

# **MBA Business Analytics e Big Data**

## **Análise Exploratória de Dados**

Prof. Dr. João Rafael Dias

2º semestre - 2022

Manipulação de *strings* com `stringr`  
Manipulação de datas com `lubridate`  
Manipulação de *dataframes* com `dplyr`  
Prática no RStudio

Técnicas de visualização  
Tipos de gráficos  
Análise univariada  
Análise bivariada  
Prática no RStudio

Apresentação do curso  
Introdução ao R/Rstudio  
Estrutura de dados  
Comando básicos  
Leitura e escrita de dados  
Prática no RStudio

Tipos de variáveis  
Medidas de Centralidade  
Medidas de Dispersão  
Medidas de Associação  
Prática no RStudio

Trabalhos práticos  
Aplicação do conteúdo

### OBJETIVOS E FOCO

- Capacitar os alunos na leitura e manipulação de bases de dados usando a ferramenta de linguagem de programação R
- Explorar o conteúdo desses dados focando em técnicas de visualização e EDA (*Exploratory Data Analysis*)
- Entender as métricas de centralidade, dispersão e associação dos dados e os tipos de variáveis
- *Problem-based approach*

### PROGRAMA

- Ciclo de vida dos dados
- Introdução à linguagem de programação R
- Leitura e escrita de dataframes
- Manipulação de strings e datas
- Manipulação de dataframes
- Tipos de variáveis
- Técnicas de visualização de dados
- Análise exploratória de dados
- Conceitos de estatística descritiva

### AVALIAÇÃO

- Trabalho I (peso **70%**): case aplicado
- Trabalho II (peso **30%**): case aplicado
- Datas de entrega: a definir

### METODOLOGIA

- Aulas expositivas, exercícios práticos e discussão de cases
- Ferramentas: R, RStudio e MS Excel

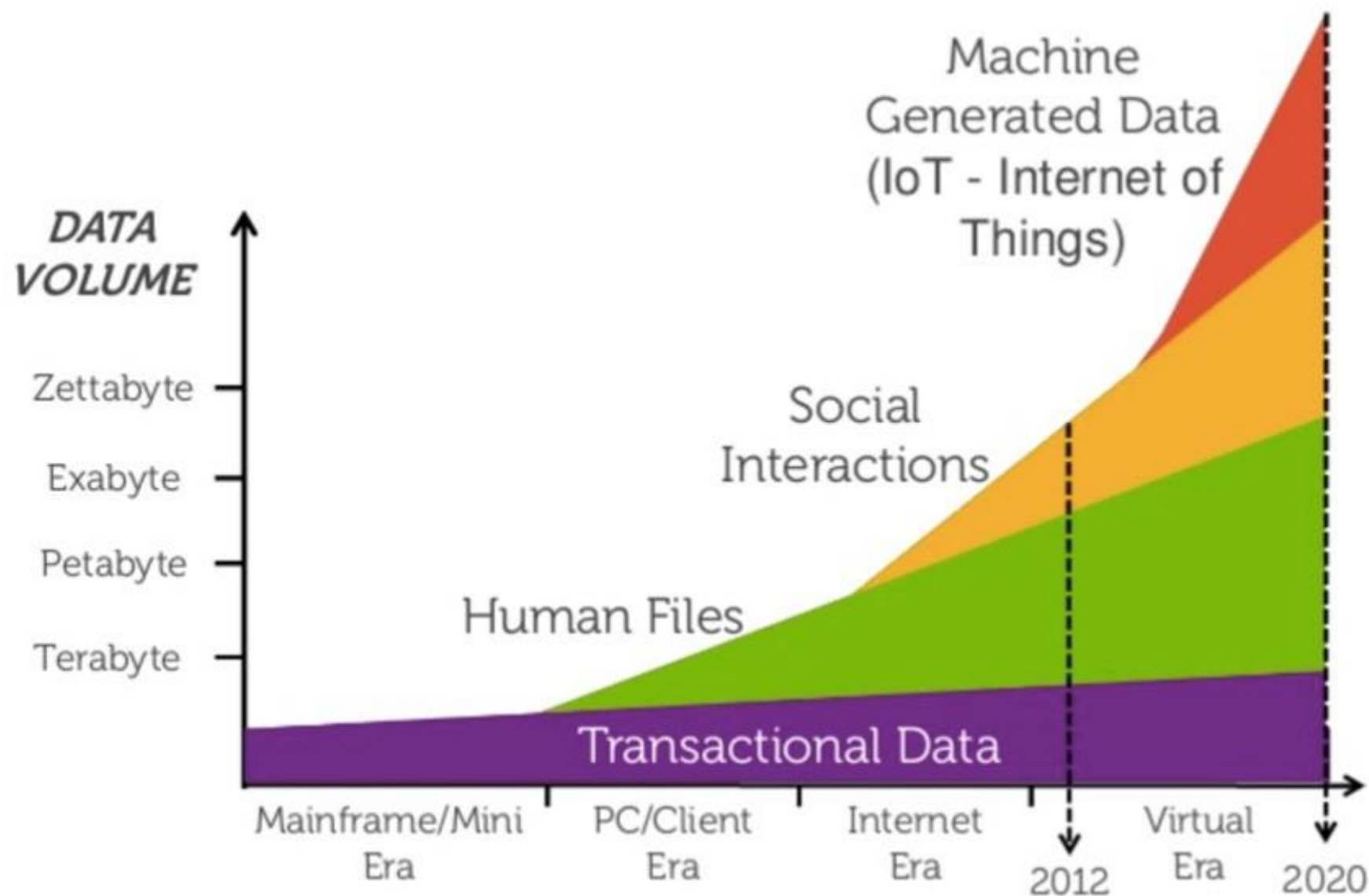
### MATERIAIS DE REFERÊNCIA

- LONG J. D. e P. TEETOR. **R Cookbook**. O'Reilly, 2019 (versão online em: <https://rc2e.com/>)
- WICKHAM H. e G. GROLEMUND. **R for Data Science: Import, Tidy, Transform, Visualize, and Model Data**. O'Reilly, 2017.
- KNAFLIC, C. N. **Storytelling com dados: Um guia sobre visualização de dados para profissionais de negócios**. Alta Books, 2019.
- VERZANI J. **Using R for Introductory Statistics**. CRC Press, 2014.
- SWEENEY D., T. A. WILLIAMS e D. R. ANDERSON. **Estatística Aplicada à Administração e Economia**. Cengage Learning, 2013.
- KAZMIER, L. **Estatística Aplicada à Administração e Economia** (Coleção Schaum). Bookman, 2006.

