

# Relatórios e apresentações automáticas



# Sobre a Curso-R

# A empresa



[www.curso-r.com](http://www.curso-r.com)



## Programação para Ciência de Dados

---

Introdução a programação com R

R para Ciência de Dados I

R para Ciência de Dados II

Pacotes

Python para quem usa R

## Web scraping

---

Web scraping

## Dashboards e Visualização de Dados

---

Visualização de dados

Relatórios e apresentações

Dashboards I

Deploy

Dashboards II

## Modelagem de Dados

---

Modelos Lineares

Introdução ao Machine Learning

Séries Temporais

Não supervisionado

# Sobre o curso

# Relatórios e apresentações automáticas

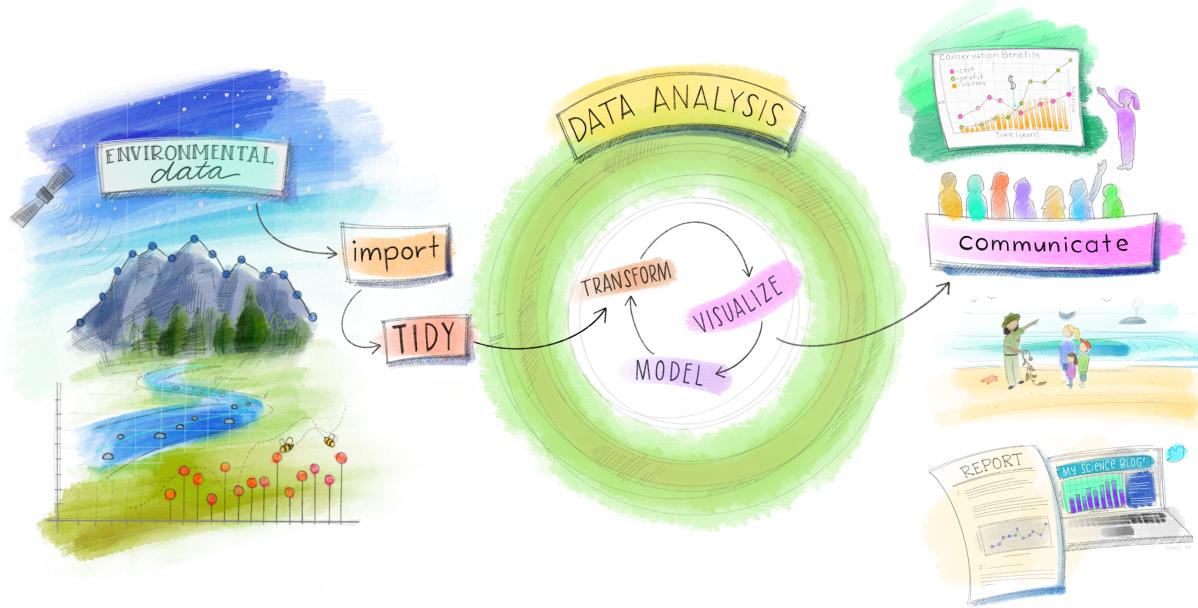


Ilustração por Allison Horst.



Ilustração por Allison Horst.

# Resultados

No final, você poderá...

- Criar documentos reproduutíveis (como relatórios e apresentações)
- Compartilhar suas análises usando R em todas as etapas do ciclo da ciência de dados

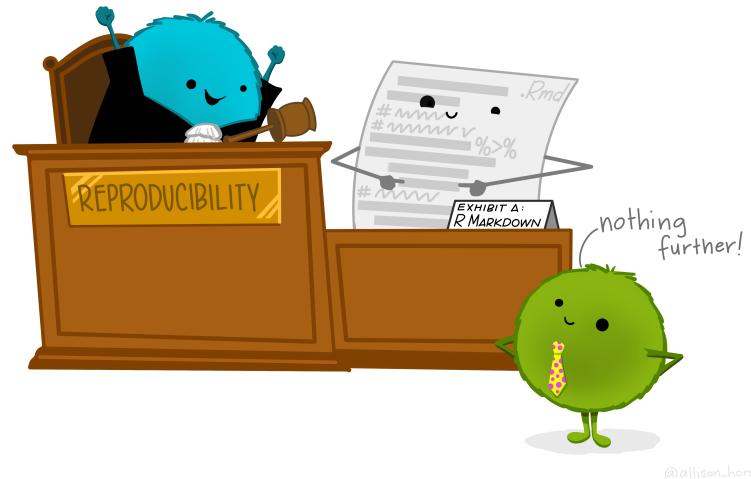


Ilustração por [Allison Horst](#).

