



R-Ladies São Paulo

Curso Básico de R

08/02/2020

# Pré-requisitos



- R e RStudio instalados no seu notebook



OU

- RStudio Cloud

- Através do link: <https://rstudio.cloud/project/876850>



# Usando a RStudio Cloud



- Exemplo de como acessar o projeto (GIF):

The screenshot shows a presentation slide with a purple header bar containing the title 'Pré-requisitos'. Below the header, there is a bulleted list of requirements:

- R e RStudio instalados no seu notebook

Below this requirement, there are two icons: a grey 'R' inside a circle and a blue 'R' inside a sphere. In the center, the word 'OU' (Or) is written in bold capital letters.

- RStudio Cloud
  - Através do link: <https://rstudio.cloud/project/876850>

In the bottom right corner of the slide, there is a watermark-like text '@BeaMilz'.

- Importante: Quando aparecer na sua tela, clique em "**Save a permanent copy**".

# Programação



## Manhã

- O que é um algoritmo?
- O que é o R?
- Introdução ao RStudio
- Boas Práticas de Programação
- Fundamentos de R

# Programação



## Tarde

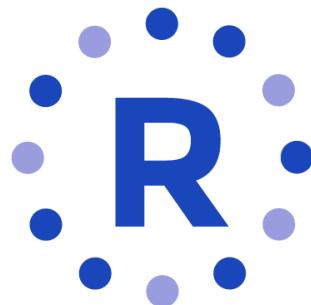
- Introdução ao Pacote Tidyverse
- O Operador Pipe
- Visualização de Data Frames no RStudio
- Introdução ao Pacote dplyr
- Para aprender mais

# Apoio



# Insper

**Insper**



**RConsortium**



# O que é o R-Ladies?

R-Ladies é uma organização mundial que promove a diversidade de gênero na comunidade da linguagem R. R-Ladies São Paulo integra, orgulhosamente, a organização R-Ladies Global, em São Paulo.

## Como?

Através de meetups e mentorias em um ambiente seguro e amigável.

Nosso principal objetivo é promover a linguagem computacional estatística R compartilhando conhecimento, assim, quem tiver interesse na linguagem será bem-vinda, independente do nível de conhecimento.

Fonte: [About us - R-Ladies, Meetup R-Ladies São Paulo](#)



## Para quem?

Nosso principal público-alvo são as pessoas que se identificam com o gênero feminino, portanto, mulheres cis, mulheres trans, bem como pessoas não-binárias e queer.

## Missão

Como uma iniciativa de diversidade, a missão das R-Ladies é alcançar uma representação proporcional de pessoas de gêneros atualmente sub-representados na comunidade R, incentivando, inspirando e capacitando-as.

Fonte: [About us - R-Ladies, Meetup R-Ladies São Paulo](#)

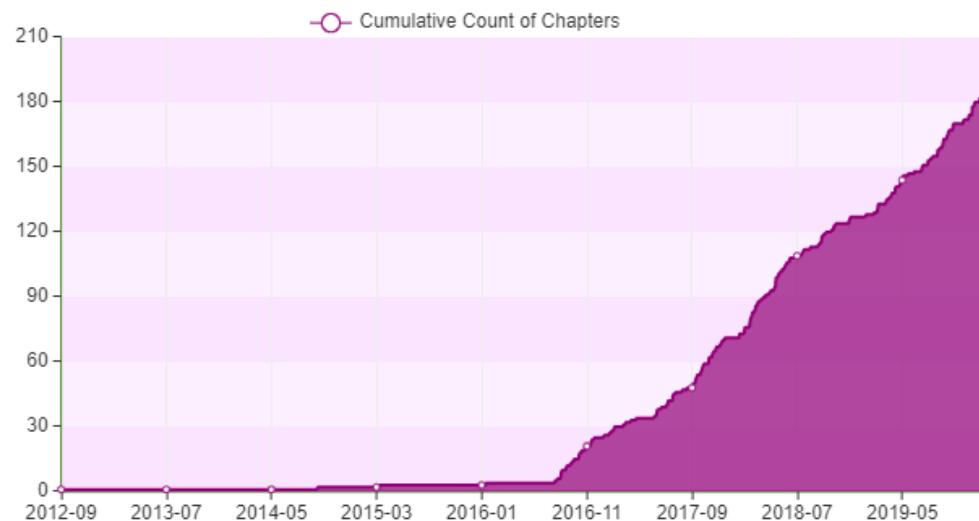
# Como o R-Ladies começou?



**Gabriela de Queiroz** fundou o R-Ladies no dia **1 de outubro de 2012**. Ela queria retribuir à comunidade depois de ir a vários encontros e aprender muito de graça. O primeiro encontro R-Ladies foi realizado em **San Francisco, Califórnia (Estados Unidos)**. Nos anos seguintes, mais capítulos do R-Ladies começaram em todo o mundo.

Fonte: [About us - R-Ladies](#)

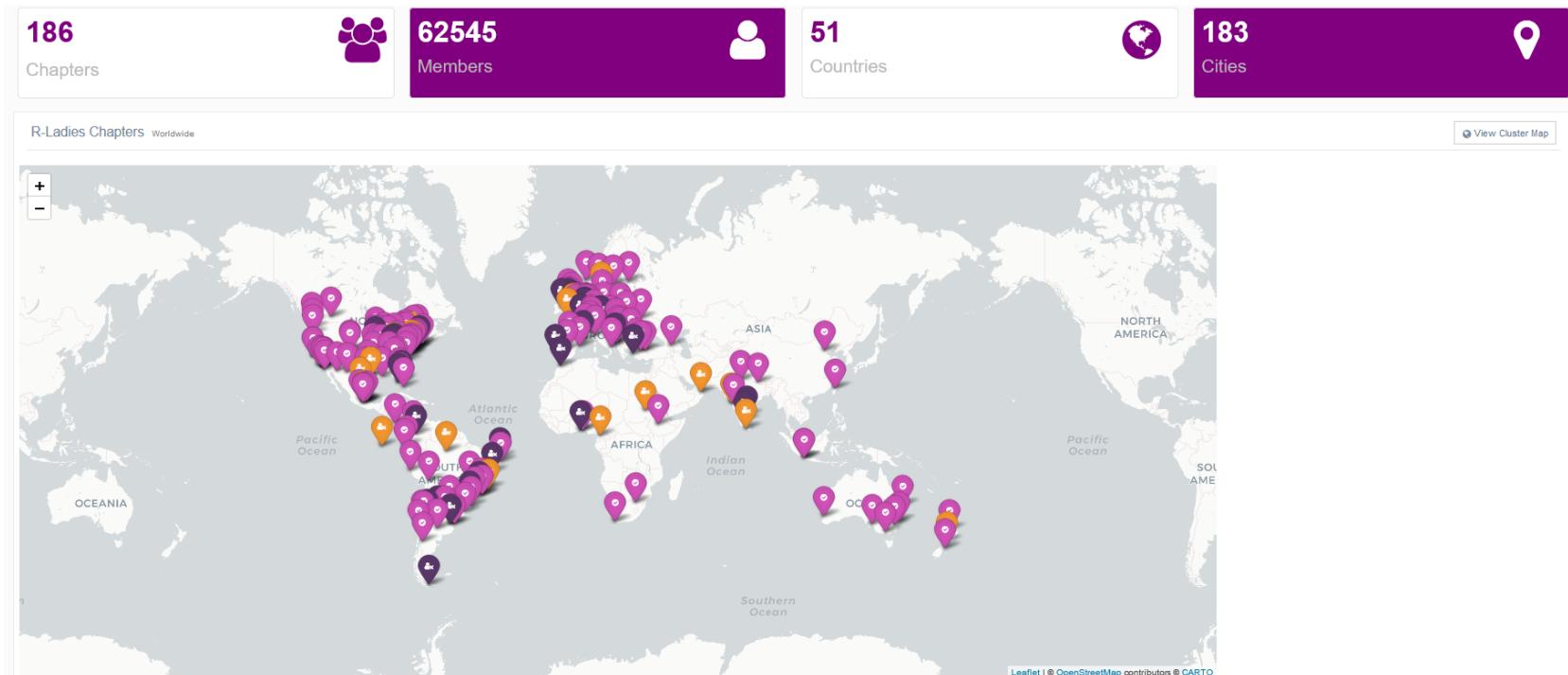
# Crescimento da R-Ladies no mundo



Atualizado em: Janeiro/2020. Fonte: [R Community Explorer](#)



# Comunidade da R-Ladies - Capítulos no mundo



Atualizado em: Janeiro/2020. Fonte: [R Community Explorer](#)



# Código de conduta

O R-Ladies dedica-se a proporcionar uma experiência livre de assédio para todas as pessoas participantes, desta forma, não é tolerada nenhuma forma de assédio. [Código de conduta - R-Ladies](#)

# R-Ladies no Brasil



Atualizado em: janeiro de 2020. Fonte: [R Community Explorer](#)

Show  entries

Search:

	<b>name</b>	<b>members</b>	<b>past_events</b>	<b>upcoming_events</b>
1	R-Ladies Belo Horizonte	758	10	0
2	R-Ladies São Paulo	608	15	0
3	R-Ladies Rio de Janeiro	465	5	2
4	R-Ladies Niterói	462	7	0
5	R-Ladies Floripa	357	7	0
6	R-Ladies Porto Alegre	313	6	0
7	R-Ladies Curitiba	203	1	0

Showing 1 to 7 of 14 entries

Previous

1

2

Next

# Cronograma da manhã



- O que é um algoritmo?
- O que é o R?
- Introdução ao RStudio
- Boas Práticas Iniciais de Programação
- Fundamentos de R
- Para aprender mais

# O que é um algoritmo?



Um algoritmo é uma sequência finita de instruções.

## Exemplo

- Receita de bolo

### Ingredientes:

- 1 xícara de chá de água;
- 1/3 de xícara de chá de óleo vegetal;
- 1 1/2 xícaras de chá de milho cru;
- 1/2 xícara de chá de coco ralado sem açúcar;
- 3/4 de xícara de chá de açúcar demerara;
- 1 xícara de chá de fubá mimoso;
- 1 colher de sopa de vinagre branco;
- 1 colher de sopa de fermento químico em pó;
- 1 pitada de sal.



### Modo de preparo:

Preaqueça o forno a 210°C e unte uma assadeira média com óleo e fubá. Em um liquidificador coloque a água, o óleo, o milho, o coco ralado, o açúcar, o fubá, o vinagre, uma pitada de sal e bata até obter uma massa uniforme. Acrescente o fermento químico em pó e misture suavemente. Despeje a massa na forma e leve para

# O que é um algoritmo?



Um algoritmo é uma sequência finita de instruções.

