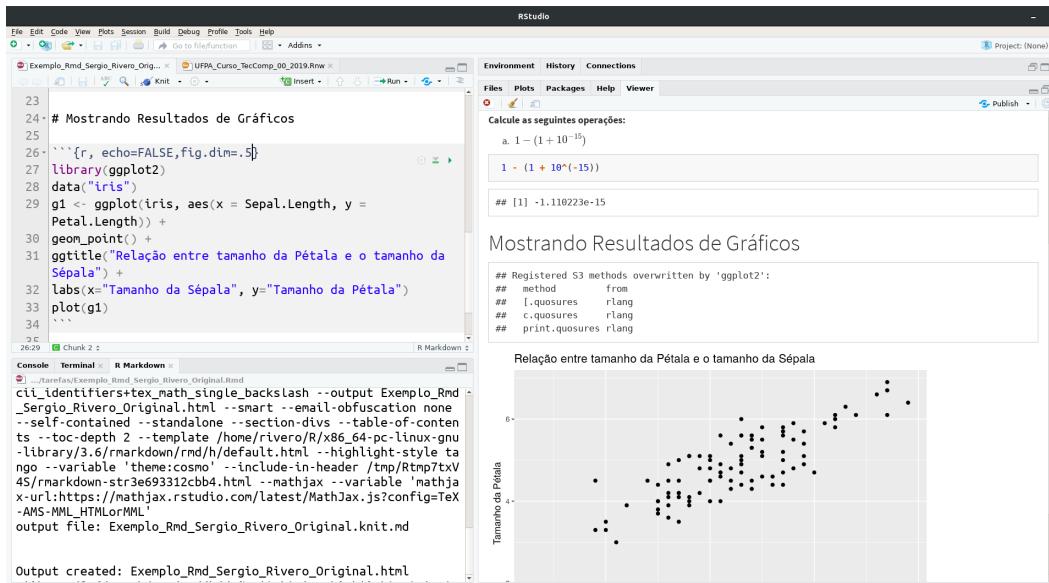


Figura 14. Um exemplo de saída de um .Rmd no Rstudio

3.1 O que é o Rmarkdown?

O *Rmarkdown* (RMD) é a implementação do **Markdown** em R. R markdown [ACX⁺16] é um formato de documento que foi introduzido inicialmente no pacote *knitr* [Xie16, Xie17]. O Knitr, na verdade suporta um conjunto grande de linguagens, como L^AT_EX, HTML, AsciiDoc, reStructuredText e Textile.

A ideia por trás do RMD (figura 16) é embutir comandos em uma linguagem qualquer (no nosso caso, o R) dentro de um documento. Assim, um arquivo .Rmd teria, ao mesmo tempo, comando que gerariam resultados como gráficos e tabelas, que seriam gerados num arquivo de resultados e compilados posteriormente num documento PDF, HTML, Word, etc. A grande vantagem do RMD é sua simplicidade.

Documentos no R Markdown são salvos com um sufixo .Rmd. Este arquivo tem textos e comandos do Markdown e código R em partes específicas chamadas *chunks*.

Para a instalação do Rmarkdown você pode simplesmente utilizar o comando *install.packages()*. Você pode instalá-lo diretamente do CRAN

```
> install.packages("rmarkdown")
```

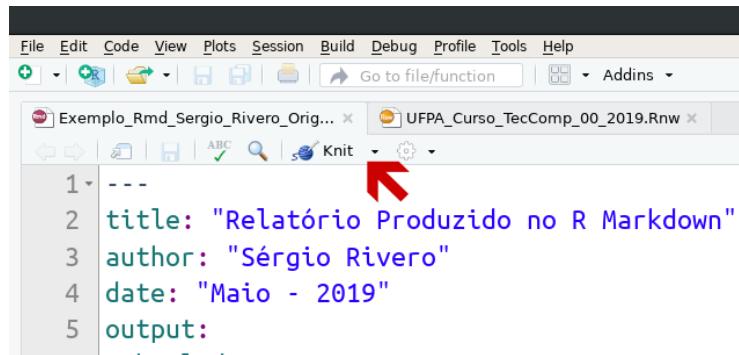


Figura 15. O Botão Knit no Rstudio

Ou instalar a versão de desenvolvimento do *GitHub*. Para isso é necessário utilizar o pacote *devtools*.

```

> if (!requireNamespace("devtools")){
+     install.packages('devtools')
+     devtools::install_github('rstudio/rmarkdown')
+ }

```

Para conseguir gerar o pdf no windows a partir de um arquivo RMD é necessário instalar o L^AT_EX. O L^AT_EX é um conjunto de pacotes que permite a edição e geração de textos formatados de alta qualidade.

3.2 Como o Rmd funciona?

O processo de geração de documentos com o RMD é feito a partir da pressão do botão **Knit** (figura 15) no RStudio. Para isso, você deve ter habilitado e configurado o Markdown no app. A partir daí, o arquivo arquivo .rmd (Rmarkdown) tem seus códigos R executados e gera um código .md com os resultados da execução do R (figuras, tabelas, etc) que é então processado pelo pandoc. Esse processamento, gera o arquivo de resultado desejado (HTML, DOC, PDF, etc)

A partir do arquivo .Rmd, você pode gerar resultados que combinem, textos, resultados de operações em R, tabelas geradas pelas bibliotecas do R (regressões, outras coisas), bem como gráficos.

Estes programas permitem um caminho mais suave entre a análise e a publicação dos resultados.

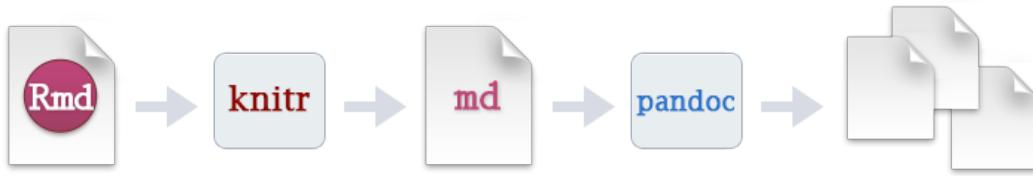


Figura 16. O processo de geração de documentos no RMD

```

---
title: "Relatório Produzido no R Markdown"
author: "Sérgio Rivero"
date: "Maio - 2019"
output:
  html_document:
    highlight: tango
    theme: cosmo
    toc: yes
    toc_depth: 2
subtitle: Exemplos de Documentos Produzidos no R Markdown
---
  
```

Figura 17. Um exemplo de Cabeçalho de um arquivo .Rmd

3.3 A estrutura de um arquivo .Rmd

Um arquivo .Rmd tem basicamente, 3 partes.

1. Um cabeçalho escrito em YAML¹³. Separados por “---” no início e no final do bloco.
2. Textos e comandos em Linguagem Markdown
3. *Chunks* em uma linguagem específica (no nosso caso, Comandos em R) - Separados por “` ` ` ” no início e no final do bloco.

O cabeçalho (figura 17) é posto uma vez e contém definições gerais do arquivo executado. Há vários componentes do cabeçalho, sendo que os principais são o título, o nome do autor, a data e qual será a saída do arquivo executado (HTML, PDF, MSWord, etc)

3.4 Sintaxe de Markdown

O texto em um documento R Markdown é gravado com a sintaxe Markdown. Precisamente falando, é o Markdown do Pandoc. Há muitos sabores de Markdown inventados por pessoas diferentes, e o sabor do Pandoc é

¹³YAML é um formato de serialização de dados legíveis por humanos inspirado em linguagens como XML, C, Python, Perl, assim como o formato de correio eletrônico especificado pela RFC 2822. YAML foi proposto por Clark Evans em 2001 em conjunto com Ingy döt Net e Oren Ben-Kiki (Wikipedia) - Mais informações em: <https://yaml.org/>

o mais abrangente para o nosso conhecimento. Você pode encontrar a documentação completa do Markdown do Pandoc em <https://pandoc.org/MANUAL.html>. É altamente recomendável que você leia esta página pelo menos uma vez para conhecer todas as possibilidades com o Markdown da Pandoc, mesmo que você não use todas elas. Esta seção é adaptada da Seção 2.1 de Xie (2016) e abrange apenas um pequeno subconjunto da sintaxe Markdown do Pandoc.

Formatação em linha

O texto embutido será em itálico se estiver cercado por sublinhados ou asteriscos, por exemplo, `_text_` ou `*text*`. O texto em negrito é produzido usando um par de asteriscos duplos (`**text**`). Um par de tis (`~~`) transformar texto para um subscrito (por exemplo, $H^3~P0^4$ torna H_3PO_4). Um par de circunflexos (`^`) produz um sobreescrito (por exemplo, Cu^{2+} renderiza Cu^{2+}).

Para marcar o texto como código *inline*, use um par de backticks, por exemplo ``code``

Os hiperlinks são criados usando a sintaxe `[text](link)`, por exemplo `[RStudio](https://www.rstudio.com)`

A sintaxe das imagens é semelhante: basta adicionar um ponto de exclamação, por exemplo `![alt text or image title](path/to/image)`,

As notas de rodapé são colocadas dentro dos colchetes após um acento circunflexo `^[]`, por exemplo `^[[This is a footnote.]]`.

Elementos de nível de bloco

Cabeçalhos de seção podem ser escritos após um número de sinais de libra, por exemplo,

```
# First-level header  
  
## Second-level header  
  
### Third-level header
```

Se você não quiser que um determinado título seja numerado, você pode adicionar `{-}` ou `{.unnumbered}` após o cabeçalho, por exemplo,

```
# Preface {-}
```

Os itens da lista não ordenada começar com `*`, `-ou +`, e você pode aninhar uma lista dentro de outra lista pelo recuo da sub-lista, por exemplo,

- um item
- um item
- um item
 - mais um item
 - mais um item
 - mais um item

A saída é:

- um item
- um item
- um item
 - mais um item
 - mais um item
 - mais um item

Os itens da lista ordenada começam com números (você também pode aninhar listas dentro de listas), por exemplo,

1. o primeiro item
2. o segundo item
3. o terceiro item
 - um item não ordenado
 - um item não ordenado

A saída não parece muito diferente com a fonte Markdown:

1. o primeiro item
2. o segundo item

3. o terceiro item
 - um item não ordenado
 - um item não ordenado

3.5 Os chunks

Um *chunk* é uma parte de código (no nosso caso, R) incluído num arquivo **Rmarkdown**. Chunks iniciam com `{{opções do chunk}} e terminam com ```. Na figura 18 podemos ver um exemplo de um chunk típico gerando um gráfico em R.

```
```{r, echo=FALSE,fig.dim=.5}
library(ggplot2)
data("iris")
g1 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
 geom_point() +
 ggtitle("Relação entre tamanho da Pétala e o tamanho da Sépala") +
 labs(x="Tamanho da Sépala", y="Tamanho da Pétala")
plot(g1)
```

```

Figura 18. Um chunk

Algumas dicas importantes sobre as opções em um *chunk*¹⁴:

1. Opções de *chunk* devem ser escritas em uma linha;
2. não são permitidas quebras de linha dentro das opções de *chunk*;
3. Evite espaços e pontos em rótulos de partes e nomes de diretórios;
4. Se sua saída for um documento TeX, esses caracteres podem causar problemas (em geral, é recomendável usar caracteres alfabéticos com palavras separadas por - e evitar outros caracteres), por exemplo, `setup-options` é um bom label, enquanto `setup.options` e `chunk 1` são ruins;
5. Todos os valores de opções devem ser expressões R válidas, assim como escrevemos argumentos de função;
6. para opções lógicas, deve-se usar TRUE ou FALSE ou T ou F mas, importante, "true"e "false"não funcionam

¹⁴O texto original pode ser encontrado em <https://yihui.name/knitr/options/>

Abaixo falamos de algumas opções relevantes do cabeçalho de um chunk

Primeiro o código pode ou não ser executado. O opção `eval = FALSE` faz com que o código seja avaliado mas não executado. Sem nenhuma opção, o código é avaliado.

