

# **Os primeiros passos na Linguagem R**

**Marias Tecnólogas - Fatec Zona Sul**

**10/11/2018**

# Maria Marinho



- Bacharel em Matemática com Informática
- Pós Graduada em Educação Matemática
- Analista Desenvolvedora nos setores farmacêutico, bancário e outros
- Professora de Matemática
- Descobriu a paixão pelos dados
- Estuda Data Science nos MOOCs Coursera, Udacity
- Participa das comunidades PyLadies e R-Ladies São Paulo
- Transição para Ciência de Dados

[mariamarinhos@gmail.com](mailto:mariamarinhos@gmail.com)

<https://github.com/MaryMS>

<https://datascienceforeverybody.tumblr.com>

# Patrícia do Nascimento



- Formada em Ciências Sociais (USP)
- Cursando Estatística (USP)
- Atuou como professora de Sociologia na rede pública estadual por 3 anos
- Participa das comunidades PyLadies e R-Ladies São Paulo
- Atualmente aspirante à Cientista de Dados

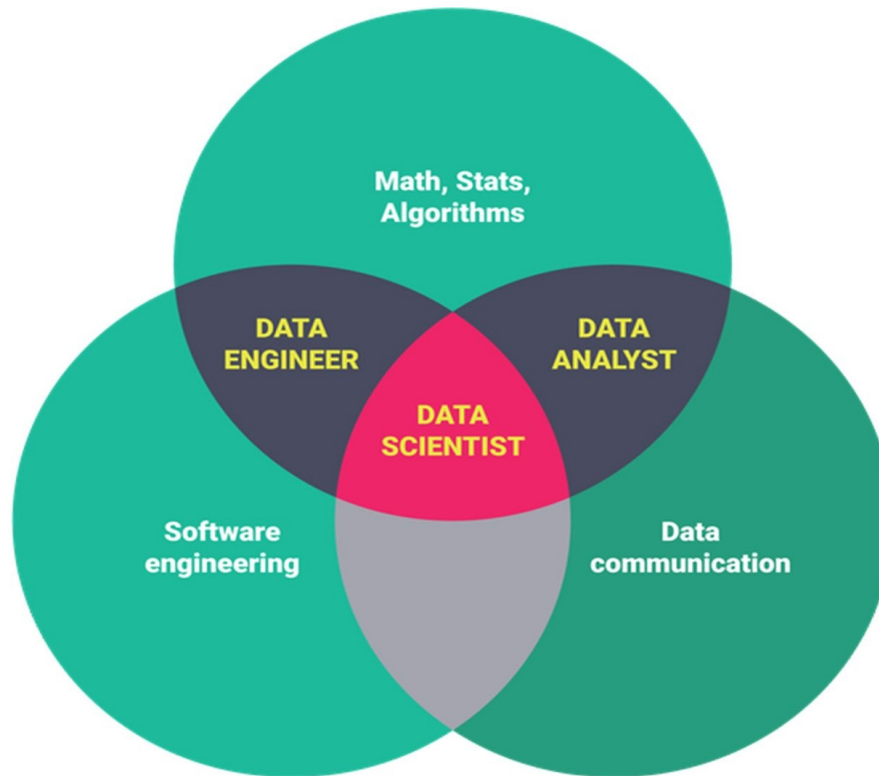