## Mais exemplos

- Construir uma estante de livros



# O que é um algoritmo?



Um algoritmo é uma sequência finita de instruções.

## Mais exemplos

- Fazer um drone caseiro com arduino



# O que é programar?



Programar um computador é escrever instruções em qualquer **linguagem** que o computador entenda.

Essa sequência de instruções pode ser executada por um humano ou um computador.

Então, **programação é a arte de fazer com que o computador execute uma sequência de instruções definidas.**

```
print('Olá!')
```

```
## [1] "Olá!"
```

**Vamos exercitar um pouco?**

# O que é o R?



"R é um ambiente de software livre para computação estatística e gráficos".  
(<https://www.r-project.org/>)

**R** é um ambiente computacional e uma linguagem de programação que vem progressivamente se especializando em manipulação, análise e visualização gráfica de dados. Na atualidade é considerado o melhor ambiente computacional para essa finalidade. O ambiente está disponível para diferentes sistemas operacionais: Unix/Linux, Mac e Windows.

- Baseada na linguagem estatística S
- 1ª versão de 1995 por Ross Ihaka e Robert Gentleman da Universidade de Auckland



# O que é o R?



- Berço na Estatística
- Muito usado por cientistas de dados, estatísticos e pesquisadores.
- Mantida pela **R Development Core Team**
- Pode ser usada para diversos fins.

# Por que usar o R?



- É uma linguagem de programação para análise de dados
- É open source
- É uma linguagem interpretada
- Possui uma comunidade ativa de desenvolvedores
- É flexível, permitindo desenvolver funções e pacotes para facilitar o trabalho
- Está disponível em diferentes plataformas: Windows, Linux e Mac
- É reproduzível!
- É compartilhável!

# O que é possível fazer com R?

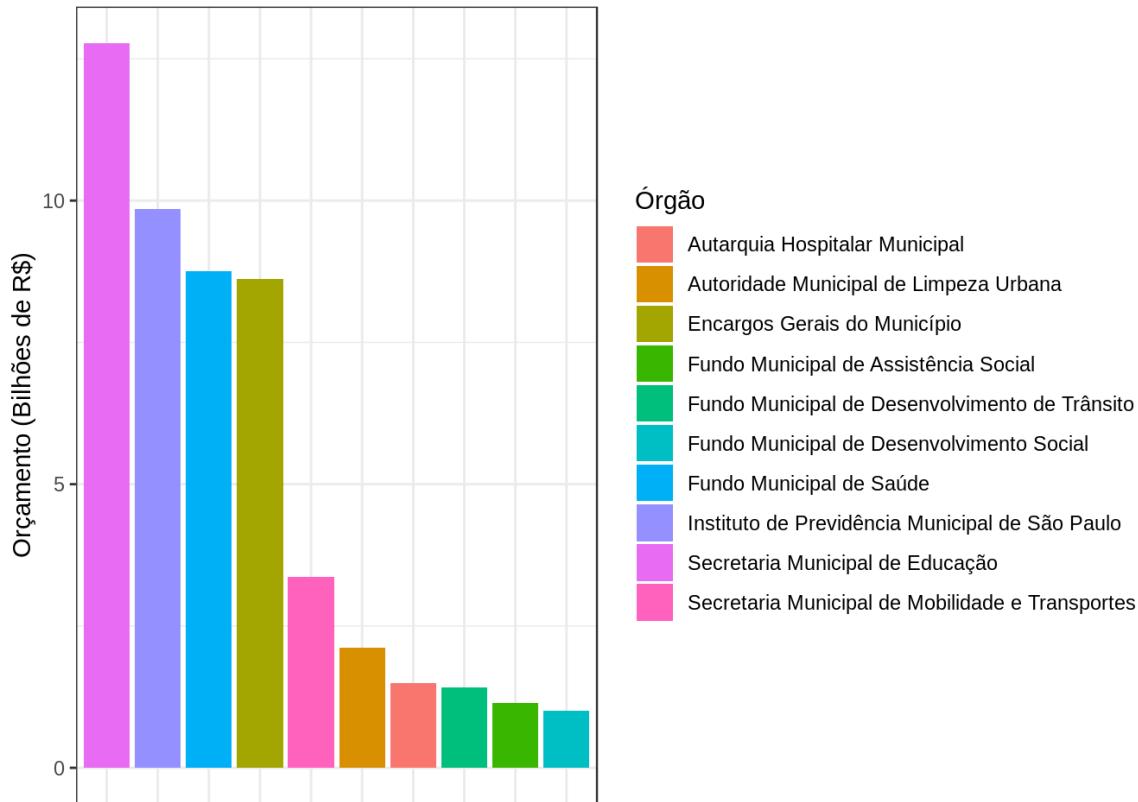


- Análise de dados - Estatística, modelagem, etc.
- Visualização de dados
- Apresentações
- Relatórios dinâmicos
- Escrever livros
- Mineração de dados
- Muito mais ...