Algumas opções que configuram os padrões de saída de um *chunk* são:

- **echo:** (TRUE) se deve incluir o código-fonte R no arquivo de saída; além de TRUE / FALSE que liga / desliga completamente o código-fonte, também podemos usar um vetor numérico para selecionar quais expressões R ecoar em um trecho, por exemplo, `echo = 2:3` significa apenas ecoar a segunda e terceira expressões, e `echo = -4` significa excluir a quarta expressão;
- **results:** ('markup'; character) pega esses valores possíveis:
 - **markup:** marque os resultados usando o gancho de saída, por ex. colocar resultados em um ambiente especial do L^AT_EX;
 - **asis:** saída como é, ou seja, grava os resultados brutos de R no documento de saída;
 - **hold:** segura todas as saídas e as empurra para o final de um *chunk*
 - **hide:** oculta resultados. esta opção só se aplica à saída R normal (não avisos, mensagens ou erros)
- **warning:** (default é VERDADEIRO;) preserva os avisos (produzidos por `warning()`) na saída como se executássemos código R em um terminal (se for FALSO, todos os avisos serão impressos no console ao invés do documento de saída);
- **error:** (default é VERDADEIRO) preserva os erros (de `stop()`). A avaliação não será interrompida mesmo em caso de erros!! Se quisermos que R pare nos erros, precisamos definir essa opção como FALSO;
- **message:** (default é VERDADEIRO) preserva as mensagens emitidas por `message()` (semelhante ao `warning`).

Um detalhamento maior das opções de chunks no Rmarkdown podem ser achados em <https://yihui.name/knitr/options/>

3.6 Algumas referências sobre o Rmarkdown

<https://daringfireball.net/projects/markdown/>

https://rmarkdown.rstudio.com/authoring_quick_tour.html

<https://www.markdownguide.org/getting-started/>

<https://ourcodingclub.github.io/2016/11/24/rmarkdown-1.html>

<https://github.com/yihui/knitr-book>

<https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf>

<https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf>

4 Acessando e Utilizando Bases de Dados

Apresentar o conceito de Dataframe, os tipos de dados utilizados no R e os principais comandos

4.1 O ciclo de tratamento e análise de dados

Atividades que utilizam dados e fazem algum tratamento formal (estatístico, matemático) destes dados são comuns tanto no mundo acadêmico quanto no mundo corporativo. Muitas vezes, os dados correspondem a milhões de observações em diversas bases de dados. Estas bases de dados, muitas vezes têm estruturas complexas e diferentes formas de organização, armazenamento, codificação, mecanismos de busca, etc. Tratar e analisar estes dados exige planejamento, avaliação, organização, o que implica em uma atividade sistemática e no uso de ferramentas adequadas para a tarefa.

Um outro aspecto relevante também é que muitos dados podem ser analisados, usando técnicas, objetivos e/ou perspectivas diferentes. O que significa que os mesmos dados, coletados e tratados, podem ter um reuso para diferentes projetos e análises. Esta possibilidade de reuso torna ainda mais importante o seu tratamento e armazenamento sistematizado.

Podemos então dizer que os dados tem um *ciclo de vida* e este ciclo pode ser pensado como um conjunto de atividades que vão do planejamento de seu uso até a sua análise. Abaixo, na figura 19, apresentamos um exemplo de ciclo de vida que pode ser encontrado em <https://www.dataone.org/data-life-cycle>

Este ciclo de vida tem oito componentes:

O ciclo de vida dos dados DataONE possui oito componentes¹⁵:

- Planejar: descrição dos dados que serão compilados e como os dados serão gerenciados e disponibilizados ao longo de sua vida útil
- Coletar: as observações são feitas manualmente, com sensores ou outros instrumentos ou os dados são adquiridos de instituições que os produzem

¹⁵<https://www.dataone.org/data-life-cycle>

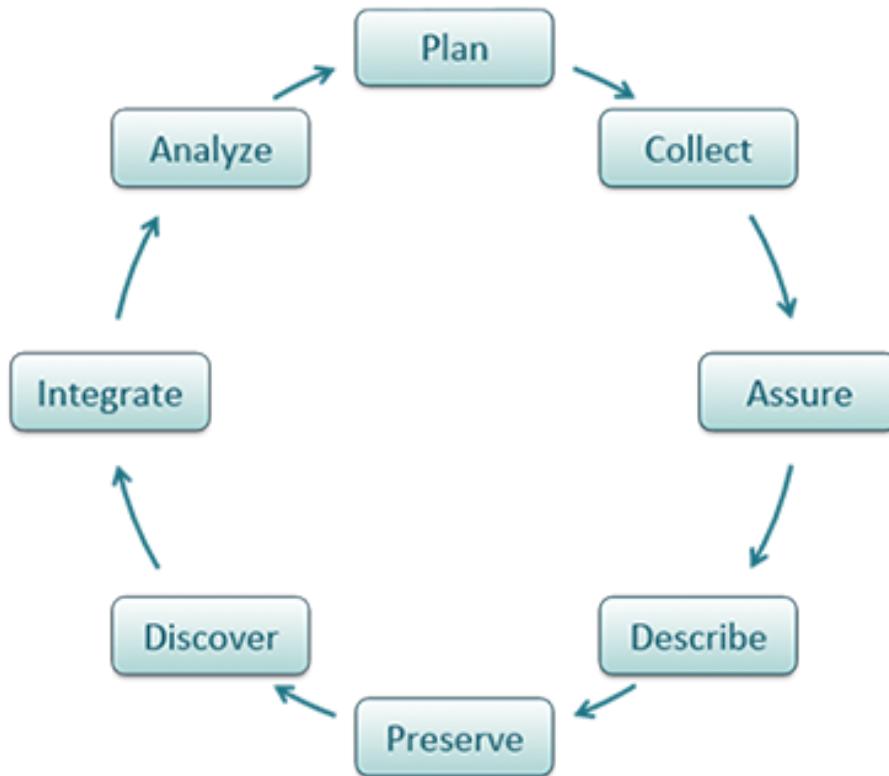


Figura 19. O Ciclo de Tratamento e Análise de Dados

- Validar: a qualidade dos dados é assegurada por meio de verificações e checagens
- Descrever: os dados são descritos com precisão e detalhadamente usando os padrões de metadados apropriados
- Preservar: os dados são submetidos a um arquivo de longo prazo apropriado (ou seja, data center)
- Descobrir: dados potencialmente úteis são localizados e obtidos, juntamente com as informações relevantes sobre os dados (metadados)
- Integrar: dados de fontes diferentes são combinados para formar um conjunto homogêneo de dados que podem ser prontamente analisados
- Analisar: os dados são analisados

Algumas atividades de pesquisa podem usar apenas parte do ciclo de vida; por exemplo, um projeto envolvendo metanálise pode se concentrar nas etapas de Descobrir, Integrar e Analisar, enquanto um projeto focado na coleta e análise de dados primários pode ignorar as etapas de Descobrir e Integrar. Além disso, outros projetos podem não seguir o caminho linear descrito aqui,

ou várias revoluções do ciclo podem ser necessárias. Além disso, alguns cientistas ou equipes (por exemplo, aqueles envolvidos em modelagem e síntese) podem criar novos dados no processo de descobrir, integrar, analisar e sintetizar os dados existentes.

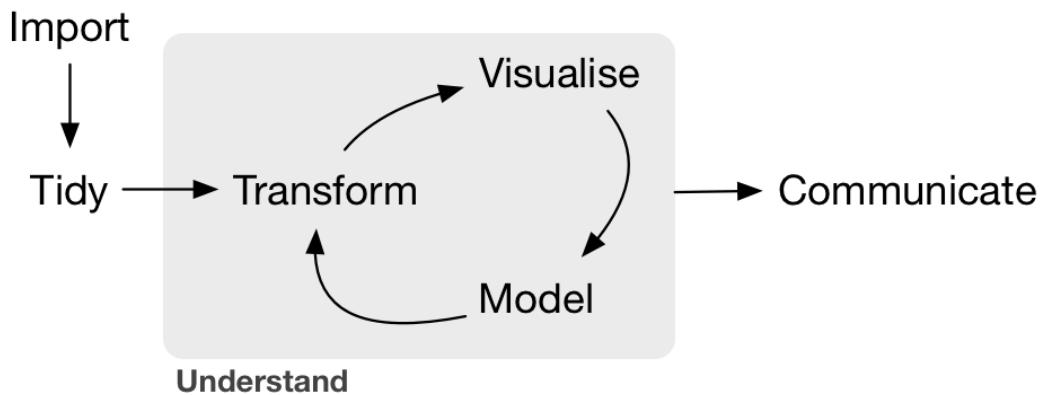


Figura 20. Tratamento de Dados Estatísticos

Uma outra abordagem para o ciclo de vida, mais focada no tratamento de dados estatísticos para modelagem é a que podemos ter em <https://paulvanderlaken.com/2017/07/07/tidyverse-101-simplifying-life-for-users/>. Neste caso, estamos falando de dados já adquiridos e digitalizados de alguma forma. Então a fase de aquisição de dados é feita a partir da importação de dados já coletados. O trabalho de economistas, é, na sua maioria das vezes, feito a partir de dados secundários adquiridos de instituições de pesquisa. Neste caso, esta abordagem (Figura 20) é mais próxima do ciclo normalmente utilizado em análises de dados secundários.

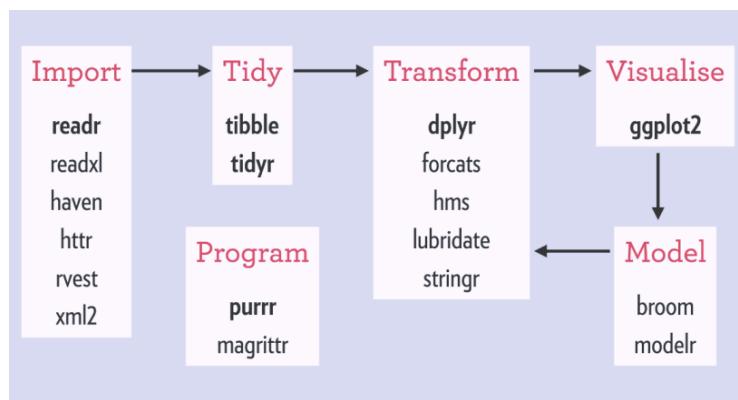


Figura 21. Ferramentas de Tratamento de Dados em R

Como podemos ver na figura 21, temos muitos pacotes para cada tipo de tratamento. A lista acima não é exaustiva. Muito provavelmente há mais pacotes que são especializados para a resolução de problemas mais específicos, como o acesso a bases de dados SQL ou a geração de páginas na internet.

para visualização ou o acesso a arquivos JSON ou YAML. Por isso, é importante sempre avaliar quais os pacotes mais adequados para a solução do seu problema de tratamento de dados.

O *readr*, por exemplo¹⁶ é um pacote genérico para dados em formato matricial (*rectangular data*), tais como arquivos csv, tsv, fwf. Para ler arquivos *xls* e *xlsx* há pacotes como o *readxl* e o *xlsx*¹⁷

O tratamento dos dados pode facilmente consumir a maior parte do tempo de um projeto de análise e modelagem. A existência de ferramentas que facilitem esta tarefa e reduzam o tempo despendido nela é uma ajuda que vem bem a calhar.

Há no R um conjunto de pacotes feitos especialmente para a fase de tratamento dos dados. O *tidyverse* [Wic14] permite a redução dos erros de codificação e a construção de um código mais simples, claro e eficaz, resduzindo assim o tempo necessário para o tratamento e limpeza dos dados¹⁸

O *tidyverse* tem pacotes que fornecem funcionalidades para executar eficazmente as atividades de tratamentos e análise de dados, reduzindo o esforço de codificação e a possibilidade de erros.

