## Programação em R

Nesta aula você aprenderá os conceitos de programação R e como lidar com dados tabulares para desenvolver análises de dados. Você aprenderá sobre:

* Variáveis e Tipos de Dados em R
* Estruturas de Controle de Fluxo
* Organização de Dados em Tabelas (Tidy)
* Aquisição e Seleção de Dados em R (dataframes)

# Introdução ao R

Neste ponto é esperado que você já tenha familiaridade com alguma linguagem de programação como (Python, C++, Java) e editores de programação. Assim faremos uma introdução baseada em exemplos, assumindo que você já tem os conceitos básicos como variáveis, tipos de dados, atribuição, controles de fluxo em programas etc.

Todos os códigos aqui foram desenvolvidos e executados no ambiente Colab e você pode acessar diretamente os códigos em <https://github.com/Rogerio-mack/Analise_Exploratoria_de_Dados> para editar e executar os programas.

Lembre-se, a execução de notebooks R no Colab requer que você inicie o Colab com <http://colab.to/r> para abrir os arquivos .ipynb com código R. Você pode verificar se a sua sessão Colab está setada para executar programas R entrando na barra de ferramentas em Runtime > Change runtime type, na qual deverá aparecer a opção R como abaixo.

## Hello World! e R Version

Esse é o primeiro programa que aprendemos em qualquer linguagem e não poderia faltar aqui também.

print('Hello!')

[1] "Hello!"

Mas esse comando é o mesmo para Python e R. Se você estiver executando esse código no Google Colab pode ser interessante, então, executar o comando abaixo para verificar que você está empregando o Kernel de R e qual a versão de R empregada.

R.version

\_   
platform x86\_64-pc-linux-gnu   
arch x86\_64   
os linux-gnu   
system x86\_64, linux-gnu   
status   
major 4   
minor 1.1   
year 2021   
month 08   
day 10   
svn rev 80725   
language R   
version.string R version 4.1.1 (2021-08-10)  
nickname Kick Things

A versão aqui é a versão R version 4.1.1 (2021-08-10) e podemos ter certeza de estar executando um Kernel de R.

## Recursos e Help

Se você já tem alguma familiaridade com Linguagens de Programação sabe da importância de consultarmos guias de referência rápida, tutoriais e a documentação da linguagem para criação de programas. Aqui alguns recursos que serão bastante úteis para você nessa jornada de aprendizado do R.

* <https://cran.r-project.org/manuals.html>

A documentação do R é bastante extensa e útil em muitos casos. O uso do help inserindo ?<comando> ou help(<comando>), ou o nome parcial do comando, exibe no prompt da console a documentação do R incluindo vários exemplos. Essa opção, entretanto, é mais útil empregando o RStudio, local ou em nuvem. Para uso no Colab, embora você possa executar o comando, a exibição é bastante limitada e é preferível consultar a documentação no site.

* <https://www.w3schools.com/r/default.asp>

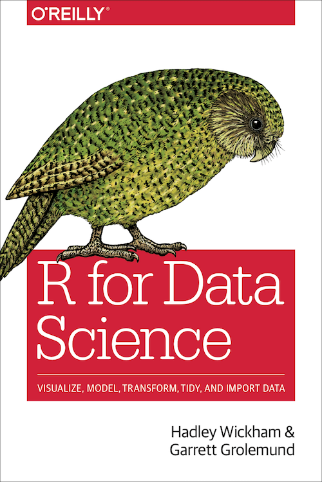
Este pode ser empregao como um guia de referência rápida. É bastante útil e prático e conta com a vantagem de podermos empregá-lo também para Python. Ele, entretanto, não cobre uma parte importante da linguagem e que trataremos aqui, os dataframes R.

* <http://www.r-tutor.com/>
* <https://www.statmethods.net/>

Esses são dois sites que você vai adorar! Eles apresentam os principais comandos em R voltados exatamente para exploração de dados. Eles são econômicos na linguagem, apresentam exemplos claros e tratam boa parte dos assuntos que iremos cobrir neste curso.

* Hadley Wickham, Garrett Grolemund, **R for Data Science: Import, Tidy, Transform, Visualize, and Model Data**.