# Exemplo



## Gráfico elaborado com R - Proposta Orçamentária PMSP 2019 - 10 maiores orçamentos



Fonte: Explorando o orçamento da Prefeitura Municipal de São Paulo

# Exemplo



## Gráfico elaborado com R - Execução Orçamentária PMSP na função Gestão Ambiental

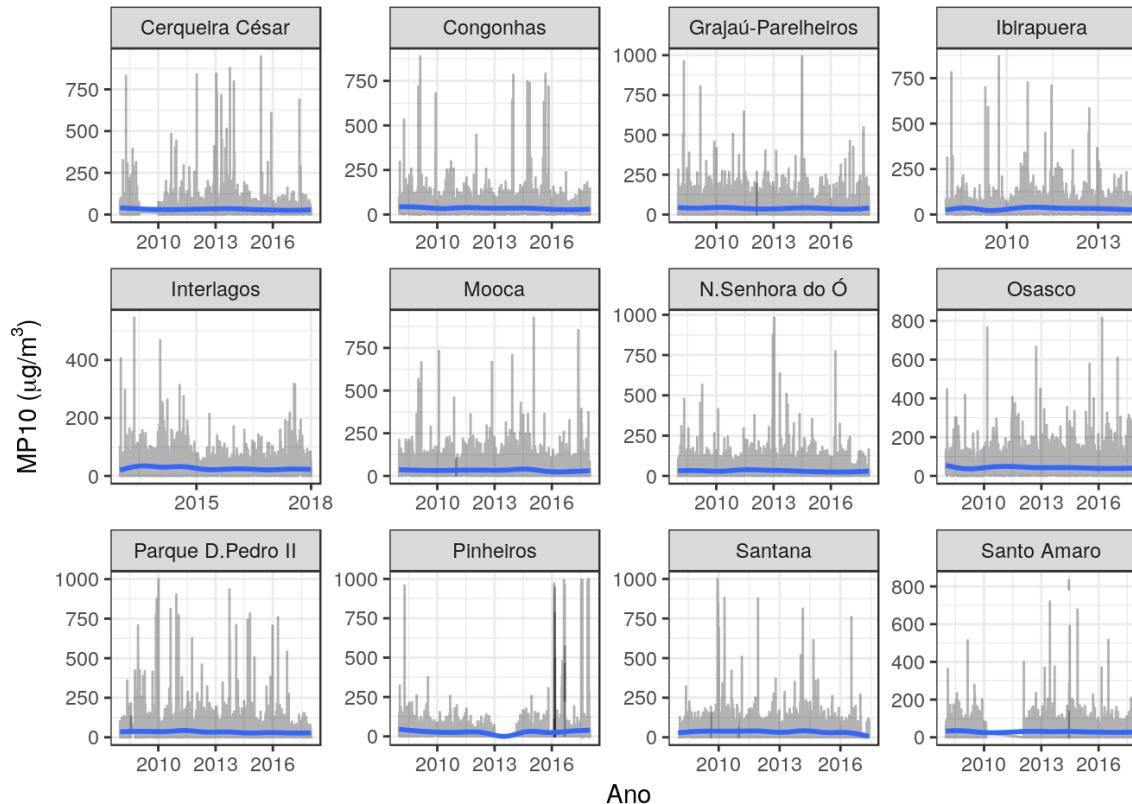


Fonte: Explorando o orçamento da Prefeitura Municipal de São Paulo

# Exemplo



## Gráfico elaborado com R - Material Particulado 10 - Dados CETESB - RPollution

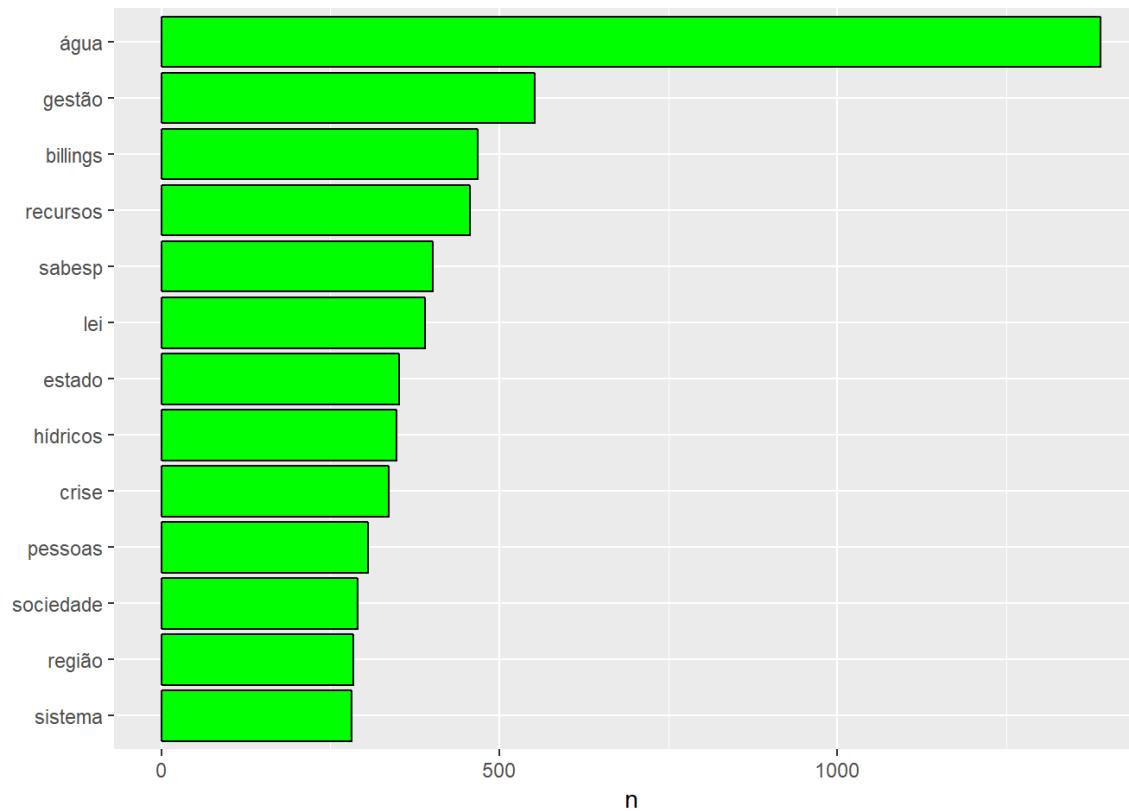


Fonte: Rpolution

# Exemplo



## Frequência de Palavras - TESE Doutorado PROCAM/USP Ana Lucia Spinola

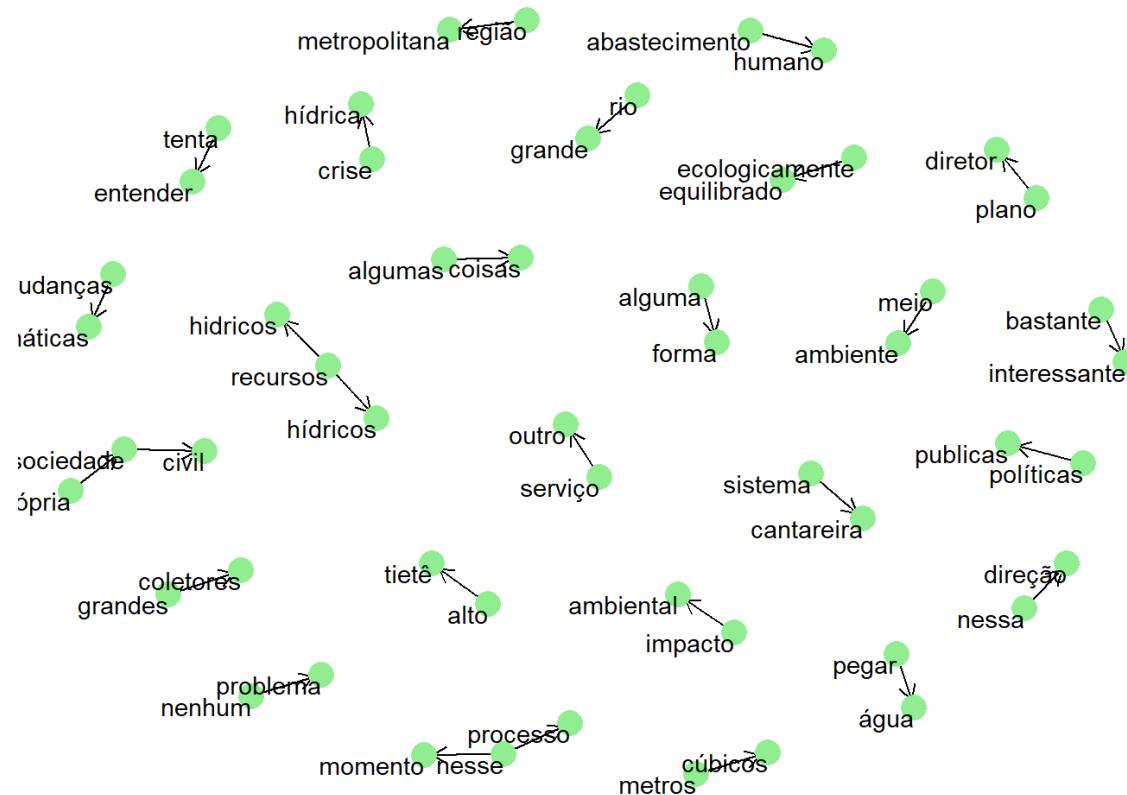


Fonte: Ana Lu Spinola

# Exemplo



## BIGRAM - TESE Doutorado PROCAM/USP Ana Lucia Spinola



Fonte: Ana Lu Spinola

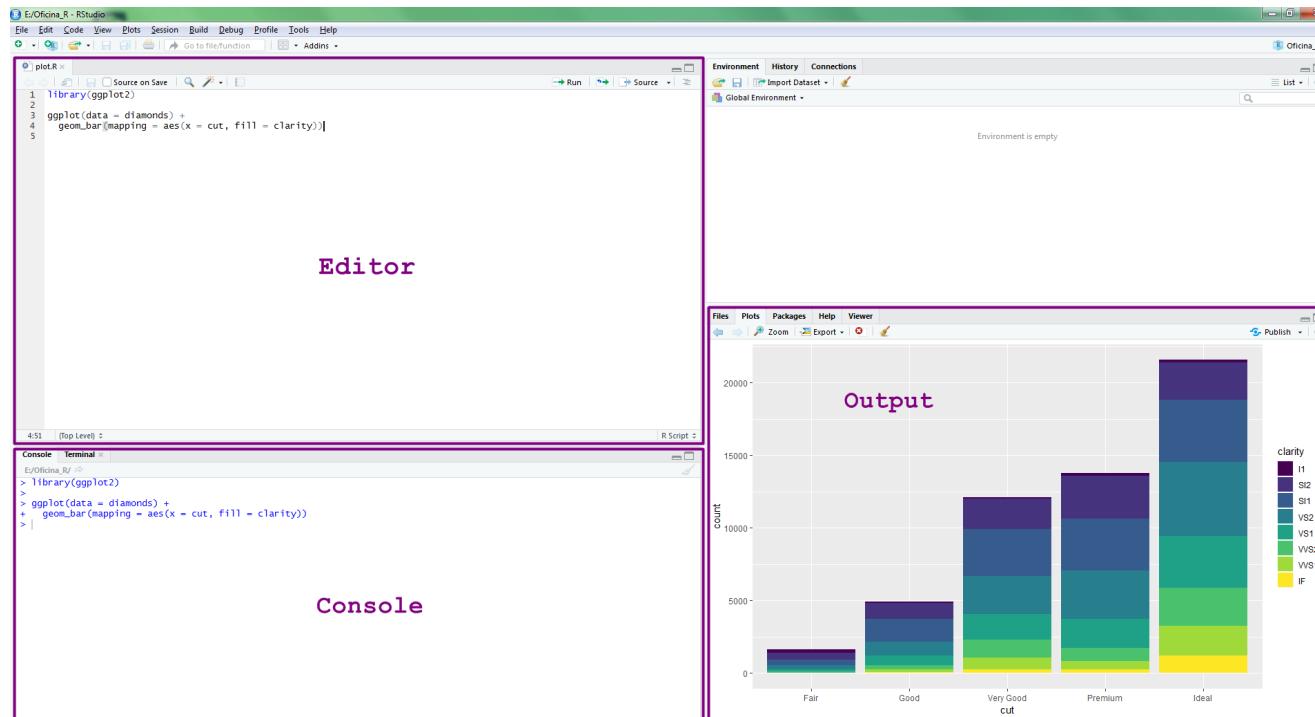


# Introdução ao RStudio

# RStudio



**RStudio** é o IDE (integrated development environment) da Linguagem R, ou seja, o ambiente que utilizamos para editar e executar os códigos em R. Tem quatro áreas, conforme a figura abaixo:



Fonte: Curso Introdução ao R - Fatec

# RStudio

A screenshot of the RStudio IDE. The top menu bar shows 'RStudio'. The top-left pane is the 'Console' showing R startup messages and a single command '> |'. The top-right pane is the 'Environment' tab of the 'Global Environment' browser, which is currently empty. The bottom-right pane is the 'Files' tab of the file browser, showing a directory structure under 'Home > Documents > SoftwareCarpentry > r\_gapminder'. The directory contains two items: an upward arrow icon labeled '..' and a folder icon labeled 'r-novice-gapminder'.

R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"  
Copyright (C) 2016 The R Foundation for Statistical Computing  
Platform: x86\_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.

> |

Fonte: SW Carpentry

# RStudio



The screenshot shows the RStudio interface. The top menu bar includes 'File', 'Edit', 'Source', 'Run', 'Help', and 'Addins'. The 'Project' dropdown shows '(None)'. The main window has several panes:

- Script Editor:** Untitled1, showing a single line of code '1 |'.
- Environment:** Global Environment, showing 'Environment is empty'.
- File Browser:** Shows a directory tree under 'r-novice-gapminder':

Name	Size	Modified
..		
r-novice-gapminder		
- Console:** Displays the R startup message, including the version (3.3.2), copyright notice, platform (x86\_64-apple-darwin13.4.0 (64-bit)), and licensing information. It also shows natural language support and collaborative project details, and ends with a prompt '> |'.

Fonte: SW Carpentry

# RStudio

A screenshot of the RStudio IDE. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main window has several panes: a top-left pane for scripts (highlighted with a red border), a top-right pane for the Environment (showing 'Environment is empty'), a bottom-left pane for the Console, and a bottom-right pane for the Files browser (showing a folder structure under 'Arquivos\_R').

Aqui escrevemos os códigos → script  
Arquivos do tipo .R

# RStudio

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main window has several panes: a left pane for files (with 'teste.R' open), a top-right pane for the Environment (labeled 'Environment is empty'), and a bottom-right pane for the File Browser (showing a directory structure). A red box highlights the 'Console' and 'B Script' panes at the bottom-left, which are used for writing and running R code. The text 'Escrever e executar comandos' is overlaid on the 'Console' area, and 'Ver resultados' is overlaid on the 'B Script' area.

Escrever e executar comandos

Ver resultados

# RStudio



A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with various icons. A file named 'teste.R' is open in the top-left panel. The bottom-left panel is a 'Console' window. The right side of the interface features a large red-bordered box highlighting the 'Environment' tab of the 'Global Environment' pane. The pane displays the message 'Environment is empty'. Inside this box, the text 'Enviroment:' and 'dados e variáveis da seção atual' is overlaid. The bottom-right panel is a 'File Browser' showing a directory structure under 'Arquivos.R'.

# RStudio



A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main window shows an R script file named "teste.R" with the following code:

```
install.packages("manipulate")
install.packages("knitr")
install.packages("markdown")
library(datasets)
data(iris)
?iris
iris
rowMeans(iris[, 1:4])
iris
rowMeans(iris[, 1:4])
apply(iris[, 1:4], 1, mean)
colMeans(iris)
apply(iris[, 1:4], 2, mean)
iris
mean(iris[iris["Species"]=="virginica"]['Sepal.Length'])
iris[iris["Species"]=="virginica"]['Sepal.Length']
iris[iris["Species"]=="virginica"]['Sepal.Length']
iris['Sepal.Length']
iris['Species']=='virginica'
iris[iris["Species"]=="virginica","Sepal.Length"]
mean(iris[iris["Species"]=="virginica", "Sepal.Length"])
library(datasets)
data(mtcars)
mtcars
apply(mtcars, 2, mean)
tapply(mtcars$cyl, mtcars$mpg, mean)
```

The "Environment" tab in the top right is selected. A red box highlights the "History" tab, which contains the command "Histórico de comandos". The bottom right panel shows a file browser with a list of files in the "Arquivos\_R" folder.

