

Linguagens de Programação: R

Aula 04: Comunicação

Profa. Deisy Morselli Gysi, Ph.D.



Comunicando Resultados

- **Comunicação** é a chave para o sucesso de qualquer projeto.
- Saber como comunicar resultados é tão importante quanto saber como obtê-los.








Quem não se comunica, se trumbica! (Chacrinha)

Quarto

- **Quarto** é uma ferramenta poderosa para comunicação de resultados.
- Permite a criação de relatórios dinâmicos e interativos, sites, apresentações, dashboards, livros, blogs, entre outros.
- Integração com R, Python, Julia, Observable, Mermaid, etc.
- Exportação para diversos formatos (HTML, PDF, Word, PowerPoint).



Instalação

 [Overview](#) [Get Started](#) [Guide](#) [Extensions](#) [Reference](#) [Gallery](#) [Blog](#) [Help](#)    




Welcome to Quarto







An open-source scientific and technical publishing system

- Author using [Jupyter](#) notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with [Python](#), [R](#), [Julia](#), and [Observable](#).
- Publish reproducible, production quality articles, presentations, dashboards, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share knowledge and insights organization-wide by publishing to [Posit Connect](#), [Confluence](#), or other publishing systems.
- Write using [Pandoc](#) markdown, including equations, citations, crossrefs, figure panels, callouts, advanced layout, and more.

Analyze. Share. Reproduce. You have a story to tell with data—tell it with Quarto.

[Get Started](#) [Guide](#)

-  Websites / Blogs
-  Dashboards
-  Articles
-  Presentations
-  Books
-  Knowledge Repos

Hello, Quarto

[Python](#) [R](#) [Julia](#) [Observable](#)

Combine Jupyter notebooks with flexible options to produce production quality output in a wide variety of formats. Author using traditional notebook UIs or with a plain text markdown representation of notebooks.



Instalação

Get Started

Tutorial: Hello, Quarto

Tutorial: Computations

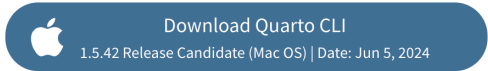
Tutorial: Authoring

Get Started

Install Quarto, then check out the tutorials to learn the basics.

Step 1

Install Quarto



Platform	Download	Size	SHA-256
Ubuntu 18+/Debian 10+	quarto-1.5.42-linux-amd64.deb	119.21 MB	6922c56
Linux Arm64	quarto-1.5.42-linux-arm64.deb	122.86 MB	a50bfd5
Mac OS	quarto-1.5.42-macos.pkg	196.78 MB	ee20548
Windows	quarto-1.5.42-win.msi	113.67 MB	bb3d436
Release notes and more downloads...			

Step 2

Choose your tool
and get started



Tutorial: Hello, Quarto →



Escolha sua ferramenta (IDE)

Tutorial: Hello, Quarto

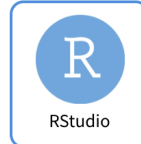
Choose
your
tool



VS Code



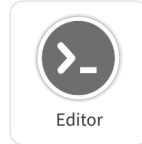
Jupyter



RStudio



Neovim



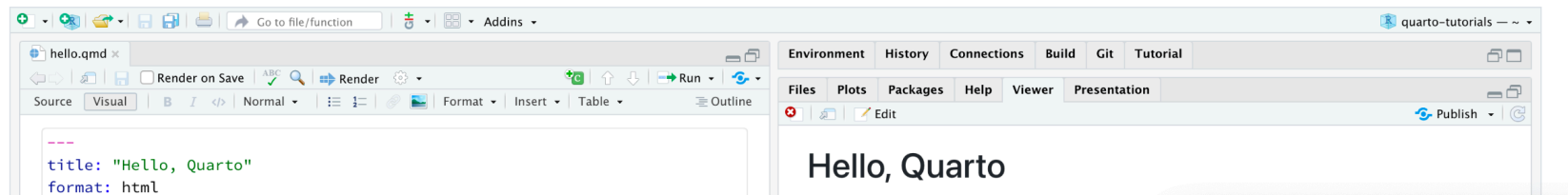
Editor

Overview

Quarto is a multi-language, [next-generation](#) version of R Markdown from Posit and includes dozens of new features and capabilities while at the same being able to render most existing Rmd files without modification.

In this tutorial, we'll show you how to use RStudio with Quarto. You'll edit code and markdown in RStudio just as you would with any computational document (e.g., R Markdown) and preview the rendered document in the Viewer tab as you work.

The following is a Quarto document with the extension `.qmd` (on the left), along with its rendered version as HTML (on the right). You could also choose to render it into other formats like PDF, MS Word, etc.