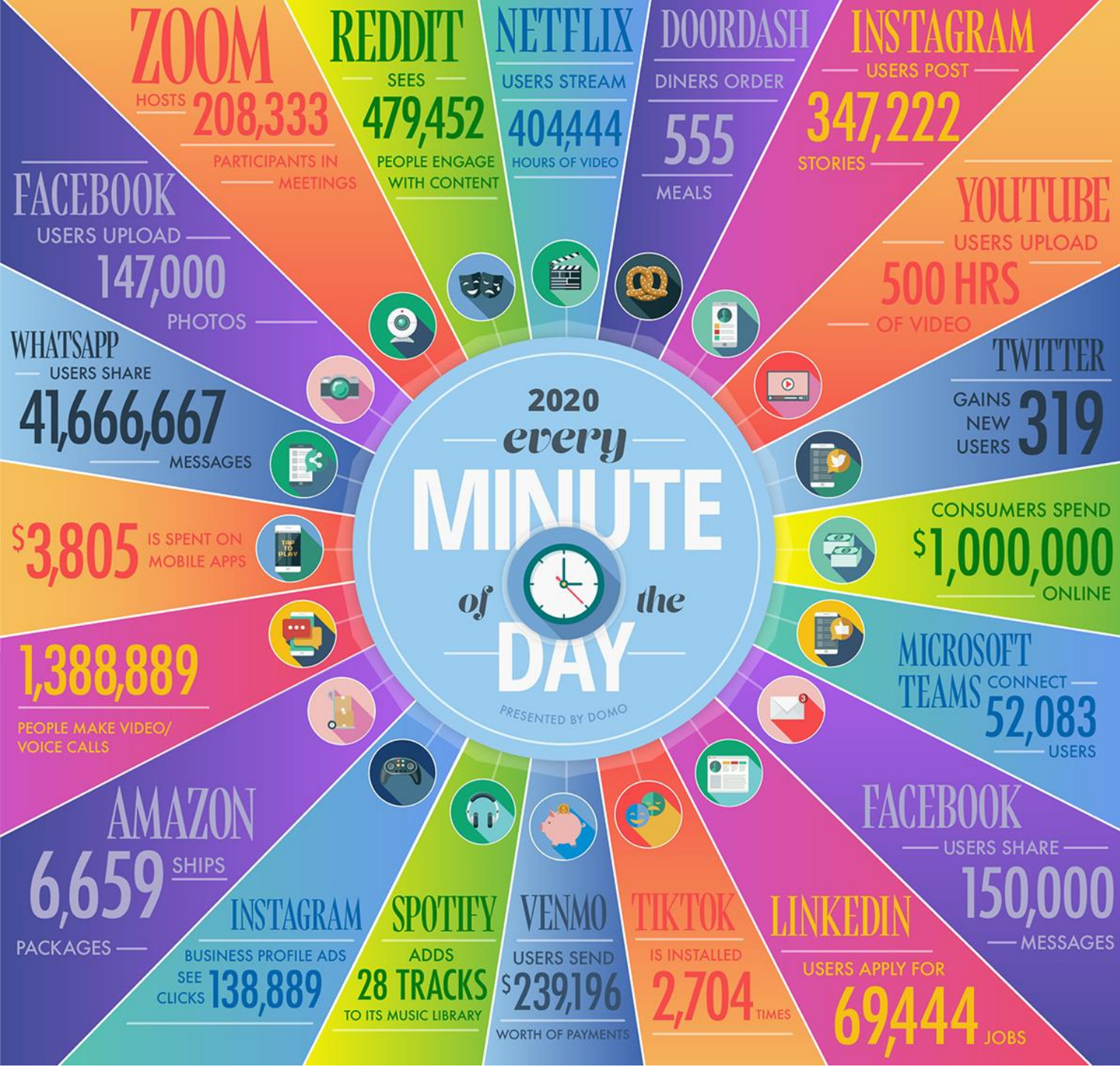
# Getting Started

# Ciclo de vida dos dados

## A explosão dos Dados







<https://www.visualcapitalist.com/every-minute-internet-2020/>

- A cada minuto é gerado um volume gigantesco de dados:

Fotos  
Reviews  
Posts  
Mensagens  
Vídeos  
Transações  
Áudio

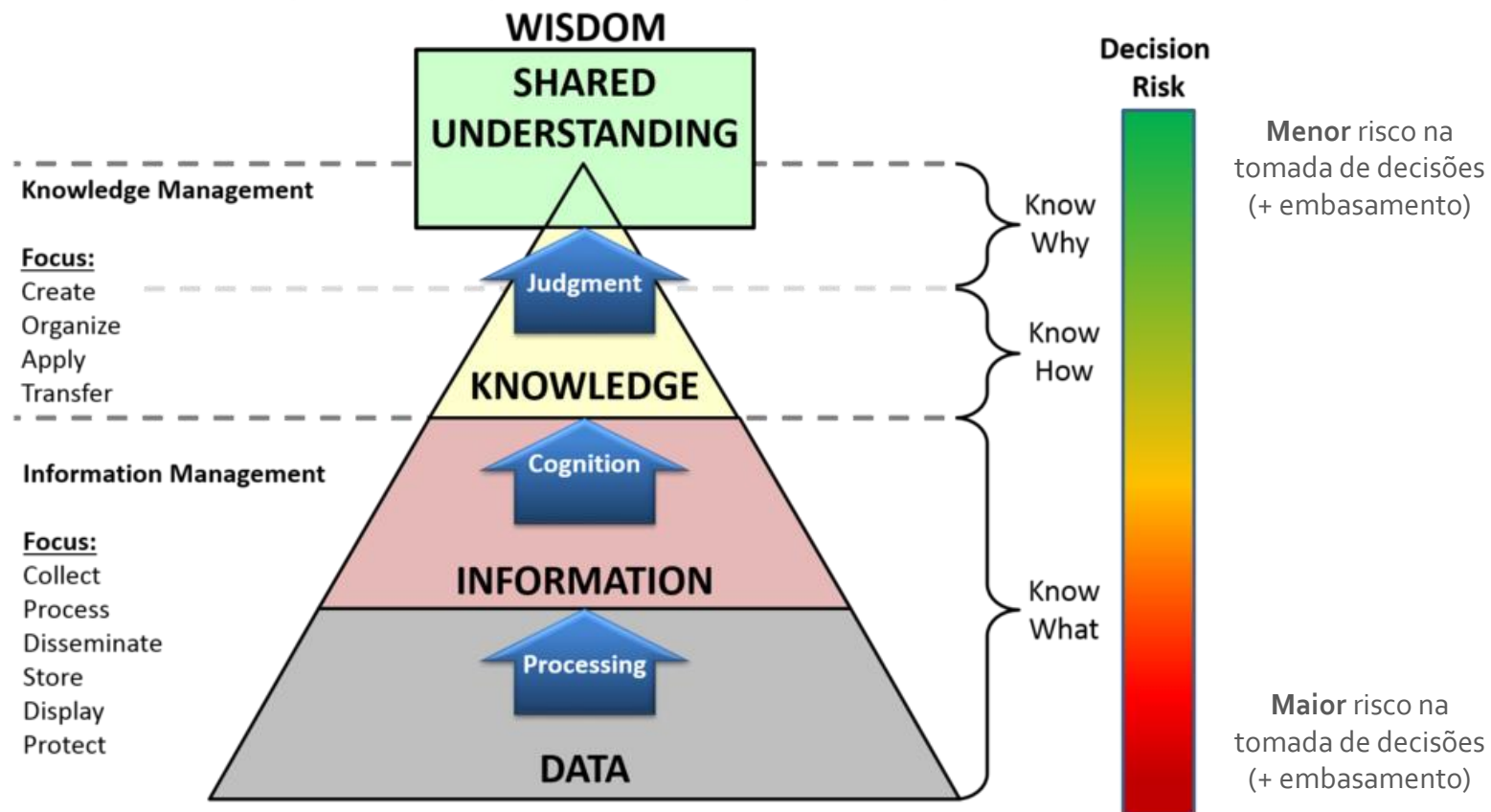
## Dados vs Informação

- Dados por si só não possuem significado relevante e não geram valor; precisam ser trabalhados para geração de *insights*.
- Informação é a ordenação e organização dos dados de forma a transmitir significado dentro de um contexto.

Quando as **informações** são interligadas elas podem ser utilizadas em um campo de atividades específico, e isso podemos chamar de **conhecimento**

Dados são passíveis de **interpretação** dentro de um contexto específico, fornecendo, desta forma, **informações** ao receptor

### Knowledge Management Cognitive Pyramid





## Dados vs Informação

- Dados por si só não possuem significado relevante e não geram valor; precisam ser trabalhados para geração de *insights*.
- Informação é a ordenação e organização dos dados de forma a transmitir significado dentro de um contexto.

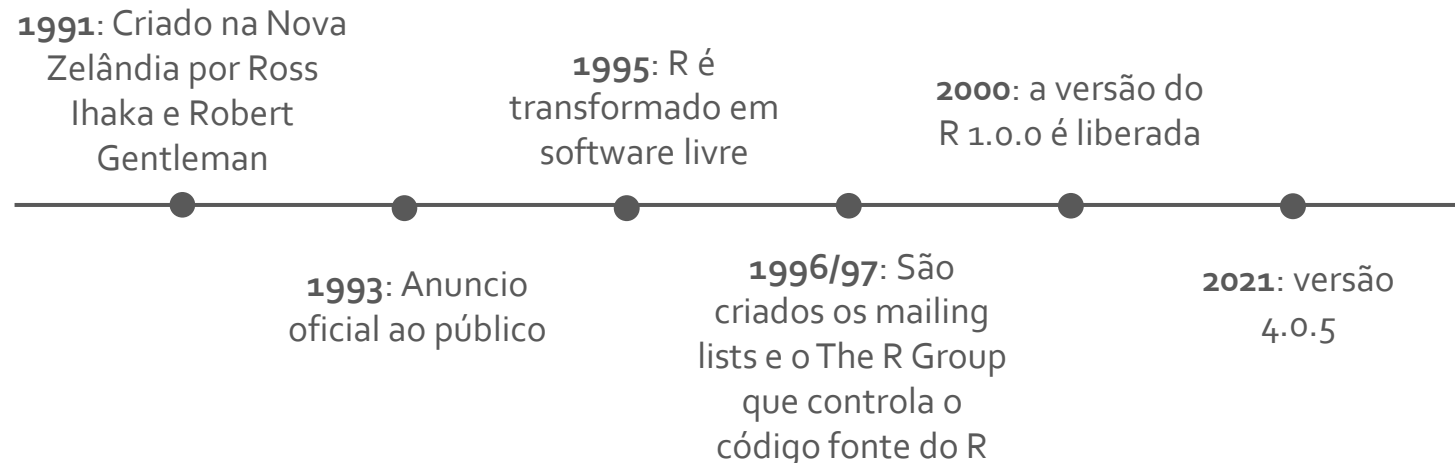


<https://medium.com/techbloghotmart/afinal-como-se-desenvolve-um-projeto-de-data-science-233472996c34>

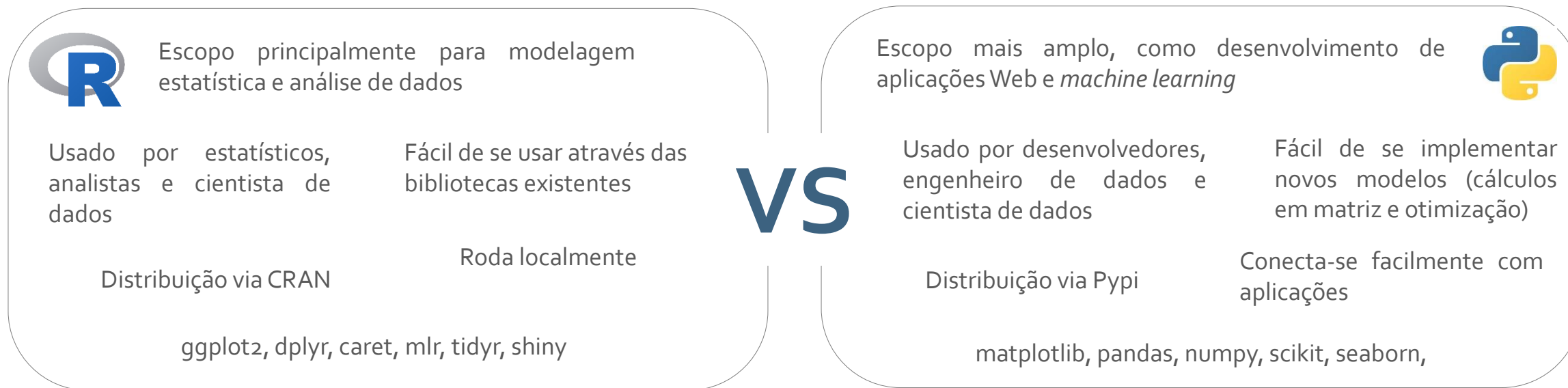
# Introdução a linguagem R e ao ambiente RStudio

- R é um dialeto da linguagem S.
- S foi inicialmente criada no Bells Labs em 1976 como uma ferramenta de análise estatística, originalmente implementada em bibliotecas de Fortran.
- As versões iniciais não continham funcionalidades para modelagem estatística.
- Em 1988 o sistema foi reescrito em C e começou a parecer com o sistema que conhecemos hoje.
- A versão 4 da linguagem S foi lançada em 1998 e é o core do R que temos hoje.