# RStudio



A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main workspace shows an R script file named 'teste.R' and an 'Environment' pane which displays 'Global Environment' and the message 'Environment is empty'. Below the workspace is a 'Console' pane. A red rectangular box highlights the 'Files' browser located at the bottom right of the interface. The 'Files' browser shows a file tree structure with 'Home &gt; Documentos &gt; Arquivos.R' selected. The browser has tabs for Files, Plots, Packages, Help, and Viewer, and includes buttons for New Folder, Delete, Rename, and More.

Files:

navegar em pastas e arquivos

# RStudio



The screenshot shows the RStudio interface. The Source editor contains R code for data manipulation and analysis, including loading packages like 'manipulate', 'knitr', and 'markdown', and datasets like 'iris' and 'mtcars'. The Environment pane shows the loaded objects. A red box highlights the Plots tab in the Viewer pane, which typically displays plots generated from the current session.

```
teste.R# <-->
install.packages("manipulate")
install.packages("knitr")
install.packages("markdown")
library(datasets)
data(iris)
?iris
iris
rowMeans(iris[, 1:4])
iris
rowMeans(iris[, 1:4])
apply(iris[, 1:4], 1, mean)
colMeans(iris)
apply(iris[, 1:4], 2, mean)
iris
mean(iris[iris["Species"]=="virginica"]["Sepal.Length"])
iris[iris["Species"]=="virginica"]["Sepal.Length"]
iris[iris["Species"]=="virginica"]["Sepal.Length"]
iris["Sepal.Length"]
iris["Species"]="virginica"
iris[iris["Species"]=="virginica","Sepal.Length"]
mean(iris[iris["Species"]=="virginica", "Sepal.Length"])
library(datasets)
data(mtcars)
mtcars
apply(mtcars, 2, mean)
tapply(mtcars$cyl, mtcars$mpg, mean)
```

Plots:

gráficos plotados no console

# RStudio

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with various icons. The main area has tabs for Environment, History, and Source. The Environment tab shows R code and its output. The History tab shows a history of operations. The Source tab shows a script named 'teste.R'. At the bottom is a Console tab. A red box highlights the Packages tab and the list of installed packages. Two arrows point from the text labels 'Pacotes habilitados' and 'Pacotes instalados no sistema' to the 'dplyr' and 'ggplot2' entries in the Packages list respectively.

Pacotes habilitados

Pacotes instalados no sistema

```
install.packages("manipulate")
install.packages("knitr")
install.packages("markdown")
library(datasets)
data(iris)
?iris
iris
rowMeans(iris[, 1:4])
iris
rowMeans(iris[, 1:4])
apply(iris[, 1:4], 1, mean)
colMeans(iris)
apply(iris[, 1:4], 2, mean)
iris
mean(iris[iris["Species"]=="virginica"]['Sepal.Length'])
iris[iris["Species"]=="virginica"]['Sepal.Length']
iris[iris['Species']=='virginica']['Sepal.Length']
iris['Sepal.Length']
iris['Species']!='virginica'
iris[iris["Species"]=="virginica","Sepal.Length"]
mean(iris[iris["Species"]=="virginica", "Sepal.Length"])
library(datasets)
data(mtcars)
mtcars
apply(mtcars, 2, mean)
tapply(mtcars$cyl, mtcars$mpg, mean)
```

Name	Description	Version
DBI	R Database Interface	0.5-1
dichromat	Color Schemes for Dichromats	2.0-0
digest	Create Compact Hash Digests of R Objects	0.6.11
<input checked="" type="checkbox"/> dplyr	A Grammar of Data Manipulation	0.5.0
evaluate	Parsing and Evaluation Tools that Provide More Details Than the Default	0.10
formatR	Format R Code Automatically	1.4
<input checked="" type="checkbox"/> ggplot2	Create Elegant Data Visualisations Using the Grammar of Graphics	2.2.1
grid	Arrange 'Grobs' in Tables	0.2.0
highr	Syntax Highlighting for R Source Code	0.6
htmltools	Tools for HTML	0.3.5
httr	Tools for Working with URLs and HTTP	1.2.1
jsonlite	A Robust, High Performance JSON Parser and Generator for R	1.2
knitr	A General-Purpose Package for Dynamic	1.15.1

# RStudio



A screenshot of the RStudio interface. The left pane shows a code editor with a section titled "Help:" containing a numbered list of function components. The right pane shows the "Environment" tab with a message "Environment is empty". A red box highlights the "Help" menu bar at the top of the interface.

- Help:
- 1 – Description -> resumo geral
  - 2 – Usage -> mostra como a função deve ser utilizada e quais argumentos podem ser especificados
  - 3 – Arguments -> explica cada um dos argumentos
  - 4 – Details -> explica alguns detalhes (poucos)
  - 5 – Value -> mostra o output da função (resultados)
  - 6 – Note -> notas sobre a função
  - 7 - Authors -> autores(as) da função
  - 8 – References -> referências para os métodos usados
  - 9 – See also -> funções relacionadas
  - 10 – Examples -> exemplos do uso da função.

A screenshot of the RStudio Help menu expanded, showing various sections like R Resources, Manuals, Reference, Packages, and Miscellaneous Material. The "R Resources" section is highlighted with a red border.

# Help!



- Pedir ajuda: **help(nome\_da\_funcao)** ou **?nome\_da\_funcao**.

```
help(sum)
```

```
?sum
```

- Se a dúvida permanecer, procure no **Stack OverFlow** ou Google.
- E se ainda tiver dúvidas, pergunte para a comunidade (há grupos no Telegram e outras redes sociais).



# Boas práticas iniciais para organizar seu projeto

**Tratar dados como somente leitura:** esse é provavelmente o objetivo mais importante da configuração de um projeto. Os dados geralmente consomem tempo e/ou são caros para coletar. Trabalhar com eles interativamente (por exemplo, no Excel), onde eles podem ser modificados, significa que você nunca tem certeza de onde os dados vieram, ou como eles foram modificados desde a coleta. Portanto, é uma boa ideia tratar seus dados como “somente leitura”.

**Qualquer coisa gerada pelos seus scripts deve ser tratada como descartável:** todos devem poder ser criados novamente a partir dos seus scripts. Existem várias maneiras diferentes de gerenciar essa saída. Acho útil ter uma pasta de saída com subdiretórios diferentes para cada análise separada. Isso fica mais fácil depois, já que muitas das análises são exploratórias e não acabam sendo usadas no projeto final, e algumas análises são compartilhadas entre os projetos.

Fonte: [SW Carpentry](#)

**Os nomes das suas variáveis devem fazer sentido:** Ao nomear suas variáveis, dê nomes que tenham significado para seres humanos. Pense que o código que você escreve hoje deve ser claro para você daqui 1 ano e também deve ser claro para algum(a) colega seu(ua).

**Comente bem o seu código:** É possível fazer comentários usando o símbolo '#'. É sempre bom explicar o que uma variável armazena, o que uma função faz, porque alguns parâmetros são passados para uma determinada função, qual é o objetivo de um trecho de código, etc.

**Evite linhas de código muito longas:** Usar linhas de código mais curtas ajuda na leitura do código.

Fonte: [SW Carpentry](#)

**Escreva um código organizado:** Por exemplo, adote um padrão no uso de minúsculas e maiúsculas, uma lógica única na organização de pastas e arquivos, pode ser adotada uma breve descrição (como comentário) indicando o que um determinado script faz.

**Carregue todos os pacotes que irá usar sempre no início do arquivo:** Quando alguém abrir o seu código será fácil identificar quais são os pacotes que devem ser instalados e quais dependências podem existir.

**Evite referência de caminho que considere seu computador ou usuário:** Faça referência ao caminho do projeto.

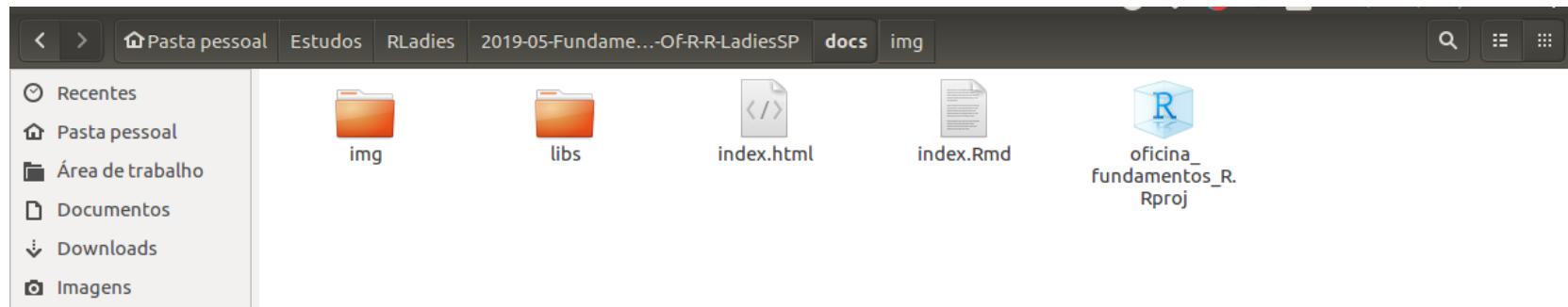
Fonte: SW Carpentry

# Antes de começar: o projeto



Ao realizar um projeto, sempre organizar os arquivos em uma **pasta** que conterá todos os arquivos de seu projeto.

Nomear novos arquivos com **nomes descritivos**



# Projetos



"Um bom layout de projeto facilitará sua vida: ajudará a garantir a integridade de seus dados; facilita o compartilhamento de seu código com outra pessoa (colega de laboratório, colaborador ou /orientador); ele permite que você facilmente faça o upload do seu código com a submissão do seu manuscrito; torna-se mais fácil recuperar o projeto depois de um intervalo."

Fonte: [SW Carpentry](#)

# Criando um projeto



1. Clique na opção “**File**” do menu, e então em “**New Project**”.
2. Clique em “**New Directory**”.
3. Clique em “**New Project**”.
4. Escreva o nome do diretório (pasta) onde deseja manter seu projeto, ex “my\_project”.
5. Clique no botão “**Create Project**”.

OBS: Crie um novo script para escrever seus códigos! **File -> New File -> RScript**

# Atalhos importantes



Os atalhos facilitam. Veja os principais:

- CTRL + ENTER: roda a linha selecionada no script.
- ALT + -: (<-) sinal de atribuição.