On this page

[Overview](#)

[Rendering](#)

[Authoring](#)

[How it works](#)

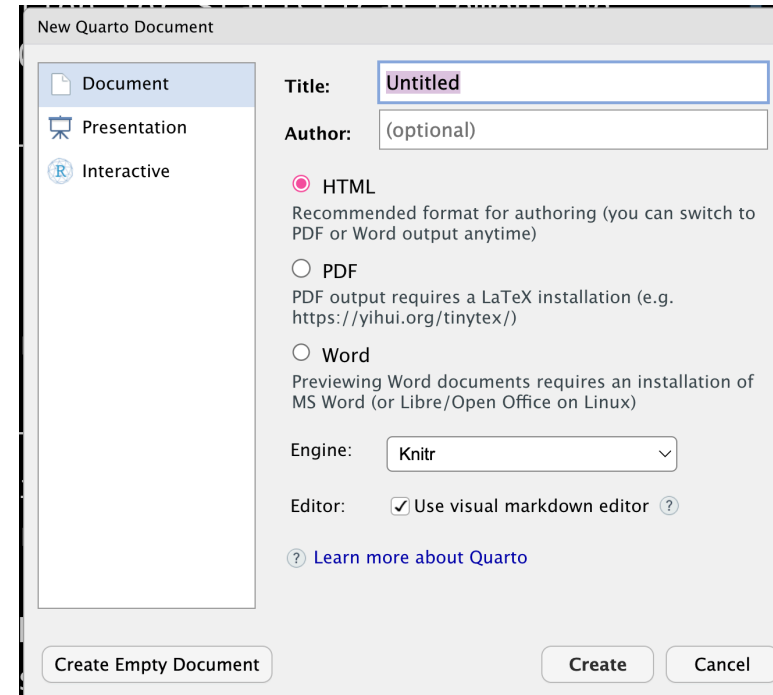
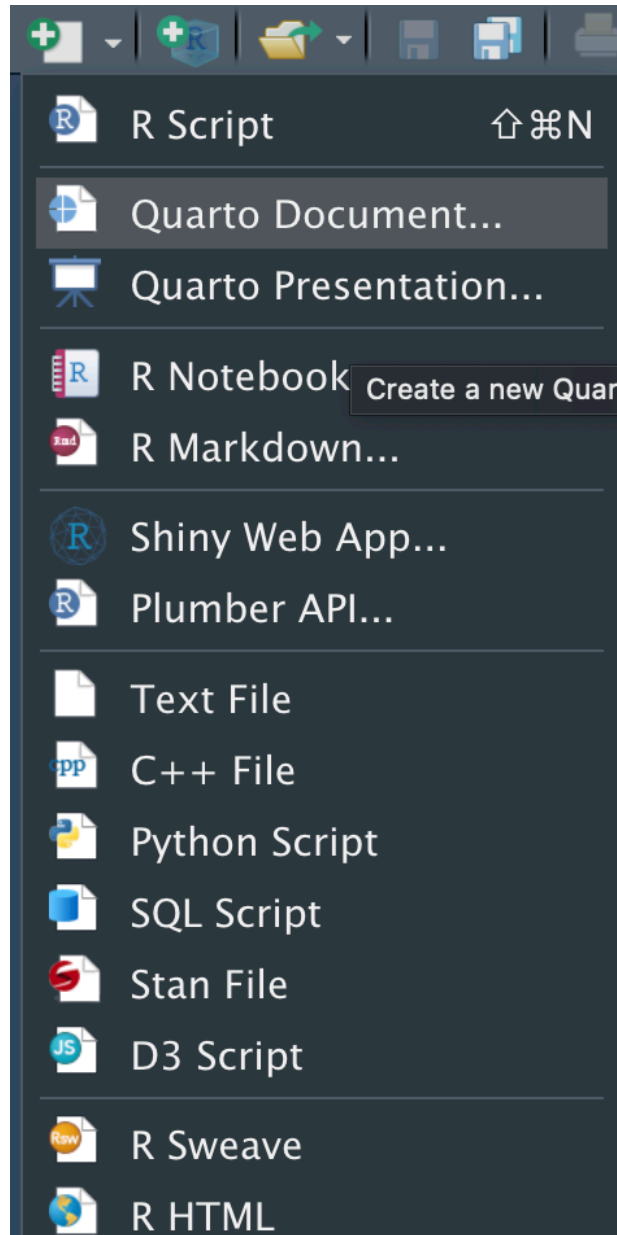
[Next Up](#)

[Edit this page](#)

[Report an issue](#)