## Linha do tempo



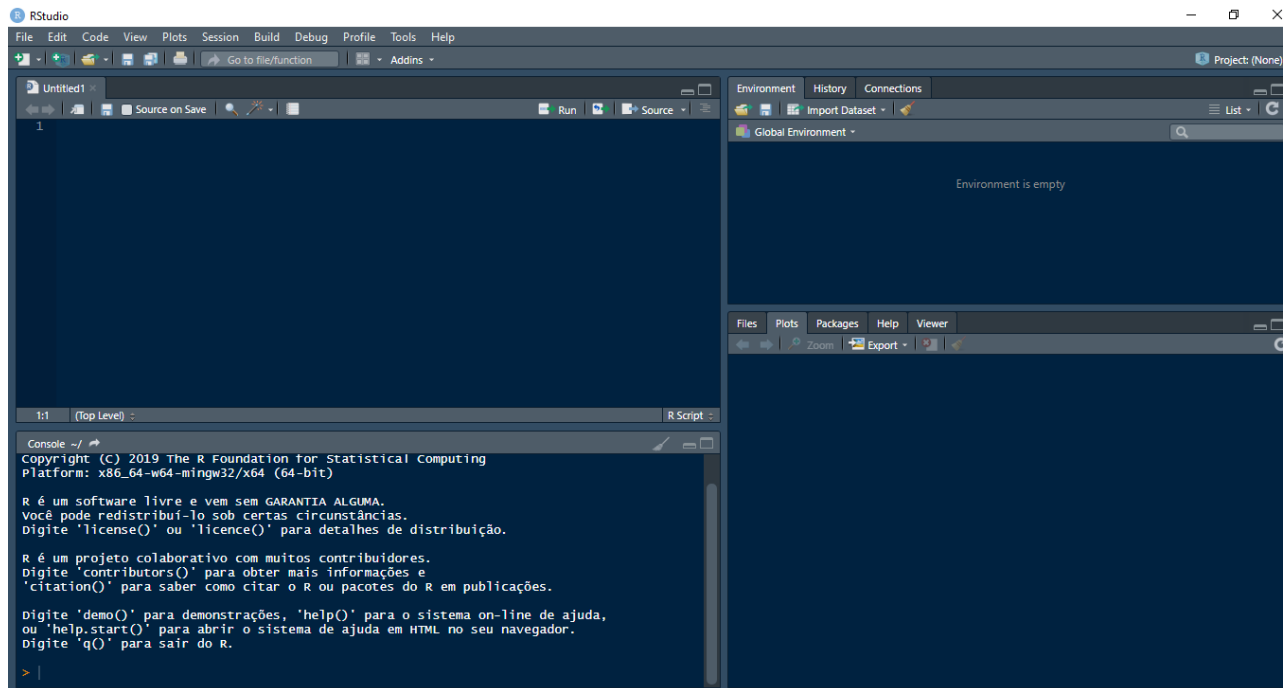
- R é uma linguagem de programação *open-source* com uma grande comunidade de usuários.
- Usado principalmente com foco em modelagem e análise de dados, com um rico ecossistema de pacotes (+ 12.000 no repositório do CRAN).
- É uma linguagem de programação muito popular no meio acadêmico, principalmente no que se refere à *machine learning* e análise estatística de dados.
- Disponível para download em: <https://cran.r-project.org/bin/windows/base/> (versão 4.1.1 para Windows e Mac, set-21).



# Introdução ao R

## RStudio

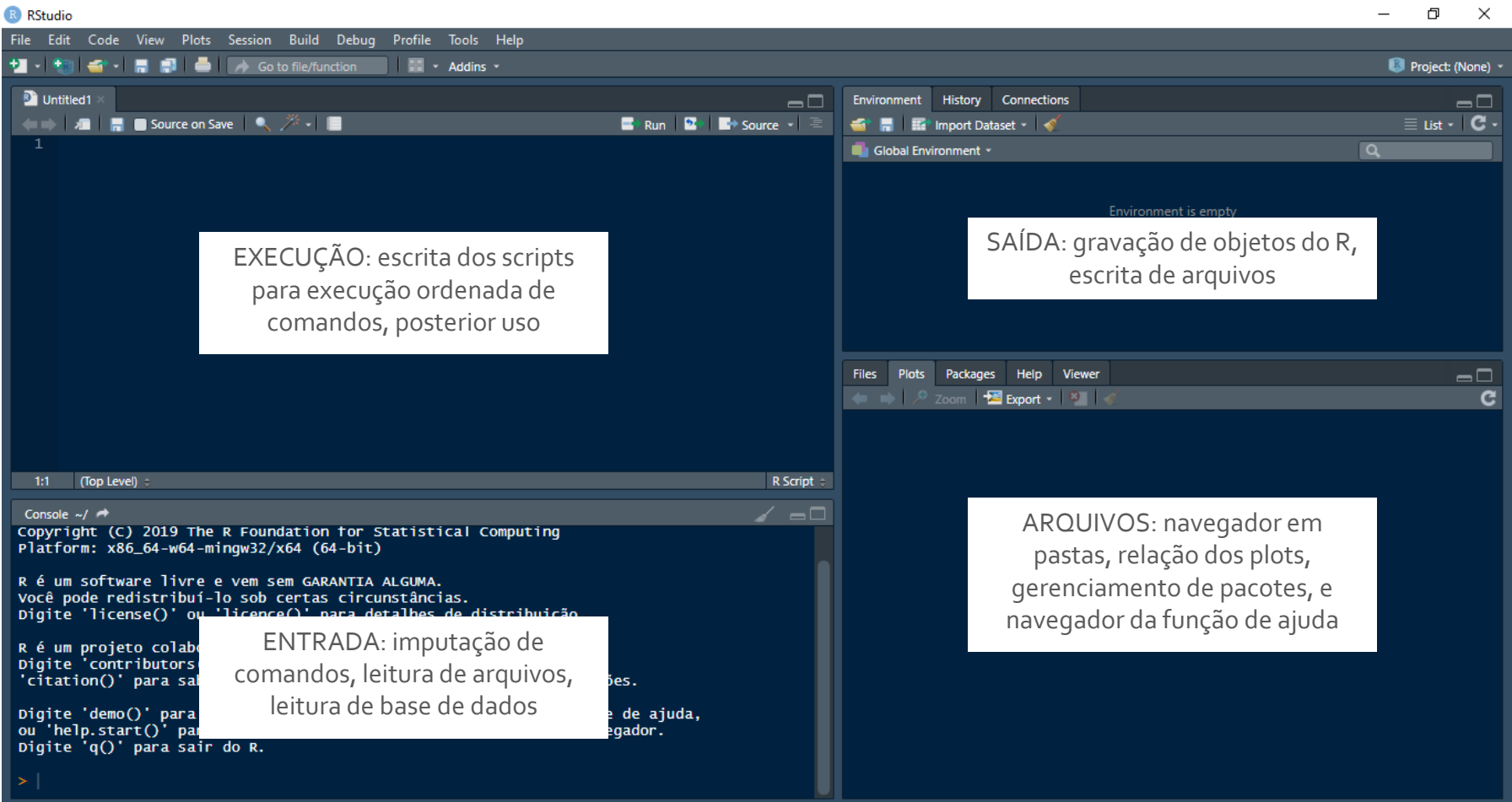
- RStudio é um Ambiente de Desenvolvimento Integrado (IDE – *Integrated Development Environment*).
- Possui console, um editor que permite execução de códigos , como também ferramentas de plotagem, *debugging* e administração do ambiente de trabalho.
- É disponível em edições open source ou comercial que rodam no desktop (Windows, Mac e Linux) ou em aplicações Web (RStudio Server).
- Disponível para download em: <https://rstudio.com/products/rstudio/download/>