# E lá vamos nós!





# R Básico

# Instruções



- Abrir o RStudio da sua máquina
- Abrir um novo R script (file -> new file -> R script)

# R como calculadora



- O R permite realizar muitas operações aritméticas em seu console!

```
2 + 5      # adição
```

```
## [1] 7
```

```
9 - 4      # subtração
```

```
## [1] 5
```

```
5 * 2      # multiplicação
```

```
## [1] 10
```

```
7 / 5      # divisão
```

```
## [1] 1.4
```

# R como calculadora



```
9 %% 4    # resto da divisão de 9 por 4
```

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

```
7 %/% 4    # parte inteira da divisão de 7 por 4
```

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

```
8 ^ 2      # potenciação
```

```
## [1] 64
```

```
sqrt(1024) # radiciação
```

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

A ordem matemática das operações também vale no R.

# Funções matemáticas



```
sin(1) # funções trigonométricas
```

```
## [1] 0.841471
```

```
log(1) # logaritmo natural (base e)
```

```
## [1] 0
```

```
log10(10) # logaritmo na base 10
```

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

```
exp(0.5) # e^(1/2)
```

```
## [1] 1.648721
```

Fonte: SW Carpentry

# Desafio 1



Haydée quer contabilizar quantas pessoas participaram dos meetups das RLadies São Paulo em 2019. Ela contabilizou o número de pessoas que participaram em cada evento:

1. Fevereiro, Outubro e Novembro - 60 pessoas
2. Abril e Agosto - 30 pessoas
3. Março, Maio e Julho - 20 pessoas
4. Junho, Setembro e Dezembro - 45 pessoas

Quantas pessoas participaram dos meetups das RLadies em 2019?

# Desafio 1 - Resposta



Haydée quer contabilizar quantas pessoas participaram dos meetups das RLadies São Paulo em 2019. Ela contabilizou o número de pessoas que participaram em cada evento:

1. Fevereiro, Outubro e Novembro - 60 pessoas
2. Abril e Agosto - 30 pessoas
3. Março, Maio e Julho - 20 pessoas
4. Junho, Setembro e Dezembro - 45 pessoas

Quantas pessoas participaram dos meetups das RLadies em 2019?

```
3*60 + 2*30 + 3*20 + 3*45
```

```
## [1] 435
```

# O que é uma variável?



- Ao se desenvolver um projeto, você irá trabalhar com diversos tipos de arquivos, além de informações que serão repetidas ao longo do script.
- Para reutilizar essas informações ao longo do script utilizamos o que chamamos de **variável**
- Uma variável é um espaço de memória que retém e representa um valor ou expressão



# Atribuindo valor a uma variável no R



- Para atribuir um valor a uma variável no R, utilizamos o operador `<-`
- O atalho ALT + - gera o operador `<-`
- Todas as declarações R onde são criadas variáveis atribuindo valores a elas, tem a mesma forma:

**nome\_da\_variável <- valor**

# Exemplos



## Variáveis e atribuição de valores

```
nome_empregado ← "Tom Cruise de Souza e Silva"  
nome_empregado
```

```
## [1] "Tom Cruise de Souza e Silva"
```

```
horas_trabalhadas ← 160  
horas_trabalhadas
```

```
## [1] 160
```

# Exemplos



## Variáveis e atribuição de valores

```
salario ← 3984.23  
salario
```

```
## [1] 3984.23
```

```
ativo ← TRUE  
ativo
```

```
## [1] TRUE
```

# Nomes de variáveis



- Os nomes devem começar com uma letra. Podem conter letras, números, \_ e .
- Recomendação do autor do livro R For Data Science: **usar\_snake\_case**, ou seja, palavras escritas em minúsculo separadas pelo underscore (\_).
- O R é *case sensitive*, isto é, faz a diferenciação entre as letras minúsculas e maiúsculas. Portanto, uma variável chamado *teste* é diferente de uma outra variável chamada *Teste*.

# Desafio 2



1) Crie variáveis para os casos abaixo:

- Restaurante com valor Rodízio Japonês
- Conta a pagar com valor 40,50
- Dinheiro na carteira com valor 60

2) Usando as variáveis criadas anteriormente, qual o valor do troco que irei receber do pagamento de uma conta de 40,50 reais com meus R\$ 60,00 reais?

# Desafio 2 - Resposta



1) Crie variáveis para os casos abaixo:

```
restaurante <- "Rodizio Japones"  
restaurante
```

```
## [1] "Rodizio Japones"
```

```
conta <- 40.50  
conta
```

```
## [1] 40.5
```

```
dinheiro <- 60  
dinheiro
```

```
## [1] 60
```

# Desafio 2 - Resposta



2) Usando as variáveis criadas anteriormente, qual o valor do troco que irei receber do pagamento de uma conta de 40,50 reais com meus R\$ 60,00 reais?

```
dinheiro - conta
```

```
## [1] 19.5
```

# Classes Básicas ou Atômicas do R



São os tipos básicos de dados que podem ser representados na linguagem R. É neles que guardamos as informações que necessitamos para um algoritmo.

- **Integer:** números inteiros
- **Numeric:** números racionais
- **Complex:** números complexos (raramente usados para Análise de Dados)
- **Logical:** TRUE, FALSE ou NA
- **Factor:** variáveis categóricas
- **Character:** texto

# Exemplos: integer



```
10L # Um número inteiro pode ser representado acompanhado de um L
```

```
## [1] 10
```

```
2019L
```

```
## [1] 2019
```

# Exemplos: numeric



```
10
```

```
## [1] 10
```

```
2019
```

```
## [1] 2019
```

```
5.44
```

```
## [1] 5.44
```

# Exemplos: complex



```
4 + 9i
```

```
## [1] 4+9i
```

# Exemplos: logical



```
TRUE
```

```
## [1] TRUE
```

```
FALSE
```

```
## [1] FALSE
```

# Exemplos: factor



```
escolaridade ← c("Médio", "Superior", "Fundamental", "Fundamental", "Médio")
fator ← as.factor(escolaridade)
fator
```

```
## [1] Médio      Superior    Fundamental Fundamental Médio
## Levels: Fundamental Médio Superior
```

A função `as.factor()` criou uma variável do tipo `factor`.

Na linha `Levels` aparecem os rótulos do fator.

Essa classe de dados pode ser trabalhada com o pacote **forcats**.

# Exemplos: character



```
"escola"
```

```
## [1] "escola"
```

```
"2019"
```

```
## [1] "2019"
```

```
"I love pinschers."
```

```
## [1] "I love pinschers."
```

# Operações simples com strings



```
animal ← "Camaleao"  
#letras maiúsculas  
toupper(animal)
```

```
## [1] "CAMALEAO"
```

```
#letras minúsculas  
tolower(animal)
```

```
## [1] "camaleao"
```

```
#número de caracteres  
nchar(animal)
```

```
## [1] 8
```

# Função class



A função **class** mostra a classe de uma variável.

```
nome_filme ← "Bohemian Rhapsody"  
class(nome_filme)
```

```
## [1] "character"
```

```
ano_inteiro ← 2018L  
class(ano_inteiro)
```

```
## [1] "integer"
```

```
ano ← 2018  
class(ano)
```

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

# Função class



```
motor ← 1.5  
class(motor)
```

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

```
passou_enem ← TRUE  
class(passou_enem)
```

```
## [1] "logical"
```

# Conversão de classes



Podemos forçar uma variável a ser de uma classe específica com as funções:

- `as.character()`
- `as.numeric()`
- `as.integer()`
- `as.logical()`

# Conversão de classes



## Exemplos de conversão de classes

```
vetor <- 0:9
```

```
vetor
```

```
## [1] 0 1 2 3 4 5 6 7 8 9
```

```
class(vetor)
```

```
## [1] "integer"
```

```
vetor_numeric <- as.numeric(vetor)  
vetor_numeric
```

```
## [1] 0 1 2 3 4 5 6 7 8 9
```

```
class(vetor_numeric)
```

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

# Conversão de classes



## Exemplos de conversão de classes

```
vetor_logical ← as.logical(vetor)
vetor_logical

## [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

class(vetor_logical)

## [1] "logical"
```

# Desafio 3



Converta a variável Conta criada anteriormente para character. Agora cheque sua classe. O que acontece com seu valor? Como explica o que aconteceu?

# Desafio 3 - Resposta



Converta a variável Conta criada anteriormente para character. Agora cheque sua classe. O que acontece com seu valor?

```
as.character(conta)
```

```
## [1] "40.5"
```

# Tipos de variáveis



Os tipos das variáveis são definidos a partir dos valores armazenados nela:

- **Vector**: armazena elementos de mesma classe.
- **Matrix**: vetores de duas dimensões que armazenam elementos de mesma classe.
- **List**: tipo especial de vetor que aceita elementos de classes diferentes.
- **Data.frame**: são tabelas de dados com linhas e colunas, como uma tabela do Excel.  
Como são listas, essas colunas podem ser de classes diferentes.

# Exemplo: Vector



A função `c()` cria um vetor.

```
semestre1 ← c("janeiro", "fevereiro", "março", "abril", "maio")
notas_alunos ← c(5, 6.5, 10, 0.5, 2.75)
```

É possível realizar operações com vetores.

```
vetor1 ← 1:5
vetor1 / 5 # variável vetor1 dividido por 5
```

```
## [1] 0.2 0.4 0.6 0.8 1.0
```

```
vetor2 ← 6:10
vetor1 * vetor2
```

```
## [1] 6 14 24 36 50
```

# Exemplo: Matrix



A função `matrix()` cria uma matriz.

```
primeira_matriz ← matrix(1:8, nrow = 2, ncol = 4)
primeira_matriz

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

A função `dim()` retorna a dimensão do variável (linha e coluna).

```
dim(primeira_matriz)

## [1] 2 4
```

# Exemplo: List



A função `list()` cria uma lista.

```
wizards ← list("Harry Potter", 18, TRUE, c("Hermione Granger", "Rony Weasley"))
class(wizards)

## [1] "list"
```

A função `is.list()` verifica se o variável é ou não uma lista.

```
harry_friends ← c("Hermione Granger", "Rony Weasley")
class(harry_friends)

## [1] "character"

is.list(harry_friends)

## [1] FALSE
```

# Exemplo: Data.frame



A função `head()` mostra as primeiras 6 linhas do dataframe.

```
data(iris)  
df ← iris  
head(df)
```

```
##   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
```

# Funções úteis



## Para trabalhar com dataframes

- `tail()`: mostra as últimas 6 linhas.
- `names()`: mostra os nomes das colunas.
- `View()`: mostra o dataframe.

# Desafio 4



1) Quantas observações tem o data.frame iris?

2) Quais são as variáveis do data.frame iris?

# Desafio 4 - Resposta



1) Quantas observações tem o data.frame iris?

```
tail(df)
```

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

2) Quais são as variáveis do data.frame iris?