# Dinâmica

- Exercícios para casa, com entrega facultativa
- **Trabalho final**, com entrega obrigatória
  - As pessoas que fizerem os trabalhos **mais legais** receberão uma **bolsa** para fazer qualquer curso da Curso-R
  - Mais detalhes sobre o trabalho final nas próximas aulas :)

# Tirando dúvidas

- **Não existe dúvida idiota.**
- Nem sempre é trivial fazer a pergunta certa para que outra pessoa esclareça a sua dúvida. Neste curso, **vamos mostrar melhores práticas na hora de fazer perguntas sobre programação**.
- Fora do horário de aula ou monitoria:
  - perguntas gerais sobre o curso deverão ser feitas no Classroom.
  - perguntas sobre R, principalmente as que envolverem código, deverão ser enviadas no [nossa discourse](#). Se envolver web scraping, é importante especificar a página que está querendo acessar e como você faria para encontrá-la manualmente.
- [Veja aqui dicas de como fazer uma boa pergunta.](#)

# Por que usar o discourse?

- Muito melhor para escrever textos que possuem códigos. Com ele, podemos usar o pacote `{reprex}`!
- Saber pesquisar sobre erros e fazer a pergunta certa é essencial para aprender e resolver problemas de programação.
- No discourse, teremos mais pessoas acompanhando e respondendo as dúvidas.
- Em um ambiente aberto, as suas dúvidas vão contribuir com a comunidade.

<https://discourse.curso-r.com/>

# Trabalho final

# Trabalho final

- Neste curso, utilizaremos uma base de dados para criar um produto para comunicar uma análise de dados (relatório, apresentação, site, etc).
- O trabalho final de vocês será criar um também!
  - Escolha uma base de dados que você ache interessante!
  - Defina um objetivo da análise.
  - Defina um público alvo: a comunicação depende de quem vai ver.
  - Defina qual formato você quer gerar.
  - Faça sua análise depois de cada aula, será possível avançar no trabalho. Tragam dúvidas nas monitorias. Isso será a tarefa semanal de vocês. Também teremos leituras adicionais.
  - Entrega: Uma pasta com o projeto `.Rproj`, o código da análise (em `.Rmd` ou `.qmd`), o arquivo final (seja PDF, HTML, Word, etc), e quaisquer outros arquivos necessários. Se quiser, deixe em um repositório público no GitHub e envie o link.
- Pontos de avaliação: Reprodutibilidade, efetividade da comunicação, atributos estéticos, engajamento na leitura.

# Introdução ao RMarkdown

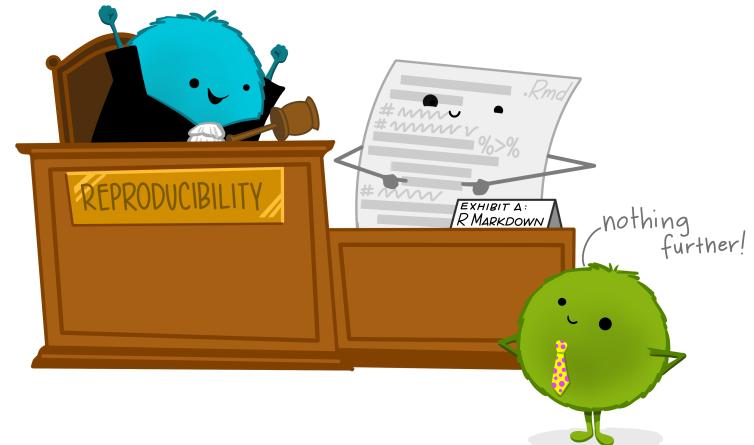
## Aula 1

# O que é RMarkdown?

- O R Markdown é um pacote para criação de **relatórios automatizados** utilizando as linguagens de programação R e de marcação Markdown.
  - | Linguagens de marcação utilizam marcadores (símbolos, tags, funções) para formatar um arquivo de texto simples. Os exemplos mais famosos de linguagem de marcação são o HTML e LaTeX.
- A linguagem de marcação Markdown serve para construirmos e formatarmos diversos formatos de arquivos (PDF, HTML, Word, entre outros) a partir de um arquivo de texto com regras bem simples.
- O **RMarkdown** é uma extensão de Markdown que nos permite **colocar código de R**.

# Por que o RMarkdown é legal?

- Possibilita automatizar a criar produtos com o R. Construindo um relatório em RMarkdown, com exceção das interpretações e conclusões, só precisamos montá-lo uma vez. A partir daí, com apenas um clique podemos:
  - replicar o relatório para diversas versões da base de dados (modificações, correções, processos periódicos);
  - replicar o relatório para diversas variáveis.
- Criar documentos reproduutíveis
- Criar diversos tipos de produtos



@allison\_horst

# Algumas aplicações

Pacote	O que podemos fazer?	Exemplos
blogdown	Blogs e sites	<ul style="list-style-type: none"><li>- Site da Curso-R</li><li>- Site da Associação Brasileira de Jurimetria (ABJ)</li></ul>
distill	Blogs e sites	<ul style="list-style-type: none"><li>- RStudio AI Blog</li><li>- Laboratório da Associação Brasileira de Jurimetria (ABJ)</li></ul>
bookdown	Livros	<ul style="list-style-type: none"><li>- Livro Ciência de Dados em R</li><li>- R for Data Science</li></ul>
xaringan	Apresentações	<ul style="list-style-type: none"><li>- Essa apresentação (e todas as usadas nos cursos da Curso-R)</li><li>- Build Your Own Universe - Garrick Aden-Buie &amp; Travis Gerke</li></ul>