# Introdução ao R

## RStudio

- A interface gráfica do RStudio é dividida em 4 painéis, que são divididos em múltiplas abas:



Aba de opções e ferramentas

Painel de programa

Painel de console

EXECUÇÃO: escrita dos scripts para execução ordenada de comandos, posterior uso

SAÍDA: gravação de objetos do R, escrita de arquivos

ARQUIVOS: navegador em pastas, relação dos plots, gerenciamento de pacotes, e navegador da função de ajuda

Painel de ambiente

Painel de arquivos



# Introdução ao R

- Para utilizar o R, é necessário baixar e carregar pacotes. O chamado “*base R*” possui apenas os operadores e funções básicas.
- Além disso, precisamos apontar um diretório de trabalho de onde serão lidos/escritos os dados.
- A função de ajuda é muito útil para entender os parâmetros de funções e sua usabilidade.

### Using Packages

**`install.packages('dplyr')`**

Download and install a package from CRAN.

**`library(dplyr)`**

Load the package into the session, making all its functions available to use.

**`dplyr::select`**

Use a particular function from a package.

**`data(iris)`**

Load a built-in dataset into the environment.

### Working Directory

**`getwd()`**

Find the current working directory (where inputs are found and outputs are sent).

**`setwd('C://file/path')`**

Change the current working directory.

**Use projects in RStudio to set the working directory to the folder you are working in.**

### Getting Help

#### Accessing the help files

**`?mean`**

Get help of a particular function.

**`help.search('weighted mean')`**

Search the help files for a word or phrase.

**`help(package = 'dplyr')`**

Find help for a package.

#### More about an object

**`str(iris)`**

Get a summary of an object's structure.

**`class(iris)`**

Find the class an object belongs to.

## Tipos de dados

- O R possui 5 tipos ou classes “atômicas” de objetos: caractere, numérico (real), inteiro, complexo, e booleano (lógico).
- Os números em R são geralmente tratados como objetos numéricos, i.e. números reais com dupla precisão
- Existem dois números especiais: `Inf` – infinito e `NaN` – valor indefinido (“*not a number*”).
- O sinal `<-` é usado para atribuição de valores (similar ao `=` usado em outras linguagens).

### Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

<code>as.logical</code>	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
<code>as.numeric</code>	1, 0, 1	Integers or floating point numbers.
<code>as.character</code>	'1', '0', '1'	Character strings. Generally preferred to factors.
<code>as.factor</code>	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

### Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

### The Environment

<code>ls()</code>	List all variables in the environment.
<code>rm(x)</code>	Remove x from the environment.
<code>rm(list = ls())</code>	Remove all variables from the environment.

**You can use the environment panel in RStudio to browse variables in your environment.**

### Exemplo

```
> x <- 1
> x
[1] 1
> print(x)
[1] 1
> msg <- 'Hello!'
> msg
[1] "Hello!"
```

- O R possui os operadores básicos aritméticos, bem como de comparação como qualquer outra linguagem de programação.
- Além disso, o R base possui os operadores e funções matemáticas
- O operador `:` é usado para criar sequências de inteiros

Operator	Description
<code>+</code>	addition
<code>-</code>	subtraction
<code>*</code>	multiplication
<code>/</code>	division
<code>^</code> or <code>**</code>	exponentiation
<code>x %% y</code>	modulus (x mod y) <code>5%%2</code> is 1
<code>x %/% y</code>	integer division <code>5%/%2</code> is 2

### Maths Functions

<code>log(x)</code>	Natural log.	<code>sum(x)</code>	Sum.
<code>exp(x)</code>	Exponential.	<code>mean(x)</code>	Mean.
<code>max(x)</code>	Largest element.	<code>median(x)</code>	Median.
<code>min(x)</code>	Smallest element.	<code>quantile(x)</code>	Percentage quantiles.
<code>round(x, n)</code>	Round to n decimal places.	<code>rank(x)</code>	Rank of elements.
<code>signif(x, n)</code>	Round to n significant figures.	<code>var(x)</code>	The variance.
<code>cor(x, y)</code>	Correlation.	<code>sd(x)</code>	The standard deviation.

