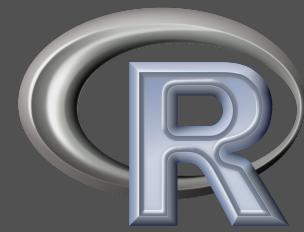




# ME115 - Linguagem R

Parte 01

1º semestre de 2023



# The R Project

# The R Project

- R: uma linguagem para análise de dados e gráfica.
- Software Livre (legalmente gratuito)
- Funciona em todas as plataformas:  
Windows, MacOS, Linux.
- Originalmente *escrito* pelos estatísticos Ross Ihaka e Robert Gentleman  
(University of Auckland).
- É uma derivação da linguagem S, desenvolvida por John Chambers.



Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

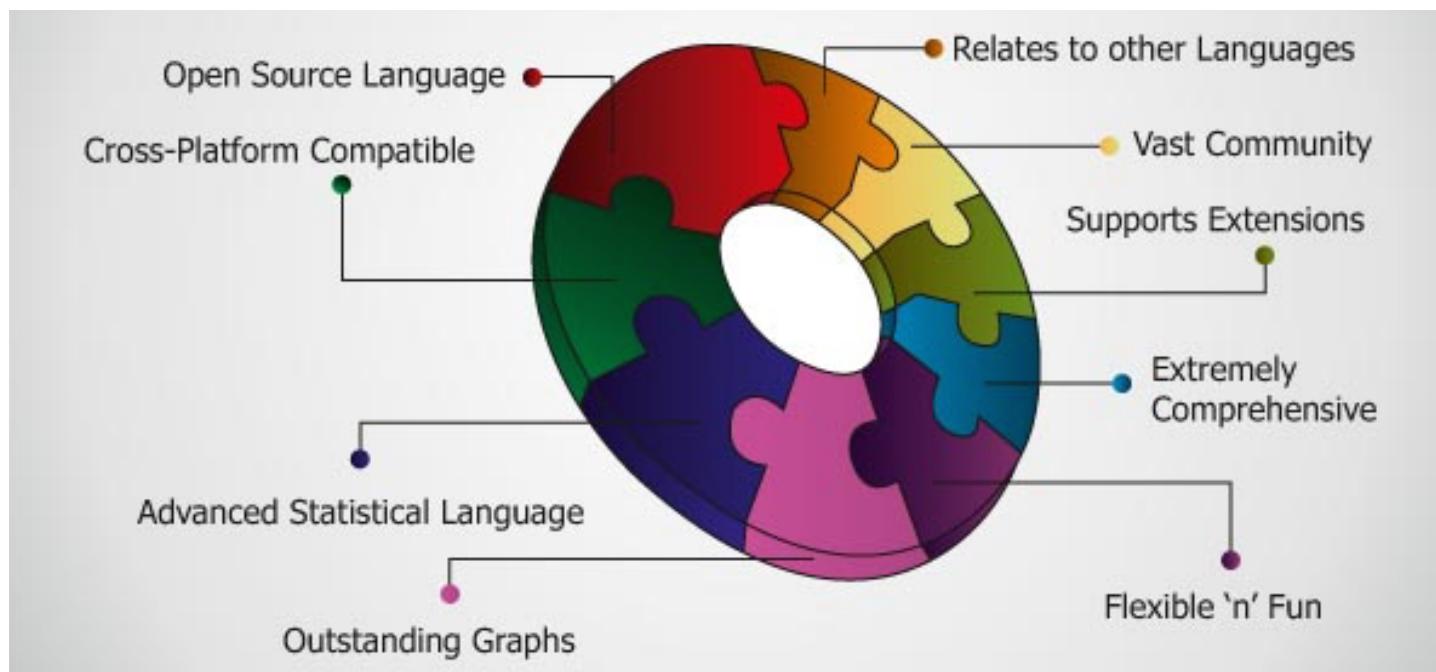
## R: A Language for Data Analysis and Graphics

Ross IHAKA and Robert GENTLEMAN

In this article we discuss our experience designing and implementing a statistical computing language. In developing this new language, we sought to combine what we felt were useful features from two existing computer languages. We feel that the new language provides advantages in the areas of portability, computational efficiency, memory management, and scoping.

**Key Words:** Computer language; Statistical computing.

# Por que usar o R?



Fonte: [http://ohi-science.org/globalfellows/why\\_rmarkdown.html](http://ohi-science.org/globalfellows/why_rmarkdown.html)

# Algumas características R



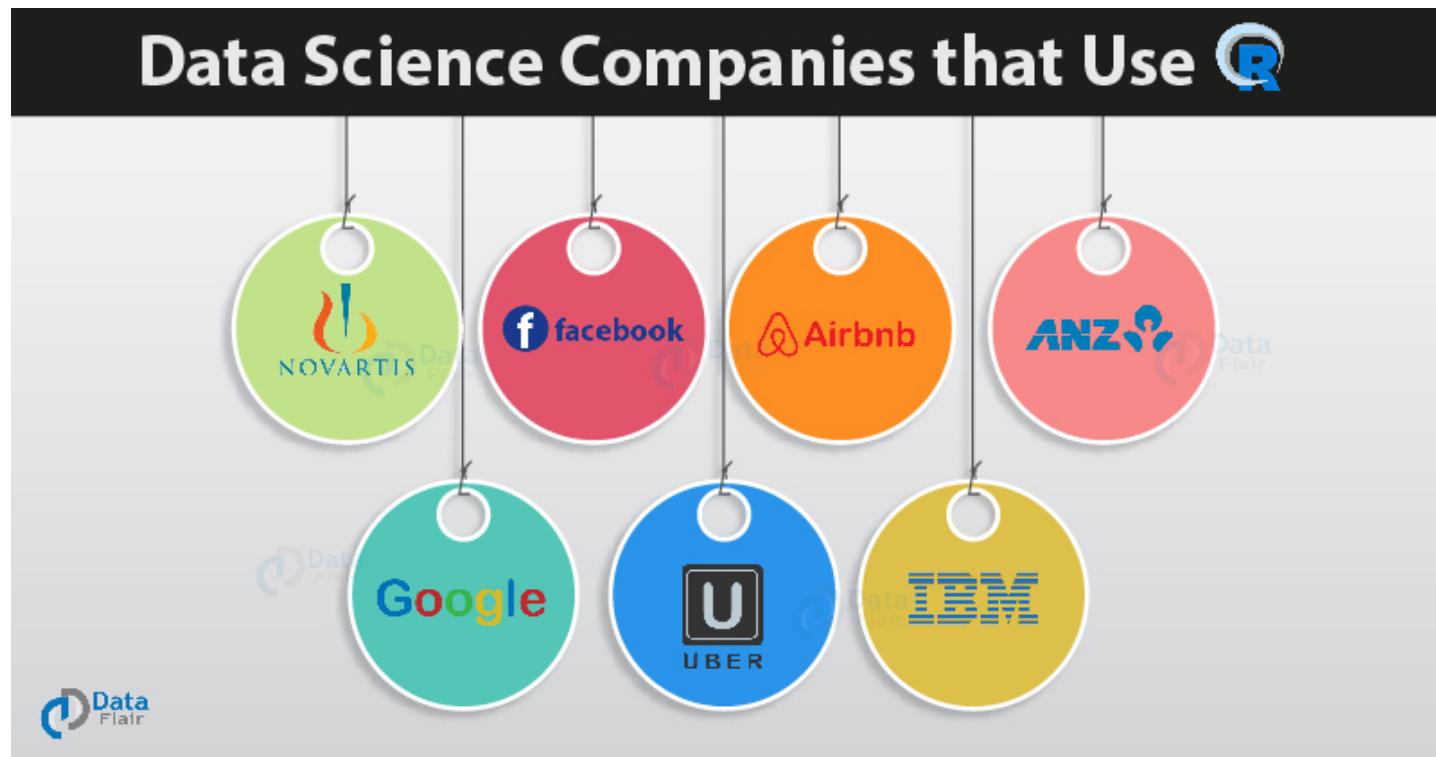
Fonte: <https://data-flair.training/blogs/using-r-for-data-science/>