pathilink@gmail.com

# Cronograma de hoje

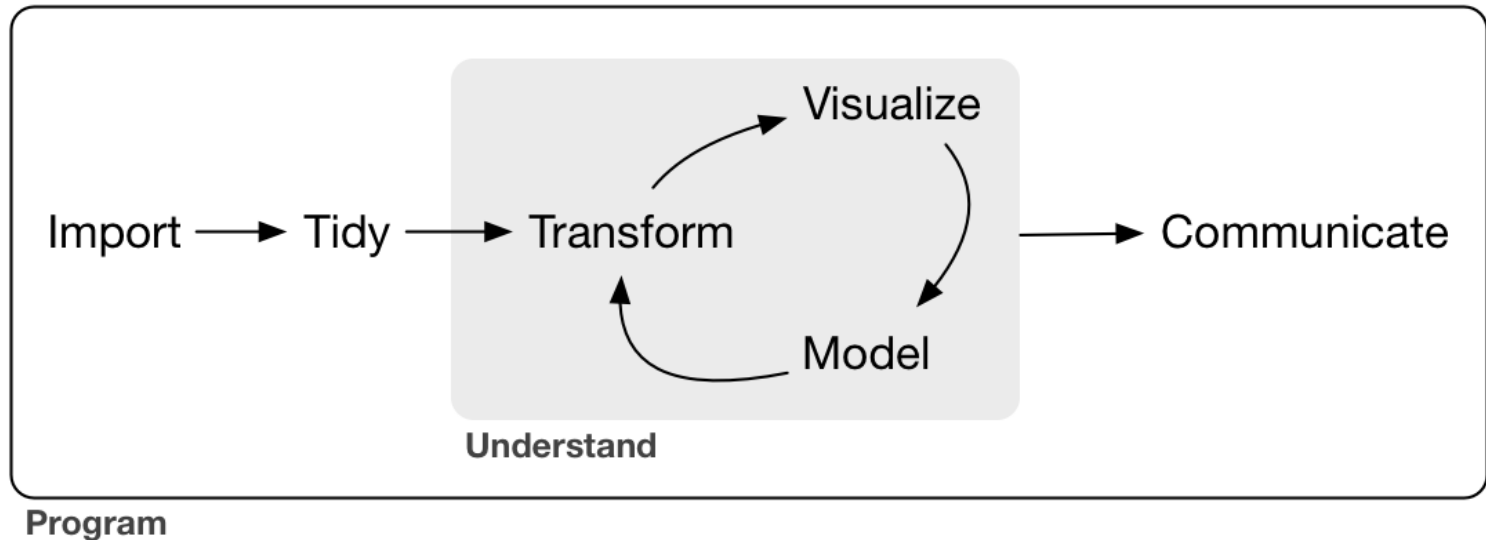
- Data Science
- Introdução ao ambiente R
- O pacote Tidyverse
- Desafios hands-on
- Para aprender mais

# O que é Data Science?

**Data Science** ou **Ciência de Dados** é uma ciência interdisciplinar sobre o processamento de grandes conjuntos de dados usando métodos estatísticos para extrair insights sobre os dados brutos.



# Ciclo da Ciência de Dados



# Ciência de Dados



TADA

DATA



SORTED



ARRANGED



PRESENTED  
VISUALLY



DATASCIENCE **GO**

Fonte: slide do Randy Lao na Conferência Data Science Go 2018

# Por que usar o R?

- É uma linguagem de programação para análise de dados
- É open source
- 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



# R Básico

# Instalação R e R Studio

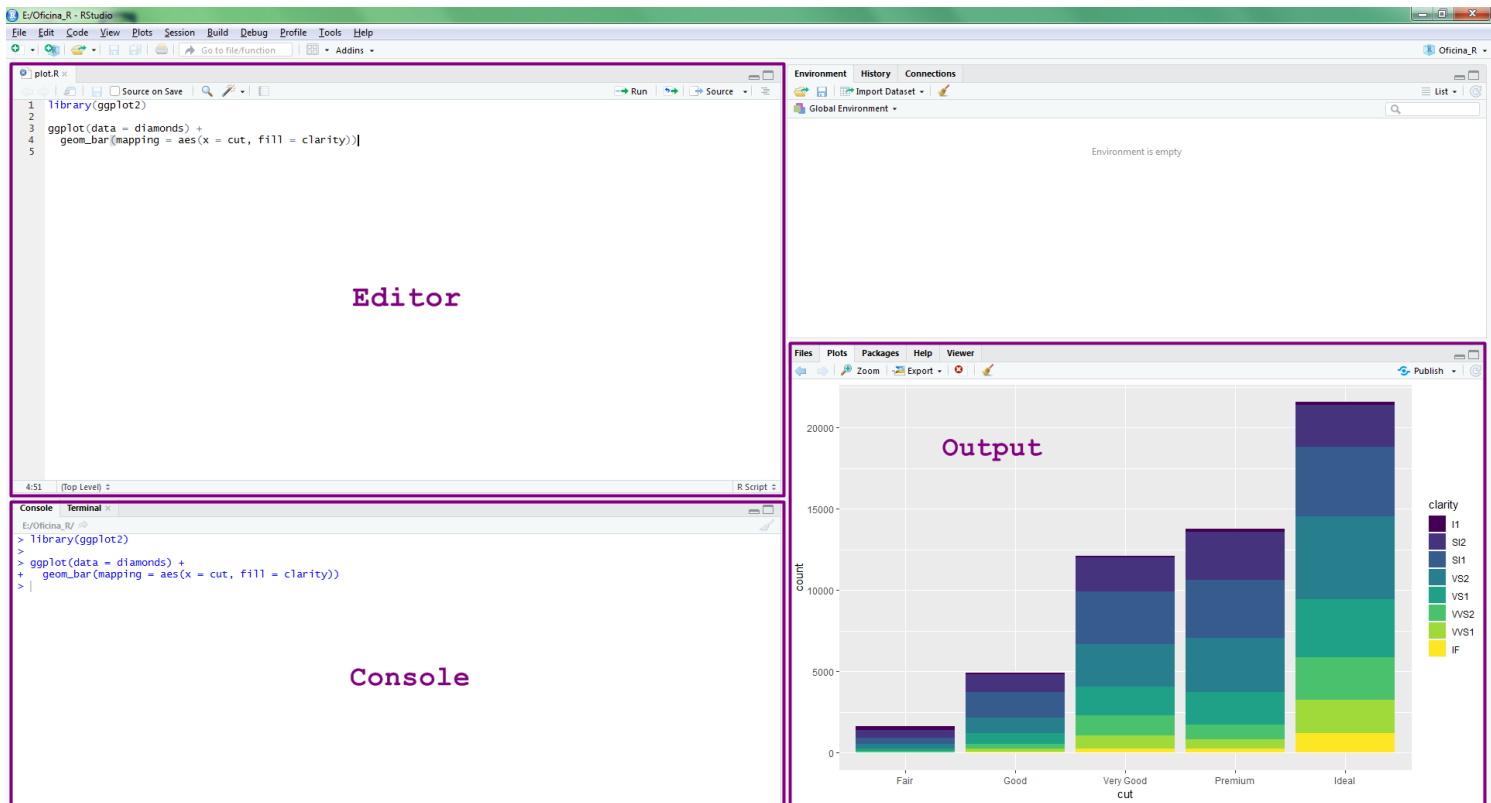
Escolha a versão para o seu sistema operacional