# Algumas aplicações

Pacote	O que podemos fazer?	Exemplos
<code>pagedown</code>	Currículos, cartas e trabalhos acadêmicos	- <a href="#">Relatório do Observatório da Insolvência</a> - <a href="#">Currículo do William Amorim</a>
<code>flexdashboard</code>	Dashboards	- <a href="#">Tesouro Nacional: Análises sobre SICONF: Despesas com educação x IDEB</a>
<code>learnr</code>	Tutoriais interativos	- <a href="#">Text mining with tidy data principles</a> - <a href="#">Teacups, Giraffes, &amp; Statistics</a>
<code>rticles</code>	Artigos científicos	
	Outros exemplos	<a href="#">Newsletter Garimpo</a>

# Próxima geração: Quarto

O RMarkdown não deixará de existir, mas hoje temos uma alternativa mais poderosa, chamada Quarto.

Para pessoas que usam R, a vantagem do Quarto é apenas ter algumas facilidades a mais (como *autocomplete* de parâmetros no RStudio) e novos templates.

No entanto, por ser um software **independente de R**, o Quarto exporta todas as qualidades do RMarkdown para outras linguagens, especialmente Python, Julia e JavaScript.

Na aula de hoje, **mostraremos exemplos em RMarkdown e Quarto**. A partir da segunda aula, **utilizaremos apenas Quarto**.

# Estrutura

Todo arquivo RMarkdown/Quarto terá a seguinte estrutura:

- Um preâmbulo com configurações
- Blocos de texto (marcados em Markdown)
- Blocos de código (em R ou outra linguagem)

```
---
```

```
title: "Relatório maravilhoso"
```

```
output: html_document
```

```
---
```

```
Texto em __RMarkdown/Quarto__!
```

```
```{r}
```

```
print("ola, codigo R")
```

```
```
```

Dependendo do formato de saída, o documento pode precisar ser dividido em vários arquivos ou precisar de arquivos adicionais.

# Sintaxe Markdown

RMarkdown é Markdown + código. Então, precisamos entender Markdown!

Principais marcadores utilizados para formatar texto:

- uma palavra entre asteriscos fica em itálico: `*texto*` é transformado em *texto*
- uma palavra entre dois asteríscos fica em negrito: `**texto**` é transformado em **texto**
- um ou mais hashtags viram títulos: `# Título muito grande, ## Título grande, ### Título médio, ##### Título pequeno, ##### Título muito pequeno`
- hiperlinks podem ser criados com a estrutura `[texto](link)`:
- `[link para o site da Curso-R](https://curso-r.com)` é transformado em [link para o site da Curso-R](https://curso-r.com).
- para deixar o texto com `esse formato` (formato de código), apenas coloque o texto entre duas crases.

Material de consulta: [Livro: Ciência de Dados em R - Seção sobre Markdown](#)

# Visual Editor

- As versões mais recentes do RStudio permitem usar o Visual Editor.

Leitura indicada:

- <https://www.rstudio.com/blog/exploring-rstudio-visual-markdown-editor/>
- <https://www.rstudio.com/blog/rstudio-v1-4-preview-visual-markdown-editing/>

The screenshot shows the RStudio interface with a document titled "relational-data.Rmd". The code block contains the following text and code:

**Filtering joins**

Filtering joins match observations in the same way as [mutating joins](#), but affect the observations, not the variables<sup>1</sup>. There are two types:

|                              |                      |  |
|------------------------------|----------------------|--|
| <code>semi_join(x, y)</code> | $x \ltimes y$        | Keeps all observations in $x$ that have a match in $y$ |
| <code>anti_join(x, y)</code> | $x \triangleright y$ | Drops all observations in $x$ that have a match in $y$ |

Graphically, a semi-join looks like this:

```
{r, echo = FALSE, out.width = NULL}
knitr::include_graphics("diagrams/join-semi.png")
```

The diagram illustrates a semi-join operation between two tables, x and y. Table x has rows x1, x2, and x3 with values 1, 2, and 3 respectively. Table y has rows y1, y2, and y3 with values 1, 2, and 3 respectively. A green circle at the top indicates a match between x1 and y1. A purple circle indicates a match between x2 and y2. A yellow circle indicates a match between x3 and y3. An arrow points from the joined rows to a resulting table:

| key | val_x |
|-----|-------|
| 1   | x1    |
| 2   | x2    |

Only the existence of a match is important; it doesn't matter which observation is matched. This means that filtering joins never duplicate rows like mutating joins do:

Fonte: RStudio

# Chunks - Campos de código

- Em um arquivo `.qmd` ou `.Rmd`, precisamos escrever nossos códigos dentro dos *chunks*. Para inserir um chunk, utilize o atalho `CTRL + ALT + I`.
- É possível adicionar campos de código utilizando a seguinte sintaxe:

```
```{r}
codigo em R aqui
```
```

- Dentro dos chunks você poderá escrever códigos em R como se fosse o nosso script `.R` tradicional.