Essa é uma referência obrigatória para quem quer aprender Análise de Dados com R e está disponível online com vários códigos exemplos. Entretanto é um texto mais avançado e em vários pontos emprega pacotes de dados mais elaborados como o tidyverse, dplyr e ggplot2 que não empregaremos aqui.



[Book](https://r4ds.had.co.nz/index.html) Acesse aqui (<https://r4ds.had.co.nz/index.html>)

## Aritmética com R

Em sua forma mais básica, R pode ser usado como uma calculadora simples empregando os operadores +, -, \*, /, ^ (potência) ou %% (módulo).

E você pode executar esses comandos na console:

# Uma adição  
5 + 3  
# Uma subtração  
5 - 3  
# Uma multiplicação  
3 \* 5  
# Uma divisão  
( 5 + 5 ) / 2

[1] 8

[1] 2

[1] 15

[1] 5

## Atribuições a uma Variável

Uma variável, como em outras linguagens, permite que você armazene um valor ou um objeto e empregar o nome dessa variável para acesso aos valores armazenados. Os valores são atribuídos com <-, ou **indiferentemente**, com = (a partir de todas as versões mais recentes de R). Prefira empregar o =.

# São equivalentes  
my\_var1 <- 4  
my\_var2 = 4

O R implementa, **tipos fracos** de dados, a exemplo do Python em que o tipo da variável é definido dinamicamente sem a necessidade de declaração da variável.

Os valores podem ser exibidos automaticamente a partir do nome da variável. Mas somente a última saída é exibida.

my\_var1  
my\_var2

[1] 4

[1] 4

Ou você pode optar pelo print dos valores.

print(my\_var1)  
print(my\_var2)

[1] 4  
[1] 4

Observe que o R é sensível a maiúsculas e minúsculas! Assim:

My\_var = 5  
my\_var = 4  
print(My\_var == my\_var)

[1] FALSE

## Tipos de dados básicos em R

Alguns dos tipos mais básicos são:

my\_numeric = 42.55  
  
my\_text = "algum texto"  
  
my\_caracter = 'a'  
  
my\_logical = TRUE # Boolean  
  
sprintf("Valores = %s , %s , %s , %d", my\_numeric, my\_text, my\_caracter, my\_logical)  
  
sprintf("Tipos = %s , %s , %s , %s", class(my\_numeric), class(my\_text), class(my\_caracter), class(my\_logical))

[1] "Valores = 42.55 , algum texto , a , 1"

[1] "Tipos = numeric , character , character , logical"

Aspas simples ou duplas são usadas indiferentemente. O comando sprintf permite a formatação da saída em C-Style e o comando class pode ser empregado para exibir a classe ou tipo de dado.

## Coleções: Vetores

Um tipo de dado essencial em R são os vetores. Lembre-se, o R é uma linguagem para trabalhar com coleções de dados, e a coleção mais simples são os vetores. Para criar esses vetores você pode empregar o comando c( ), que significa **combine**, e são estruturas bastante semelhantes as listas em Python (embora não possuam os mesmos métodos).

vetor\_numerico = c(1, 2, 3)  
  
vetor\_caracter = c("a", "b", "c")  
  
vetor\_mix = c(1, 'a', TRUE)  
  
cat('Tipos = ', class(vetor\_numerico), class(vetor\_caracter), class(vetor\_mix))

Tipos = numeric character character

Como você pode ver acima os vetores podem ter vários tipos de dados e assumem o tipo mais abrangente. O comando cat( ) 'Concatenate and Print' é empregado aqui para exibir os tipos de dados como alternativa ao print() e sprintf() usados anteriormente.

Como nos arrays de outras linguagens os vetores são listas ordenadas e indexadas e você pode acessar e alterar os valores referenciando o índice dos elementos.

cat('Antes: ', vetor\_numerico, '\n')  
vetor\_numerico[2] = 99  
cat('Depois: ', vetor\_numerico, '\n')  
  
for (i in 1:3) cat('\n', vetor\_numerico[i])

Antes: 1 2 3   
Depois: 1 99 3   
  
 1  
 99  
 3

Você deve notar que os índices começam por 1 (diferentemente de listas em Python que iniciam com 0), e abaixo você encontra algumas formas que serão úteis para inicializar vetores.