A maioria das vezes neste tipo de cenário os dados estão armazenados em formatos externos ao R e devem ser importados¹⁹.

Uma segunda etapa do trabalho é garantir a correção dos dados. Para isso é necessário ”limpar” e ”arrumar” os dados. Nesta fase é necessário eliminar erros, inconsistências de tipos de dados, problemas de codificação e representação dos dados e, eventualmente, gerar novas variáveis a partir dos próprios dados que sejam necessárias para a análise. Este processo é cíclico e também parte da análise. Durante o processo de análise dos dados é possível que surja a necessidade de novas variáveis ou que se descubra erros ainda existentes nos dados de entrada, o que implica em sua correção.

O processo então continua até que se tenha o modelo suficientemente eficaz e testado para que resultados significativos possam ser comunicados.

¹⁶<https://cran.r-project.org/web/packages/readr/index.html>

¹⁷<https://cran.r-project.org/web/packages/readxl/index.html> e <https://cran.r-project.org/web/packages/xlsx/index.html>

¹⁸<https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html>

¹⁹por exemplo, Excel, SPSS, Spyder, Jupyter

Mas uma pergunta é, o que são dados limpos? Os dados limpos têm algumas características fundamentais:

1. Cada coluna da sua base de dados tem apenas um tipo de dado e corresponde a apenas uma variável;
2. Cada linha de sua base de dados corresponde a apenas uma observação;
3. Cada elemento da base corresponde a um dado de uma variável em uma observação.
4. Preferencialmente, o número de observações faltantes é mínimo.

Alguns dos pacotes utilizados para importar dados no R são:

- *haven* - para importar arquivos SPSS, Stata, e SAS;
- *readxl* e *xlsx* - arquivos excel (.xls e .xlsx);
- *DBI* - arquivos de bancos de dados;
- *jsonlite* - arquivos *json*;
- *xml2* - arquivos padrão *XML*
- *httr* - *APIs* na *Web*

Há um conjunto grande de pacotes em R para importação de dados. O ideal é, na fase do planejamento do tratamento dos dados, você definir qual estratégia utilizar para importar os dados para o R.

4.2 Tipos de Dados em R

Os tipos de dados do R incluem: dados escalares, vetores, matrizes, listas e quadro de dados (Dataframe).

4.2.1 Escalares

Determinada variável pode ser um escalar, ou seja, simplesmente um número:

Exemplo de escalares:

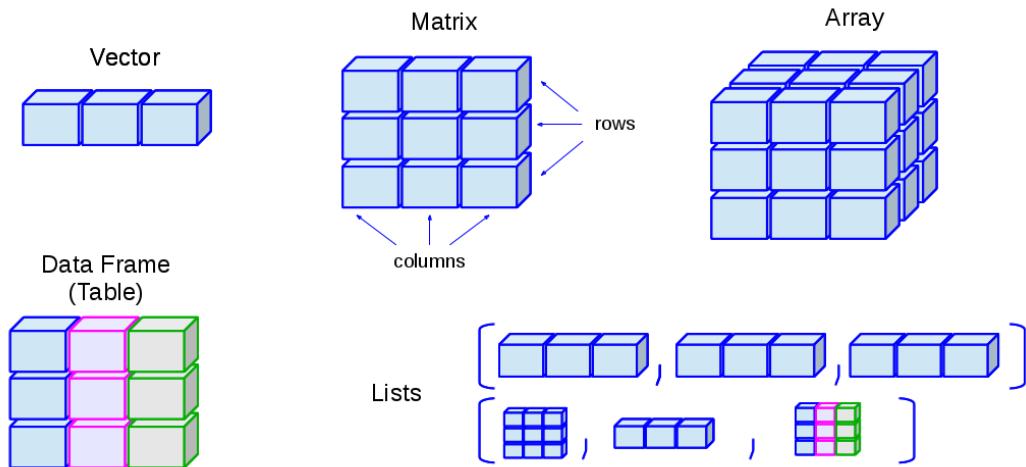


Figura 22. Uma representação gráfica dos tipos de dados em R.

Fonte: <http://venus.ifca.unican.es/Rintro/index.html>

```
> a <- 2
> b <- 2*a
> a
```

```
[1] 2
```

```
> b
```

```
[1] 4
```

4.2.2 Vetores

O vetor é um objeto matemático caracterizado em um conjunto de segmentos orientados de reta que possuem o mesmo módulo, direção e sentido. Ele contém elementos de classes diferentes, conforme apresentado abaixo:

Vetor de classe numérica:

```
> a <- c(1,3500,5.3,543,-2,4000)
```

Vetor de classe de caractér:

```
> b <- c("taxa de juros","taxa de câmbio","reservas bancárias")
```

Vetor de classe lógica:

```
> c <- c(FALSE, TRUE, FALSE, TRUE)
```

4.2.3 Matrizes

São compostas por linhas (valores ordenados na horizontal) representadas pela letra "m" e colunas (valores ordenados na vertical) representadas pela letra "n", onde os dados são convertidos segundo essas disposições.

```
> matriz <- matrix(data=1:16, nrow=4, ncol=4)
> matriz
```

```
 [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Onde:

data: parâmetro que representa os dados usados para criar matriz;

nrow: parâmetro para número de linhas;

ncol: parâmetro para número de colunas.

4.2.4 Array

O dado pode ser um **Array**. Essa estrutura de dados possui três dimensões, as linhas, as colunas e as camadas.

Exemplo de Array:

```
> cubo <- array(data = 1:27, dim=c(3,3,3))
> cubo
```

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

```
, , 2  
  
[,1] [,2] [,3]  
[1,] 10 13 16  
[2,] 11 14 17  
[3,] 12 15 18
```

```
, , 3  
  
[,1] [,2] [,3]  
[1,] 19 22 25  
[2,] 20 23 26  
[3,] 21 24 27
```

data: parâmetro que representa os dados usados para criar matriz;

dim: parâmetro para determinar as dimensões do array, sendo **dim** um vetor;

<https://www.statmethods.net/input/datatypes.html>

<https://swcarpentry.github.io/r-novice-inflammation/13-supply-data-structures/>

https://www.tutorialspoint.com/r/r_data_types.htm

<http://www.r-tutor.com/r/introduction/basic-data-types>

<https://www.cyclismo.org/tutorial/R/types.html>

<https://stat.ethz.ch/R-manual/R-devel/library/base/html/typeof.html>

4.3 Dataframes

Data frame é uma formatação de tabela presente no R que comporta duas dimensões. A primeira dimensão compreende as linhas(Observações) e a segunda dimensão compreende as colunas(variáveis).

Vetor dos meses:

```
> DATA <- c("ago/2018", "set/2018", "out/2018",
+           "nov/2018", "dez/2018", "jan/2019")
> DATA
```

```
[1] "ago/2018" "set/2018" "out/2018" "nov/2018" "dez/2018" "jan/2019"
```

Vetor do IPCA para os respectivos meses do vetor DATA:

```
> IPCA <- c(-0.09, 0.48, 0.45, -0.21, 0.15, 0.32)
> IPCA
```

```
[1] -0.09 0.48 0.45 -0.21 0.15 0.32
```

Vetor do Pib mensal em milhões(R\$) para os respectivos meses do vetor DATA:

```
> PIBmensalMilhoes <- c(583011.3, 551215.6, 597218.7,
+                         604073.9, 624464.1, 591715.7)
> PIBmensalMilhoes
```

```
[1] 583011.3 551215.6 597218.7 604073.9 624464.1 591715.7
```

Unimos os três vetores criados anteriormente em uma única estrutura de dados e conseguinte transformamos essa estrutura em um dataframe por meio dos comandos a baixo:

```
> Dados <- data.frame(cbind(DATA, IPCA, PIBmensalMilhoes))  
> Dados
```

| | DATA | IPCA | PIBmensalMilhoes |
|---|----------|-------|------------------|
| 1 | ago/2018 | -0.09 | 583011.3 |
| 2 | set/2018 | 0.48 | 551215.6 |
| 3 | out/2018 | 0.45 | 597218.7 |
| 4 | nov/2018 | -0.21 | 604073.9 |
| 5 | dez/2018 | 0.15 | 624464.1 |
| 6 | jan/2019 | 0.32 | 591715.7 |

https://www.tutorialspoint.com/r/r_data_frames.htm

<https://www.datamentor.io/r-programming/data-frame/>

<http://www.r-tutor.com/r-introduction/data-frame>

<https://stat.ethz.ch/R-manual/R-devel/library/base/html/data.frame.html>

<https://www.tutorialgateway.org/data-frame-in-r/>

<https://datacarpentry.org/R-ecology-lesson/02-starting-with-data.html>

<https://www.statmethods.net/input/importingdata.html>

4.4 Acessando Arquivos no computador

Para acessar um conjunto de documentos e manusear seus dados no R, primeiramente salve o arquivo no computador ou então obtenha os dados através da internet ou outras fontes. Importante prestar atenção quanto ao tipo de arquivo que se esta trabalhando, pois a prática de importar os dados para o R requer uma própria função a depender do tipo de arquivo, que pode ser: HTML, XML, Json, dados espaçados por TAB, linhas de arquivo de texto, HDF5, SPSS, Stata e outros, com inúmeros tutoriais disponíveis a pesquisa

para mais informações, ficando este material no enfoque da apresentação de dois formatos de arquivos que são bastante utilizados: o csv e o xlsx..

4.4.1 Importando arquivo CSV (Comma separated values) em R

Ao salvar um arquivo em formato csv, execute o comando abaixo no R:

```
> MyData <- read.csv(file="aux/PIB_BR_1996-2016.csv", header=TRUE, sep=",")
```

Onde:

file = Parâmetro que representa o arquivo que se vai importar no R.

Observação: Copie e cole o arquivo ou endereço, conforme o exemplo, que vai ser lido em um quadro de dados chamado MyData

header = Parâmetro que especifica que esses dados incluem uma linha de cabeçalho.

Observação: no trabalho com planilhas, a primeira linha é geralmente reservada para o cabeçalho.

sep = Parâmetro de indicativo que os dados vão ser separados, neste caso, por vírgulas, pois arquivos separados por vírgulas são mais fáceis de trabalhar.

Observação: em seu conjunto de dados evite nomes, valores ou campos com espaços em branco, pois cada palavra será interpretada como uma variável separada, o que resulta em erros relacionados ao número de elementos por linha em seu conjunto de dados. Para concatenar palavras insira . entre as palavras em vez de espaço. Nomes curtos são mais adequados em nomes mais longos. Evitar nomes que contenham símbolos como os seguintes: (\$%^&*() -#?<>/|\[\]{}). Estes símbolos não poderão ser utilizados.

4.5 Importando arquivo xlsx em R

Ao salvar um arquivo em formato xlsx, execute os comandos abaixo no R:

```
> library(xlsx)
> xlsxFile = "aux/PIB_BR_1996-2016.xlsx"
> myFile = read.xlsx(xlsxFile, 1, rowIndex = 1)
```

Onde:

`xlsxFile` = Parâmetro que representa o arquivo `xlsx`, objeto de pasta de trabalho ou URL para o arquivo `xlsx`.

o segundo parâmetro é o número da planilha que você vai importar

o parâmetro ”`rowIndex`”, indica onde você vai começar a ler o arquivo.

A documentação do pacote está em: <https://cran.r-project.org/web/packages/xlsx/xlsx.pdf>

4.5.1 Mais informações

Informações mais detalhadas sobre importação de dados, podem

https://www.datacamp.com/community/tutorials/r-data-import-tutorial?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=1t1&utm_creative=332602034364&utm_targetid=dsa-473406573035&utm_loc_interest_ms=&utm_loc_physical_ms=1001610&gclid=Cj0KCQiA5NPjBRDDARIIsAM9X1GLgVEiwAqyW9CPisvAqFv2mNXzwarSIIaAgdZEALw_wcB

<http://rprogramming.net/read-csv-in-r/>

<https://www.rdocumentation.org/packages/gdata/versions/2.18.0/topics/read.xls>

<https://stat.ethz.ch/R-manual/R-devel/library/utils/html/read.fwf.html>

<https://riptutorial.com/r/example/31447/importing-fixed-width-files>

4.6 Acessando Bases de dados via APIs

APIs são abreviatura para Application Programming Interface (Interface de Programação de Aplicativos), permite à máquina o acesso à funcionalidade do programa dentro de outro programa, realizando a ação do programa automaticamente, para o uso de funcionalidades e leitura de dados.

4.6.1 Pacotes Obrigatórios

O acesso a APIs em R, pode ser feito com o pacote httr²⁰ atua criando chamadas de APIs e lidando com autenticação destas;

O pacote jsonlite, realiza suporte ao trabalho com dados JASON, para traduzir as estruturas de dados aninhadas do JSON em objetos R sensíveis;

O pacote lubridate, atua na transformação e extração de datas, funções úteis para trabalhar com datas, fusos horáriros e operações aritméticas com datas;
<http://material.curso-r.com/lubridate/>

Para obtê-los

```
install.packages(c("httr", "jsonlite", "lubridate"))
```

Carregar os pacotes

A funcionalidade do R em transformar automaticamente as cadeias de caracteres em variáveis de fator (de utilidade estatística), opera de maneira indesejada,e deverá ser efetuada uma chamada de desligamento (que atuará somente nesta sessão e depois retornará à funcionalidade anterior quando reiniciado, o R).

Argumentos de consulta:

- GET(): Recupera o arquivo;
 - POST(): Adiciona um arquivo;
 - DELETE(): Remove um arquivo;
1. As API's possuem estruturas de solicitação, para isso uma solicitação de HTTP referente à base de dados com que será trabalhada, segue as seguintes descrições:
 - (a) Verbo HTTP (GET, POST, DELETE, etc.);
 - (b) O URL base da API;
 - (c) O caminho da URL ou o endpoint;
 - (d) Argumentos de consulta de URL (por exemplo, ?foo=bar);
 - (e) Cabeçalhos opcionais;

²⁰produzido po Hadley Wickham - <http://hadley.nz/>

- (f) Um corpo de solicitação opcional;
2. usa-se a função download.file() para baixar um arquivo para o seu computador e trabalhar com ele mantendo salva uma cópia dos dados no seu computador).
 3. É necessário fazer analise a saída em um objeto R. podendo verificar O pacote que realiza suporte ao trabalho com dados JASON (para traduzir as estruturas de dados JSON em objetos R),faça a intalação do pacote jasolite.
 4. Em seguida use a função fromJSON() no pacote rjson para importar esses dados para um objeto data.frame.”
 5. Com os dados em um formato data.frame, é possível limpá-los.

Exemplo de base de dados JASON retirada do Banco Central do Brasil:

Instala o pacote (apenas uma vez)

```
install.packages("jsonlite")
```

Comando para fazer o download

```
download.file("http://api.bcb.gov.br/dados/serie/bcdata.sgs.4329/dados?formato=json&dataI
```

```
> library(jsonlite)
> ICMS_json <- jsonlite::fromJSON("ICMS")
> ICMS <- as.data.frame(ICMS_json)
> ICMS$valor <- as.numeric(ICMS$valor)
> str(ICMS)
> ICMS$data <- as.Date(ICMS$data)
> str(ICMS)
> download.file("http://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?formato=json&dataI
> IPCA <- jsonlite::fromJSON("IPCA")
> as.numeric(IPCA[,2])
> ICMS <- ICMS[31:325,]
> IPCA <- IPCA[175:469,]
> IPCA <- IPCA[,2]
> IPCA_new <- as.numeric(IPCA)
> IPCA <- IPCA_new
> IPCS_new <- NULL
```

```
> data <- cbind(ICMS, IPCA)
> deflator <- as.numeric(data[295,3])
> VR <- (deflator/(as.numeric(data[,3])))*data[,2]
> plot.ts(VR)
```

<https://www.r-bloggers.com/accessing-apis-from-r-and-a-little-r-programming/>

<https://cran.r-project.org/web/packages/httr/vignettes/api-packages.html>

<https://zapier.com/learn/apis/>

<https://www.earthdatascience.org/courses/earth-analytics/get-data-using-apis/API-data-access-r/>

4.7 Trabalhando com bases de dados muito grandes

<http://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/sql.html>

<https://datacarpentry.org/R-ecology-lesson/05-r-and-databases.html>

<https://db.rstudio.com/>

http://www.columbia.edu/~sjm2186/EPIC_R/EPIC_R_BigData.pdf

<https://www.rstudio.com/resources/webinars/working-with-big-data-in-r/>

https://rpubs.com/msundar/large_data_analysis

5 Limpando e organizando seus dados

5.1 O que é uma boa base de dados e que tipos de bases existem?

Boas bases de dados para trabalhar, sobretudo com tidy, são bases que possuem variáveis nas colunas e observações nas linhas. Podemos identificar tal estrutura na figura 23 abaixo:

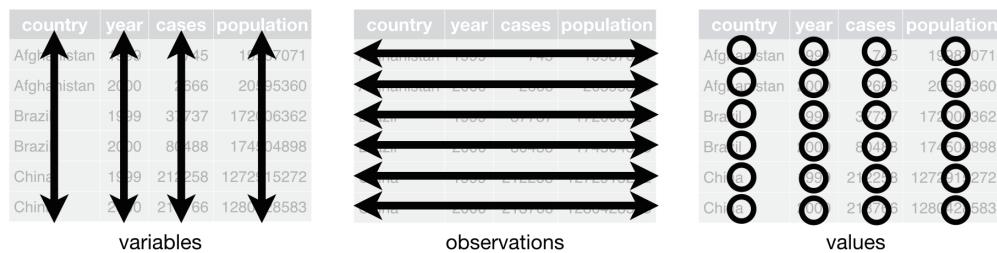


Figura 23. Organização ideal para base de dados

Há três tipos bases de dados principais que comportam a estrutura de variáveis nas colunas e observações nas linhas que são:

- Séries Temporais
- Cortes Transversais
- Dados em Painel

5.1.1 Série Temporal

Séries temporais(ou longitudinais) são identificadas como observações ordenadas ao longo do "tempo". Exemplos:

- PIB distribuído anualmente;
- Arrecadação mensal de um ou mais Estados;
- Taxa de câmbio diária.

Um bom exemplo de fonte de bases de dados de series temporais é o SGS - Banco Central. Segue o link abaixo:

[https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.
do?method=prepararTelaLocalizarSeries](https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries)

| Séries selecionadas | | Parâmetros Informados |
|---|--|--|
| 20783 - Spread médio das operações de crédito - Total | | |
| Período | | Função |
| 01/03/2018 a 27/03/2019 | | Linear |
| Registros encontrados por série: 12 | | |
| | | Lista de valores (Formato numérico: Europeu - 123.456.789,00) |
| Data
mês/AAAA | | 20783 |
| mar/2018 | | 0.p. |
| abr/2018 | | 19,93 |
| mai/2018 | | 19,56 |
| jun/2018 | | 18,46 |
| jul/2018 | | 17,68 |
| ago/2018 | | 17,69 |
| set/2018 | | 17,58 |
| out/2018 | | 17,26 |
| nov/2018 | | 17,68 |
| dez/2018 | | 18,08 |
| jan/2019 | | 16,94 |
| fev/2019 | | 18,58 |
| Fonte | | 19,04 |
| | | BCB-DSTAT |

Figura 24. Exemplo de série temporal

5.1.2 Corte Transversal

Dados transversais(ou cross-section) são identificados quando não há importância de ordenação das observações, além de representar dados em um único ponto do "tempo". Exemplos:

- Arrecadação de ICSM para todos Estados do Brasil para o ano de 2018;
- Dados demográficos para todos os municípios do Pará para o ano de 2010.

Um bom exemplo de fonte de bases de dados de cortes transversais é o Sidra - IBGE. Segue o link abaixo:

<https://sidra.ibge.gov.br/home>

```
> library(dplyr)
```

Nesta seção utilizaremos a base de dado "starwars", que é uma base nativa do pacote dplyr. Para que possamos ter acesso a base, devemos utilizar a função a seguir:

```
> data("starwars")
```

| Tabela 3939 - Efetivo dos rebanhos, por tipo de rebanho | |
|--|---------|
| Variável - Efetivo dos rebanhos (Cabeças) | |
| Ano - 2017 | |
| Tipo de rebanho - Bovino | |
| Município | |
| Abaetetuba (PA) | 2.800 |
| Abel Figueiredo (PA) | 46.364 |
| Acará (PA) | 10.740 |
| Afuá (PA) | 1.571 |
| Água Azul do Norte (PA) | 643.946 |
| Alenquer (PA) | 188.400 |
| Almeirim (PA) | 23.414 |
| Altamira (PA) | 656.430 |
| Anajás (PA) | 960 |

Figura 25. Base de dados de Série Temporal

5.1.3 Funções de Remodelagem

A primeira função apresentada é a função **arrange**, que consiste em ordenar uma coluna(variável) de forma crescente ou decrescente. As função é escrita da seguinte forma:

```
> starwars11 <- arrange(starwars, mass) #Para ordenamento crescente  
> starwars12 <- arrange(starwars, desc(height)) #Para ordenamento decrescente
```

Estão apresentadas a seguir as funções utilizadas para renomear rótulos(nomes) de colunas(variáveis).

```
> starwars21 <- rename(starwars, nome = name)  
> starwars22 <- rename(starwars, altura = height)  
> starwars23 <- rename(starwars, massa = mass)
```

5.1.4 Funções de Filtragem

A função **filter** extrai linhas de uma base de dados a partir de um critério lógico. Essa função é bastante útil quando é preciso extrair um conjunto de dados que apresentam determinadas características. Por exemplo quando é preciso criar um novo objeto sem os outliers. A função está descrita abaixo:

```
> starwars31 <- filter(starwars, mass < 70)
```

A função **distinct** remove todas as linhas que se encontram duplicadas na base de dados, permanecendo apenas uma de cada linha duplicada. Utilizamos a função da seguinte forma:

Observação: Para que possamos usar a função **distinct**, criamos o objeto *starwars32* com as linhas 31 até a 60 duplicadas.

```
> starwars32 <- rbind(starwars, starwars[31:60,])  
> starwars32 <- distinct(starwars32)
```

A função **sample_frac** seleciona aleatoriamente o número de linhas correspondentes a fração determinada na função. O exemplo abaixo seleciona aleatoriamente 40% das linhas da base de dados.

```
> starwars41 <- sample_frac(starwars, 0.4, replace = TRUE)
```

A função **sample_n** seleciona um número N de linhas. Onde N é o número de linhas que devem ser extraídas.

```
> starwars42 <- sample_n(starwars, 25, replace = TRUE)
```

A função **slice** extrai linhas de acordo com as posições das linhas. Para o exemplo abaixo, criamos um novo objeto com as linhas 10 até a linha 50.

A função **select** seleciona colunas a partir dos rótulos especificados na função.

```
> starwars61 <- select(starwars, name, height, mass)
> starwars62 <- select(starwars, name, gender, species)
```

Para combinação de duas ou mais bases de dados temos as funções de **"join"**. A primeira é a função **left_join** que insere uma base de dados a outra pela esquerda.

```
> starwars71 <- left_join(starwars61, starwars62, by = "name")
```

A função **right_join** insere uma base a outra pela direita.

```
> starwars72 <- right_join(starwars61, starwars62, by = "name")
```

A função **full_join** insere uma base a outra a partir de uma chave comum nas duas bases.