### Exemplo

```
> x <- 1:5
> x
[1] 1 2 3 4 5
>
```

#### Conditions

<code>a == b</code>	Are equal	<code>a &gt; b</code>	Greater than	<code>a &gt;= b</code>	Greater than or equal to	<code>is.na(a)</code>	Is missing
<code>a != b</code>	Not equal	<code>a &lt; b</code>	Less than	<code>a &lt;= b</code>	Less than or equal to	<code>is.null(a)</code>	Is null

- Estrutura de dados é a forma de organizar os dados contendo, dessa forma, os itens armazenados e a relação entre si.
- O R possui 5 formas diferentes de estruturação de dados.
- Vetor é a forma primordial; podemos pensar que nas formas restantes como derivações do conceito de vetor. O que muda são as características de ordem e homogeneidade
- Possuem diversos atributos: dimensão, classe, comprimento, nomes, etc

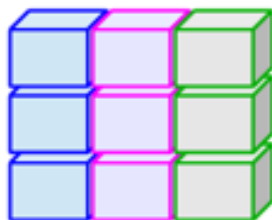
Forma mais básica de estruturação de dados

Vector

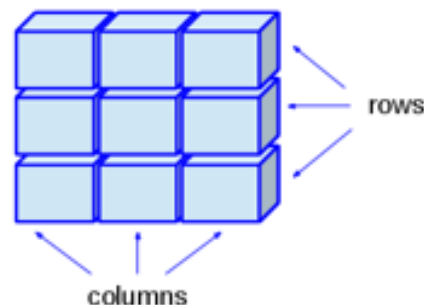


Forma mais usada para modelagem e análise de dados

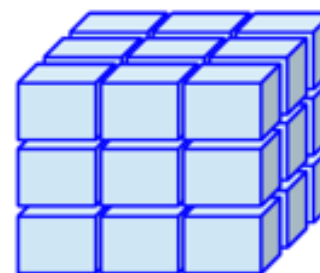
Data Frame (Table)



Matrix

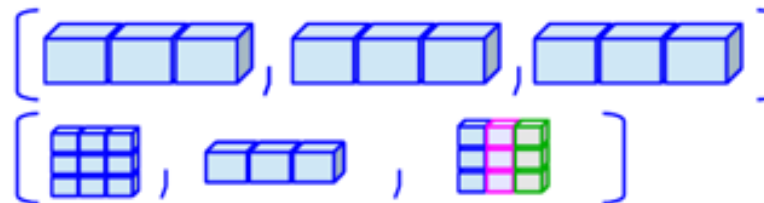


Array



Matrizes são vetores com duas dimensões. Arrays são multi-dimensionais

Lists



Coleção de elementos de formas distintas

<https://medium.com/analytics-vidhya/advanced-data-structures-in-r-2-635cbbedbc8c>

# Introdução ao R

## Vetores (*vectors*)

- Vetores é a forma mais básica de estruturação de dados em R.
- É uma coleção de dados homogêneos (i.e. mesmo tipo) ordenados.
- O operador `c` é um concatenador que cria um vetor na sua forma mais básica.

Creating Vectors		
<code>c(2, 4, 6)</code>	2 4 6	Join elements into a vector
<code>2:6</code>	2 3 4 5 6	An integer sequence
<code>seq(2, 3, by=0.5)</code>	2.0 2.5 3.0	A complex sequence
<code>rep(1:2, times=3)</code>	1 2 1 2 1 2	Repeat a vector
<code>rep(1:2, each=3)</code>	1 1 1 2 2 2	Repeat elements of a vector
Vector Functions		
<b><code>sort(x)</code></b> Return x sorted.	<b><code>rev(x)</code></b> Return x reversed.	
<b><code>table(x)</code></b> See counts of values.	<b><code>unique(x)</code></b> See unique values.	

**Exemplo**

```
> x <- c(0.5, 0.6)      ## numerico
> x <- c(TRUE, FALSE)   ## logico
> x <- c(T, F)          ## logico
> x <- c("a", "b", "c") ## caractere
> x <- 9:29             ## inteiro
> x <- c(1+0i, 2+4i)    ## complexo
>
> x <- vector('numeric', length = 8)
> x
[1] 0 0 0 0 0 0 0 0
> x <- 0:6
> x
[1] 0 1 2 3 4 5 6
> class(x)
[1] "integer"
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

Selecting Vector Elements	
By Position	
<code>x[4]</code>	The fourth element.
<code>x[-4]</code>	All but the fourth.
<code>x[2:4]</code>	Elements two to four.
<code>x[-(2:4)]</code>	All elements except two to four.
<code>x[c(1, 5)]</code>	Elements one and five.
By Value	
<code>x[x == 10]</code>	Elements which are equal to 10.
<code>x[x &lt; 0]</code>	All elements less than zero.
<code>x[x %in% c(1, 2, 5)]</code>	Elements in the set 1, 2, 5.
Named Vectors	
<code>x['apple']</code>	Element with name 'apple'.



- Uma matriz é um vetor com informação na forma bidimensional.
- Além disso, ela pode ser entendida como uma coleção de elementos distribuídos por uma quantidade fixa de linhas e colunas.
- Arrays é uma estrutura de dados multidimensional; em R é uma estrutura que armazena dados em mais de 2 dimensões.
- Passível de operações vetorizadas.

### Matrizes

```
m <- matrix(x, nrow = 3, ncol = 3)
```

Create a matrix from x.



`m[2, ]` - Select a row



`m[, 1]` - Select a column



`m[2, 3]` - Select an element

`t(m)`

Transpose

`m %*% n`

Matrix Multiplication

`solve(m, n)`

Find x in:  $m \cdot x = n$

### Exemplo

```
> m <- matrix(nrow = 2, ncol = 3)
> m                                     # geracao de uma matriz vazia 2x3
      [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
>
> dim(m)                               # dimensao da matriz
[1] 2 3
> attributes(m)                        # atributos da matriz
$dim
[1] 2 3
> nrow(m)                              # numero de linhas
[1] 2
> ncol(m)                              # numero de colunas
[1] 3
> is.array(m)                          # logico
[1] TRUE
> is.matrix(m)                         # logico
[1] TRUE
> is.data.frame(m)                    # logico
[1] FALSE
```

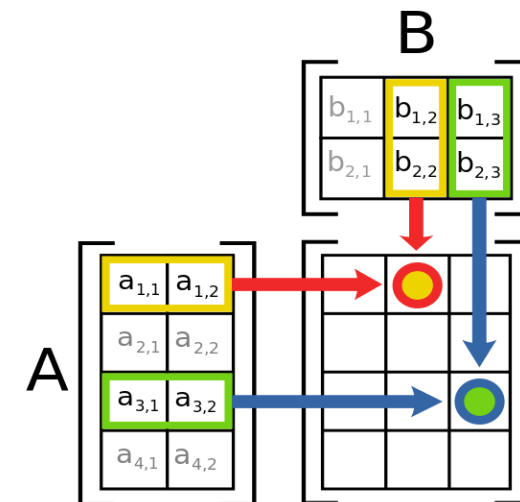
## Operações com vetores e matrizes

- Muitas operações no R são vetorizadas tornando o código mais eficiente, limpo e de fácil leitura.

### Exemplo