# Aplicações do R



Fonte: <https://data-flair.training/blogs/using-r-for-data-science/>

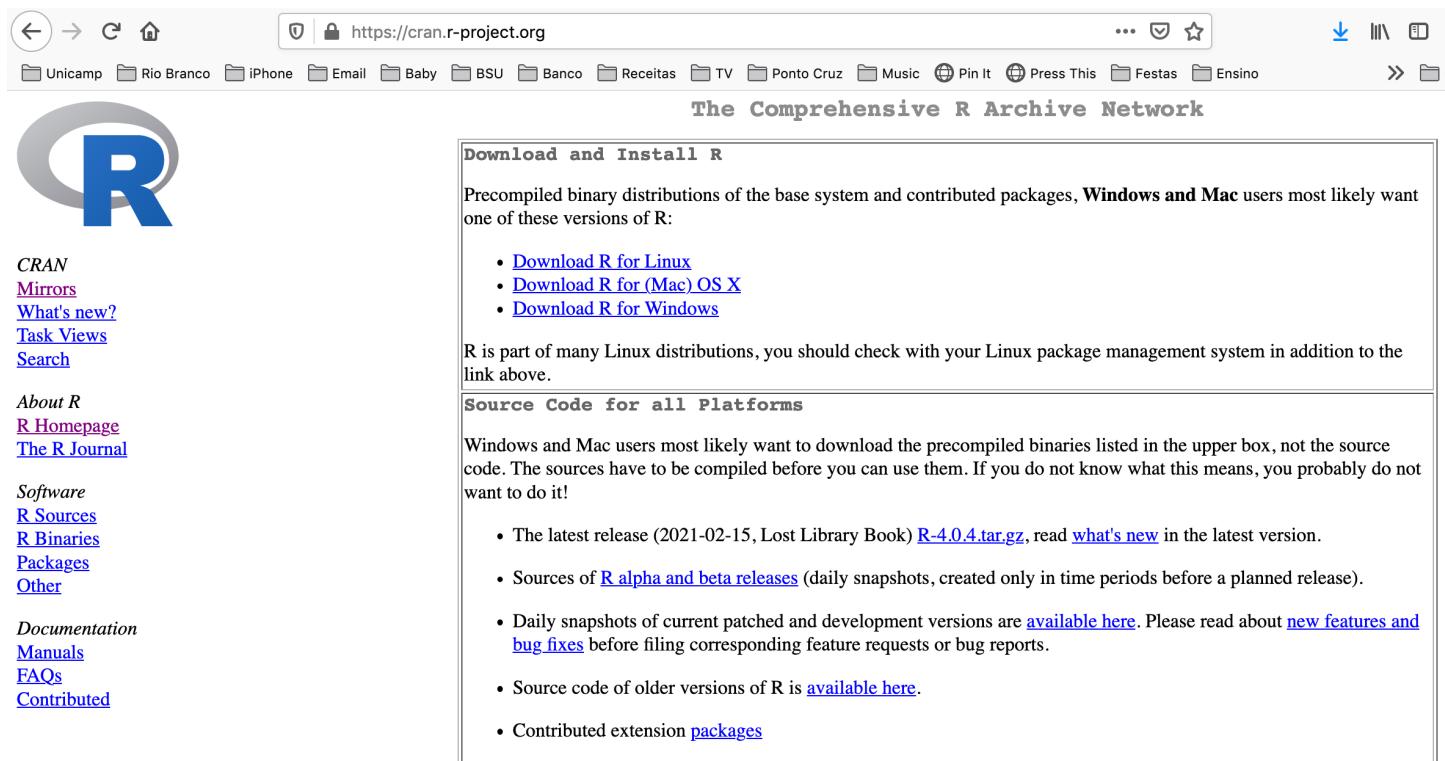
# Empresas de Ciência de Dados que usam R



Fonte: <https://data-flair.training/blogs/using-r-for-data-science/>

# Instalando o R e RStudio

Para instalar o R vá na página principal [The R Project](https://cran.r-project.org), escolha um servidor do CRAN (Comprehensive R Archive Network) e siga as instruções.



The screenshot shows a web browser displaying the CRAN homepage at <https://cran.r-project.org>. The page features the R logo and navigation links for CRAN mirrors, what's new, task views, search, about R, and documentation. The main content area is titled "The Comprehensive R Archive Network" and contains two sections: "Download and Install R" and "Source Code for all Platforms". The "Download and Install R" section provides links for precompiled binary distributions for Linux, Mac OS X, and Windows. The "Source Code for all Platforms" section explains that users should download precompiled binaries for Windows and Mac, while source code is intended for compilation. It also lists sources for alpha and beta releases, daily snapshots, and older versions.

**Download and Install R**

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

**Source Code for all Platforms**

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2021-02-15, Lost Library Book) [R-4.0.4.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

# Instalando o RStudio

Somente depois de instalar o R no seu computador, instale o [RStudio](#)