Material de consulta: [Livro: Ciência de Dados em R - Seção sobre R Markdown](#)

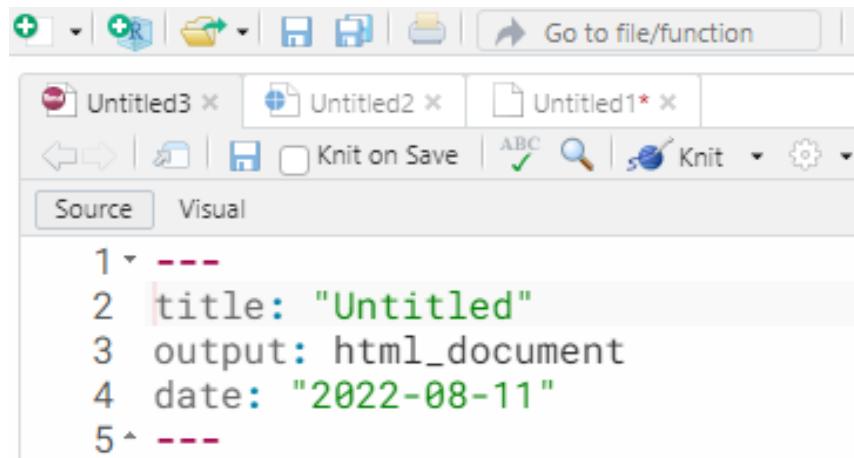
# YAML

- Os arquivos `.qmd` ou `.Rmd` começam com códigos em yaml/yml.
- Esse "bloco" de código é delimitado por `---`.
- Apresenta metadados e parâmetros utilizados para gerar o documento final.

```
---
title: "Relatórios automáticos"
subtitle: "O poder do RMarkdown"
author: "Curso-R"
date: "Março de 2021"
output: xaringan::moon_reader
---
```

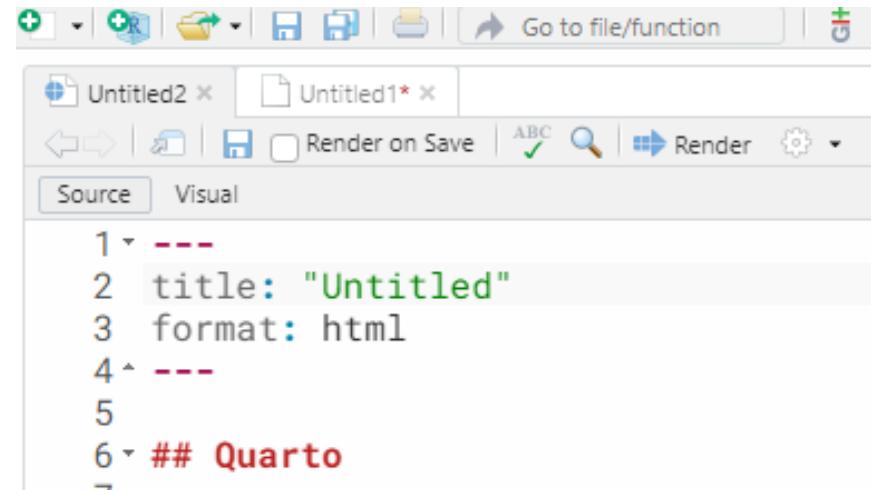
# Knit!

- Para gerarmos o relatório na extensão desejada, precisamos *renderizá-lo*, isto é, transformar o arquivo `.qmd` ou `.Rmd` em um PDF, HTML ou Word.
- Isso pode ser feito no RStudio a partir dos botões `Knit` (`.Rmd`) ou `Render` (`.qmd`), que fica logo acima do script, ou pelo atalho `CTRL + SHIFT + K`.



A screenshot of the RStudio interface showing a .Rmd file named "Untitled1". The "Knit" button in the toolbar is highlighted with a red box. The code in the editor is:

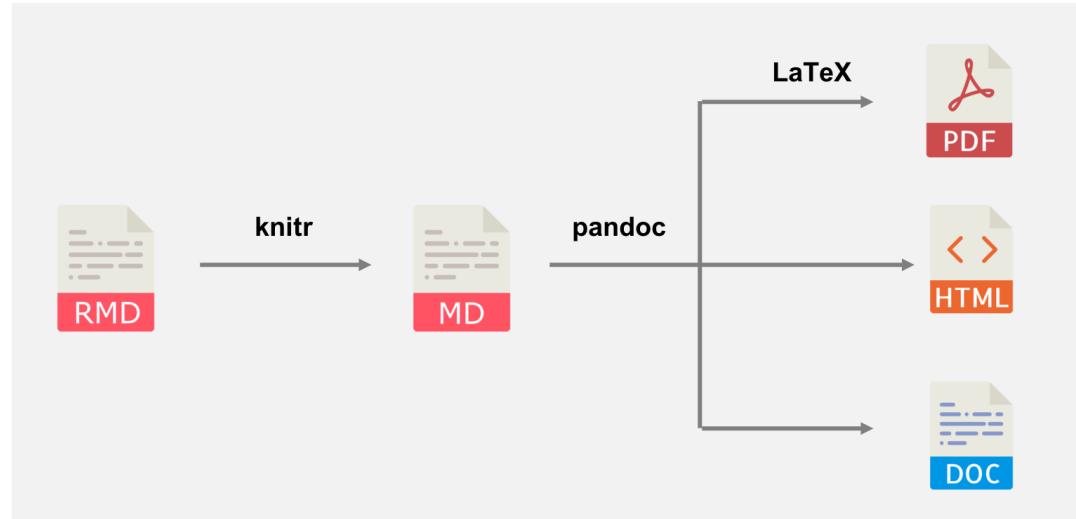
```
1 ---  
2 title: "Untitled"  
3 output: html_document  
4 date: "2022-08-11"  
5 ---
```



A screenshot of the RStudio interface showing a .qmd file named "Untitled1". The "Render" button in the toolbar is highlighted with a red box. The code in the editor is:

```
1 ---  
2 title: "Untitled"  
3 format: html  
4 ---  
5  
6 ## Quarto
```

# O que acabamos de fazer

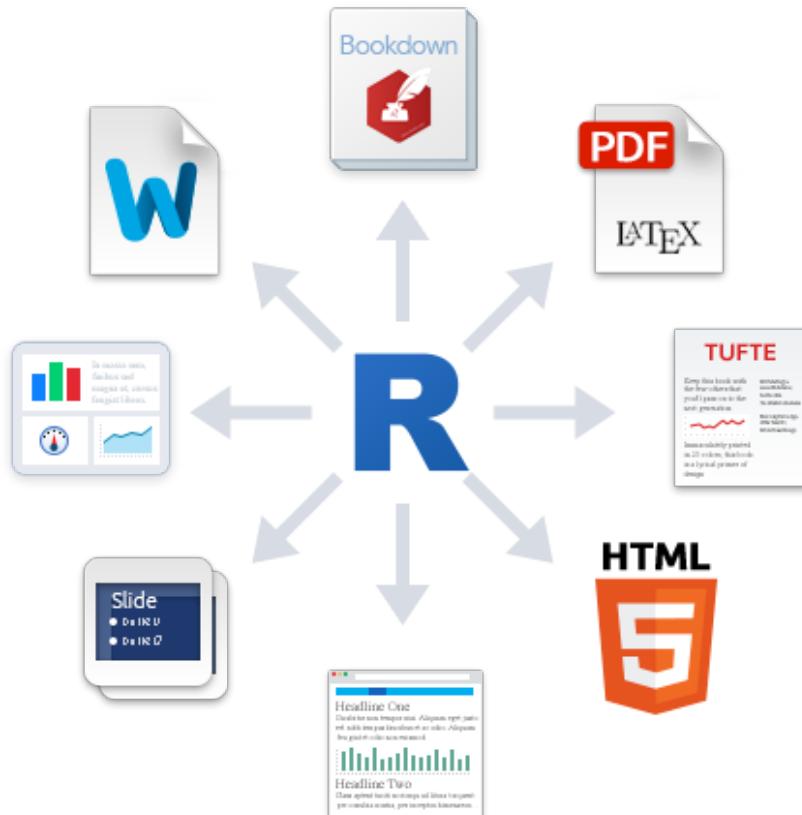


Fontr: Blog do Yihui Xie



Fonte: quarto.org

# Principais saídas



Fonte: Pacote *rmarkdown*

# Relatórios (.Rmd)

HTML

```
output: html_document
```

Word (.docx)

```
output: word_document
```

# Relatórios (.qmd)

HTML

```
format: html
```

Word (.docx)

```
format: docx
```

# Relatórios (.Rmd)

## PDF (com $TEX$ )

```
output: pdf_document
```

Caso não tenha o  $TEX$ :

```
tinytex::install_tinytex()
```

# Relatórios (.qmd)

## PDF (com *TEX*)

```
format: pdf
```

Caso não tenha o *TEX*:

```
tinytex::install_tinytex()
```

# Relatórios

## PDF com o pacote Pagedown

```
output: html_document  
knit: pagedown::chrome_print
```

Ou utilize a função `pagedown::chrome_print()` em um relatório em HTML:

```
pagedown::chrome_print("relatorio.html")
```

Obs: Por enquanto não existe uma alternativa Quarto a isso.