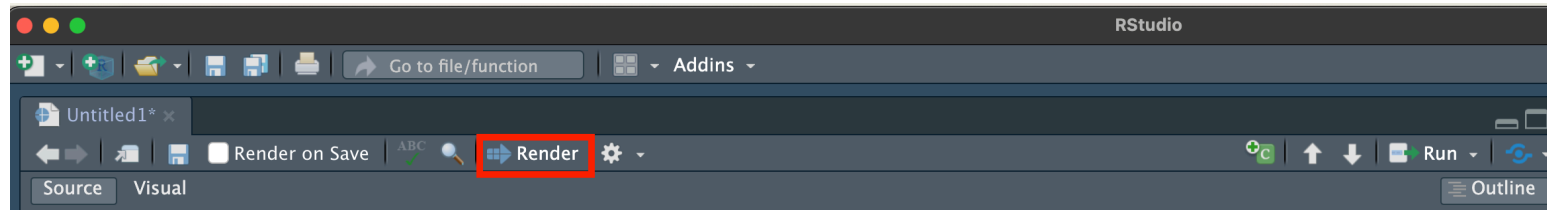


Criando um relatório



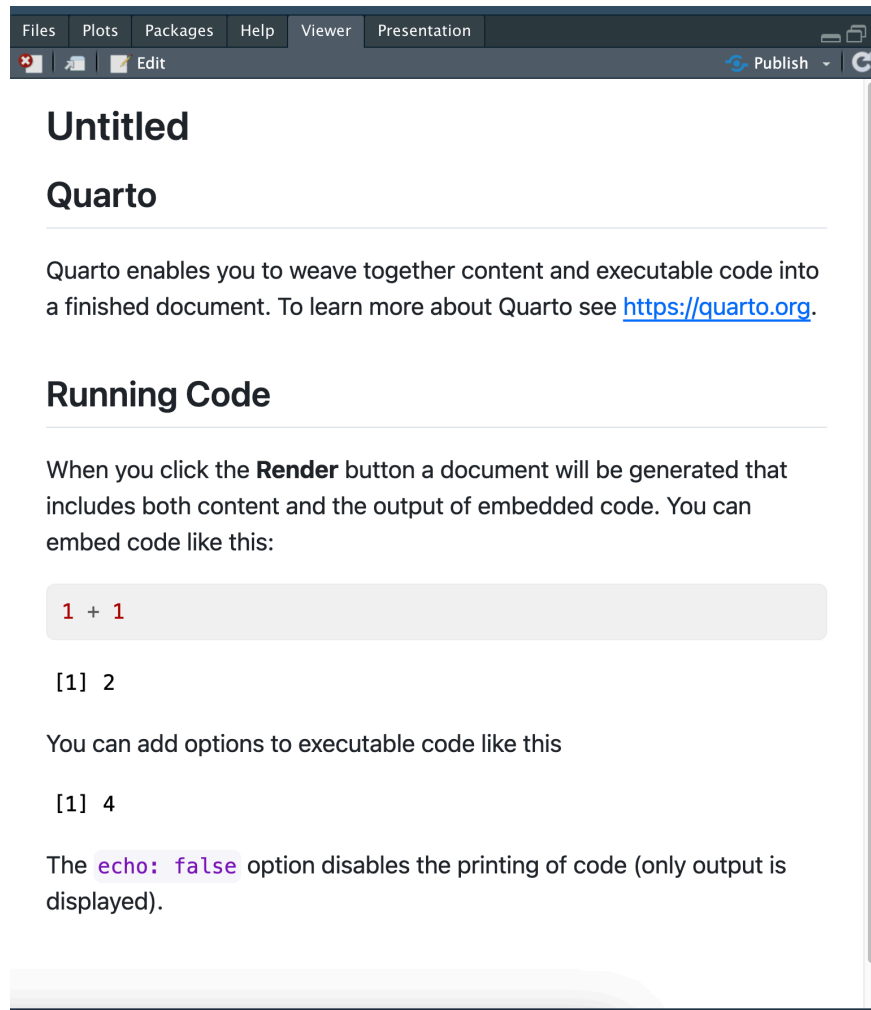
Renderizando o relatório

- Salve seu relatório (`.qmd`).
- Para renderizar o relatório, basta clicar no botão **Render**.



Visualizando o relatório

- O relatório será renderizado no painel **Viewer**.



YAML (YAML Ain't Markup Language)

- A primeira parte do documento é o cabeçalho YAML.
- O cabeçalho YAML é delimitado por três traços (—) no início e no final do documento.
- O cabeçalho YAML é usado para definir metadados, como título, autor, data, tema, etc.
- *Markup Language* é uma linguagem de marcação, como HTML, XML, markdown, etc.

```
---  
title: "Título do Documento"  
subtitle: "Subtítulo do Documento"  
author: "Meu Nome Aqui"  
date: "2024-06-06"  
date-format: long  
output: html_document  
---
```



Temas e Estilos

- O tema padrão é `default`.
- Você pode criar seu próprio tema ou usar um dos temas disponíveis.
- Para criar seu próprio tema, crie um arquivo `.SCSS` e adicione-o ao cabeçalho YAML.

```
title: "Seu Título Aqui"
subtitle: "Seu Subtítulo Aqui"
author: "Seu Nome aqui"
date: 06/06/2024
date-format: long
lang: pt
theme:
  light: yeti
  dark: slate
```

• Mais temas para o relatório HTML

Guide > Documents > HTML > HTML Theming

HTML Theming

Overview

HTML documents rendered with Quarto use Bootstrap 5 by default. This can be disabled or customized via the `theme` option:

```
theme: default # bootstrap 5 default
theme: cosmo  # cosmo bootswatch theme
theme: pandoc  # pandoc default html treatment
theme: none    # no theme css added to document
```

Quarto includes 25 themes from the [Bootswatch](#) project (for example, this website uses the [cosmo](#) theme). Available themes include:

- [default](#)
- [cerulean](#)
- [cosmo](#)
- [cyborg](#)
- [darkly](#)
- [flatly](#)
- [journal](#)
- [litera](#)
- [lumen](#)
- [lux](#)
- [materia](#)
- [minty](#)
- [morph](#)



Formatação de código

- Podemos incluir códigos no texto usando crases (`).
- Para formatar sua aparência, devemos realizar as definições no cabeçalho YAML.

```
---
title: "Seu Título Aqui"
subtitle: "Seu Subtítulo Aqui"
author: "Seu Nome aqui"
date: 10/10/2023
date-format: long # date-format: dddd MMM D, YYYY
lang: pt
title-block-style: none
theme:
  light: yeti
  dark: slate

format:
  html:
    code-fold: true
```



Formatando Figuras

- Podemos adicionar figuras ao texto usando a sintaxe `![]` (`caminho/para/imagem.png`).
- Para formatar a figura, devemos realizar as definições no cabeçalho YAML.

```
---  
title: "Seu Título Aqui"  
subtitle: "Seu Subtítulo Aqui"  
author: "Seu Nome aqui"  
date: 10/10/2023  
date-format: long  
lang: pt  
title-block-style: none  
  
fig-align: center  
fig-width: 10  
fig-height: 12  
  
format:  
  html:
```



Inserindo TOC (Table of Contents)

- Para inserir um índice no seu documento, basta adicionar `toc: true` no cabeçalho YAML.

```
---  
title: "Seu Título Aqui"  
subtitle: "Seu Subtítulo Aqui"  
author: "Seu Nome aqui"  
date: 10/10/2023  
date-format: long  
lang: pt  
title-block-style: none  
  
fig-align: center  
fig-width: 10  
fig-height: 12  
  
toc: true  
toc-depth: 5
```



Inserindo Citações

- Para inserir citações no seu documento, basta adicionar `bibliography: caminho/para/arquivo.bib` no cabeçalho YAML.
- Para definir a formatação de citações no seu documento, basta adicionar `csl: caminho/para/arquivo.csl` no cabeçalho YAML.
- Vários tipos de citações podem ser encontrados em [Citation Styles](#).

```
---
title: "Seu Título Aqui"
subtitle: "Seu Subtítulo Aqui"
author: "Seu Nome aqui"

bibliography: references.bib
csl: nature.csl
---
```

- Exemplo de `.bib`:

```
@book{wickham2016,
  title = {R for Data Science: Import, Tidy, Transform, Visualize, and Model Data},
  author = {Wickham, H. and Grolemund, G.},
  year = {2016},
  date = {2016},
  publisher = {O'Reilly Media},
}
```



Incluindo parâmetros

- Podemos incluir parâmetros no relatório para que o usuário possa interagir com o documento.

```
---  
params:  
  alpha: 0.1  
  ratio: 0.1  
---
```

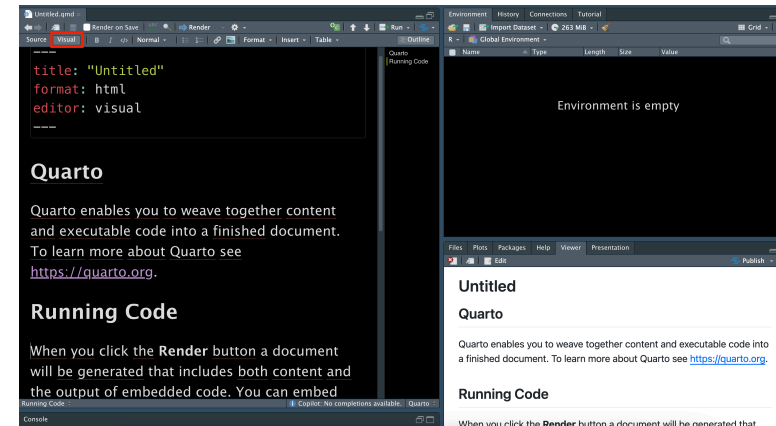
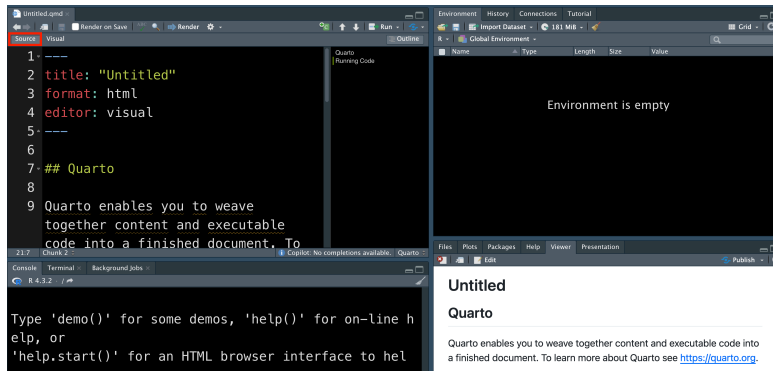
- A lista de parâmetros pode ser acessada na lista `params`.

```
```{r}  
params$alpha
```
```