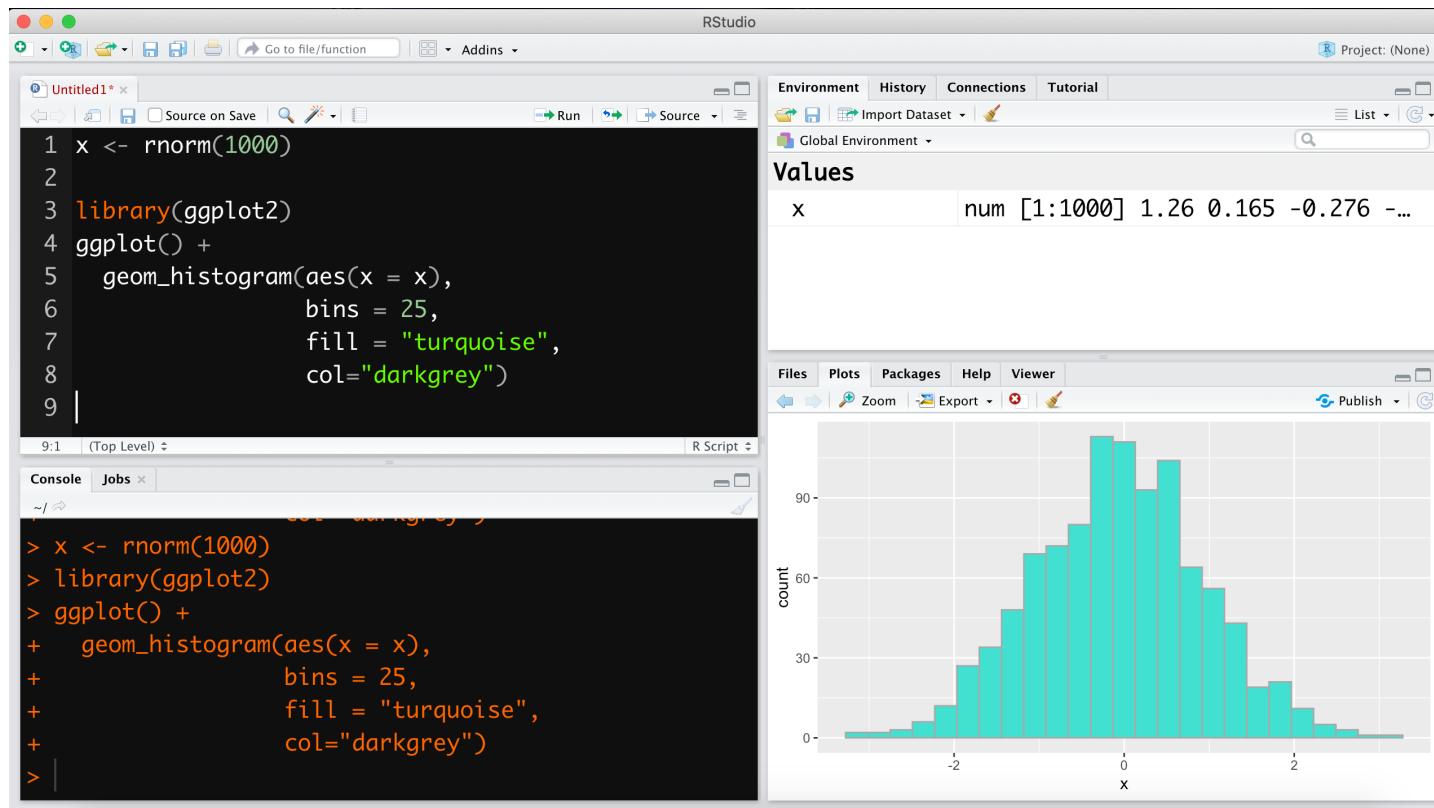
RStudio: ambiente de desenvolvimento integrado (IDE) que torna as tarefas de programar, testar, analisar e visualizar mais conveniente, gerenciável e divertida.

Existem muitos vídeos e tutoriais com as instruções de como fazer o download e instalar. Selecioneamos alguns:

- Windows: [https://youtu.be/cfwn\\_aj2o7s](https://youtu.be/cfwn_aj2o7s)
- Mac: <https://www.youtube.com/watch?v=7rFMLnm3sAE>

# RStudio

RStudio: composto por um editor, um console, ambiente de trabalho e ferramentas para gerenciamento de gráficos, histórico, debugging e ajuda.



# Pacotes (*R Packages*)

Para ter sucesso ao utilizar o R, você precisa entender o conceito de “pacotes” (ou em inglês *R packages*).

**Pacotes:** coleção de funções e conjuntos de dados definidos pelo usuário, acompanhados de documentação.

Existe um conjunto de pacotes padrão que são considerados parte do código fonte do R e já são instalados automaticamente ao instalar o R.

Outros, são criados pelos contribuidores e estão disponíveis para que outros usuários possam utilizá-los.

Fonte da Figura: <https://6chaoran.wordpress.com/2019/05/03/write-your-own-r-packages/>



# Pacotes (*R Packages*)

Os pacotes são armazenados em **repositórios**, tipicamente online e acessíveis para todos, ou você pode ter seu repositórios local.