```
> starwars73 <- full_join(starwars61, starwars62, by = "name")
```

Além de todas essas funções apresentadas anteriormente, o dplyr permite concatenar funções por meio de um mecanismo chamado **pipe**. Para utilizar o pipe é preciso colocar tais símbolos:

- %>%

Abaixo temos um exemplo da utilização do pipe. O exemplo consiste em calcular média e desvio padrão a partir de um agrupamento por espécie da base de dados starwars.

```
> starwars81 <- starwars %>%
+   group_by(species) %>%
+   summarise(desvpad = sd(birth_year, na.rm = TRUE),
+             avg = mean(birth_year, na.rm = TRUE))
```

5.2 tidyverse

Tidyr é um pacote de tratamento de dados que fornece funções baseadas na organização de observações nas linhas e variáveis nas colunas, além de possuir boa relação com outros pacotes, como o **dplyr** para tratamento de dados e **ggplot** para apresentação de resultados.

A fonte para parte das figuras desta seção está em <https://github.com/rstudio/cheatsheets/blob/master/data-import.pdf>

Para utilizar o pacote precisamos efetuar a instalação do pacote e depois executar o mesmo.

```
> install.packages("tidyverse")
```

```
> library(tidyverse)
```

Para os exemplos desta seção utilizaremos as bases de dados **population** e **table2** do próprio tidyverse. Para utilizar as duas bases precisamos efetuar dois comandos, que são:

```
> data("population")
> data("table2")
```

O **tidyverse** possui quatro funções de remodelagem que são:

5.2.1 Gather

A função **gather** é oposta a função **spread**, visto que **gather** transforma os rótulos das variáveis(nome das colunas) em observações(linhas). Para tal exemplo utilizaremos a base de dados (table2).

Primeiro transformaremos o fator(cases, count) em colunas.

```
> View(table2)
> population12 <- table2 %>% spread(type, count)
> View(population12)
```

| table4a | | |
|---------|------|------|
| country | 1999 | 2000 |
| A | 0.7K | 2K |
| B | 37K | 80K |
| C | 212K | 213K |

| country | year | cases |
|---------|------|-------|
| A | 1999 | 0.7K |
| B | 1999 | 37K |
| C | 1999 | 212K |
| A | 2000 | 2K |
| B | 2000 | 80K |
| C | 2000 | 213K |

Figura 26. Exemplo da função gather: `gather(table4a, '1999', '2000', key = "year", value = "cases")`

Depois retornaremos com o fator(cases, count) para as linhas, retornando ao painel de dados anteriormente.

```
> population13 <- population12 %>% gather(type, count, cases:population)
> View(population13)
```

5.2.2 Spread

A função **spread** transforma linhas em colunas. É bastante útil quando é preciso transformar dados categóricos em variáveis.

No exemplo abaixo transformamos um painel de dados em uma série temporal, sendo as variáveis os países presentes na base de dados **population**.

```
> View(population)
> population11 <- population %>% spread(country, population)
> View(population11)
```

table2

| country | year | type | count | | country | year | cases | pop |
|---------|------|-------|-------|---|---------|------|-------|------|
| key | | value | | → | A | 1999 | 0.7K | 19M |
| A | 1999 | cases | 0.7K | | A | 2000 | 2K | 20M |
| A | 1999 | pop | 19M | | B | 1999 | 37K | 172M |
| A | 2000 | cases | 2K | | B | 2000 | 80K | 174M |
| A | 2000 | pop | 20M | | C | 1999 | 212K | 1T |
| B | 1999 | cases | 37K | | C | 2000 | 213K | 1T |
| B | 1999 | pop | 172M | | | | | |
| B | 2000 | cases | 80K | | | | | |
| B | 2000 | pop | 174M | | | | | |
| C | 1999 | cases | 212K | | | | | |
| C | 1999 | pop | 1T | | | | | |
| C | 2000 | cases | 213K | | | | | |
| C | 2000 | pop | 1T | | | | | |

Figura 27. Exemplo da função spread: spread(table2, type, count)

5.2.3 Unite

A função **unite** agrupa o conteúdo de todas as linhas de duas colunas distintas em uma única coluna. Para o exemplo dessa função utilizaremos o dataset **table1** para unir o conteúdo de duas colunas.

Primeiramente acessamos o dataset **table1**:

```
> data("table1")
> head(table1)
```

Após ter acesso ao dataset, podemos executar a função **unite** para juntar os conteúdos **country** e **year** em uma única coluna. Fazemos a ação da seguinte forma:

```
> table2 <- unite(table1, "country_year", c("country", "year"))
> head(table2)
```

5.2.4 Separate

A função **separate** desagrega o conteúdo de uma coluna em duas novas colunas. Para o exemplo dessa função utilizaremos o dataframe **table2** criado no exemplo anterior.

A função de separação é a seguinte:

table5

| country | century | year | | country | year |
|---------|---------|------|---|---------|------|
| Afghan | 19 | 99 | → | Afghan | 1999 |
| Afghan | 20 | 0 | | Afghan | 2000 |
| Brazil | 19 | 99 | | Brazil | 1999 |
| Brazil | 20 | 0 | | Brazil | 2000 |
| China | 19 | 99 | | China | 1999 |
| China | 20 | 0 | | China | 2000 |

Figura 28. Exemplo da função unite: `unite(table5, century, year, col = "year", sep = "")`

```
> table3 <- separate(table2, country_year, c("country", "year"), sep = "_")
> head(table3)
```

Como resultado da separação é preciso que obtenhamos um dataframe **table3** exatamente igual ao dataframe **table1**. Para checarmos o resultado está conforme o esperado, podemos executar o respectivo comando:

```
> table3 == table1
```

table3

| country | year | rate | | country | year | cases | pop |
|---------|------|----------|---|---------|------|-------|-----|
| A | 1999 | 0.7K/19M | → | A | 1999 | 0.7K | 19M |
| A | 2000 | 2K/20M | | A | 2000 | 2K | 20M |
| B | 1999 | 37K/172M | | B | 1999 | 37K | 172 |
| B | 2000 | 80K/174M | | B | 2000 | 80K | 174 |
| C | 1999 | 212K/1T | | C | 1999 | 212K | 1T |
| C | 2000 | 213K/1T | | C | 2000 | 213K | 1T |

Figura 29. Exemplo da função separate: `separate(table3, rate, into = c("cases", "pop"))`

table3

| country | year | rate | | country | year | rate |
|---------|------|----------|---|---------|------|------|
| A | 1999 | 0.7K/19M | → | A | 1999 | 0.7K |
| A | 2000 | 2K/20M | | A | 1999 | 19M |
| B | 1999 | 37K/172M | | A | 2000 | 2K |
| B | 2000 | 80K/174M | | B | 1999 | 20M |
| C | 1999 | 212K/1T | | B | 1999 | 37K |
| C | 2000 | 213K/1T | | B | 2000 | 172M |
| | | | | C | 1999 | 80K |
| | | | | C | 2000 | 174M |
| | | | | C | 1999 | 212K |
| | | | | C | 1999 | 1T |
| | | | | C | 2000 | 213K |
| | | | | C | 2000 | 1T |

Figura 30. Exemplo da função separate_rows: `separate_rows(table3, rate)`

5.2.5 Lidando com valores omissos

| X | |
|----|----|
| x1 | x2 |
| A | 1 |
| B | NA |
| C | NA |
| D | 3 |
| E | NA |



| x1 | x2 |
|----|----|
| A | 1 |
| D | 3 |

drop_na(x, x2)

Figura 31. Exemplo da função drop_na: *drop_na(x, x2)*

| X | |
|----|----|
| x1 | x2 |
| A | 1 |
| B | NA |
| C | NA |
| D | 3 |
| E | NA |



| x1 | x2 |
|----|----|
| A | 1 |
| B | 1 |
| C | 1 |
| D | 3 |
| E | 3 |

fill(x, x2)

Figura 32. Exemplo da função fill: *fill(x,x2)*

| X | |
|----|----|
| x1 | x2 |
| A | 1 |
| B | NA |
| C | NA |
| D | 3 |
| E | NA |



| x1 | x2 |
|----|----|
| A | 1 |
| B | 2 |
| C | 2 |
| D | 3 |
| E | 2 |

replace_na(x, list(x2 = 2))

Figura 33. Exemplo da função replace_na: *replace_na(x, list(x2 = 2))*

6 Gráficos em R

6.1 ggplot

O pacote **ggplot2** é uma ferramenta de visualização gráfica bastante útil para apresentação de dados com muitos componentes. A estrutura de construção de gráficos utilizando o ggplot2 é basicamente definir os dados a serem plotados e o sistema de coordenadas. Uma boa forma de entender como funciona o **ggplot2** é por meio da metáfora de camadas. Cada detalhe do gráfico é uma camada de comando definido a partir das camadas principais(dados e coordenadas).

A figura 34 mostra cada camada do processo de construção de gráficos utilizando o ggplot2

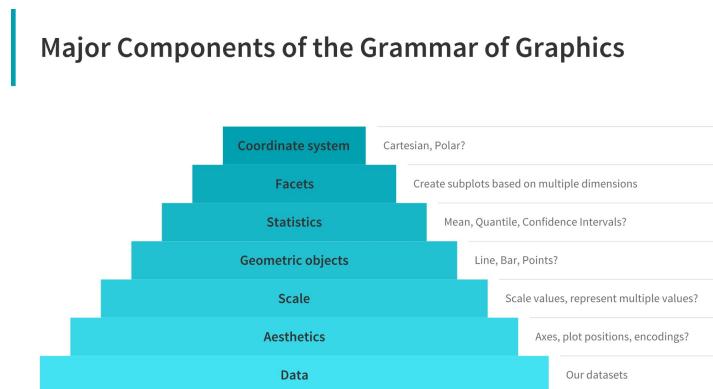


Figura 34. Camadas do ggplot2

Abaixo estão as descrições de cada camada da figura 34

Data: Base de dados que será utilizada;

Asthetics: Definição do conjunto de dados que compõem o gráfico, bem como a posição de cada dado;

Scale: Definição dos escala dos valores do gráfico;

Objeto Geométrico: Definição da geometria do gráfico;

Statistics: Estatística descritiva dos dados utilizados para gerar o gráfico;

Facets: Criação de subplots(divisão da janela de plots);

Sistema de Coordenada: Definição do sistema de coordenadas.

Para os exemplos do pacote **ggplot2** utilizaremos o dataset Iris. Essa base de dados possui 5 variáveis que são:

Sepal.Length: Comprimento da sépala;

Petal.Length: Comprimento da pétala;

Sepal.Width: Largura da sépala;

Petal.Width: Largura da pétala;

Species: Espécies da plantas.

```
> data("iris")
> 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 primeiro gráfico apresentará a relação entre o comprimento da pétala e da sépala como pode ser visto na figura 35.