```
> x <- 1:4; y <- 6:9
> x; y
[1] 1 2 3 4
[1] 6 7 8 9
> x + y
[1] 7 9 11 13
> x - y
[1] -5 -5 -5 -5
> x > 2
[1] FALSE FALSE TRUE TRUE
> x >= 2
[1] FALSE TRUE TRUE TRUE
> y == 8
[1] FALSE FALSE TRUE FALSE
> x * y
[1] 6 14 24 36
> x / y
[1] 0.1666667 0.2857143 0.3750000 0.4444444
```

```
> x <- matrix(1:4, nrow = 2, ncol = 2)
> y <- matrix(rep(10, 4), nrow = 2, ncol = 2)
> x; y
      [,1] [,2]
[1,]     1     3
[2,]     2     4
      [,1] [,2]
[1,]    10    10
[2,]    10    10
>
> x * y # multiplicacao elemento por elemento
      [,1] [,2]
[1,]    10    30
[2,]    20    40
> 0.5*x # multiplicacao por uma constante
      [,1] [,2]
[1,]   0.5   1.5
[2,]   1.0   2.0
> x / y
      [,1] [,2]
[1,]   0.1   0.3
[2,]   0.2   0.4
> x %*% y # multiplicacao entre matrizes
      [,1] [,2]
[1,]    40    40
[2,]    60    60
```



<https://machinelearningmastery.com/introduction-matrices-machine-learning/>

A multiplicação **real** entre matrizes envolve **linha e coluna**

# Introdução ao R

## Dataframes

- Referem-se ao armazenamento de dados na forma tabular, onde os casos (instâncias) representam cada linha, e as observações (medições) representam as colunas.
- Representa um caso de uma lista com vetores heterogêneos de mesmo comprimento.
- É a estrutura de dados mais utilizada.

### Exemplo

```
> df <- data.frame(ID = 1:3, SEXO = c('M','F','M'),  
+                  GASTOS = c(100.5,30.2,274.2))  
> df  
  ID SEXO GASTOS  
1  1    M  100.5  
2  2    F   30.2  
3  3    M  274.2  
>  
> dim(df)           # dimensoes do df  
[1] 3 3  
> names(df)[3]      # nome da terceira coluna de df  
[1] "GASTOS"  
>  
> df$GASTOS         # elementos contidos em GASTOS  
[1] 100.5  30.2 274.2  
>
```

Also see the  
**dplyr** package.




### Data Frames

```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
```

A special case of a list where all elements are the same length.

x	y
1	a
2	b
3	c

#### Matrix subsetting

df[ , 2]	
df[2, ]	
df[2, 2]	

#### List subsetting

df\$x		df[[2]]	
-------	---	---------	---

#### Understanding a data frame

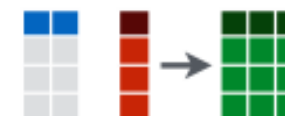
<b>View(df)</b>	See the full data frame.
<b>head(df)</b>	See the first 6 rows.

**nrow(df)**  
Number of rows.

**ncol(df)**  
Number of columns.

**dim(df)**  
Number of columns and rows.

**cbind** - Bind columns.



**rbind** - Bind rows.



- Consistem em objetos que contém elementos de diferentes tipos, como por exemplo caracteres, números, vetores, etc.
- Podem conter matrizes, *arrays*, listas dentro de si.
- O tamanho desses elementos pode ser diferente, ou seja, possui uma estrutura heterogênea.
- É um importante formato de dados no R.

### Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
```

A list is a collection of elements which can be of different types.

`l[[2]]`

Second element  
of l.

`l[1]`

New list with  
only the first  
element.

`l$x`

Element named  
x.

`l['y']`

New list with  
only element  
named y.

### Exemplo

```
> l <- list(1, c('A','b','#'), TRUE)
> l                                # imprimindo a lista
[[1]]
[1] 1

[[2]]
[1] "A" "b" "#"

[[3]]
[1] TRUE

> l[2]                            # acessando o 2o elemento da lista (vetor)
[[1]]
[1] "A" "b" "#"

> l[[2]][3] # acessando a 3a posicao do 2o elemento
[1] "#"

>
> class(l) # classe de l
[1] "list"
> length(l) # dimensao da lista
[1] 3

>
> l <- list(VAR1 = c(1:3),
+          VAR2 = 'B&@sj1',
+          VAR3 = c(TRUE,FALSE))
> l$VAR1 # acessando os elementos de VAR1
[1] 1 2 3
```

- Fatores em R são usados para representar dados categóricos. Pode-se considerar fatores como um vetor de inteiros onde cada valor representa um rótulo de classe (*label*)
- Valores omissos (inexistentes) no R são denotados por NA ou NaN para operações indefinidas matematicamente. Valores NA também possuem classe: existem números NA, caracteres NA, etc

### Exemplo

```
> x <- factor(c('Yes', 'No', 'Yes', 'No', 'No'))
> x
[1] Yes No  Yes No  No
Levels: No Yes
> table(x)
x
  No Yes
   3   2
> unclass(x)
[1] 2 1 2 1 1
attr(,"levels")
[1] "No" "Yes"
>
> y <- factor(c('Yes', 'No', 'Yes', 'No', 'No'),
+             levels = c('Yes', 'No'))
> y
[1] Yes No  Yes No  No
Levels: Yes No
> table(y)
y
Yes  No
   2   3
```

### Exemplo

```
> x <- c(1, 2, NA, 10, 3)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
> x <- c(1, 2, NaN, NA, 4)
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE
> is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE
>
> x <- c(1, 2, NA, 4, NA, 5)
> miss <- is.na(x)
> x[!miss]
[1] 1 2 4 5
```

- Estruturas de controle em R permite que controle o fluxo de execução do programa, dependendo das condições de execução. As estruturas mais comuns são: `if – else`, `while`, e `for`.
- São importantes para escrita de programas (*scripts*), e possuem duas finalidades:
  - Tomada de decisão: permite que o script tome decisões baseados em resultados em alguma condição estabelecida.