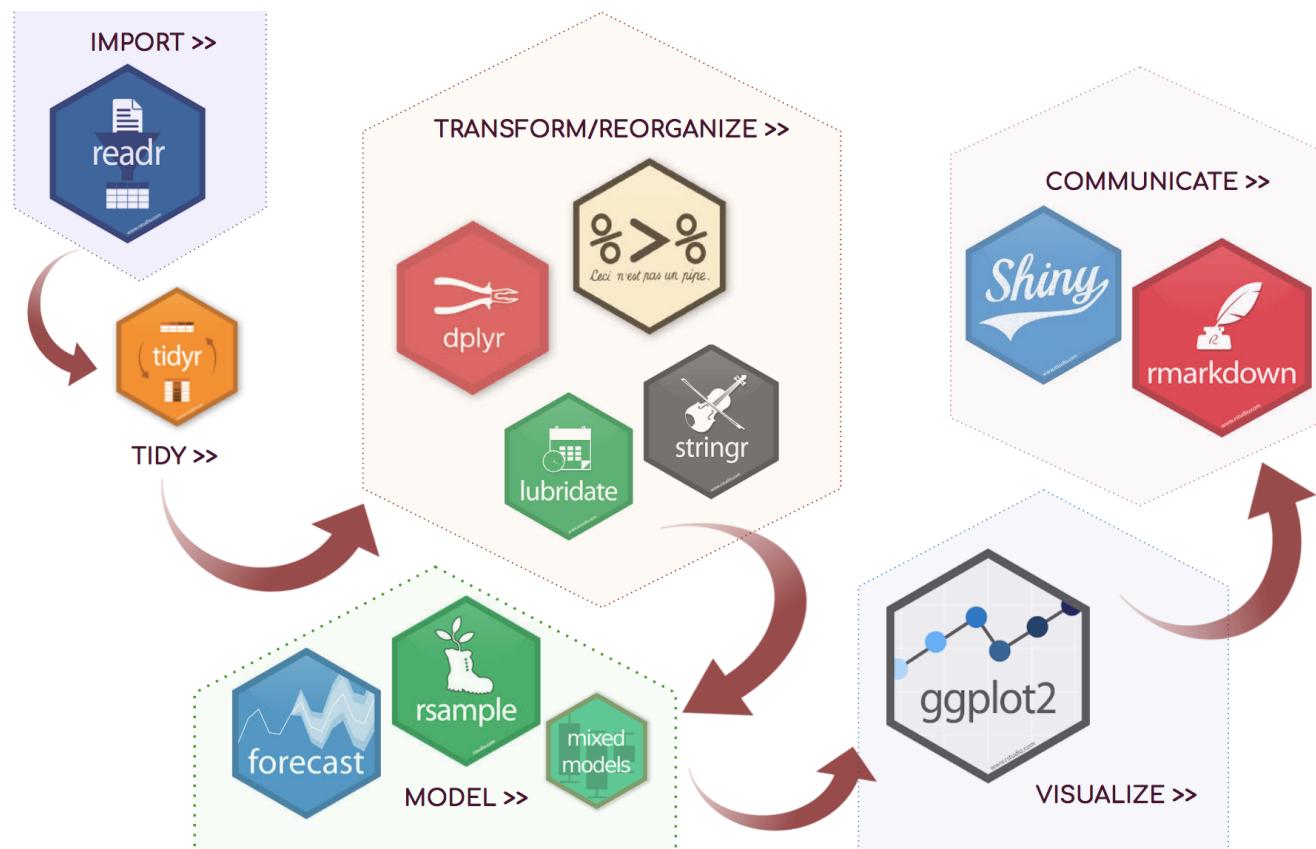


Três repositórios bem populares para obter pacotes do R:

- CRAN: repositório oficial
- Bioconductor: voltado para bioinformática
- Github: tem se tornado bem popular entre os usuários para armazenar seus projetos



# Para cada tarefa um pacote



Fonte: [http://ohi-science.org/globalfellows/why\\_rmarkdown.html](http://ohi-science.org/globalfellows/why_rmarkdown.html)

# Instalando pacotes

Para instalar um pacote no R, use a função `install.packages()` no console:

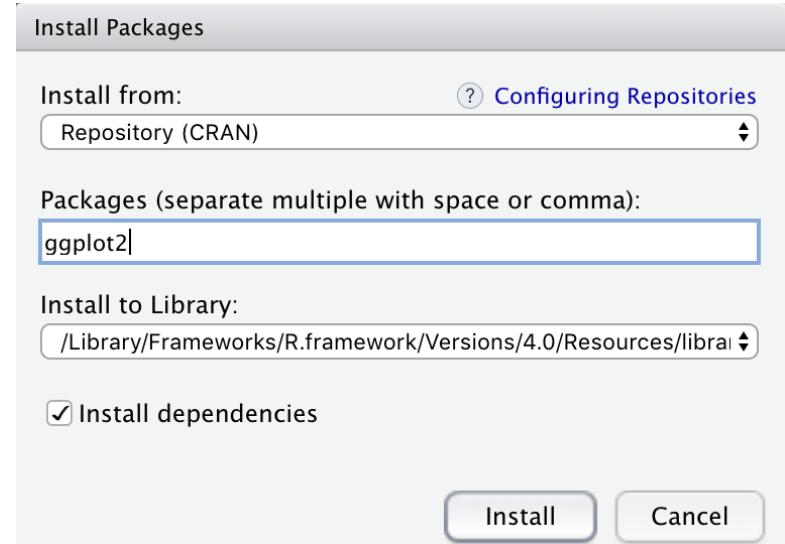
```
install.packages("ggplot2")
install.packages("ggplot2", repo = "https://cran.fiocruz.br/")
install.packages(c("nycflights13", "gapminder"))
```

No RStudio, temos dois caminhos:

- Tools > Install Packages...
- No canto inferior direito, na aba Packages > Install...

Em ambos os casos, abrirá uma janela como essa que temos no lado direito.

O que a opção *Install dependencies* faz?



# Instalando pacote *from source*

Um pacote também pode ser instalado de um arquivo local.

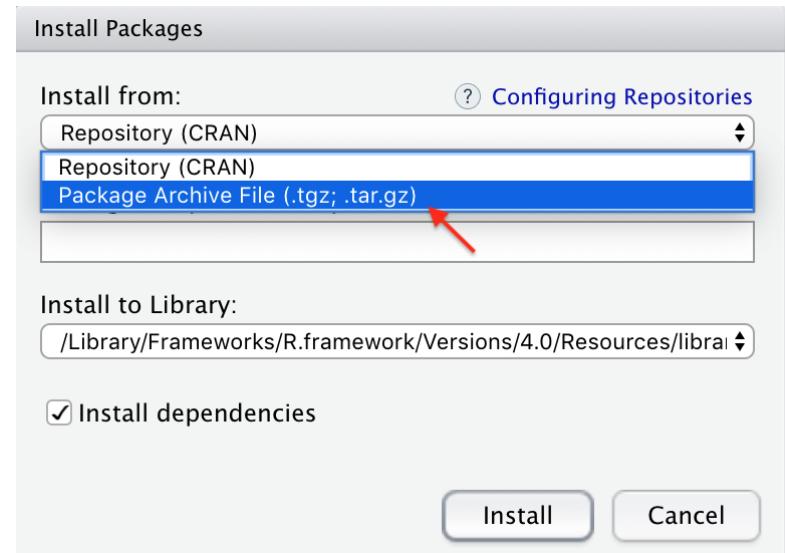
Isso pode ser útil se você está sem acesso à internet. Geralmente, esse arquivos estão com a extensão **.tgz** ou **.tar.gz**.

No RStudio, você seleciona (como na figura ao lado):

Packages > Install > Install from > ...

Ou diretamente no console:

```
# NÃO EXECUTE!
install.packages( "~/Downloads/ggplot2_1.0.1.tar.gz", type="source", repos=NULL)
```



# Atualizar ou Remover Pacotes

Para verificar quais pacotes você tem instalado no seu computador:

```
installed.packages()
```