```
names(df)
```

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

# Operadores Relacionais

- Igual a: **==**

Exemplos:

```
TRUE == TRUE
```

```
## [1] TRUE
```

```
TRUE == FALSE
```

```
## [1] FALSE
```

# Operadores Relacionais

- Diferente de: `!=`

Exemplos:

```
TRUE != TRUE
```

```
## [1] FALSE
```

```
TRUE != FALSE
```

```
## [1] TRUE
```

# Operadores Relacionais

- Maior que: **>**
- Maior ou igual:  **$\geq$**
- Menor que: **<**
- Menor ou igual:  **$\leq$**



# Operadores Relacionais

Exemplos:

```
3 < 5
```

```
## [1] TRUE
```

```
10 ≥ 10
```

```
## [1] TRUE
```

```
10 > 10
```

```
## [1] FALSE
```

# Operadores Lógicos

- E: &

Será verdadeiro se os dois forem TRUE

```
x ← 5
```

```
x ≥ 3 & x ≤ 7
```

```
## [1] TRUE
```

```
y ← 2
```

```
y ≥ 3 & y ≤ 7
```

```
## [1] FALSE
```

# Operadores Lógicos

- OU: |

Será verdadeiro se um dos dois forem TRUE

```
y ← 2  
y ≥ 3 | y ≤ 7
```

```
## [1] TRUE
```

```
y ← 1  
y ≥ 3 | y = 0
```

```
## [1] FALSE
```

# Operadores Lógicos

- Negação: !

```
(!x < 4)
```

```
## [1] TRUE
```

Uma característica importante do R que pode dificultar a comparação são os valores ausentes ou **NAs** (não disponíveis).

**NA** representa um valor desconhecido.

# NA



Quase qualquer operação envolvendo um valor desconhecido também será desconhecido.

```
NA > 10
```

```
## [1] NA
```

```
10 == NA
```

```
## [1] NA
```

```
NA + 10
```

```
## [1] NA
```

```
NA / 2
```

```
## [1] NA
```

E o mais confuso:

```
NA == NA
```

```
## [1] NA
```

**is.na()** é a função que testa se uma variável é NA.

# Índices



**Início:** o início da contagem é 1

Vetores: característica linear

```
v ← c(10:25)
```

```
v[2]
```

```
## [1] 11
```

*Data Frames:* o primeiro número indica a linha (observação) e o segundo a coluna (variável)

```
df[145, 2]
```

```
## [1] 3.3
```

# Desafio 5



- 1) No data.frame iris, testar se o Petal.Length da 5ª observação é igual ao da 6ª observação. E se o Petal.Length da 5ª observação é igual ao da 7ª observação.
- 2) A 10ª observação tem Petal.Width maior ou menor que a seguinte (11ª)? E qual é a espécie da menor?

# Desafio 5 - Resposta



1) No data.frame iris, testar se o Petal.Length da 5<sup>a</sup> observação é igual ao da 6<sup>a</sup> observação.  
E se o Petal.Length da 5<sup>a</sup> observação é igual ao da 7<sup>a</sup> observação.

```
df[5,3]==df[6,3]
```

```
## [1] FALSE
```

```
df[5,3]==df[7,3]
```

```
## [1] TRUE
```

# Desafio 5 - Resposta



2) A 10<sup>a</sup> observação tem Petal.Width maior ou menor que a seguinte (11a)? E qual é a espécie da menor?

```
df[10,4] < df[11,4] # Petal.Width da 10a linha é menor que da 11a
```

```
## [1] TRUE
```

```
df[10,5]
```

```
## [1] setosa
## Levels: setosa versicolor virginica
```

# Pacotes no R



**Pacotes** são coleções de funções, dados e documentação que estendem as capacidades do R básico.

Eles precisam ser instalados e carregados.



# Instalação de Pacotes:

- Via CRAN: `install.packages("nome-do-pacote")`.

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

- Via Github: `devtools::install_github("nome-do-repo/nome-do-pacote")`.

```
# devtools::install_github("tidyverse/dplyr")
```

# Carregar pacotes:

- `library(nome-do-pacote)`

```
library(tidyverse)
```

# Dicas sobre Pacotes



1. Você só precisa instalar o pacote uma vez, mas precisa carregá-lo sempre que começar uma nova sessão;
2. Para instalar o pacote use as aspas;
3. Para carregar o pacote, **não** utilize as aspas.

# Pacotes - CRAN Task View



Relação de pacotes encontrados no CRAN por áreas de interesse.

- CRAN Task View

<a href="#">Bayesian</a>	Bayesian Inference
<a href="#">ChemPhys</a>	Chemometrics and Computational Physics
<a href="#">ClinicalTrials</a>	Clinical Trial Design, Monitoring, and Analysis
<a href="#">Cluster</a>	Cluster Analysis & Finite Mixture Models
<a href="#">Databases</a>	Databases with R
<a href="#">DifferentialEquations</a>	Differential Equations
<a href="#">Distributions</a>	Probability Distributions
<a href="#">Econometrics</a>	Econometrics
<a href="#">Environmetrics</a>	Analysis of Ecological and Environmental Data
<a href="#">ExperimentalDesign</a>	Design of Experiments (DoE) & Analysis of Experimental Data
<a href="#">ExtremeValue</a>	Extreme Value Analysis
<a href="#">Finance</a>	Empirical Finance
<a href="#">FunctionalData</a>	Functional Data Analysis
<a href="#">Genetics</a>	Statistical Genetics
<a href="#">Graphics</a>	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
<a href="#">HighPerformanceComputing</a>	High-Performance and Parallel Computing with R
<a href="#">Hydrology</a>	Hydrological Data and Modeling
<a href="#">MachineLearning</a>	Machine Learning & Statistical Learning
<a href="#">MedicalImaging</a>	Medical Image Analysis
<a href="#">MetaAnalysis</a>	Meta-Analysis
<a href="#">MissingData</a>	Missing Data
<a href="#">ModelDeployment</a>	Model Deployment with R
<a href="#">Multivariate</a>	Multivariate Statistics
<a href="#">NaturalLanguageProcessing</a>	Natural Language Processing
<a href="#">NumericalMathematics</a>	Numerical Mathematics
<a href="#">OfficialStatistics</a>	Official Statistics & Survey Methodology

# Cronograma da tarde



- Introdução ao Pacote Tidyverse
- O Operador Pipe
- Visualização de Data Frames no RStudio
- Introdução ao Pacote dplyr
- Para aprender mais



# Tidyverse

# O Tidyverse



É uma coleção de pacotes R projetados para a ciência de dados. Todos os pacotes compartilham uma mesma filosofia de desenvolvimento, sintaxe e estruturas de dados.



# Pacotes do Tidyverse



- **ggplot2**: cria gráficos
- **dplyr**: manipulação de dados
- **tidyverse**: arruma os dados
- **readr**: leitura dos dados
- **purrr**: ferramentas para programação funcional, trabalha com funções e vetores
- **tibble**: dataframes moderno, mais simples de manipular
- **magrittr**: facilita a escrita e leitura do código
- **stringr**: trabalha com strings
- **forcats**: trabalha com fatores



# O operador %>%, o Pipe

Imagine uma receita que tenha as instruções: junte os ingredientes, misture e leve ao forno.  
Na forma usual do R, essas instruções provavelmente seriam assim:

```
forno(misture(junte(ingredientes)))
```

Dessa forma temos que pensar “de dentro para fora”. O primeiro comando que lemos é forno, sendo que essa é a última operação que será realizada.

Com o operador pipe seria algo assim:

```
ingredientes %>% junte %>% misture %>% forno
```

É mais intuitivo!

# O operador %>%, o Pipe



Para ficar mais fácil: pense no Pipe %>% como um operador que efetua as operações à direita nos valores que estão à esquerda.

Ou ainda, o operador %>% passa o que está à esquerda como argumento para a operação da direita.

**Atalho:** CTRL + SHIFT + M

# Dataframes - Tidy data



country	year	cases	population
Afghanistan	1990	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	17206362
Brazil	2000	80488	174504898
China	1999	21258	127291272
China	2000	21666	128042583

variables

This diagram shows a wide data frame with four columns: country, year, cases, and population. Four black arrows point from the column headers to the corresponding columns. Below the table, the word "variables" is centered.

country	year	cases	population
Afghanistan	1990	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	17206362
Brazil	2000	80488	174504898
China	1999	21258	127291272
China	2000	21666	128042583

observations

This diagram shows a long data frame with six rows, each representing an observation. Each row contains four values: country, year, cases, and population. Six black arrows point from the row labels to the corresponding rows. Below the table, the word "observations" is centered.

country	year	cases	population
Afghanistan	1990	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	17206362
Brazil	2000	80488	174504898
China	1999	21258	127291272
China	2000	21666	128042583

values

This diagram shows a tidy data frame where each value is represented by a black circle. The circles are arranged in a grid corresponding to the values in the original wide and long frames. Below the table, the word "values" is centered.

Fonte: Data Science with R by Garrett Grolemund

# Importação de arquivos



Pacote **readr**: funções para ler arquivos texto

- read\_csv
- read\_csv2
- read\_delim
- read\_log
- read\_rds

Pacote **readxl**: função para ler arquivo Excel

- read\_excel

Pacote **haven**: funções para ler outros softwares estatísticos

- read\_sas
- read\_spss
- read\_stata

# E lá vamos nós!



# Instruções



- Abrir o RStudio da sua máquina
- Fazer o download do repositório: <https://github.com/MaryMS/2019-02-R-Basics-R-LadiesSP>
- Descompactar na pasta Documentos
- Abrir o arquivo oficina\_R\_basico.Rproj
- Abrir um novo R script (file -> new file -> R script)

# Importação de arquivos



```
library(tidyverse)
```

*# Uma outra opção é carregar somente o(s) pacote(s) que irá utilizar.*

```
library(dplyr)
```

*# Importa o arquivo csv para o objeto df\_titanic*

```
df_titanic ← read_csv("data/titanic.csv")
```

# View e glimpse



- Para visualizar um objeto: **View**(nome-do-objeto)
- **glimpse()**: mostra informações como o número de observações (linhas) e variáveis (colunas), o nome das colunas, o tipo delas e os primeiros dados de cada coluna.