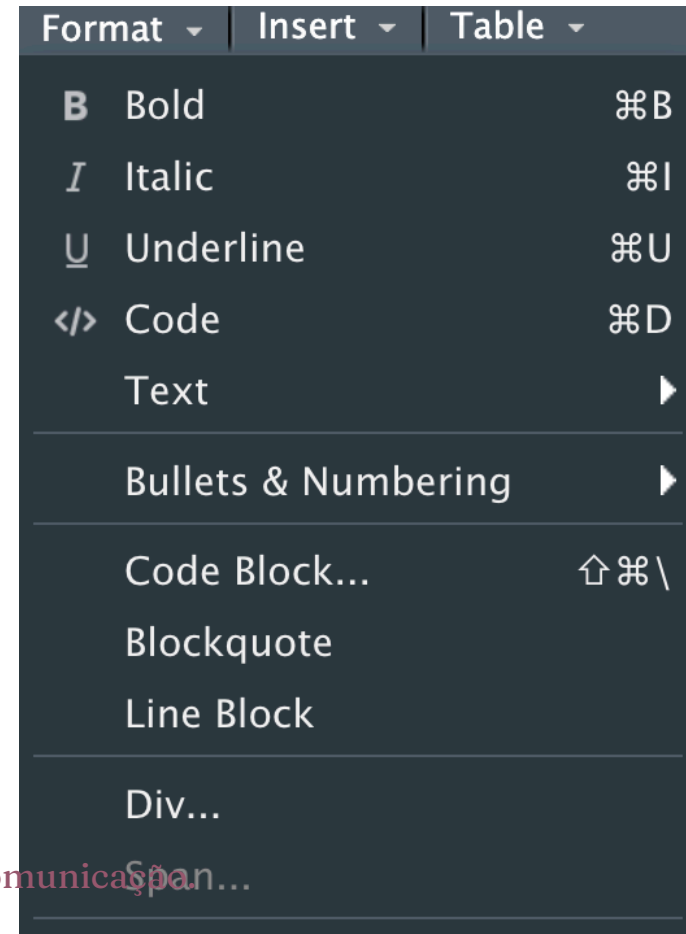
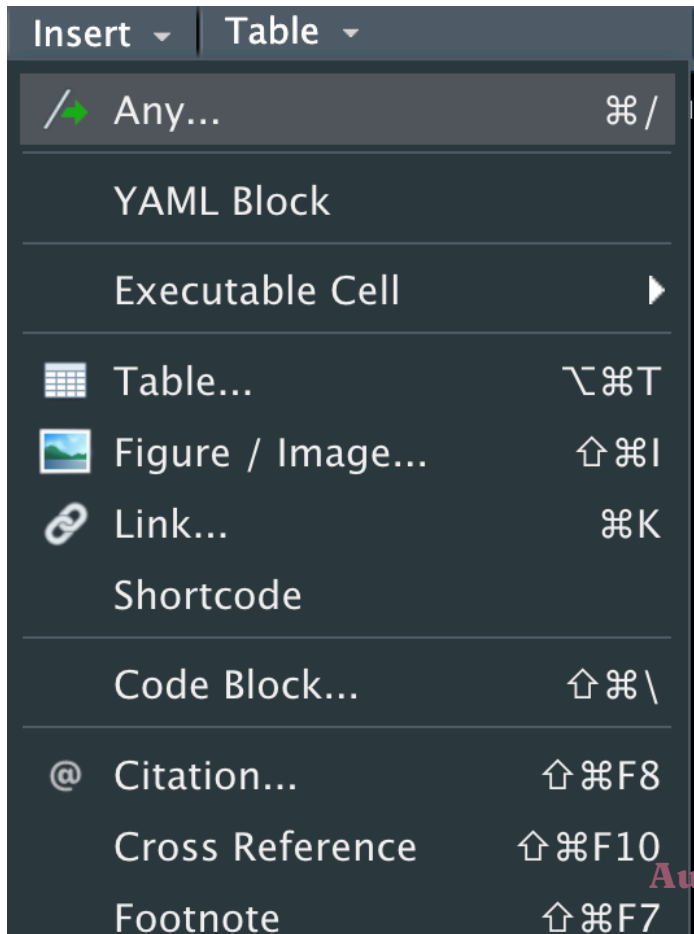
Editando o Relatório

- Podemos editar um relatório em quarto de duas maneiras:
- Editando o arquivo `.qmd` diretamente, em **Source**, neste caso, escrevemos em `markdown`.
- Editando o relatório no painel **Visual**.



Inserindo e Formatando Elementos

- Quando utilizamos o painel no modo **Visual**, podemos inserir elementos clicando nos ícones disponíveis.
- Quando utilizamos o painel no modo **Visual**, podemos formatar texto clicando nos ícones disponíveis.