# Apresentações (.Rmd)

- **Powerpoint**

```
output: powerpoint_presentation
```

- **HTML**

```
output: xaringan::moon_reader
```

# Apresentações (.qmd)

- **Powerpoint**

```
format: pptx
```

- **HTML**

```
format: revealjs
```

# Resumo

- Podemos escrever arquivos com código, resultados, textos, imagens, etc.
- O mesmo conteúdo pode ser disponibilizado em diferentes tipos de saída.
- Nos metadados (yaml) do arquivo `.Rmd` ou `.qmd`, definimos a saída no parâmetro `output`.
- Mais saídas e comparação completa:
  - [RMarkdown](#)
  - [Quarto](#)

# Avançado e automatização

## Aula 2

# Chunks de Código em R

- Os chunks são campos onde podemos inserir código de R (ou Python, SQL, Bash...) em um arquivo.
- Os chunks podem ter rótulos/nomes, para identificar o conteúdo gerado.
- Existe um atalho do teclado para criar chunks no RStudio: `Ctrl + Alt + I`.
- Também é possível criar um chunk clicando no botão  do RStudio.

# Opções de Chunk

- Opções de chunk que podem afetar como os chunks de código são compilados. Exemplos:
- echo: false - evita que o próprio código apareça
- eval: false - mostra o código, mas ele não é executado
- warning: false e message: false - oculta mensagens de avisos produzidas
- out-width - controla a largura das figuras, gráficos, tabelas geradas (Ex: out-width: "100%")
- Ex de configuração do chunk

```
```{r}
#| warning: false
#| message: false
```
```

**Obs:** O RMarkdown permite outra forma de incluir opções de chunk, mas ela não é mais recomendada.

# Códigos em R - Opções de chunk

## Código + Resultado

- echo: true

```
nrow(mtcars)
```

```
## [1] 32
```

## Apenas código

- echo: true, eval: false

```
nrow(mtcars)
```

## Apenas resultado

- echo: false

```
## [1] 32
```

# Opções globais de Chunk

As opções globais de chunk são opções de chunk que são válidas para o documento inteiro. Algumas opções são úteis, como `fig.align = "center"`. Para configurar as opções globais de chunk, modifique o código abaixo e insira após o código YAML (retire os # no início de cada linha):

- Ex:
  - Configurações do chunk: `{r setup, include=FALSE}`
  - Conteúdo do chunk: `knitr::opts_chunk$set(...)`
- [Alternativa com Quarto aqui](#)

# Adicionando imagens usando o knitr

- Função: `knitr:::include_graphics()`
- Opções de Chunk:
  - `out-width: "30%"` - Largura da imagem
  - `fig-align: center` - Alinhamento da imagem
  - `fig-cap: "Logo R"` - Legenda da imagem

Exemplo:

```
knitr:::include_graphics("https://www.r-
```



Logo R

Sugestão de leitura: [Opções do knitr](#)

# Adicionar gráficos

- Você pode adicionar um chunk de código com o código que gera o gráfico.
- Controle como isso aparecerá no relatório usando as opções de chunk.

# Tabelas com R

- Várias opções: `knitr::kable()`, `DT::datatable()` , `reactable::reactable()`, entre outras.

```
mtcars |>  
head(3) |>  
knitr::kable()
```

|               | 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    |

# Código inline

- É útil para adicionar resultados de código em R dentro de parágrafos.

## Exemplo:

A base mtcars possui 32 carros. As colunas presentes na base são mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, e carb.

## Código Markdown:

```
A base mtcars possui `r nrow(mtcars)` carros.  
As colunas presentes na base são  
`r knitr:::combine_words(names(mtcars), and = "e ")`.
```

# Parâmetros

Devemos adicionar os parâmetros (e algum valor padrão) no YAML.

Ex:

```
---
```

```
title: "Relatório com Parâmetros"
author: "Beatriz"
date: "Fevereiro de 2022"
output: html
params:
  mes: 12
  ano: 2021
---
```

Para acessar os parâmetros, usamos:  
`params$nome_parametro`.

Ex: `params$mes` para acessar o mês, ou  
`params$ano`.

Recomendação de leitura: [R Markdown: The Definitive Guide - Chapter 15 Parameterized reports](#)

# Adicionar equações

- Podemos adicionar equações em LaTeX. Dica: [Equações em LaTeX no Mettzer](#), ou pesquise no google por "Equações em LaTeX".
- Equação centralizada: Envolver a equação por dois \$.

$$\text{Média} = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

- Equação junto ao texto: Envolver a equação por um \$.

Ou também na linha  $\text{Média} = \frac{a_1 + a_2 + \cdots + a_n}{n}$ , junto ao texto!

## Código Markdown:

```
$$\text{Média}=\frac{a_1+a_2+\cdots+a_n}{n}$$
```

# Adicionar referências

- Podemos adicionar referências no texto usando um arquivo `.bib`.
- Podemos gerar um arquivo `.bib` usando um gerenciador de referências. Eu utilizo o [Zotero](#), a extensão [Better Bibtex for Zotero](#) e a extensão do Zotero para o navegador.
- No arquivo `.bib`, cada referência deverá ter um rótulo, que usaremos para citar dentro do arquivo `.qmd`, precedido de `@`.
- Devemos indicar os arquivos `.bib` no YAML do relatório. Exemplo em um caso com 2 arquivos diferentes:

```
bibliography: [referencias/packages.bib, referencias/zotero.bib]
```

# Citar o R - Função citation()

```
> citation()
```

To cite R in publications use:

R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

A BibTeX entry for LaTeX users is

```
@Manual{,  
  title = {R: A Language and Environment for Statistical Computing},  
  author = {{R Core Team}},  
  organization = {R Foundation for Statistical Computing},  
  address = {Vienna, Austria},  
  year = {2021},  
  url = {https://www.R-project.org/},  
}
```

We have invested a lot of time and effort in creating R, please cite it when using it for data analysis. See also ‘citation("pkgname")’ for citing R packages.