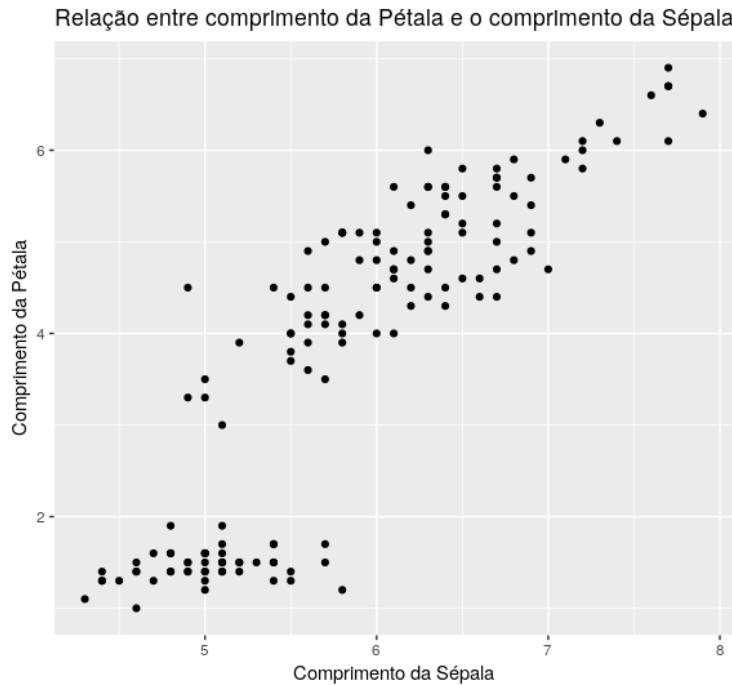


Figura 35. Relação de pontos entre os comprimentos Pétala e Sépala

Os comandos para gerar o gráfico são:

```
> g1 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
+ geom_point() +
+ ggtitle("Relação entre tamanho da Pétala e o tamanho da Sépala") +
+ labs(x="Tamanho da Sépala", y="Tamanho da Pétala")
> plot(g1)
```

O objeto **g1** recebe a função que trata do acesso a base aos dados **iris** e a definição da posição das variáveis. Tal função está descrita abaixo:

```
> g1 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length))
> #iris indica a base de dados e aes() indica a posição das variáveis
```

próxima camada indica qual é a geometria do gráfico. Tal função é dada por:

```
> geom_point()
```

As próximas camadas apresentam apenas títulos e legendas presentes nos gráficos, além da função **plot** para apresentar o gráfico na região de plot do **Rstudio**

```
> ggtile("Relação entre tamanho da Pétala e o tamanho da Sélula") +
+   labs(x="Tamanho da Sélula", y="Tamanho da Pétala")
> plot(g1)
```

É possível incluir mais de uma geometria no gráfico.

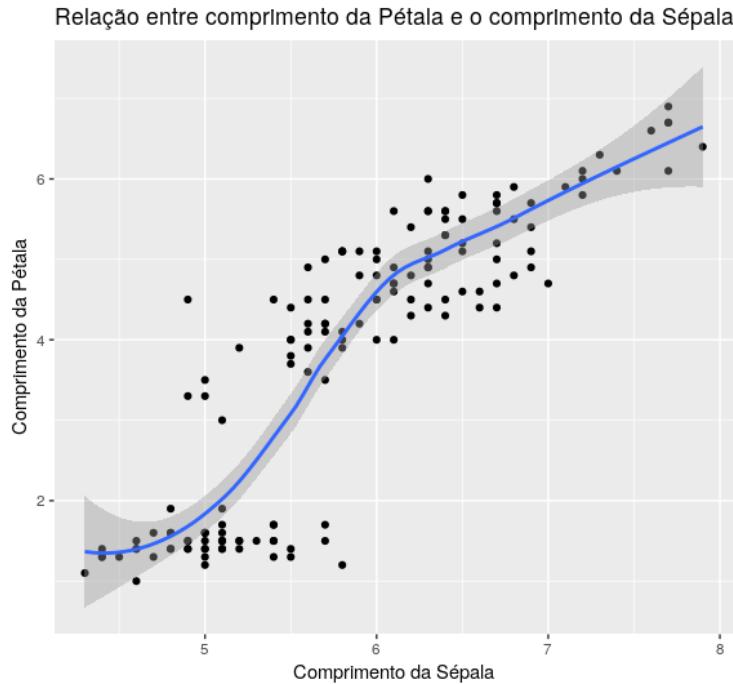


Figura 36. Relação de comprimento entre Pétala e Sélula

Para fazer o gráfico da figura 36 podemos utilizar os comandos abaixo:

```
> g2 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
+   geom_point() +
+   geom_smooth() +
+   ggtile("iris") +
+   labs(x="Sepal length", y="Petal Length")
> plot(g2)
```

Podemos fazer uma separação por variáveis categóricas e apresentá-las com cores distintas. Na **aes()** adicionamos o parâmetro *color = Species*. Para o nosso exemplo, a variável categórica é **Species**.

O exemplo para construir a figura 37 está exemplificado abaixo:

```
> g3 <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
+   geom_point() +
```

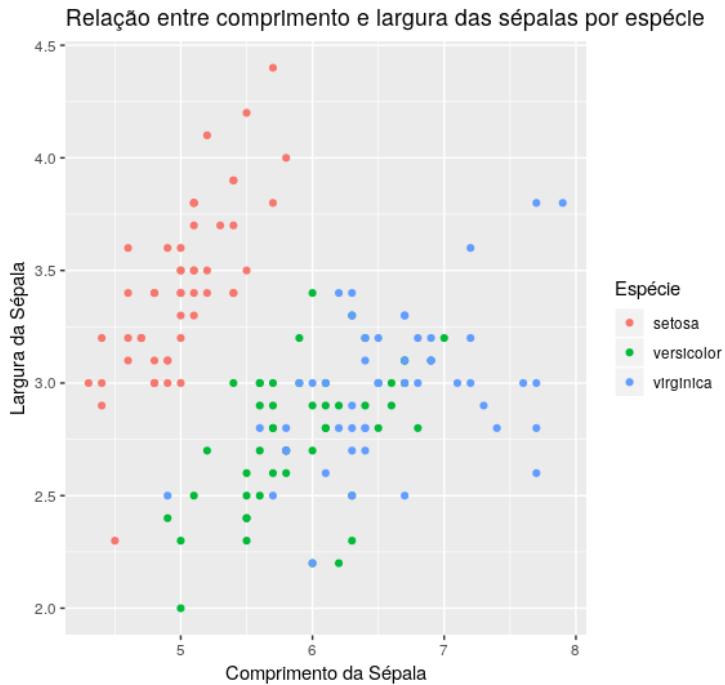


Figura 37. Relação de comprimento e largura das Sépalas

```
+ ggtitle("Relação entre comprimento e largura das sépalas por espécie") +
+ labs(x="Comprimento da Sélpa", y="Largura da Sélpa", color = "Espécie")
> plot(g3)
```

Podemos incluir mais uma camada e criar vários ambientes de plot dividindo o ambiente principal. Para que isso seja possível utilizaremos a função `facet_wrap()`.

O código para gerar o gráfico da figura 38 está exposto abaixo:

```
> gf4 <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
+ geom_point() +
+ facet_wrap(~Species) +
+ ggtitle("Relação entre o comprimento e a largura das sépalas") +
+ labs(x="Comprimento da Sélpa", y="Largura da Sélpa", color = "Espécie")
> plot(gf4)
```

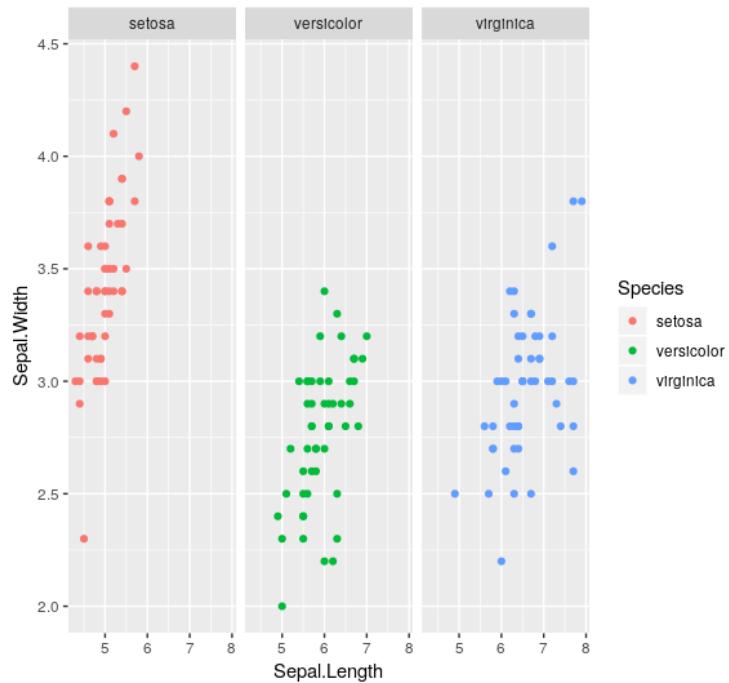


Figura 38. Relação entre comprimento e largura das sépalas por espécie

Percebam que na figura 38 utilizamos a mesma variável para o parâmetro **color** e para o parâmetro **facet_wrap**. Em um novo exemplo usando a base de dados **mpg**, utilizaremos os dois parâmetros citados anteriormente, mas com variáveis distintas para cada um.

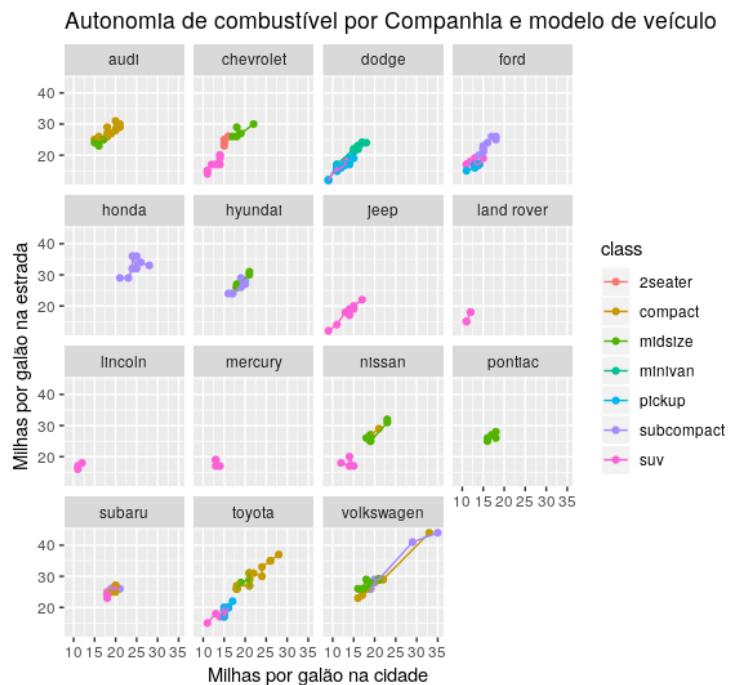


Figura 39. Autonomia de combustível de veículos

O código que gera a figura 39 está escrito abaixo:

```
> gf8 <- ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
+ geom_point() +
+ geom_line() +
+ facet_wrap(~manufacturer) +
+ ggtitle("Autonomia de combustível por Companhia e modelo de veículo") +
+ labs(x="Milhas por galão na cidade", y="Milhas por galão na estrada")
> plot(gf8)
```

É possível adicionar os tamanhos dos pontos pra indicar o impacto de outra variável. Para fazer isso basta adicionar uma camada na função **geom_point(size = cyl)**. O código fica da seguinte forma:

```
> gf9 <- ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
+ geom_point(aes(size = cyl)) +
+ geom_line() +
+ facet_wrap(~year) +
+ ggtitle("Autonomia de combustível") +
+ labs(x="Autonomia na Cidade", y="Autonomia na Estrada",
+ color = "Classe do Veículo", size = "Cilindro")
> plot(gf9)
```

É possível determinar novas posições para as legendas. Para o nosso exemplo, as legendas serão movidas para baixo:

```
> gf11 <- ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
+ geom_point() +
+ geom_line() +
+ facet_wrap(~year) +
+ ggtitle("Autonomia de combustível") +
+ labs(x="Autonomia na Cidade", y="Autonomia na Estrada",
+ color = "Classe do Veículo", size = "Cilindro") +
+ theme(legend.position = "bottom") # "bottom", "top", "left", ou "right"
> plot(gf11)
```

Por último, podemos mudar o tema dos gráficos, para isso adicionamos uma nova camada:

```
> gf10 <- ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
+ geom_point() +
```

```
+ geom_line() +
+ facet_wrap(~year) +
+ ggtitle("Autonomia de combustível") +
+ labs(x="Autonomia na Cidade", y="Autonomia na Estrada",
+ color = "Classe do Veículo", size = "Cilindro") +
+ theme_minimal()
> plot(gf10)
```

Todos os temas disponíveis podem ser checados em <https://ggplot2.tidyverse.org/reference/ggtheme.html>

7 Analisando dados com R

Neste capítulo veremos como é possível gerar um grande conjunto de tabelas em R (utilizando o *Markdown* bem como discutiremos a apresentação de resultados de modelos com estas bibliotecas.