Markdown

- Markdown é uma linguagem de marcação leve.
- Criada por John Gruber e Aaron Swartz em 2004.
- É uma maneira de adicionar formatação a um texto de forma rápida e direta.
- Não possui estruturas de controle ou funções, como linguagens de programação.
- Quando digitamos algo em Markdown, o texto é formatado para a saída que temos interesse, mas ainda assim vemos o código do texto.
- WYSIWYG: What You See Is What You Get é um conceito que se refere a editores de texto que mostram o texto formatado enquanto você o digita, por exemplo quando estamos na aba **Visual**.



Formatação de Texto

- **Negrito:** `**texto**` ou `__texto__`
 - **texto**
- **Itálico:** `*texto*` ou `_texto_`
 - *texto*
- **Riscado:** `~~texto~~`
 - ~~texto~~
- **Sublinhado:** `<u>texto</u>`
 - texto



Formatação de Texto

- **Título:**

```
# Título de Nível 1  
## Título de Nível 2  
### Título de Nível 3
```

- **Citação:** `> texto`

`> texto`

- **Referência Bibliográfica:** `[@wickham2016]`

- `1`

- **Imagem:** ``

- **Link:** `[texto](url)`

- `texto`



Formatação de Texto

Lista Ordenada:

1. texto
2. texto
3. texto

1. texto
2. texto
3. texto

Lista Não Ordenada:

- texto
- texto
- texto

- texto
- texto
- texto



Inserindo código

- Códigos podem ser inseridos de diversas maneiras:

- **Código em linha:**

```
`texto`
```

- `texto`

- **Código em linha com linguagem:**

```
variavel = 1234
```

O valor da variavel de interesse é `r`
variavel `.

- O valor da variavel de interesse é 1234.

- **Código em bloco:** 31b8e172-b470-440e-83d8-e6b185028602:dAB5AHAAZQA6AE8AUQBCAGoAQQFb470-440e-83d8-e6b185028602



Configurações de Bloco de Código

- Podemos configurar se um bloco de código será executado, utilizando o parâmetro `eval:true` ou `eval:false`.
- Podemos definir se a saída de um código será exibida no relatório ou não, utilizando o parâmetro `echo:true` ou `echo:false`.

```
```{r}  
#| eval: true
#| echo: true
x = 1 + 1
x
```
```



Configurações de Gráficos

- Podemos configurar o tamanho de um gráfico utilizando o parâmetro `fig-width` e `fig-height`.

```
```{r}
#| fig-width: 6
#| fig-height: 4
#| fig.cap: "Gráfico de exemplo"

ggplot(datasets::iris) +
 aes(
 x = Sepal.Length,
 y = Sepal.Width,
 fill = Species,
 colour = Species
) +
 geom_point(shape = "circle", size = 1.5) +
 geom_smooth(span = 0.75) +
 scale_fill_hue(direction = 1) +
```

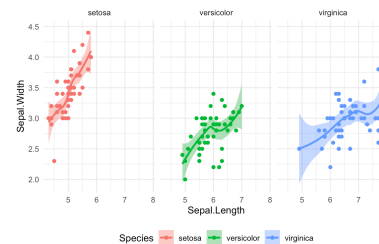


Gráfico de exemplo

# Criando Apresentações

- Podemos criar apresentações utilizando o Quarto.
- Para isso, precisamos alterar o formato do arquivo para `revealjs`, `pptx` ou `beamer`.
- O separador de slides é `##`, onde adicionamos o título do slide.
- A maioria das configurações de um relatório também podem ser utilizadas em uma apresentação.



# Criando Apresentações

```

title: "Apresentação"
author: "Autor"
output: revealjs

```

```
Slide 1
```

- Texto 1
- Texto 2

```
Slide 2
```

- Texto 3
- Texto 4

# Criando Apresentações

- Para criarmos slides de título, basta adicionar o título do slide com # antes do texto.

```

title: "Apresentação"
author: "Autor"
output: revealjs

```

```
Título 1
```

```
Slide 1
```

```
- Texto 1
```

```
- Texto 2
```

```
Slide 2
```

```
- Texto 3
```

```
- Texto 4
```



# Listas incrementais

- Podemos criar listas incrementais alterando o YAML, adicionando o parâmetro `incremental: true`. Neste caso, todas as listas serão incrementais.

```
title: "Minha apresentação"
format:
 revealjs:
 incremental: true
```

- Podemos alterar um slide específico para ser incremental, adicionando um div `incremental` no início do slide.

```
::: {.incremental}
- Texto 1
- Texto 2
:::
```

- Para um slide específico não ser incremental:

```
::: {.nonincremental}
- Texto 1
- Texto 2
:::
```



# Criando colunas

- Para criarmos múltiplas colunas em um slide, podemos utilizar o `div columns`.