Para remover um pacote:

```
remove.packages("ggplot2")
```

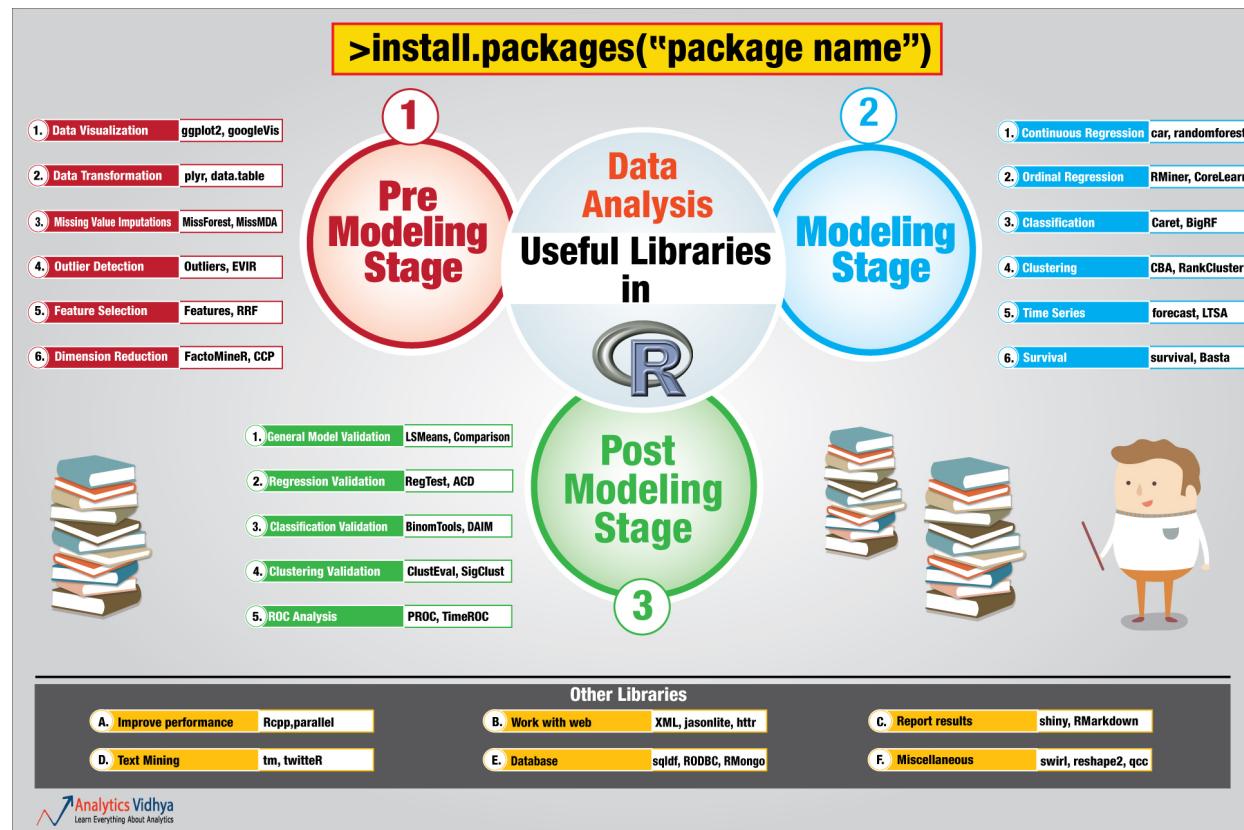
Para checar quais pacotes precisam ser atualizados:

```
old.packages()
```

Para atualizar todos os pacotes ou um pacote específico:

```
update.packages() # atualiza todos os pacotes  
update.packages("MASS") # atualiza o pacote MASS
```

# Mais exemplos de pacotes



Fonte: <https://chen115yaohua.wordpress.com/2017/01/10/r-packages-for-data-science-2016-december/>

# Carregando pacotes

Para usar as funções, objetos e arquivos de ajuda de um pacote, você precisa carregá-lo, usando a função `library()`:

```
library(tidyverse)
#> — Attaching packages ————— tidyverse 1.3.0 —
#> ✓ ggplot2 3.3.2    ✓ purrr   0.3.4
#> ✓ tibble  3.0.3    ✓ dplyr   1.0.2
#> ✓ tidyr   1.1.2    ✓ stringr 1.4.0
#> ✓ readr   1.4.0    ✓forcats 0.5.0
#> — Conflicts ————— tidyverse_conflicts() —
#> ✘ dplyr::filter() masks stats::filter()
#> ✘ dplyr::lag()   masks stats::lag()
```

Existe uma confusão entre os termos pacote e biblioteca.

# Usando as funções de diferentes pacotes

Quando você carrega um pacote usando `library()`, as funções contidas nele ficam disponível para uso diretamente no R:

```
library(dplyr)  
filter(starwars, species == "Human")
```

Alguns pacotes têm funções com os mesmos nomes, por exemplo, a função `filter()` está tanto no pacote `dplyr` quanto no `stats`. E são funções completamente distintas. Veja:

```
?filter
```

Então, para se certificar que você está usando a função do pacote que você quer, use a seguinte sintaxe `pacote::função()`:

```
dplyr::filter(starwars, species == "Human")
```

# Pacote versus Biblioteca

Existe um conjunto de pacotes padrão que são considerados parte do código fonte do R e já são instalados automaticamente ao instalar o R. E você, como usuário, também instala seus pacotes de interesse, como vimos anteriormente.

**Bibliotecas (*libraries*):** são diretórios no seu computador onde os pacotes são armazenados.

*"... a package is a like a book, a library is like a library; you use `library()` to check a package out of the library." — Hadley Wickham, Dec. 8, 2014.*



Tente os seguintes comandos e observe a saída de cada um:

```
library()  
sessionInfo()
```

# Como obter ajuda no R

O R tem uma função interna para obter informações sobre funções/pacotes que até já usamos anteriormente: `help()` ou `?`

```
help(mean) # ajuda sobre uma função  
?mean  
  
help(package = "ggplot2") # ajuda sobre um pacote  
help(ggplot2::geom_histogram) # ajuda sobre uma função dentro de um pacote
```

Para funções representadas por caracteres especiais ou palavras com significado sintático como `if`, `for` e `function`.

```
help("[[")  
help("if")
```

# Help.search e example

O comando `help.search()` ou “?” permite busca de diversas maneiras:

```
??histrogram
```

Para rodar os exemplos de um tópico da ajuda use `example()`:

```
example(mean)
```

```
##  
## mean> x <- c(0:10, 50)  
##  
## mean> xm <- mean(x)  
##  
## mean> c(xm, mean(x, trim = 0.10))  
## [1] 8.75 5.50
```

# Introdução aos Básicos

# Operações Matemáticas

Antes de qualquer coisa, o R pode ser usado como uma calculadora simples:

```
2 + 2 # adição
```

```
## [1] 4
```

```
8 * 2 # multiplicação
```

```
## [1] 16
```

```
4 ^ 2 # exponenciação
```

```
## [1] 16
```

**Nota:** o símbolo “#” é utilizado para adicionar comentários no seu código. Comentários na linha de comando é necessário dois espaços antes do “#”.

# Operadores do R

Experimente e veja qual é o resultado dessas operações:

Operação	Símbolo	Exemplo
Adição	+	$10 + 20$
Subtração	-	$10 - 20$
Multiplicação	*	$5 * 4$
Divisão	/	$5/2$
Exponenciação	$\wedge$	$2^5$
Módulo	$\%\%$	$5 \% 2$
Divisão por inteiro	$\%/\%$	$5 \%/\% 2$

# Objetos e Variáveis

**Objeto:** coisas armazenadas no R e que podem ser usadas posteriormente.

**Variável:** é um nome criado para armazenar um objeto do R.

Veja abaixo que nós usamos <- para assinalar o valor 1.5 à variável x, mas existem outras formas. Também podemos atribuir valores usando =, mas recomendamos usar <- para evitar confusão.

```
x <- 1.5 # forma preferida  
1.5 -> x  
x = 1.5  
assign("x", 1.5)
```

Para ver o valor armazenado em uma variável, basta digitar o nome da variável ou usar print():

```
x  
print(x)
```

# Como nomear suas variáveis

- Escolha um nome que diga algo à respeito daquele objeto;
- Não use hífen (-) em nomes de variáveis;
- Use letras minúsculas separadas por ponto;
- Uso de underscore (\_);
- Uso de uma mistura de letras minúsculas e maiúsculas (camelCase).

```
my-age <- 20 # não aceitável  
my.age <- 20 # preferida pelo Google's R Style Guide  
my_age <- 20 # comum em alguns pacotes como ggplot  
myAge <- 20 # camelCase
```

# Tipos Básicos de Dados

O R tem cinco tipos básicos de dados:

- **Numérico (numeric)**: valores numéricos;
- **Inteiro (integer)**: números inteiros;
- **Texto (character)**: texto (caracteres ou *string*);
- **Lógico (logical)**: TRUE ou FALSE (*booleano*);
- **Complexo (complex)**: números complexos.