x = c(1,-2,3)  
y = seq(1,10,0.5)  
z = c(1:10)  
t = rep(c(1,2),each=3) # bem sofisticado aqui... :-)  
  
cat('x= ', x, '\ny= ', y, '\nz= ', z, '\nt= ', t)

x= 1 -2 3   
y= 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10   
z= 1 2 3 4 5 6 7 8 9 10   
t= 1 1 1 2 2 2

O R, muito mais orientado a trabalhar com coleções de dados que com variáveis simples, permite que uma série de funções sejam aplicadas diretamente às coleções sem a necessidade de iterações (loops), sendo assim mais próximo do comportamento de listas em Python do que o que seria como seria comum em arrays de C++ ou Java.

Assim, no lugar de:

x = seq(1,10,1)  
cat('Antes ', x)  
  
for (i in 1:10){ # C++ e Java arrays funcionam desse modo  
 x[i] = x[i] + 1  
}  
  
cat('\nDepois ', x)

Antes 1 2 3 4 5 6 7 8 9 10  
Depois 2 3 4 5 6 7 8 9 10 11

Você pode simplesmente escrever:

x = seq(1,10,1)  
cat('Antes ', x)  
  
x = x + 1 # em R a operação + é feita para todos elementos da coleção sem a necessidade das iterações  
  
cat('\nDepois ', x)

Antes 1 2 3 4 5 6 7 8 9 10  
Depois 2 3 4 5 6 7 8 9 10 11

E de modo análogo outras operações:

x = seq(1,10,1)  
cat('Antes ', x)  
  
cat('\nDepois... \n')  
  
print(2\*x)  
print(x+1)  
print(sum(x))  
print(max(x), min(x))  
print(sqrt(x))

Antes 1 2 3 4 5 6 7 8 9 10  
Depois...   
 [1] 2 4 6 8 10 12 14 16 18 20  
 [1] 2 3 4 5 6 7 8 9 10 11  
[1] 55  
[1] 10  
 [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427  
 [9] 3.000000 3.162278

O que se aplica também para funções lógicas:

x = seq(1,10,1)  
  
print(x > 5)

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

o que fornece uma forma bastante prática para você fazer a seleção de valores em uma coleção:

x = seq(1,10,1)  
  
L = x > 5 # vetor lógico selecionando os índices que correspondem a valores > 5  
  
print(x[L])

[1] 6 7 8 9 10

## Matrizes e Arrays

Matrizes e Arrays também podem ser criados. As matrizes são coleções de 2 dimensões, enquanto os arrays são generalizações de matrizes para mais (ou menos) de 2 dimensões. Mas essas estruturas serão pouco empregadas aqui, sendo muito mais frequente o uso de vetores.

X = matrix(c(1,2,3,4,5,6),nrow=2,ncol=3); X # Uma matriz 2x3  
  
X = array(c(1,2,3,4,5,6,7,8,9,10,1,1),dim=c(3,2,2)); X # Um array: 2 matrizes 3x2

[,1] [,2] [,3]  
[1,] 1 3 5   
[2,] 2 4 6

, , 1  
  
 [,1] [,2]  
[1,] 1 4  
[2,] 2 5  
[3,] 3 6  
  
, , 2  
  
 [,1] [,2]  
[1,] 7 10  
[2,] 8 1  
[3,] 9 1

## Fatores

Para lidar com medições nas escalas nominais e ordinais o R fornece vetores de tipo fator. Um fator é uma variável que pode assumir apenas um número discreto (finito) de valores distintos (os levels). Para criar um fator você pode aplicar a função factor a um vetor de qualquer classe:

# gender com 5 "male" e 10 "female"  
gender = c(rep("male",5), rep("female", 10))  
  
sprintf(gender, '\n', class(gender) , '\n') # como vetor  
  
gender = factor(gender) # como factor  
sprintf(as.character(gender), '\n', class(gender) , '\n')  
  
# factor faz com que os valores sejam armazenados internamente 1=female, 2=male internally (ordem alfabética) e o R tratará gender como uma variável nominal e não como caracteres!  
  
cat(gender, '\n')  
summary(gender)

Warning message in sprintf(gender, "\n", class(gender), "\n"):  
“3 arguments not used by format”

[1] "male" "male" "male" "male" "male" "female" "female" "female"  
 [9] "female" "female" "female" "female" "female" "female" "female"