```
df_titanic %>% glimpse()
```

```
## Observations: 891
## Variables: 12
## $ id_passageiro <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1 ...
## $ sobreviveu     <chr> "nao", "sim", "sim", "sim", "nao", "nao"
## $ classe        <dbl> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2 ...
## $ nome           <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradl ...
## $ sexo           <chr> "masculino", "feminino", "feminino", "feminino", "ma ...
## $ idade          <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39 ...
## $ irmaos_conjuge <dbl> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0 ...
## $ pais_criancas   <dbl> 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0 ...
## $ passagem        <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803 ...
## $ tarifa          <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51 ...
## $ cabine          <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, "G ...
## $ embarque        <chr> "Southampton", "Cherbourg", "Southampton", "Southamp ...
```

# dplyr



A ideia do pacote **dplyr** é tornar a manipulação de dados explícita utilizando verbos que indicam a ação a ser realizada.

O encadeamento dos verbos com o banco de dados é realizado com o operador **pipe: %>%**

O dplyr foi desenhado para trabalhar com o operator pipe **%>%** do pacote magrittr.

# Os 6 verbos do dplyr



- **filter()**: seleciona linhas
- **arrange()**: ordena de acordo com uma ou mais colunas
- **select()**: seleciona colunas
- **mutate()**: cria/modifica colunas
- **summarise()**: sumariza/agrega colunas
- **group\_by()**: agrupa colunas

# filter



Selecionar linhas da base de dados.

OBS: **Tibble** é uma releitura moderna do data.frame.

```
df_titanic %>% filter(sobreviveu == "sim") # Seleciona os sobreviventes.
```

```
## # A tibble: 342 x 12
##   id_passageiro sobreviveu classe nome sexo  idade irmaos_conjuge
##       <dbl>      <chr>    <dbl> <chr> <chr> <dbl>           <dbl>
## 1          1       sim      1 Cumi~ femi~    38             1
## 2          2       sim      3 Heik~ femi~    26             0
## 3          3       sim      4 Futr~ femi~    35             1
## 4          4       sim      9 John~ femi~    27             0
## 5          5       sim     10 Nass~ femi~    14             1
## 6          6       sim     11 Sand~ femi~     4             1
## 7          7       sim     12 Bonn~ femi~    58             0
## 8          8       sim     16 Hewl~ femi~    55             0
## 9          9       sim     18 Will~ masc~    NA             0
## 10         10      sim     20 Mass~ femi~    NA             0
## # ... with 332 more rows, and 5 more variables: pais_criancas <dbl>,
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```

# filter



# Cria um objeto e atribui a ele as linhas com os sobreviventes.

```
sobreviventes ← df_titanic %>% filter(sobreviveu = "sim")
```

# Crianças com menos de 12 anos que sobreviveram.

```
criancas_sobreviventes ← df_titanic %>% filter(sobreviveu = "sim" & idade < 12)
```

# Embarque realizado nos locais: Southampton ou Queenstow.

```
embarque ← df_titanic %>% filter(embarque = "Southampton" |  
                                     embarque = "Queenstow")
```

# A instrução acima pode ser reescrita com o operador %in%:

```
embarque ← df_titanic %>% filter(embarque %in% c("Southampton", "Queenstow"))
```

# filter



```
# Pessoas sem informação de local de embarque.  
# is.na() - função que retorna TRUE se o valor for NA e FALSE se não for.  
sem_embarque ← df_titanic %>% filter(is.na(embarque))  
  
# Pessoas que tem "Elizabeth" em qualquer posição do campo nome.  
# str_detect - função que retorna TRUE se detectou o valor dado e  
# FALSE, caso não tenha encontrado.  
nome ← df_titanic %>% filter(str_detect(nome, "Elizabeth"))
```

# Desafio 1



- 1) Criar um objeto chamado passageiras que seleciona somente as passageiras.
- 2) Criar um objeto chamado criancas\_Cherbourg que seleciona as crianças com menos de 12 anos que embarcaram na cidade de Cherbourg.

# Desafio 1 - Resposta



1) Criar um objeto chamado passageiras que seleciona somente as passageiras.

```
passageiras <- df_titanic %>% filter(sexo = "feminino")
```

2) Criar um objeto chamado criancas\_Cherbourg que seleciona as crianças com menos de 12 anos que embarcaram na cidade de Cherbourg.

```
criancas_Cherbourg <- df_titanic %>% filter(idade < 12 & embarque = "Cherbourg")
```

# arrange



Ordenar as linhas da base de dados conforme uma ou mais variáveis.

```
# Ordena por ordem crescente da coluna nome.  
passageiros_ordenados ← df_titanic %>% arrange(nome)  
passageiros_ordenados
```

```
## # A tibble: 891 x 12  
##   id_passageiro sobreviveu classe nome   sexo   idade irmaos_conjuge  
##       <dbl>      <chr>    <dbl> <chr> <chr>   <dbl>          <dbl>  
## 1         846  nao        3 Abbi~ masc~    42        0  
## 2         747  nao        3 Abbo~ masc~    16        1  
## 3         280  sim        3 Abbo~ femi~    35        1  
## 4         309  nao        2 Abel~ masc~    30        1  
## 5         875  sim        2 Abel~ femi~    28        1  
## 6         366  nao        3 Adah~ masc~    30        0  
## 7         402  nao        3 Adam~ masc~    26        0  
## 8          41  nao        3 Ahli~ femi~    40        1  
## 9         856  sim        3 Aks,~ femi~    18        0  
## 10        208  sim        3 Albi~ masc~    26        0  
## # ... with 881 more rows, and 5 more variables: pais_criancas <dbl>,  
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```

# arrange



É possível ordenar na ordem descrecente e também por mais de uma variável.

```
# Ordena por ordem descrecente de idade e por ordem crescente de nome.  
passageiros_ordenados ← df_titanic %>% arrange(desc(idade), nome)  
passageiros_ordenados
```

```
## # A tibble: 891 x 12  
##   id_passageiro sobreviveu classe nome   sexo   idade irmaos_conjuge  
##       <dbl>      <chr>    <dbl> <chr> <chr>   <dbl>          <dbl>  
## 1         631 sim        1 Bark~ masc~   80            0  
## 2         852 nao        3 Sven~ masc~   74            0  
## 3         494 nao        1 Arta~ masc~   71            0  
## 4          97 nao        1 Gold~ masc~   71            0  
## 5        117 nao        3 Conn~ masc~ 70.5           0  
## 6        746 nao        1 Cros~ masc~   70            1  
## 7        673 nao        2 Mitc~ masc~   70            0  
## 8          34 nao        2 Whea~ masc~   66            0  
## 9        281 nao        3 Duan~ masc~   65            0  
## 10       457 nao        1 Mill~ masc~   65            0  
## # ... with 881 more rows, and 5 more variables: pais_criancas <dbl>,  
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```

# filter & arrange



```
# Filtra os sobreviventes homens e ordena por classe.  
df_titanic %>%  
  filter(sexo == "masculino" & sobreviveu == "sim") %>%  
  arrange(classe)
```

```
## # A tibble: 109 x 12  
##   id_passageiro sobreviveu classe nome   sexo   idade irmaos_conjuge  
##       <dbl>     <chr>    <dbl> <chr>   <chr>   <dbl>           <dbl>  
## 1          24 sim        1 "Slo~ masc~ 28            0  
## 2          56 sim        1 "Woo~ masc~ NA            0  
## 3          98 sim        1 "Gre~ masc~ 23            0  
## 4         188 sim        1 "Rom~ masc~ 45            0  
## 5         210 sim        1 "Bla~ masc~ 40            0  
## 6         225 sim        1 "Hoy~ masc~ 38            1  
## 7         249 sim        1 "Bec~ masc~ 37            1  
## 8         299 sim        1 "Saa~ masc~ NA            0  
## 9         306 sim        1 "All~ masc~ 0.92          1  
## 10        371 sim        1 "Har~ masc~ 25            1  
## # ... with 99 more rows, and 5 more variables: pais_criancas <dbl>,  
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```

# Desafio 2



- 1) Ordenar os passageiros por ordem decrescente de classe e nomeie o objeto.
- 2) Ordenar somente as passageiras por ordem de idade e dê um nome para o objeto.

# Desafio 2 - Resposta



1) Ordenar os passageiros por ordem decrescente de classe e nomeie o objeto.

```
passageiros <- df_titanic %>% arrange(desc(classe))
```

2) Ordenar somente as passageiras por ordem de idade e dê um nome para o objeto.

```
mulheres <- df_titanic %>%
  filter(sexo = "feminino") %>%
  arrange(idade)
```

# select



Selecionar colunas (variáveis) da base de dados.

```
# Seleciona as colunas indicadas.  
df_titanic %>% select(nome, idade, classe, embarque)
```

```
## # A tibble: 891 x 4  
##   nome                      idade classe embarque  
##   <chr>                     <dbl>  <dbl>  <chr>  
## 1 Braund, Mr. Owen Harris      22     3 Southampton  
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)    38     1 Cherbourg  
## 3 Heikkinen, Miss. Laina       26     3 Southampton  
## 4 Futrelle, Mrs. Jacques Heath (Lily May Peel)        35     1 Southampton  
## 5 Allen, Mr. William Henry     35     3 Southampton  
## 6 Moran, Mr. James             NA     3 Queenstow  
## 7 McCarthy, Mr. Timothy J      54     1 Southampton  
## 8 Palsson, Master. Gosta Leonard      2     3 Southampton  
## 9 Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)    27     3 Southampton  
## 10 Nasser, Mrs. Nicholas (Adele Achem)     14    2 Cherbourg  
## # ... with 881 more rows
```

# select



O select tem várias funções úteis, como por exemplo:

- **starts\_with("cla")**: seleciona colunas que começam com "cla"
- **ends\_with("ifa")**: seleciona colunas que terminam com "ifa"
- **contains("ssa")**: seleciona colunas que contêm "ssa"

# select



```
# Seleciona as colunas que começam com "id".  
df_titanic %>% select(starts_with("id"))
```

```
## # A tibble: 891 x 2  
##   id_passageiro  idade  
##       <dbl>     <dbl>  
## 1 1             1     22  
## 2 2             2     38  
## 3 3             3     26  
## 4 4             4     35  
## 5 5             5     35  
## 6 6             6     NA  
## 7 7             7     54  
## 8 8             8      2  
## 9 9             9     27  
## 10 10           10    14  
## # ... with 881 more rows
```

# select



Ao inserir o caracter - na frente da coluna, você estará excluindo as colunas da seleção.