Esses tipos de dados são combinados para formar as estruturas de dados: vetores, matrizes, listas, arrays e data frames.

# Tipos Básicos de Dados

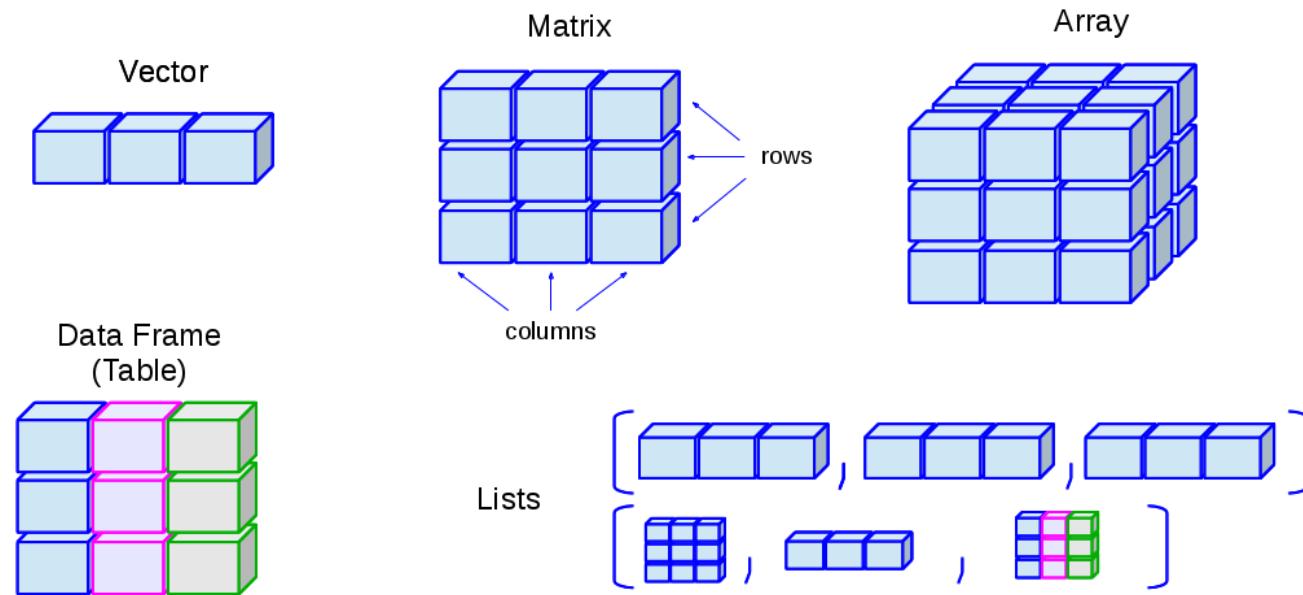
Teste alguns exemplos:

```
var1 <- 3
var2 <- as.integer(3)
b <- "Olá mundo"
c <- TRUE
d <- 12 + 3i
```

Para verificar o tipo de variável, use a função **class()**:

```
class(var1)
class(var2)
class(b)
class(c)
class(d)
```

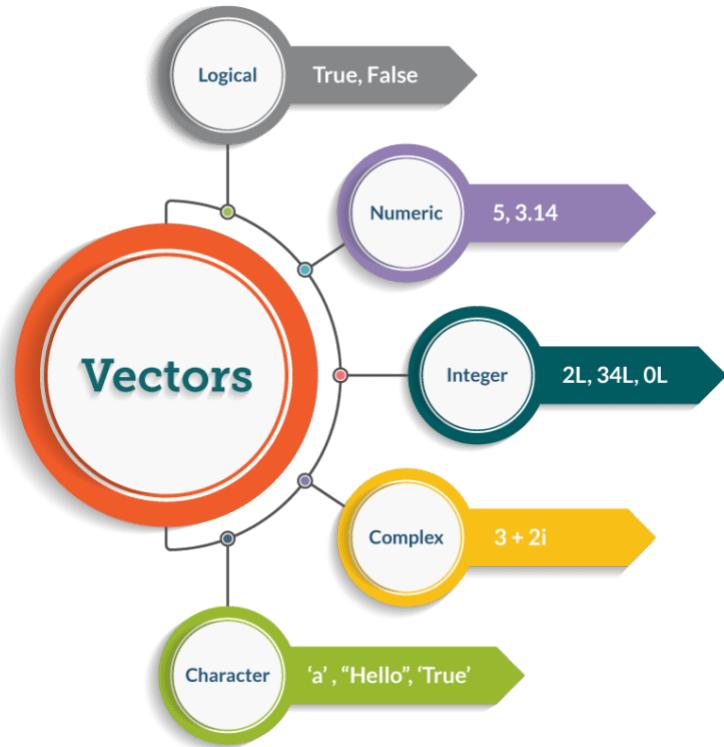
# Estruturas de Dados no R



**Vetores, matrizes e arrays:** armazenam dados do mesmo tipo.  
**Listas e data frames:** podem armazenar dados de tipos variados.

Fonte: ource: <http://venus.ifca.unican.es>

# Vetores



**Vetor** é a estrutura básica de dados no R.

**Vetores**: uma série de elementos de mesmo tipo (números, textos, lógicos, etc).

Vetores podemos ser criados com:

- **c()**: para concatenar elementos ou subvetores;
- **rep()**: para repetir elementos ou padrões;
- **seq()** ou **n:m**: para gerar sequências.