Warning message in sprintf(as.character(gender), "\n", class(gender), "\n"):  
“3 arguments not used by format”

[1] "male" "male" "male" "male" "male" "female" "female" "female"  
 [9] "female" "female" "female" "female" "female" "female" "female"

2 2 2 2 2 1 1 1 1 1 1 1 1 1 1

female male   
 10 5

O uso de fatores envolve uma série detalhes e por hora é suficiente você saber que existem essas coleções que criam valores nominais e ordinais.

rating = c(rep("large",1), rep("medium",2), rep("small",3))  
rating = ordered(rating)  
print(rating)

[1] large medium medium small small small   
Levels: large < medium < small

## Caracteres

Vale a pena também você olhar algumas funções úteis para a manipulação de caracteres em R como a conversão para caracteres, a concatenação de strings e substrings:

x = as.character(3.14) # converte para caracteres  
cat(x, class(x), '\n')  
  
y = paste('Universidade','Presbiteriana','Mackenzie') # concatena strings  
print(y)  
  
print( substr (y, start = 14, stop = 26) ) # substring

3.14 character   
[1] "Universidade Presbiteriana Mackenzie"  
[1] "Presbiteriana"

## Input pelo teclado

nome = readline(prompt="Entre com seu nome: ")  
idade = as.numeric(readline(prompt="Entre com sua idade: "))  
  
cat('nome: ', nome, 'idade: ', idade, '\n')

Entre com seu nome: Adriana  
Entre com sua idade: 1984  
nome: Adriana idade: 1984

## Estruturas de Controle de Fluxo

Estruturas e controle de fluxo como for, while, if são bastante análogas às outras linguagens inclusive na sintaxe:

if(cond) expr  
if(cond) cons.expr else alt.expr  
  
for(var in seq) expr  
  
while(cond) expr  
  
repeat expr  
  
break  
next

Você pode ver alguns exemplos abaixo, mas vamos deixar para você explorar melhor esses comandos sugerindo que você comece pelo help('for').

for(i in 1:5) print(1:i)  
  
for(n in c(2,5,10,20,50)) {  
 x = stats::rnorm(n)  
 cat(n, ": ", sum(x^2), "\n", sep = "")  
}  
  
f = factor(sample(letters[1:5], 10, replace = TRUE))  
for(i in unique(f)) print(i)  
  
a = 5; b = 6  
if (a < b) print('menor') else print('maior')

[1] 1  
[1] 1 2  
[1] 1 2 3  
[1] 1 2 3 4  
[1] 1 2 3 4 5  
2: 1.256652  
5: 6.95564  
10: 11.0504  
20: 14.42704  
50: 51.29902  
[1] "e"  
[1] "d"  
[1] "a"  
[1] "menor"

E você pode empregar {} para definir blocos de instruções que são executadas em conjunto.

## Pacotes

Existem muitos pacotes que implementam funcionalidades adicionais ao R e você empregará alguns deles ao longo do curso. Você deve empregar o comando library() para 'importar' as funcionalidades de uma biblioteca já instalada. Caso o pacote ainda não tenha sido instalado você deverá usar o install.packages(), mas este último comando, em geral, precisa ser feito uma única vez no seu ambiente.

# install.packages('ggplot2') # comentado aqui  
library(ggplot2)

O comando acima instala e importa um pacote bastante popular para gráficos e visualização de dados em R.

## Outros recursos

O R possui ainda muito mais recursos do quê você aprendeu aqui, como recursos para orientação a objetos, suporte a operações em arquivos etc. Mas o aprendizado de uma linguagem, assim como aprender uma língua, não pode ser feito de uma única vez e requer o uso e a prática, e aqui você tem mais do que o suficiente para prosseguir no uso do R. É também importante você notar que nosso propósito é o de empregar R para a Análise de Dados e, sendo um propósito específico, alguns aspectos da linguagem não serão aqui de interesse.

# Organização dos Dados

É importante ao trabalharmos com dados empregar uma forma consistente de organização dos dados. Colocar os dados neste formato pode requerer um grande trabalho, mas é essencial para desenvolvermos correta Análise dos Dados.