# Citar pacotes

- Podemos gerar um `.bib` com as referências de pacotes utilizados usando a função `knitr::write_bib()`:

```
pacotes <- c("tidyverse")
knitr::write_bib(pacotes, # pacotes para gerar a referencia
                 # local para salvar o arquivo
                 'packages.bib')
```

```
@Manual{R-tidyverse,
  title = {tidyverse: Easily Install and Load the Tidyverse},
  author = {Hadley Wickham},
  year = {2021},
  note = {R package version 1.3.1},
  url = {https://CRAN.R-project.org/package=tidyverse},
}

@Article{tidyverse2019,
  title = {Welcome to the {tidyverse}},
  author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy D'Agostino
  year = {2019},
  journal = {Journal of Open Source Software},
  volume = {4},
  number = {43},
  pages = {1686},
  doi = {10.21105/joss.01686},
}
```

# Como citar no .qmd

- Formas de usar os rótulos gerados no arquivo .qmd:

| forma_de_citar   | resultado                                     |
|--|---|
| @tidyverse2019   | Wickham et al. (2019)                         |
| [@tidyverse2019; @R-tidyverse]                           | (Wickham et al. 2019; Wickham 2021)           |
| Em -@tidyverse2019, Hadley e colaboradores...            | Em 2019, Hadley e colaboradores...            |
| Segundo Hadley Wickham e colaboradores [-@tidyverse2019] | Segundo Hadley Wickham e colaboradores (2019) |

# Formatação das referências

- O conteúdo do arquivo `.bib` é formatado a partir de um arquivo `.csl` (Citation Style Language).
- Devemos informar caminho até o arquivo CSL no YAML, por exemplo:

```
csl: template/abnt.csl
```

- Lugares para baixar arquivos CSL:
  - <https://github.com/citation-style-language/styles>
  - <https://citationstyles.org/>

# Referências cruzadas

- **Figuras, tabelas, equações, seções:**
  - Usamos o rótulo do chunk (ou nome) onde a tabela ou imagem foi criada para fazer a referência cruzada.
  - Ex: @fig-rotulo-da-imagem , @tab-rotulo-da-tabela, @eq-rotulo-da-equacao, @sec-rotulo-da-secao.
- Para fazer referência cruzada em capítulos/seções, devemos adicionar um rótulo para o capítulo, que deverá ser escrito logo após a cabeçalho (que delimitamos com #, ##, etc).
- Obs: No RMarkdown, funciona com outputs que são derivados do pacote {bookdown}. Ex:  
bookdown::html\_document2, bookdown::word\_document2
  - Referência: <https://bookdown.org/yihui/rmarkdown-cookbook/cross-ref.html>

# Compartilhando os resultados

## Aula 3

# Compartilhando os resultados

Podemos facilmente enviar os arquivos PDF e .docx via e-mail. Porém e no caso do output ser um HTML?

No HTML, muitas vezes o arquivo depende de outros arquivos (ou seja, ele não é auto-contido).

Podemos usar uma opção que permite fazer um arquivo autocontido, sem dependências (porém isso deixará seu arquivo grande, e também pode demorar para ser gerado):

```
---
```

```
format:
  html:
    self-contained: true
```

```
--
```

<https://bookdown.org/yihui/rmarkdown/html-document.html>

# Disponibilizar na internet

Outra forma é disponibilizar o HTML na internet, com serviços como Netlify Drop ou GitHub !

DICA IMPORTANTE: Busque nomear seus arquivos como `index.html`

## Disponibilizar utilizando o Netlify Drop

- <https://app.netlify.com/drop>
- Gratuito
- Para quem não usa GitHub.
- Podemos subir uma pasta com todo o conteúdo dentro.
- Mais fácil, mas não dá pra ficar atualizando o material depois de um jeito fácil.