Loops: permite que um bloco de código possa ser executado diversas vezes, diferente da sequencial.

### If Statements

```
if (condition){  
  Do something  
} else {  
  Do something different  
}
```

#### Example

```
if (i > 3){  
  print('Yes')  
} else {  
  print('No')  
}
```

### While Loop

```
while (condition){  
  Do something  
}
```

#### Example

```
while (i < 5){  
  print(i)  
  i <- i + 1  
}
```

### For Loop

```
for (variable in sequence){  
  Do something  
}
```

#### Example

```
for (i in 1:4){  
  j <- i + 10  
  print(j)  
}
```



- Funções são um bloco de código organizado e reutilizável que é usado para realizar uma tarefa determinada em um *script* em R.
- Elas podem ser *built-in* ou *user defined*.
- Elas são armazenadas como objeto em R, podendo ser usadas como argumentos de outras funções ou definidas dentro de outras funções.
- O valor de retorno de uma função é a última expressão a ser calculada no corpo da função.

### Functions

```
function_name <- function(var){  
  Do something  
  return(new_variable)  
}
```

### Example

```
square <- function(x){  
  squared <- x*x  
  return(squared)  
}
```

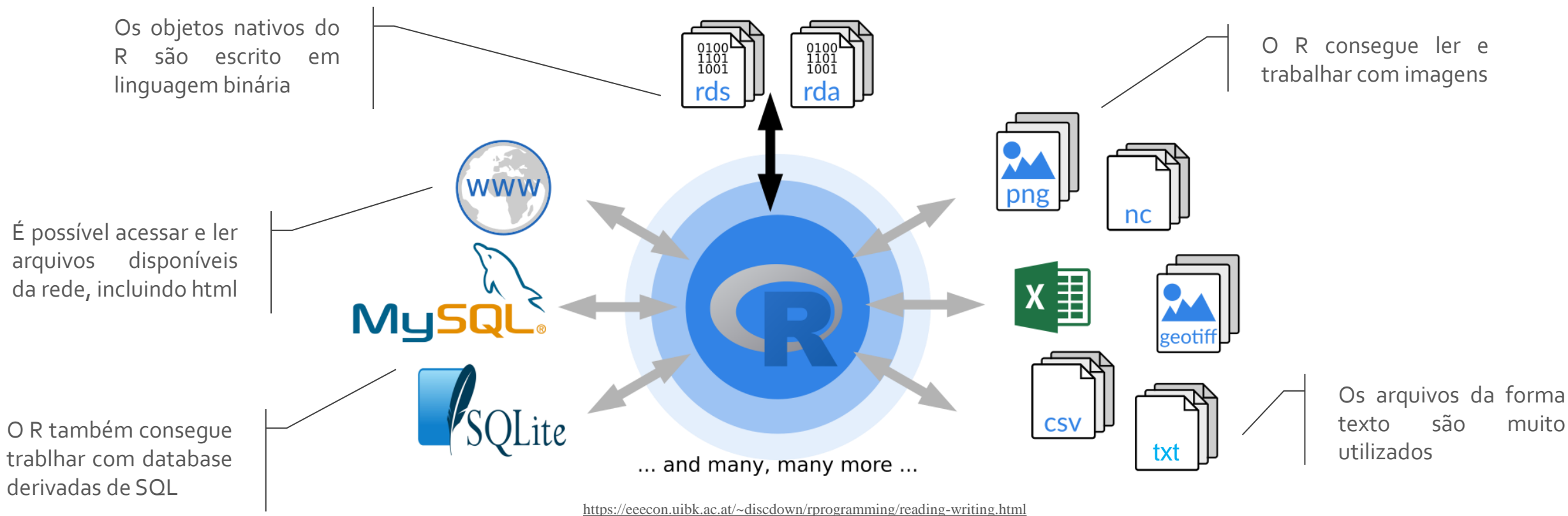
### Exemplo

```
> func <- function(a, b){  
+   a**2 + b  
+ }  
>  
> func(2,3)  
[1] 7  
>  
> func <- function(n, y = seq(0.05, 1, by = 0.05)){  
+  
+   n**y  
+ }  
>  
> func(2)  
[1] 1.035265 1.071773 1.109569 1.148698 1.189207 1.231144 1.274561  
[8] 1.319508 1.366040 1.414214 1.464086 1.515717 1.569168 1.624505  
[15] 1.681793 1.741101 1.802501 1.866066 1.931873 2.000000
```

# Introdução ao R

## Leitura e escrita de arquivos

- Uma das habilidades mais importantes de qualquer linguagem é a capacidade de se comunicar com o mundo exterior, i.e., importar e exportar dados.
- Isso inclui desde a leitura e escrita de arquivos simples de texto no HD, *download* e leitura de dado via Web, e outros formatos como xlsx, txt, SAS, Stata, SPSS, etc.



## Leitura e escrita de arquivos

- Em R existem diversas funções nativas capazes da leitura e escrita de dados tabulares e texto.
- Existem também funções próprias para escrever e ler objetos nativos em R, bem como bibliotecas para tipos de dados específicos.
- Para a maior parte das aplicações de modelagem os dados são no formato de texto e tabulares.

### Reading and Writing Data

Input	Output	Description
<code>df &lt;- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	Read and write a delimited text file.
<code>df &lt;- read.csv('file.csv')</code>	<code>write.csv(df, 'file.csv')</code>	Read and write a comma separated value file. This is a special case of read.table/write.table.
<code>load('file.RData')</code>	<code>save(df, file = 'file.Rdata')</code>	Read and write an R data file, a file type special for R.

### Exemplo

```
# Leitura de arquivo
data <- read.table('foo.txt')
data <- read.table('D:/datafiles/mydata.txt')
data <- read.csv('D:/datafiles/mydata.csv')

# Escrita de arquivo
x <- data.frame(a = 5, b = 10, c = pi)

write.table(x, file = 'data1.csv', sep = ',')
write.table(x, 'c:/mydata.txt', sep = '\t')
write.csv(x, file = 'data.csv')
```

# Prática no RStudio

## ...foco de hoje

- **CASE 1.1: Sintaxes e comandos básicos do R**

Comandos básicos, operadores, trabalhando com os formatos de dados, e criando alguns fluxos com *loops* e regras de condições

- **CASE 1.2: Lendo e escrevendo dados no R**

Leitura e escrita de um *dataframes* no R