Aqui nós iremos trabalhar essencialmente com dados tabulares, isto é, no formato de tabelas. Mesmo se pensarmos unicamente em tabelas existem várias formas de organizarmos um mesmo conjunto de dados.

Por exemplo, as tabelas abaixo trazem 4 diferentes formas de organizar os mesmos dados de país, população, ano e número de casos de Tuberculose documentados pela Organização Mundial de Saúde para o Afeganistão, Brasil e China entre os anos de 1999 e 2000.

library(tidyverse)

Warning message in system("timedatectl", intern = TRUE):  
“running command 'timedatectl' had status 1”  
── [1mAttaching packages[22m ─────────────────────────────────────── tidyverse 1.3.1 ──  
  
[32m✔[39m [34mtibble [39m 3.1.3 [32m✔[39m [34mdplyr [39m 1.0.7  
[32m✔[39m [34mtidyr [39m 1.1.3 [32m✔[39m [34mstringr[39m 1.4.0  
[32m✔[39m [34mreadr [39m 2.0.1 [32m✔[39m [34mforcats[39m 0.5.1  
[32m✔[39m [34mpurrr [39m 0.3.4   
  
── [1mConflicts[22m ────────────────────────────────────────── tidyverse\_conflicts() ──  
[31m✖[39m [34mdplyr[39m::[32mfilter()[39m masks [34mstats[39m::filter()  
[31m✖[39m [34mdplyr[39m::[32mlag()[39m masks [34mstats[39m::lag()

table1

country year cases population  
1 Afghanistan 1999 745 19987071  
2 Afghanistan 2000 2666 20595360  
3 Brazil 1999 37737 172006362  
4 Brazil 2000 80488 174504898  
5 China 1999 212258 1272915272  
6 China 2000 213766 1280428583

table2

country year type count   
1 Afghanistan 1999 cases 745  
2 Afghanistan 1999 population 19987071  
3 Afghanistan 2000 cases 2666  
4 Afghanistan 2000 population 20595360  
5 Brazil 1999 cases 37737  
6 Brazil 1999 population 172006362  
7 Brazil 2000 cases 80488  
8 Brazil 2000 population 174504898  
9 China 1999 cases 212258  
10 China 1999 population 1272915272  
11 China 2000 cases 213766  
12 China 2000 population 1280428583

table3

country year rate   
1 Afghanistan 1999 745/19987071   
2 Afghanistan 2000 2666/20595360   
3 Brazil 1999 37737/172006362   
4 Brazil 2000 80488/174504898   
5 China 1999 212258/1272915272  
6 China 2000 213766/1280428583

table4a  
table4b

country 1999 2000   
1 Afghanistan 745 2666  
2 Brazil 37737 80488  
3 China 212258 213766

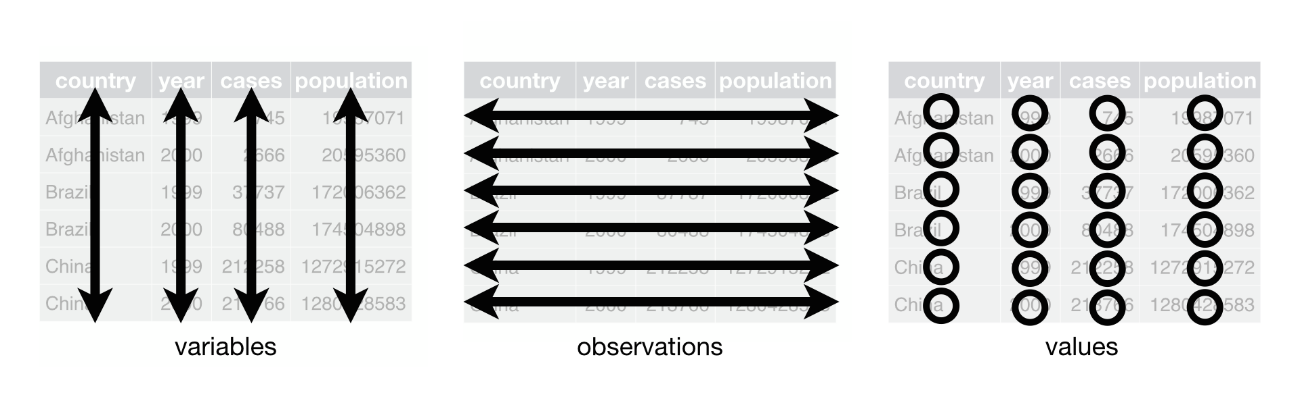
country 1999 2000   
1 Afghanistan 19987071 20595360  
2 Brazil 172006362 174504898  
3 China 1272915272 1280428583

São os mesmos dados, mas nem todas as representações, ainda que no formato de tabelas, são igualmente fáceis de usar.