# Vetores

Veja alguns exemplos de como criar vetores:

```
vetor_numero <- c(140, -50, 20, -120, 240)
vetor_texto <- c("Segunda", "Terça", "Quarta", "Quinta", "Sexta")
vetor_logico <- c(TRUE, FALSE, TRUE, TRUE, FALSE)

vetor_seq <- 1:6
vetor_seqby <- seq(2, 5, by=0.5)
vetor_fixlenght <- seq(2, 5, length.out = 5) # Veja a diferença entre `by` e `length.out`
vetor_rep <- rep(c(3, 6, 9), times=3)
vetor_rep <- rep(c(3, 6, 9), each=3) # Veja a diferença entre `times` e `each`
```

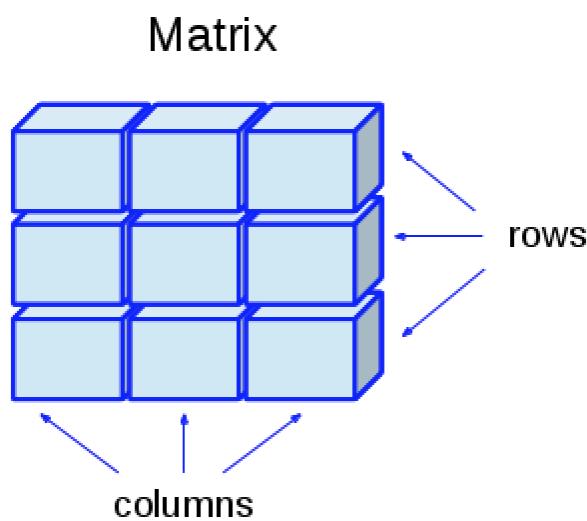
A maioria das funções matemáticas e operadores podem ser aplicados a vetores, sem precisar usar *loops*.

Na próxima aula, falaremos mais sobre vetores, como indexar seus elementos, operações, etc.

# Matrizes

Matriz é uma coleção de elementos no formato retangular bidimensional (linhas e colunas), sendo estes do mesmo tipo (numéricos, caracteres, lógicos).

Pense em uma matriz como vários vetores de mesmo comprimento dispostos lado a lado (horizontal ou vertical).



# Como criar uma matriz

Podemos criar uma matriz usando a função `matrix()`:

```
matrix(data, nrow, ncol, byrow = FALSE, dimnames = NULL)
```

Teste o seguinte exemplo:

```
m <- matrix(c(2, 4, 3, 1, 5, 7), nrow=2, ncol=3, byrow = TRUE)
```

Para assinalar nomes às colunas e linhas, respectivamente:

```
colnames(m) <- c("C1", "C2", "C3")
rownames(m) <- c("R1", "R2")
```

```
##      C1 C2 C3
## R1    2  4  3
## R2    1  5  7
```

# Como criar uma matriz

Também pode-se criar uma matriz à partir de vetores individuais do mesmo tipo e comprimento, usando as funções `rbind()` ou `cbind()`:

```
vetor1 <- c(3, 4, 5)
vetor2 <- c(9, 10, 11)
m2 <- rbind(vetor1, vetor2)
print(m2)
```

```
##          [,1] [,2] [,3]
## vetor1     3     4     5
## vetor2     9    10    11
```

```
class(m2)
```

```
## [1] "matrix" "array"
```

# Listas

Listas são um tipo especial de vetor, que podem conter elementos de diferentes tipos, incluindo outros vetores ou mesmos outras listas.

```
a <- c(3, 6, 9)
b <- c("a", "b", "c", "d")
c <- c(TRUE, FALSE, TRUE, TRUE)
my.list <- list(a, b, c)
print(my.list)
```

```
## [[1]]
## [1] 3 6 9
##
## [[2]]
## [1] "a" "b" "c" "d"
##
## [[3]]
## [1] TRUE FALSE TRUE TRUE
```

# Como criar uma lista

A função `list()` cria uma lista do mesmo modo que `c()` cria um vetor.

```
lista <- list(10:20, "R", list(TRUE, FALSE))
lista
```

```
## [[1]]
## [1] 10 11 12 13 14 15 16 17 18 19 20
##
## [[2]]
## [1] "R"
##
## [[3]]
## [[3]][[1]]
## [1] TRUE
##
## [[3]][[2]]
## [1] FALSE
```

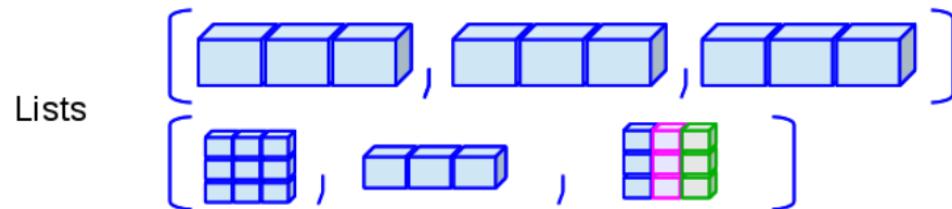
# Como criar uma lista

É possível nomear os elementos dentro de uma lista:

```
lista <- list(numeros = 10:20, software = "R", resultados = list(TRUE, FALSE))
lista
```

```
## $numeros
## [1] 10 11 12 13 14 15 16 17 18 19 20
##
## $software
## [1] "R"
##
## $resultados
## $resultados[[1]]
## [1] TRUE
##
## $resultados[[2]]
## [1] FALSE
```

# Listas



A estrutura de listas pode se tornar bem complicada, mas sua flexibilidade em armazenar dados de diferentes tipos e tamanho faz com que as listas sejam uma ferramenta muito útil, já que você pode agrupar qualquer coisa de uma lista.

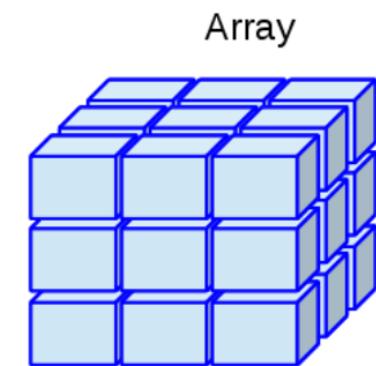
Na próxima aula, iremos estudar como acessar, extrair e manipular elementos de uma lista. Também veremos como converter listas para vetores.

# Arrays

Arrays são objetos que podem armazenar dados em mais que duas dimensões. Por exemplo, uma matriz tridimensional é um array.

A sintaxe básica para criar um array é:

```
array(data, dim, dimnames)
```



# Como criar um array

```
myArray <- array(c(1:9), dim=c(3, 3, 2))  
myArray
```

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

# Data Frames

Data frames é a estrutura de dados que vocês mais irão utilizar como Estatísticos.

Data frames servem para tabular conjuntos de dados das mais diversas áreas.

Em R, data frames é um estrutura de dados nos quais temos um número de observações (linhas) e as variáveis/medidas (colunas).

Veja o exemplo de um data frame ao lado.

São bem semelhantes a matrizes no sentido de serem retangulares, mas com a diferença que as colunas não necessariamente são do mesmo tipo de dados.