- Instalação do R
- Instalação do R Studio

# RStudio

**RStudio** é o IDE da Linguagem R, ou seja, o ambiente que utilizamos para editar e executar os códigos em R.

Tem quatro áreas, conforme a figura abaixo:



# Help!

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

```
help(sum)  
?sum
```

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

# 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.

# RMarkdown

No R é possível criar documentos que podem, por exemplo, explicar os dados em textos e também apresentar os códigos e os resultados destes. Tudo em um só lugar.

**RMarkdown** é um documento que contém textos e códigos em R.

O trecho de código do RMarkdown é chamado de **chunk**.

Atalho para criar um chunk: CTRL + ALT + I



# Atalhos importantes

Os atalhos facilitam. Veja os principais:

- CTRL + ENTER: roda a linha selecionada no script.
- ALT + -: (<-) sinal de atribuição.
- CTRL + SHIFT + M: (%>%) operador pipe.
- CTRL + ALT + I: cria um chunk do RMarkdown.

# R como calculadora

```
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 4 por 3
```

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

```
8 ^ 2      # potenciação
```

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

```
sqrt(1024) # radiciação
```

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

A ordem matemática das operações também vale no R.

# Nomes de objetos e 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, um objeto chamado *teste* é diferente de um outro objeto chamado *Teste*.

# Criando objetos no R

- Para atribuir um valor a um objeto no R, utilizamos o operador <-
- O atalho ALT + - gera o operador <-
- Todas as declarações R onde são criados objetos atribuindo valores a eles, tem a mesma forma:

**nome\_do\_objeto <- valor**

## Exemplos de objetos 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
```

```
salario <- 3984.23  
salario
```

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

```
ativo <- TRUE  
ativo
```

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

# Classes Básicas ou Atômicas do R

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

# Tipos de objetos:

- **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.



# Operadores Lógicos

- Igual a: ==
- Diferente de: !=
- Maior que: >
- Maior ou igual: >=
- Menor que: <
- Menor ou igual: <=
- Negação: !
- E: &
- OU: |

# NA

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
```

# NA

E o mais confuso:

```
NA == NA
```

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

**is.na()** é a função que testa se um objeto é NA.

# Estrutura Condicional: IF

```
mes <- "Novembro"

if(mes == "Novembro") {
  print("O Natal é mês que vem!") # mostra essa mensagem.
}
```

```
## [1] "O Natal é mês que vem!"
```

# Estrutura Condicional: IF e ELSE

```
mes <- "Dezembro"

if(mes == "Novembro") {
  print("O Natal é mês que vem!") # mostra essa mensagem.
} else if(mes == "Dezembro") {
  print("O Natal chegou!") # mostra essa mensagem.
}
```

```
## [1] "O Natal chegou!"
```