Muitas vezes, como já vimos aqui, é necessário tratar e sintetizar os dados em um dataframe para podermos analisá-los. Então, faremos, neste capítulo um conjunto de tabelas que serão apresentadas utilizando algumas ferramentas específicas para geração de tabelas em R.

7.1 Gerando Tabelas com Xtable e Stargazer

A geração de tabelas no R pode ser feita utilizando alguns pacotes específicos adequados para tal. Estes pacotes leem dataframes ou matrizes e geram tabelas a partir destes. Muitos destes pacotes têm também a capacidade de ler objetos específicos do R (como resultados de regressões, análise de cluster, etc) e gerar tabelas correspondentes às saídas normais de apresentação destes resultados.

Abaixo aqui uma seção utilizando xtable

Aqui temos o nosso velho conhecido banco de dados MTCARS. Utilizando a opção auto, o xtable define o tipo de saída em termos de casas decimais, automaticamente.

```
> library(xtable)
> mtc2 <- xtable(head(mtcars),
+                  auto = TRUE,
+                  caption = 'Base mtcars, primeiras linhas')
> print.xtable(mtc2, caption.placement='top')

>
```

A apresentação da tabela nestes termos, com este nível de detalhamento não faz nenhum sentido para a nossa análise. Claramente o gostaríamos de ter é

Tabela 1. Base mtcars, primeiras linhas

| | 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 |

uma tabela mais sintética. Para isso, precisamos gerar um novo dataframe que sintetize os nossos resultados de interesse e, a partir daí, gera-se a tabela.

```
> library(dplyr)
> data(mtcars)
> sintesePot <- as.data.frame(mtcars %>% group_by(cyl) %>%
+                               summarize(potenciaMedia = mean(hp,na.rm = TRUE),
+                                         sigmaPotencia = sd(hp, na.rm = TRUE),
+                                         n = n())
+ )
> tabelaPot <- xtable(sintesePot,
+                         caption = 'Média e Desvio Padrão da potência por no. de cilindros')
> print(tabelaPot, type = "latex", caption.palcement='bottom')
```

| cyl | potenciaMedia | sigmaPotencia | n |
|-----|---------------|---------------|-------|
| 1 | 4.00 | 82.64 | 20.93 |
| 2 | 6.00 | 122.29 | 24.26 |
| 3 | 8.00 | 209.21 | 50.98 |
| | | | 14 |

Tabela 2. Média e Desvio Padrão da potência por no. de cilindros

Para a análise, no entanto, podemos aqui pensar na relação entre consumo e número de hp.

Podemos então pensar em uma regressão que olhe para a relação entre potência e consumo

```
> library(ggplot2)
> ggplot(mtcars, aes(x=hp, y=mpg)) +
+   geom_point() +
+   geom_smooth() +
+   labs(x='Potência (HP)', y = 'Consumo (MPG)')
```

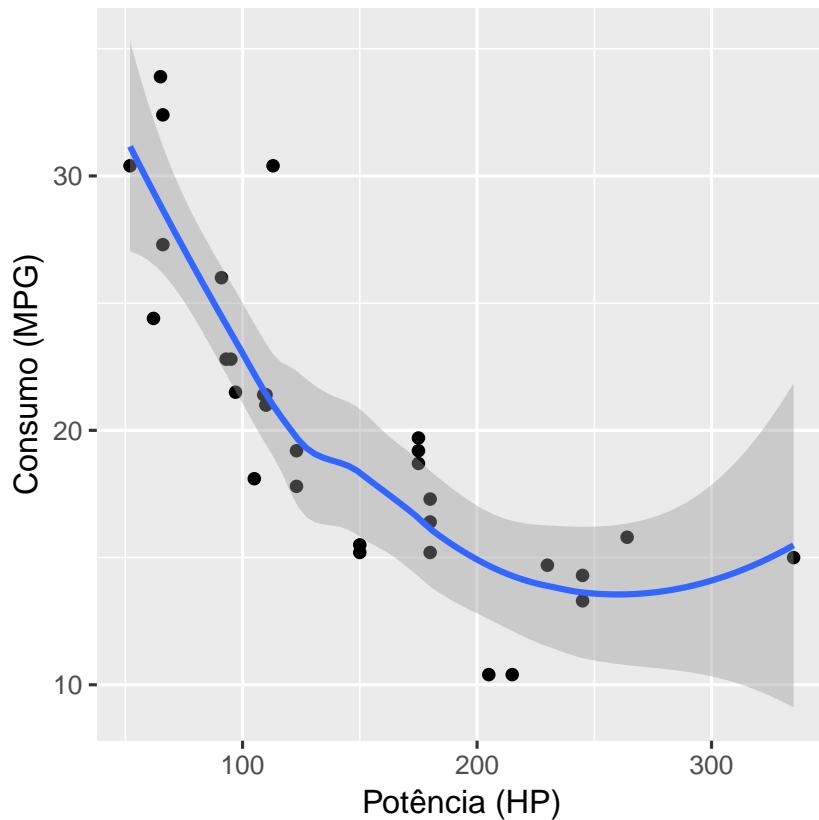


Figura 40. Comparação de Potência em HP com Consumo (mpg)

```

> hpmpg <- lm(mpg~hp, data = mtcars)
> summary(hpmpg)

Call:
lm(formula = mpg ~ hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.7121 -2.1122 -0.8854  1.5819  8.2360 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 30.09886   1.63392 18.421 < 2e-16 ***
hp          -0.06823   0.01012 -6.742 1.79e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.863 on 30 degrees of freedom
Multiple R-squared:  0.6024,        Adjusted R-squared:  0.5892 
F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07

```

A saída da regressão, porém, não está adequada para apresentação. Então, utilizamos o xtable para gerar uma tabela bem mais apresentável.

```

> lm.hpmpg.tab <- xtable(hpmpg,
+                           caption = 'Resultado do Modelo Consumo versus HP',
+                           label   = 'tabreg:consumohp'
+                         )
> print.xtable(lm.hpmpg.tab, caption.placement = 'top')

```

Tabela 3. Resultado do Modelo Consumo versus HP

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|----------|------------|---------|----------|
| (Intercept) | 30.0989 | 1.6339 | 18.42 | 0.0000 |
| hp | -0.0682 | 0.0101 | -6.74 | 0.0000 |

Note que podemos incluir outros detalhes na tabela, como títulos, por exemplo (usando `caption` no `xtable`) bem como colocar um identificador da tabela (o `label`) e, na função `print.xtable()` modificar a posição do título.

```
> library(stargazer)
> stargazer(hpmpg, type='latex',
+             title = 'Resultado do Modelo Consumo versus HP'
+           )
```

Tabela 4. Resultado do Modelo Consumo versus HP

| <i>Dependent variable:</i> | |
|----------------------------|------------------------|
| mpg | |
| hp | -0.068***
(0.010) |
| Constant | 30.099***
(1.634) |
| <hr/> | |
| Observations | 32 |
| R ² | 0.602 |
| Adjusted R ² | 0.589 |
| Residual Std. Error | 3.863 (df = 30) |
| F Statistic | 45.460*** (df = 1; 30) |

Note: *p<0.1; **p<0.05; ***p<0.01

Uma outra função que podemos utilizar que apresenta os resultados da nossa análise em uma forma um pouco diferente e bem mais detalhada é o stargazer. Abaixo um exemplo de tabela utilizando esta biblioteca.

É o mesmo resultado da tabela anterior, apenas utilizando outra biblioteca de geração de tabelas.

xtable e stargazer têm uma variedade imensa de possibilidade que permitem a geração de tabelas de diversos tipos.

7.2 Gerando Dashboards utilizando o Shiny

O *shiny* é um pacote R que facilita a criação de aplicativos interativos com uma interface web direto do R.

Para baixar o *shiny* usamos nosso velho conhecido *install.packages()*

```
> install.packages('shiny')
>
```

7.2.1 Estrutura de um aplicativo shiny

Esta seção está fortemente baseada na *cheat sheet* do *shiny* publicada em <https://shiny.rstudio.com/images/shiny-cheatsheet.pdf>

Aplicativos *shiny* estão contidos em um único script chamado app.R. O script app.R reside em um diretório (por exemplo, newdir/) e o aplicativo pode ser executado com runApp("newdir").

app.R tem três componentes:

1. um objeto de interface do usuário
2. uma função de servidor
3. uma chamada para a função shinyApp

O objeto *ui* da interface com o usuário controla o layout e a aparência do seu aplicativo. A função *server* contém as instruções que seu computador precisa para criar seu aplicativo. Finalmente, a função *shinyApp* cria objetos do *shiny* a partir de um par de interface do usuário/servidor explícito.

O objeto de interface de usuário e a função servidor podem tanto estar em dois arquivos ("ui.R" e "server.R") quanto em um arquivo único. Utilizando um arquivo único permite um compartilhamento muito mais simples do código escrito em *shiny*.

Um arquivo de aplicação em *shiny*, então, conterá 3 partes (Figura 41). Uma primeira parte é a descrição da estrutura da página a ser publicada, bem




```

library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)

```

Figura 41. Estrutura de um arquivo de aplicação shiny

como com os elementos da interface do usuário (ui). Este componentes terão os elementos de entrada e também indicarão onde ficariam os gráficos, tabelas e textos que resultariam nas **reações** as mudanças nos valores que estiverem conectados com estas saídas.

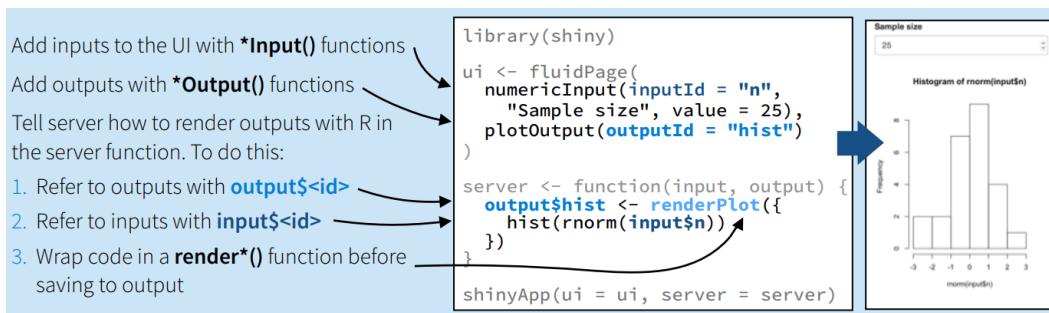


Figura 42. Entrada, servidor e aplicação em detalhe

A segunda parte é a descrição das ações que ocorrem quando há uma mudança nos valores observados e quais as funções específicas serão utilizadas. Nesta segunda parte também é incluída a estratégia de renderização dos resultados (Figura 42).

7.2.2 Layouts do shiny

A interface de usuário do shiny permite uma quantidade razoável de opções de layout, podendo também combinar conjuntos diferentes de elementos em entradas únicas. Abaixo (Figura 43) vemos o exemplo de um painel com um campo de entrada tipo data e um botão para incluir o valor da variável.

Na definição da interface, ao utilizarmos a função *fluidPage()* (Figura 44) podemos montar um conjunto grande de layouts com uma flexibilidade que permite diversos padrões de distribuição das informações na sua home page gerada pelo *shiny*.