```
::: {.columns}
::: {.column width="40%"}
- Texto 1
:::
::: {.column width="60%"}
- Texto 2
:::
:::
```

# Organização de Projetos

- Uma boa maneira organizar um projeto é em pastas e subpastas.
- Crie pastas separadas para dados, scripts, relatórios e apresentações.
- Dentro da pasta de dados, crie subpastas para dados brutos e dados processados.
- Dentro da pasta de scripts, crie scripts separados para cada análise.
- Nomeie os scripts de forma que seja fácil identificar o que cada um faz, e inclua sempre a ordem que devem ser executados.



# Boas Práticas de Programação

- Utilize comentários para explicar o que cada parte do código faz.
- Nomeie as variáveis de forma que seja fácil entender o que elas representam.
- Utilize funções para evitar repetição de código.
- Utilize pacotes para facilitar a leitura e escrita do código.
- Mantenha seus códigos versionados em um repositório Git.





# Versionamento de Código

- O versionamento de código é uma prática essencial para manter o histórico de alterações de um projeto.
- Isto é, mantenha seu código na versão mais recente, sem perder o histórico de alterações.
- Diversos sistemas de versionamento existem, o mais comum é o Git.
- O Git é um sistema de controle de versão distribuído, que permite que você mantenha o histórico de alterações de um projeto.
- O GitHub é uma plataforma que permite hospedar repositórios Git de forma gratuita.



# Integrando o Rstudio com o Git

- O Rstudio possui uma integração com o Git, que permite que você versiona seus códigos diretamente do Rstudio.
- Para isso, é necessário instalar o Git no seu computador e configurar o Rstudio para utilizar o Git.



# Configurando o Git

**GitHub** é uma plataforma popular para hospedar projetos versionados usando o Git. Siga os seguintes passos para criar uma conta:

1. Acesse o site oficial do GitHub em <https://github.com/>.
2. Clique em “Sign Up” ou “Criar conta”.
3. Preencha seus dados, incluindo nome de usuário, endereço de e-mail e senha.
4. Selecione um plano de conta.
5. Conclua o processo de criação de conta seguindo as instruções na tela.



# Git

1. **Repositórios:** Após criar uma conta no GitHub, você pode criar um novo repositório ou clonar (fazer uma cópia local) um repositório existente para começar a trabalhar no projeto.
2. **Commits:** Os commits são registros das alterações feitas no código. Ao fazer um commit, é importante fornecer uma mensagem clara e concisa que explique o que foi alterado. Isso ajuda na compreensão das mudanças realizadas em cada versão.
3. **Branches:** Os branches permitem que você crie cópias do seu repositório para trabalhar em funcionalidades específicas ou correções. Branches são úteis para desenvolver recursos em paralelo sem afetar o projeto principal. Quando um trabalho em um branch estiver concluído, ele pode ser mesclado ao projeto principal através de um `pull request`.



# Git

4. **Pull Requests:** Um pull request é uma solicitação para mesclar as alterações de um branch ao projeto principal.
5. **Forks:** Quando você deseja contribuir para um projeto mantido por outra pessoa, pode criar um fork, que é uma cópia independente do repositório original em sua conta do GitHub. Após fazer as alterações em seu fork, você pode enviar um `pull request` para o repositório original, sugerindo que suas mudanças sejam incorporadas ao projeto principal.



# Git: Principais Comandos

- `git init`: Inicializa um repositório Git em um diretório.
- `git add <arquivo>`: Adiciona um arquivo ao índice.
- `git commit -m "mensagem"`: Salva as alterações no repositório.
- `git push`: Envia as alterações para o repositório remoto.
- `git pull`: Atualiza o repositório local com as alterações do repositório remoto.
- `git clone`: Clona um repositório remoto para o repositório local.



# Integrando o Git ao Rstudio

- Suporte da [Posit](#) para [sincronização do Git & RStudio](#).

## Instalação

- Instale o **Git** e/ou **Subversion** em seu sistema.



# Instalando o Git

- Windows e OS X: <http://git-scm.com/downloads>
- Debian/Ubuntu: `sudo apt-get install git-core`
- Fedora/RedHat: `sudo yum install git-core`





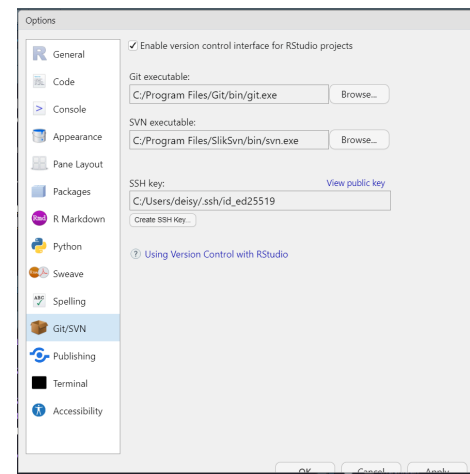
# Instalando o Subversion

- Windows: [SilkSVN](#)
- OSX ( $\leq$  v10.7): Não é necessário - já está incluído nas versões do OSX 10.7 e anteriores
- OSX (v10.8+): Instale as Ferramentas de Linha de Comando do Xcode a partir dos downloads do desenvolvedor da Apple
- Debian/Ubuntu: `sudo apt-get install subversion`
- Fedora/RedHat: `sudo yum install subversion`



# Ativando Git no RStudio

- Para ativar o sistema de versionamento de código no RStudio, siga os seguintes passos:
  - Acesse as Opções Globais (no menu Ferramentas)
  - Clique em Git/SVN
  - Clique em Habilitar interface de controle de versão para projetos do RStudio
  - Se necessário, insira o caminho para o executável do Git ou SVN, se fornecido.



Tools -> Global Options -> Git/SNV



# Configurar o Git

- Após ativar o Git no RStudio, declare para o Git seu nome e e-mail associado a sua conta do GitHub. Isso pode ser feito da seguinte forma:

1. Pelo **Terminal** no RStudio, digite esses comandos substituindo seu nome/e-mail:

```
git config --global user.name 'Seu Nome'
git config --global user.email 'seu.email@email.com'
```

2. Usando o pacote `usethis`, no **Console** do RStudio:

```
library(usethis)
use_git_config(user.name = "Seu Nome",
 user.email = "seu.email@email.com")
```



# Criar um token do GitHub

- Para que a interação entre o seu computador e o GitHub funcione corretamente, é necessário autenticação.
- Uma maneira fácil de implementar essa autenticação é configurando um token do GitHub, também conhecido como PAT (Personal Access Token).
- No **console** do RStudio:

```
usethis::create_github_token()
```

Esse comando abrirá uma página no seu navegador. Clique em *Generate token* e copie o código do token gerado.

## Note

Não feche a página. Ao fechar a página não será possível visualizá-lo novamente.

Para garantir que o RStudio irá se recordar do seu token e suas credenciais:

```
gitcreds::gitcreds_set()
```

Em seguida, cole o código obtido anteriormente.



# Teste

## Criar um repositório no GitHub

A seguir estão as etapas para criar um novo repositório no GitHub:

1. Acesse o site do GitHub e verifique se você já fez o login na sua conta.
2. Próximo à seção “Repositories”, clique no botão verde “New”.
3. Na nova página, preencha o nome do repositório (R\_Prog) e adicione uma breve descrição, se desejar.
4. Marque as opções “Public” (público) e “Add a README file” (adicionar um arquivo README).
5. Por fim, clique em “Create repository”.

Após criar o repositório, você será direcionado para a página do mesmo no GitHub. Agora, clique no botão verde escrito “Code” e copie a URL que aparece na opção “HTTPS” (recomendado).

Essas etapas permitirão que você crie um novo repositório no GitHub e obtenha a URL necessária para interagir com ele através de comandos Git.



# Clonar repositório pelo RStudio

## 1. Via Menu:

- Navegue para File > New Project > Version Control > Git.
- Cole a URL do GitHub em “Repository URL”.
- Mantenha o campo “Project directory name” inalterado e escolha um local (pasta) onde o projeto será criado.
- Marque a opção “Open in new session”.
- Clique em “Create Project”.



# Clonar repositório pelo RStudio

2. Via Console usando o pacote usethis:

- Utilize a função `create_from_github` do pacote `usethis`.
- Informe o nome de usuário (que criou o repositório) e o nome do repositório.
- Exemplo:

```
usethis::create_from_github("nome_do_usuario/nome_do_repositorio").
```

Após clonar o repositório, você estará trabalhando em um novo projeto do RStudio que contém o repositório do GitHub. No painel “Files” do RStudio, procure pelo arquivo “README.md” e abra-o para visualizá-lo diretamente no RStudio.



# Fazer alterações locais, salvar, commitar

Para adicionar uma linha com uma frase informativa ao arquivo README.md e salvar essas alterações, siga os passos abaixo para realizar o commit no seu repositório local:

1. Clique na aba “Git” do RStudio.
2. Em seguida, clique no botão “Commit”.
3. Na coluna “Staged”, marque a caixinha ao lado de README.md para incluir o arquivo no commit.
4. No campo “Commit message”, digite uma mensagem informativa que descreva as alterações no código.
5. Clique no botão “Commit”, aguarde a mensagem de sucesso e depois clique em “Close”.

Caso existam outros arquivos no repositório criados automaticamente pelo RStudio, não é necessário fazer alterações neles. Apenas o arquivo README.md, contendo a nova linha com sua frase criativa, será incluído no commit.





# Enviar alterações (**push**) para o GitHub

Para enviar suas alterações realizadas no repositório local para o repositório remoto (GitHub), siga o passo abaixo:

1. Clique no botão verde “Push”.

Caso todas as etapas e configurações tenham sido executadas corretamente, você não será solicitado a digitar o token e suas alterações locais serão enviadas para a página do repositório no GitHub (recomendo atualizar a página para verificar).



# Exercícios

1. Utilizando os dados em `cheeses.csv`, crie um relatório descrevendo os dados. Algumas informações devem estar presentes no seu relatório:
  - a. Quantidade de observações e variáveis.
  - b. Inclua gráficos e tabelas para descrever os dados.
2. Crie uma apresentação utilizando os dados em `winequality-red.csv` e `winequality-white.csv`. A sua apresentação deve conter comparações entre os dois tipos de vinho.



# Referências

1. Wickham, H. & Grolemund, G. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. (O'Reilly Media, 2016).