# Estrutura de Repetição: FOR

Imprime os 10 primeiros nomes dos personagens da base starwars usando o laço for.

```
df <- starwars  
  
lista_personagens <- head(df$name, 10)  
  
for(i in seq_along(lista_personagens)) {  
  print(lista_personagens[i])  
}
```

```
## [1] "Luke Skywalker"  
## [1] "C-3PO"  
## [1] "R2-D2"  
## [1] "Darth Vader"  
## [1] "Leia Organa"  
## [1] "Owen Lars"  
## [1] "Beru Whitesun lars"  
## [1] "R5-D4"  
## [1] "Biggs Darklighter"  
## [1] "Obi-Wan Kenobi"
```

# 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
- **tidyr**: 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
- **lubridate**: trabalha com datas

# 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 operador 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

# 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

# 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!**





# 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    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
## $ sobreviveu      <chr> "nao", "sim", "sim", "sim", "nao", "nao", "nao"...
## $ classe          <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2,...
## $ nome            <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John ...
## $ sexo            <chr> "masculino", "feminino", "feminino", "feminino"...
## $ idade           <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 2...
## $ irmaos_conjuge   <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0,...
## $ pais_crianças    <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0,...
## $ passagem        <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "1...
## $ tarifa          <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.458...
## $ cabine          <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, N...
## $ embarque        <chr> "Southampton", "Cherbourg", "Southampton", "Sou..."
```

# filter

Selecionar linhas da base de dados.

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

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

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

# 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 passageiros que seleciona somente as passageiras.
- 2) Criar um objeto chamado crianas\_Chernbourg que seleciona as crianças com menos de 12 anos que embarcaram na cidade de Chernbourg.

# 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
##   <int> <chr>         <int> <chr> <chr> <dbl>         <int>
## 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_crianças <int>,
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```

# arrange

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

```
# Ordena por ordem decrescente 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
##   <int> <chr>         <int> <chr> <chr> <dbl>         <int>
## 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_crianças <int>,
## #   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
##   <int> <chr>         <int> <chr> <chr> <dbl>         <int>
## 1          24 sim             1 Slop~ masc~ 28             0
## 2          56 sim             1 Wool~ masc~ NA             0
## 3          98 sim             1 Gree~ masc~ 23             0
## 4         188 sim             1 "Rom~ masc~ 45             0
## 5         210 sim             1 Blan~ masc~ 40             0
## 6         225 sim             1 Hoyt~ masc~ 38             1
## 7         249 sim             1 Beck~ masc~ 37             1
## 8         299 sim             1 Saal~ masc~ NA             0
## 9         306 sim             1 Alli~ masc~ 0.92            1
## 10        371 sim             1 Hard~ masc~ 25             1
## # ... with 99 more rows, and 5 more variables: pais_crianças <int>,
## #   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.

# 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>  <int> <chr>  
## 1 Braund, Mr. Owen Harris              22      3 Southampt~  
## 2 Cumings, Mrs. John Bradley (Florence Briggs Th~ 38      1 Cherbourg  
## 3 Heikkinen, Miss. Laina               26      3 Southampt~  
## 4 Futrelle, Mrs. Jacques Heath (Lily May Peel)    35      1 Southampt~  
## 5 Allen, Mr. William Henry             35      3 Southampt~  
## 6 Moran, Mr. James                     NA      3 Queenstow  
## 7 McCarthy, Mr. Timothy J             54      1 Southampt~  
## 8 Palsson, Master. Gosta Leonard         2      3 Southampt~  
## 9 Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Be~ 27      3 Southampt~  
## 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  
##           <int> <dbl>  
## 1             1    22  
## 2             2    38  
## 3             3    26  
## 4             4    35  
## 5             5    35  
## 6             6    NA  
## 7             7    54  
## 8             8     2  
## 9             9    27  
## 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_crianças passagem
##   <chr>         <int> <chr> <dbl>         <int>         <int> <chr>
## 1 nao             3 masc~   22             1             0 A/5 211~
## 2 sim             1 femi~   38             1             0 PC 17599
## 3 sim             3 femi~   26             0             0 STON/02~
## 4 sim             1 femi~   35             1             0 113803
## 5 nao             3 masc~   35             0             0 373450
## 6 nao             3 masc~   NA             0             0 330877
## 7 nao             1 masc~   54             0             0 17463
## 8 nao             3 masc~    2             3             1 349909
## 9 sim             3 femi~   27             0             2 347742
## 10 sim            2 femi~   14             1             0 237736
## # ... with 881 more rows, and 3 more variables: tarifa <dbl>,
## #   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_crianças` e `passagem`.

# 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  
##         <int> <chr>         <int> <chr> <chr> <dbl>         <int>  
## 1             1 nao             3 Brau~ masc~    22             1  
## 2             2 sim             1 Cumi~ 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_crianças <int>,  
## #   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.