# Disponibilizar utilizando o GitHub

- Gratuito
- Precisa ter o Git e GitHub configurado com o seu RStudio (esse vídeo é útil: <https://youtu.be/fiZStofJqMQ?t=2415>)
- O projeto precisa estar vinculado a um repositório no GitHub
- No repositório do pacote, clique em **Settings**, e no menu lateral escolha **Pages**
- Em **Source**, selecione a sua branch principal (provavelmente chama de **main** ou **master**), escolha o diretório onde seus arquivos estão: `docs/` ou `root/` (a pasta raíz) e clique em **save**.
- O endereço do site será `https://seu-username.github.io/nome-repositorio/caminho-do-arquivo.html`

# Customização

## Aula 3

# Usar um modelo para word

- Em alguns casos, somos obrigados a enviar arquivos .docx (ex: algumas revistas científicas apenas aceitam esse tipo de arquivo).
- Podemos informar um template de arquivo word como referência. As regras de formatação serão copiadas. Não funciona perfeitamente!

format:

word:

```
reference_doc: template/modelo.docx
```

# Exportar um .docx tunado

- Em alguns casos, somos obrigados a enviar arquivos .docx (ex: algumas revistas científicas apenas aceitam esse tipo de arquivo).
- Exemplo de YAML:

```
format:  
  word:  
    reference_doc: template/modelo.docx  
    number-sections: false  
bibliography: [referencias/packages.bib, referencias/zotero.bib]  
csl: template/abnt.csl
```

# Introdução à HTML e CSS

- Quando criamos um arquivo RMarkdown que gera um output em `.html`, esse arquivo `.html` é interpretado pelo navegador (ex. Chrome), utilizando também os arquivos `.css` e `.js`
  - **HTML** (HyperText Markup Language - Linguagem de Marcação de Hipertexto): é uma linguagem de marcação, é usado para estruturar páginas da internet (websites).
  - **CSS** (Cascading Style Sheets - Folha de Estilo em Cascata): é usado para estilizar os elementos escritos no HTML.
  - **Javascript**: é uma linguagem de programação, permite que as páginas sejam dinâmicas.





- HTML descreve a estrutura de uma página web.
- HTML consiste em uma série de elementos. Estes elementos mostram para o navegador (o browser) como apresentar o conteúdo.
- Um elemento em HTML consiste em uma tag inicial, algum conteúdo que será marcado, e a tag final (que é diferente da tag inicial por possuir a barra / )

```
<nomedatag>Conteúdo a ser marcado vai aqui...</nomedatag>
```

- Elementos em HTML podem ter atributos de classe (`class`). As classes geralmente são usadas para referenciar um estilo.

```
<nomedatag class="nomedaclasse">  
Conteúdo a ser marcado vai aqui...  
</nomedatag>
```

# Exemplo



---

Código em Markdown

Código HTML

Como aparece

---

Este é o curso de \*\*Relatórios e visualização de dados\*\*,  
oferecido pela [Curso-R] (<https://curso-r.com/>) .

# CSS



- CSS é uma sigla para *Cascading Style Sheets* (tradução literal: folha de estilo em cascada).
- CSS descreve como os elementos em HTML serão apresentados.
- Você pode criar estilos no seu arquivo `.qmd`, ou também salvar em um arquivo externo. As folhas de estilo são salvas em arquivos com a extensão `.css`.
- Você pode criar um arquivo CSS e reaproveitar em diversos arquivos HTML.
- Criar estilos no seu arquivo `.Rmd`:
  - Colocando o estilo dentro de uma tag html `<style>`
  - Criando um Chunk de código CSS

# Exemplo de código CSS



---

Exemplo de elementos

Exemplo de classe

---

- Para elementos, usamos a seguinte estrutura:

```
nomedoelemento {  
    propriedade: valor da propriedade;  
}
```

Exemplo:

```
p {  
    font-size: 20px;  
}
```

# Exemplo 1



---

Código HTML

Como aparece

---

```
<style>
.textorosa{
color: pink;

}
</style>
<p>
Este é o curso de
<b class="textorosa">Relatórios e visualização de dados</b>,
oferecido pela <a href='https://curso-r.com/'>Curso-R</a>.
</p>
```

## Exemplo 2



Código em Markdown

Como aparece

```
```{css}
.textoroxo{
  color: #6A0DAD;
}
````
```

Este é o curso de `.textoroxo[**Relatórios e visualização de dados**]`, oferecido pela [Curso-R] (<https://curso-r.com/>).

# Dica de ouro



Use **muito** as ferramentas de desenvolvedor(a) do navegador.

Permite explorar o conteúdo de uma página web.

Atalho costuma ser: **CTRL + SHIFT + C**

- Firefox - Page inspector
- Chrome

# Extensões

## Aula 4

# Extensões

- Quarto: [Quarto Guide](#)

- ! : muita coisa!

- Relatórios:

- [pagedown](#)
  - [rticles](#)
  - [bookdown](#)

- Apresentações:

- [xaringan](#)

- Dashboards:

- [flexdashboard](#)

# Referências

- Site do Quarto
- Site do pacote RMarkdown
  - Página com diversas referências
- Folha de cola - *cheatsheet*
- Livro: *R Markdown: The Definitive Guide*
- Livro: *R Markdown Cookbook*
- Pessoas para seguir no Twitter:
  - Yihui Xie
  - Garrick Aden-Buie
  - Alison Presmanes Hill
- Livro da Curso-R - Capítulo sobre relatórios

# Referências

- Mozilla Web Docs
- w3schools.com - HTML
- w3schools.com - CSS
- Free Code Camp
- Palestra do Garrick Aden-Buie: Using xaringan to learn web development