Na verdade, são um caso especial de *listas*, onde todos os elementos tem o mesmo comprimento.

nome	idade	cidade	fumante
João	21	Campinas	FALSE
Maria	20	São Paulo	FALSE
Tiago	18	Campinas	TRUE
Ana	20	Ibaté	FALSE
Priscila	19	Araraquara	FALSE

# Criando Data Frames

Para criar um data frame no R, usamos a função `data.frame()`.

```
nome <- c("João", "Maria", "Tiago", "Ana", "Priscila")
idade <- c(21, 20, 18, 20, 19)
cidade <- c("Campinas", "São Paulo", "Campinas", "Ibaté", "Araraquara")
fumante <- c(FALSE, FALSE, TRUE, FALSE, FALSE)

df <- data.frame(nome, idade, cidade, fumante)
df
```

```
##      nome  idade     cidade fumante
## 1    João    21  Campinas   FALSE
## 2   Maria    20 São Paulo   FALSE
## 3   Tiago    18  Campinas    TRUE
## 4     Ana    20     Ibaté   FALSE
## 5 Priscila    19 Araraquara  FALSE
```

# Data Frame

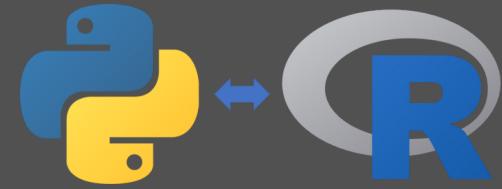
Para verificar a estrutura de um data frame, com uma breve descrição de cada variável/coluna, usamos a função `str()`:

```
str(df)
```

```
## 'data.frame': 5 obs. of 4 variables:  
## $ nome : chr "João" "Maria" "Tiago" "Ana" ...  
## $ idade : num 21 20 18 20 19  
## $ cidade : chr "Campinas" "São Paulo" "Campinas" "Ibaté" ...  
## $ fumante: logi FALSE FALSE TRUE FALSE FALSE
```

Voltaremos a falar de data frames muito em breve para aprender como manipular esse tipo de objeto, preparar para análise, etc.

nome	idade	cidade	fumante
João	21	Campinas	FALSE
Maria	20	São Paulo	FALSE
Tiago	18	Campinas	TRUE
Ana	20	Ibaté	FALSE
Priscila	19	Araraquara	FALSE



# R vs Python

# História e Propósito



- Foi criado em 1995 pelos estatísticos Ross Ihaka e Robert Gentleman.
- R foi uma implementação da linguagem S da Bell Labs.
- R foi desenvolvido por estatísticos, para estatísticos.
- Foi criado em 1991 pelo cientista da computação Guido van Rossum.
- Python foi inspirada na linguagem C, Modula-3 e ABC.
- Python é uma linguagem de uso geral, com características inerentes à outras linguagens de programação de computadores.

# Usabilidade



- Temos como IDE mais famosa
- Modelos estatísticos podem ser escritos com poucas linhas de código. (Simples para modelos estatísticos).
- R base (sem pacotes instalados) já possui implementado comandos para análises e gráficos.



- Existem diversas IDE's (Jupyter, Google Colab)



- Codificação e Debugging é mais fácil, pois a sintaxe é bem simples.
- Depende de bibliotecas instaladas para fazer análises e gráficos.

# Análise de Dados



- R é extremamente eficiente em análise de dados, devido seu grande número de pacotes com modelos, fórmulas e testes estatísticos.
- Alguns Pacotes: `dplyr`, `ggplot2`, `car`, `readr`, `tidyverse`.
- Python não foi criada inicialmente para análise de dados, mas a linguagem tem crescido rapidamente e cada vez oferece mais recursos (novas bibliotecas).
- Alguns Pacotes: `Pandas`, `Scikit-Learn`, `NumPy`, `Matplotlib`, `Seaborn`.

# R vs Python

Se olharmos lado a lado, ambos funcionam de maneira similar e são bem parecidos, apenas com algumas pequenas diferenças em suas sintaxes.



Tipos de Dados:

```
type()  
type(5)      # int  
type(5,5)    # float  
type('Hello') # string  
type(True)   # bool
```



Tipos de Dados:

```
class()  
class(5)      # numeric  
class(5,5)    # numeric  
class('Hello') # character  
class(True)   # logical
```

Assinalando variáveis:

```
a = 5
```

Assinalando variáveis:

```
a <- 5
```

# R vs Python

Operadores algébricos e lógicos são iguais!



Operadores algébricos:

```
+ - / *
```

Operadores lógicos:

```
< # less than
> # greater than
<= # less less than or equal to
== # is equal to
& # and
| # or
```

```
a == 5
a < 3
```



Operadores algébricos:

```
+ - / *
```

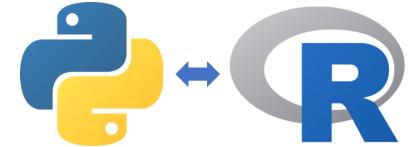
Operadores lógicos:

```
< # less than
> # greater than
<= # less less than or equal to
== # is equal to
& # and
| # or
```

# Listas e Vetores

Em Python, uma **lista** é uma coleção mutável de elementos de qualquer tipo. O índice de uma lista em Python começa em 0 e é não inclusivo.

Em R, um **vetor** é uma coleção mutável de elementos do mesmo tipo. O índice de um vetor em R começa em 1 e é inclusivo.



```
ls = [1, 'a', 3, False]  
  
# Python indexing starts at 0  
b = ls[0]  
print(b) # returns 1  
  
c = ls[0:1]  
print(c) # returns 1
```

```
vc <- c(1,2,3,4)  
  
# R indexing starts at 1  
b = vc[0]  
print(b) # returns 1  
  
c = vc[1:2]  
print(c) # returns 1, 2
```

# Semelhança nos Códigos



```
# Ler dados

import pandas
campeonato = pandas.read_csv("campeonato.csv")
campeonato.shape
campeonato.head(1)

# Ajustar um modelo de regressão

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train[["fg"]],train[["ast"]])
predictions <- lr.predict(test[["fg"]])
```



```
# Ler dados

campeonato <- read.csv("campeonato.csv")
dim(campeonato)
head(campeonato,1)

# Ajustar um modelo de regressão

fit <- lm(ast ~ fg, data=train)
predictions <- predict(fit, test)
```

# Comparação



Objetivo	Análise de dados e estatística	Implantação e produção
Principais usuários	Acadêmico	Programadores e desenvolvedores
Flexibilidade	Biblioteca disponível fácil de usar	Fácil de construir novos modelos a partir do zero, ou seja, cálculo de matriz e otimização
Curva de aprendizado	Difícil no começo	Linear e suave

# Vantagens e Desvantagens



## Desvantagens

- Curva de aprendizado lenta e alta
- Dependências entre bibliotecas

Não tem tantas bibliotecas quanto o R

## Vantagens

- Gráficos são feitos para falarem por si só. O R faz isso muito bem e bonito
- Grande catálogo para análise de dados
- Interface com o GitHub
- Rmarkdown
- Shiny

- Jupyter notebook: notebooks ajudam a compartilhar dados com colegas
- Computação matemática
- Implantação
- Legibilidade do código
- Velocidade
- Função em Python



[Fonte](#)

# Referências

Algumas referências online usadas para a construção desse material:

[Introdução à Ciência de Dados - Rafael A. Irizarry](#)

[Base R Cheat Sheet](#)

[Curso de R online para iniciantes - Didática Tech](#)

[Data Flair - R Tutorial Series](#)

[R Programming – Beginners Guide To R Programming Language](#)

Slides produzidos pelas professoras:

- Samara Kiihl
- Tatiana Benaglia