# 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 x 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.

# 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(media_tarifa = median(tarifa, na.rm = TRUE))
```

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



## Desafios 6

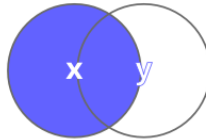
- 1) Criar uma tabela com a quantidade de pessoas por classe.
- 2) Criar uma tabela com a mediana da tarifa por sexo.

# Join

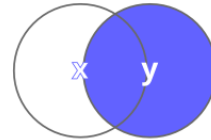
Com o **dplyr** também é possível fazer joins.

## dplyr *joins*

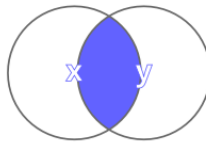
left\_join(x, y)



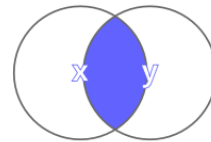
right\_join(x, y)



inner\_join(x, y)

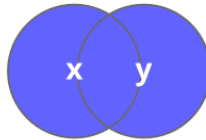


semi\_join(x, y)

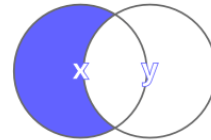


(never duplicate rows of x)

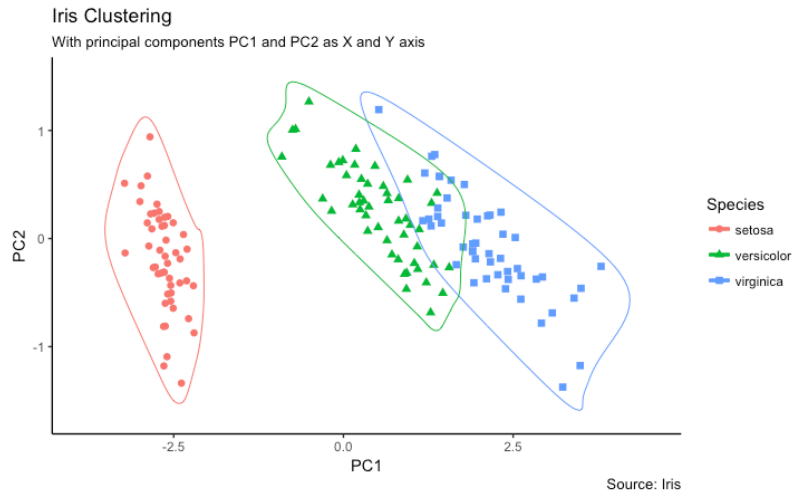
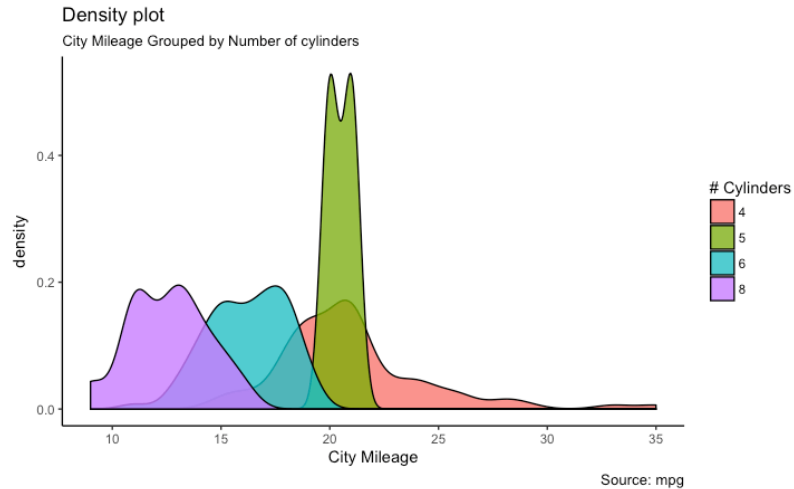
full\_join(x, y)



anti\_join(x, y)



# ggplot2 - o pacote dos gráficos



# Para aprender mais

- [Livro R for Data Science](#)
- [Material do Curso-R](#)
- [Site do Tidyverse](#)
- [R-Bloggers](#)

# Referências

- <https://r4ds.had.co.nz>
- <https://www.curso-r.com/material/>
- <https://www.tidyverse.org>
- <http://brunaw.com/slides/rladies-sp/13-08-2018/meetup.html#1>
- [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://analysereal.com/tag/introducao-a-analise-de-dados-com-o-r-2/>
- <http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html>

A apresentação foi feita em **xaringan**, um pacote do R! :)

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

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

- African Proverb -

**Muito obrigada!**

[mariamarinhos@gmail.com](mailto:mariamarinhos@gmail.com)

[pathilink@gmail.com](mailto:pathilink@gmail.com)