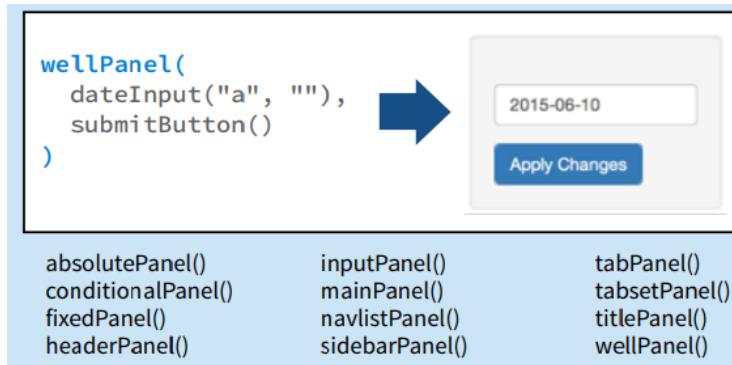


Figura 43. Um painel composto e as diversas funções de painéis compostos

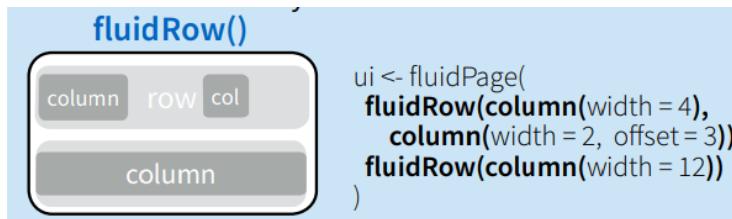


Figura 44. Exemplo da função `fluidRow()`

A figura 44 mostra um exemplo da função `fluidRow()` com duas linhas e duas colunas de tamanhos diferentes dentro do layout de uma página.

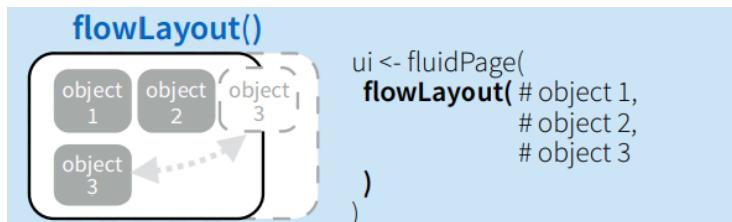


Figura 45. Exemplo da função `flowLayout()`

A função `flowLayout()` (Figura 44) permite uma distribuição automática dos objetos dentro da página.

A função `sidebarLayout()` (Figura 46) é extremamente útil para termos painéis onde opções em um painel alteram a saída em um painel principal. Muitas apresentações e aplicações em páginas que mostram dados têm esse layout.

Um outro conjunto de funções bastante útil que aumenta a densidade de informações apresentadas na página são as que colocam painéis (As funções `tabsetPanel()`, `navlistPanel()` combinadas com a função `tabPanel()`). Estas funções permitem a apresentação de resultados selecionados a partir de um conjunto de painéis com diferentes conteúdos para cada painel (Figura 47)

Os objetos que permitem diversos tipos de entradas no `shiny` são incluídos na seção ui. Na figura 48 vemos alguns dos tipos de objetos que podem ser

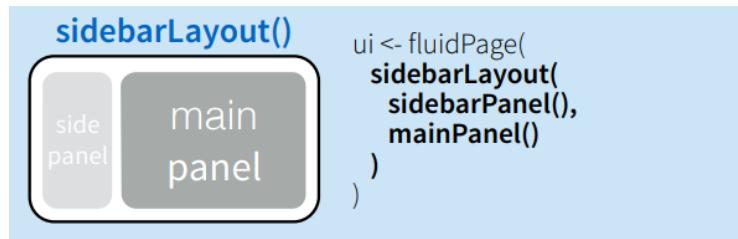


Figura 46. Exemplo da função `sidebarLayout()`

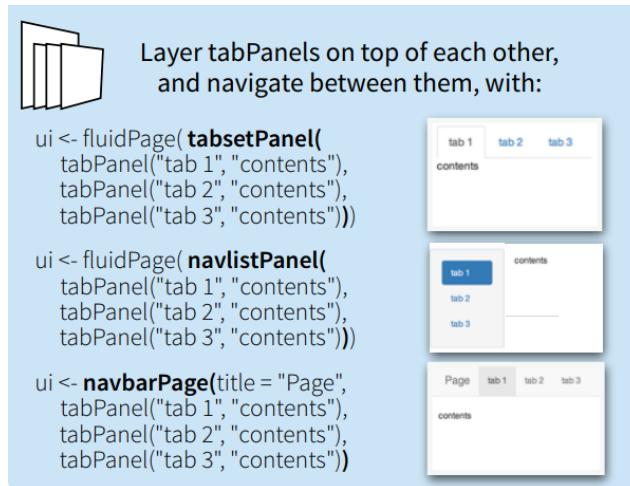


Figura 47. Painéis no shiny

utilizados. É possível fazer uma entrada de um valor numérico com `numericInput()`, de um valor que permite escolha entre alternativas com (`radioButtons()` e `selectInput()`) entre diversas outras possibilidades.

Finalmente, o *shiny* combina funções de renderização e saída para produzir as suas páginas. As funções de renderização funcionam combinadas com as funções de saída (Figura 49) e são também combinadas ou disparadas por eventos que ocorrem na interface de usuário. Este padrão é uma das características do modelo de programação das páginas geradas pelo *shiny* chamado de *programação reativa*. Mais informações sobre o modelo de programação reativa, podem ser encontradas em <https://shiny.rstudio.com/articles/reactivity-overview.html>

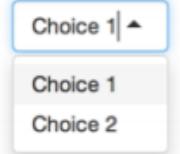
7.2.3 Alguns links úteis

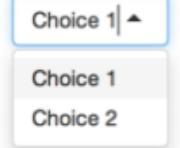
<https://cran.r-project.org/web/packages/xtable/vignettes/xtableGallery.pdf>

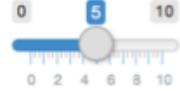
<https://www.rdocumentation.org/packages/xtable/versions/1.8-4/topics/xtable>

 **numericInput**(inputId, label, value, min, max, step)

 **passwordInput**(inputId, label, value)

 **radioButtons**(inputId, label, choices, selected, inline)
 Choice A
 Choice B
 Choice C

 **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)
 Choice 1
 Choice 1
 Choice 2

 **sliderInput**(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)
 0 5 10
 0 2 4 6 8 10

 **submitButton**(text, icon)
 (Prevents reactions across entire app)

 **textInput**(inputId, label, value)

Figura 48. Alguns Tipos de Entradas

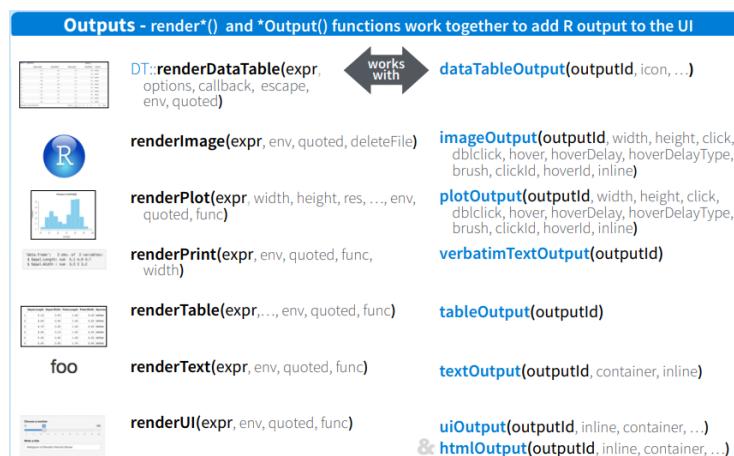


Figura 49. Funções de Renderização e Saída

<http://bcb.dfci.harvard.edu/~aedin/courses/ReproducibleResearch/RR/exampleSweave.Rnw>

<https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdf>

<https://cran.r-project.org/web/packages/stargazer/index.html>

<https://www.rdocumentation.org/packages/stargazer/versions/5.2.2/topics/stargazer>

<https://www.princeton.edu/~otorres/NiceOutputR.pdf>

<https://www.r-statistics.com/2013/01/stargazer-package-for-beautiful-latex-tables/>

<https://shiny.rstudio.com/tutorial/written-tutorial/lesson1/>

<https://shiny.rstudio.com/>

<http://material.curso-r.com/shiny/>

<https://www.rstudio.com/products/shiny/shiny-user-showcase/>

<https://www.showmeshirey.com/category/topics/finance/>

<https://shiny.rstudio.com/images/shiny-cheatsheet.pdf>

<https://www.rstudio.com/wp-content/uploads/2015/02/shiny-cheatsheet.pdf>

8 Onde Aprender Mais?

Este material é apenas um conjunto bem pequeno de informações básicas sobre tratamento de dados em R.

O R é uma ferramenta que evoluiu de um conjunto de programas execução de modelos estatísticos para uma suíte bastante completa de análise e tratamento de dados, com boa parte do que há de mais sofisticado neste campo atualmente.

É também um projeto colaborativo, onde milhares de pessoas contribuem com seu conhecimento e expertise para a construção de um ambiente aberto e em rápida evolução. É um exemplo do que se pode fazer a partir a ideia de software livre.

No decorrer deste livro, sempre que foi necessário, apresentamos as fontes de maior informação para aqueles que são curiosos ou que têm necessidades mais complexas e específicas.

Há uma profusão de sites na internet com informação sobre o R. Quando da ocorrência de erros e de problemas específicos ou dúvidas vale sempre a pena fazer a busca com palavras chaves o mais específicas que for possível. Isso muito provavelmente lhe levará a uma lista de discussão onde alguém eventualmente já postou o mesmo problema recebeu a ajuda de usuários mais experientes. Esta é uma forma bastante eficaz de encontrar respostas.

Uma boa sugestão é tentar a pergunta em inglês (pode-se usar o Google translator ou outra app de tradução). Há uma quantidade muito grande de sites e respostas prontas nesta língua.

Abaixo alguns sites que são referência para a comunidade R.

<https://cran.r-project.org/>

O CRAN (The Comprehensive R Archive Network) é o principal site do R. Lá que se encontram as versões mais atualizadas do R e as bibliotecas aprovadas pelo pessoal que controla o desenvolvimento do programa. No CRAN você também pode encontrar a documentação das bibliotecas do R.

<https://www.rstudio.com/>

O Rstudio parece ter se tornado a IDE preferida no R. Também o grupo que o desenvolve produziu um conjunto de ferramentas de análise e tratamento de dados que rapidamente se tornaram um padrão. Há também muitos tutoriais extremamente úteis.

<https://www.rdocumentation.org/>

O RDocumentation contém informações sobre mais de 17 mil pacotes em R, listados no CRAN, Biocondutor e GitHub. Uma fonte importante de referência para as funções.

<https://www.bioconductor.org/>

O Biocondutor é um site com pacotes específicos para Biologia, mas tem muita coisa útil para diversas outras aplicações lá.

Divirtam-se.

Referências Bibliográficas

- [ACX⁺16] J Allaire, Joe Cheng, Yihui Xie, Jonathan McPherson, Winston Chang, Jeff Allen, Hadley Wickham, Aron Atkins, Rob Hyndman, and Ruben Arslan. rmarkdown: Dynamic documents for r, 2016.
- [Sho19] Show me shiny: Gallery of r web apps, 2019.
- [VS11] W N Venables and D M Smith. *An Introduction to R*, volume 2. CRAN - Comprehensive R Archive Network, 2011.
- [Wic14] Hadley Wickham. Tidy data. *The Journal of Statistical Software*, 59, 2014.
- [Xie16] Y Xie. knitr: A general-purpose package for dynamic report generation in r [software], 2016.
- [Xie17] Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, 2017.