```
df_titanic %>% select(-id_passageiro, -nome)

## # A tibble: 891 x 10
##   sobreviveu classe sexo  idade irmaos_conjuge pais_criancas passagem tarifa
##   <chr>      <dbl> <chr> <dbl>           <dbl>           <dbl> <chr>     <dbl>
## 1 nao          3 masc~    22            1             0 A/5 211~    7.25
## 2 sim          1 femi~   38            1             0 PC 17599   71.3 
## 3 sim          3 femi~   26            0             0 STON/02~   7.92
## 4 sim          1 femi~   35            1             0 113803    53.1 
## 5 nao          3 masc~   35            0             0 373450    8.05
## 6 nao          3 masc~   NA            0             0 330877    8.46
## 7 nao          1 masc~   54            0             0 17463     51.9 
## 8 nao          3 masc~   2              3             1 349909   21.1 
## 9 sim          3 femi~   27            0             2 347742    11.1 
## 10 sim         2 femi~   14            1             0 237736    30.1 
## # ... with 881 more rows, and 2 more variables: cabine <chr>, embarque <chr>
```

# Desafio 3



- 1) Criar um objeto para salvar o resultado com as colunas nome, tarifa e classe.
- 2) Mostrar uma tabela com as tarifas maiores que 50 por ordem decrescente de tarifa e ordem crescente de classe. A tabela não deverá conter os campos irmaos\_conjuge, pais\_criancas e passagem.

# Desafio 3 - Resposta



1) Criar um objeto para salvar o resultado com as colunas nome, tarifa e classe.

```
tarifa ← df_titanic %>% select(nome, tarifa, classe)
```

2) Mostrar uma tabela com as tarifas maiores que 50 por ordem decrescente de tarifa e ordem crescente de classe. A tabela não deverá conter os campos irmaos\_conjuge, pais\_criancas e passagem.

```
tarifa_classe ← df_titanic %>%
  filter(tarifa > 50) %>%
  select(-irmaos_conjuge, -starts_with("p")) %>%
  arrange(desc(tarifa), classe)
```

# mutate



Criar ou modificar colunas de uma base de dados.

Supondo que o valor da tarifa no dataset está em libras, e que 1£ = R\$ 4.93, vamos descobrir qual é o valor das tarifas em reais.

```
# Altera a coluna tarifa para o valor da tarifa em reais.  
tarifa_conversao ← df_titanic %>% mutate(tarifa = tarifa * 4.93)
```

```
# Retorna a coluna tarifa para o valor da época.  
tarifa_conversao ← df_titanic %>% mutate(tarifa = tarifa / 4.93)
```

# mutate



```
# Cria no dataset uma nova variável chamada tarifa_reais.  
tarifa_conversao ← df_titanic %>% mutate(tarifa_real = tarifa * 4.93)  
tarifa_conversao
```

```
## # A tibble: 891 x 13  
##   id_passageiro sobreviveu classe nome  sexo  idade irmaos_conjuge  
##       <dbl>      <chr>    <dbl> <chr> <chr> <dbl>          <dbl>  
## 1 1             nao        3  Brau~ masc~  22            1  
## 2 2             sim        1  Cum~ femi~  38            1  
## 3 3             sim        3  Heik~ femi~  26            0  
## 4 4             sim        1  Futr~ femi~  35            1  
## 5 5             nao        3  Alle~ masc~  35            0  
## 6 6             nao        3  Mora~ masc~  NA            0  
## 7 7             nao        1  McCa~ masc~  54            0  
## 8 8             nao        3  Pals~ masc~  2             3  
## 9 9             sim        3  John~ femi~  27            0  
## 10 10           sim        2  Nass~ femi~  14            1  
## # ... with 881 more rows, and 6 more variables: pais_criancas <dbl>,  
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>,  
## #   tarifa_real <dbl>
```

# Desafio 4



Criar uma tabela com um novo campo que contenha a tarifa em dólar seguindo essa cotação: 1£ = \$ 1.31. Classifique por ordem decrescente de tarifa.

# Desafio 4 - Resposta



Criar uma tabela com um novo campo que contenha a tarifa em dólar seguindo essa cotação: 1£ = \$ 1.31. Classifique por ordem decrescente de tarifa.

```
tarifa_conversao ← df_titanic %>%  
  mutate(tarifa_dolar = tarifa * 1.31) %>%  
  arrange(desc(tarifa))
```

# summarize



Sumariza colunas da base de dados, ou seja, resume os valores das colunas em um só valor, podendo ser a média, mediana, min, max, etc.

```
# Calcula a média da variável idade  
# na.rm = TRUE remove os NAs  
df_titanic %>% summarize(mean(idade, na.rm=TRUE))
```

```
## # A tibble: 1 × 1  
##   `mean(idade, na.rm = TRUE)`  
##                 <dbl>  
## 1                  29.7
```

# summarize



```
# Calcula: número de mulheres, mediana geral da tarifa e número de passageiros.  
# No caso abaixo a função sum() retorna o número de mulheres.  
# A função n() mostra o número de linhas (em cada grupo) e  
# costuma ser bastante usada com o summarize.  
df_titanic %>%  
  summarize(  
    mulheres = sum(sexo == "feminino", na.rm = TRUE),  
    mediana_tarifa = median(tarifa, na.rm = TRUE),  
    num_passageiros = n()  
)  
  
## # A tibble: 1 x 3  
##   mulheres  mediana_tarifa  num_passageiros  
##     <int>        <dbl>            <int>  
## 1       314         14.5           891
```

# summarize



```
# Filtra os passageiros homens e calcula a mediana da tarifa.  
df_titanic %>%  
  filter(sexo == "masculino") %>%  
  summarize(  
    mediana_tarifa = median(tarifa, na.rm = TRUE)  
)
```

```
## # A tibble: 1 x 1  
##   mediana_tarifa  
##       <dbl>  
## 1      10.5
```

# Desafio 5



- 1) Calcular a média da tarifa.
- 2) Filtrar as passageiras mulheres e calcular a mediana da tarifa.

# Desafio 5 - Resposta



1) Calcular a média da tarifa.

```
media_tarifa ← df_titanic %>% summarize(mean(tarifa, na.rm=TRUE))
```

2) Filtrar as passageiras mulheres e calcular a mediana da tarifa.

```
mulheres_tarifa ← df_titanic %>%
  filter(sexo = "feminino") %>%
  summarize(
    mediana_tarifa = median(tarifa, na.rm = TRUE)
)
```

# group\_by + summarize



Agrupa as colunas de uma base de dados.

O group\_by é bastante utilizado com o summarize.

```
# Agrupa pela variável sobreviveu e calcula
# o número de passageiros por grupo (sim/nao).
df_titanic %>%
  group_by(sobreviveu) %>%
  summarize(num_passageiros = n())
```

```
## # A tibble: 2 x 2
##   sobreviveu num_passageiros
##   <chr>          <int>
## 1 nao              549
## 2 sim              342
```

# group\_by + summarize



```
# Agrupa pelo local de embarque e calcula a mediana da tarifa de cada grupo.  
df_titanic %>%  
  group_by(embarque) %>%  
  summarize(mediana_tarifa = median(tarifa, na.rm = TRUE))
```

```
## # A tibble: 4 x 2  
##   embarque    mediana_tarifa  
##   <chr>          <dbl>  
## 1 Cherbourg     29.7  
## 2 Queenstown    7.75  
## 3 Southampton    13  
## 4 <NA>           80
```

# Desafios 6



- 1) Criar uma tabela com a quantidade de pessoas por classe.
- 2) Criar uma tabela com a mediana da tarifa por sexo.

# Desafio 6 - Resposta



1) Criar uma tabela com a quantidade de pessoas por classe.

```
df_titanic %>%
  group_by(classe) %>%
  summarize(qtd_classe = n())
```

```
## # A tibble: 3 x 2
##   classe qtd_classe
##   <dbl>     <int>
## 1 1         216
## 2 2         184
## 3 3         491
```

# Desafio 6 - Resposta



2) Criar uma tabela com a mediana da tarifa por sexo.

```
df_titanic %>%
  group_by(sexo) %>%
  summarize(mediana_tarifa = median(tarifa, na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##   sexo      mediana_tarifa
##   <chr>          <dbl>
## 1 feminino       23
## 2 masculino     10.5
```

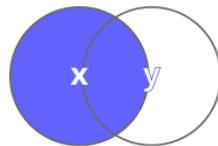
# Join



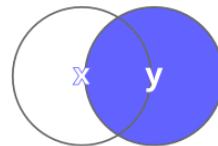
Com o **dplyr** também é possível fazer joins.

## dplyr *joins*

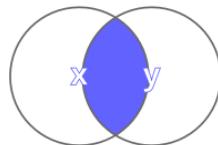
left\_join(x, y)



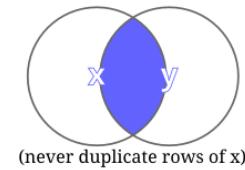
right\_join(x, y)



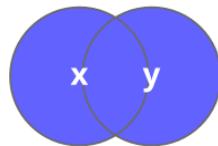
inner\_join(x, y)



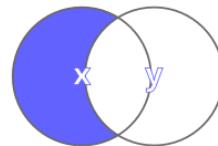
semi\_join(x, y)



full\_join(x, y)



anti\_join(x, y)





If you want to go fast,  
*go alone.*

If you want to go far,  
*go together.*

- African Proverb -

# Para aprender mais



- Repositório RLadies São Paulo
- Livro R for Data Science
- Software Carpentry
- Material do Curso-R
- R-Bloggers

# Referências



- <https://r4ds.had.co.nz>
- <https://www.curso-r.com/material/>
- <https://www.tidyverse.org>
- <https://software-carpentry.org/lessons/>
- <http://brunaw.com/slides/r-ladies-sp/13-08-2018/meetup.html#1>
- <https://github.com/MaryMS/2018-11-R-Course-FatecZS>
- [https://beatrizmilz.github.io/talk/oficina\\_intro\\_r\\_ufabc\\_2018/](https://beatrizmilz.github.io/talk/oficina_intro_r_ufabc_2018/)
- [https://bookdown.org/wevsena/curso\\_r\\_tce/curso\\_r\\_tce.html](https://bookdown.org/wevsena/curso_r_tce/curso_r_tce.html)
- [https://rstudio-pubs-static.s3.amazonaws.com/279878\\_c7634fb5fe9e40b7abc7c35aa724a2a0.html](https://rstudio-pubs-static.s3.amazonaws.com/279878_c7634fb5fe9e40b7abc7c35aa724a2a0.html)
- <https://analisereal.com/tag/introducao-a-analise-de-dados-com-o-r-2/>

# Como saber mais?



- Website RLadies Global: <https://rladies.org/>
- Twitter: [@RLadiesGlobal](#), [@RLadiesSaoPaulo](#)
- Instagram: [@RLadiesSaoPaulo](#)
- Facebook: [@RLadiesSaoPaulo](#)
- MeetUp: <https://www.meetup.com/pt-BR/R-Ladies-Sao-Paulo>
- Github: [https://github.com/rladies/meetup-presentations\\_sao-paulo](https://github.com/rladies/meetup-presentations_sao-paulo)
- R-Ladies LATAM Blog (Latin America) - Em breve!

**Não tem capítulo na sua cidade e quer iniciar um?**

Saiba como em [R-Ladies - How do get involved](#)

- Apresentação feita com [Xaringan](#), com o tema `metropolis` modificado por [Bea Milz](#).