A ideia é que um conjunto de dados organizado em um formato Tidy será muito mais fácil de trabalhar. Um formato de tabela Tidy é construído de forma a atender 3 regras:

1. **Cada variável deve ter sua própria coluna.**
2. **Cada observação, caso ou instância de dados, deve ter sua própria linha.**
3. **Cada valor deve ter sua própria célula.**

representadas abaixo.



Uma forma mais formal de definirmos esse formato poderia ser *um conjunto multivariado de dados está na forma de uma matriz de dados (Tidy) se linhas correspondem aos indivíduos da amostra e as colunas às variáveis, de modo que a entrada na i-ésima linha e j-ésima coluna dá o valor da j-ésima variável conforme medido ou observado no i-ésimo indivíduo.* Mas realmente o formato Tidy aparece forma bastante natural para um grande número de conjuntos de dados e é suficiente verificarmos as 3 regras acima.

Dos conjuntos de dados do exemplo anterior você poderá verificar que somente table1 atende as 3 regras e, portanto, organiza adequadamente os dados:

1. Cada variável deve ter sua própria coluna: country, year, cases e population.
2. Cada observação deve ter sua própria linha: de fato a população e casos observados aparece para cada país e ano em uma única linha da tabela.
3. Cada valor deve ter sua própria célula: cada valor encontra-se em uma única célula, linha coluna.

table1

country year cases population  
1 Afghanistan 1999 745 19987071  
2 Afghanistan 2000 2666 20595360  
3 Brazil 1999 37737 172006362  
4 Brazil 2000 80488 174504898  
5 China 1999 212258 1272915272  
6 China 2000 213766 1280428583

Essa organização de dados encontra uma representação própria em R conhecida como dataframe e que deu origem a outras estrutura dados semelhantes como os dataframes do pacote Pandas do Python.

# Dataframes em R

Os dataframes em R (a função é data.frame) são o formato de dados mais importante e empregado do R para Análise de Dados tabulares. Nele cada coluna pode então representar uma variável e cada linha uma observação respeitando o formato Tidy dos dados que vimos acima. Os dataframes em R permitem importar dados tabulares de vários formatos como .csv, excel, json, sql, SAS, SPSS etc. Aqui vamos nos concentrar em empregar dados *built-in* do R e seus pacotes (dados que já constam da base do pacote R ou de outros pacotes como as tabelas do pacote tidyverse que empregamos acima) ou dados em formato .csv, e você pode consultar a documentação do R se estiver interessado em explorar dados em outros formatos.

## Criando data.frame

Um Dataframe (vamos empregar aqui o nome no lugar da função R) pode ser criado explicitamente como abaixo e isso é útil para criarmos novos conjuntos de dados que podem ser então gravados e exportados para os mesmos formatos que citamos antes (.csv, excel, json, sql, SAS, SPSS etc.).

nr = c(1, 2, 3)  
nome = c("Ana", "Adriana", "Daniel")  
estudante = c(TRUE, FALSE, TRUE)  
idade = c(15, 18, 17)  
df = data.frame (nr, nome, estudante, idade)  
  
print(df)

nr nome estudante idade  
1 1 Ana TRUE 15  
2 2 Adriana FALSE 18  
3 3 Daniel TRUE 17

### Vetor de Coluna

De modo semelhante ao SQL você pode referenciar os atributos (as colunas) a partir do nome da coluna como vetores.

print(df$nome)  
print(df$idade)

[1] "Ana" "Adriana" "Daniel"   
[1] 15 18 17

Note que em R empregamos o símbolo '$' referenciar o atributo no lugar de '.', como é empregado em geral para fazer referências de objetos.

Cada atributo sendo um vetor você pode aplicar todas as funções que são aplicáveis a vetores para esses atributos, e você pode ver algumas das funções que empregamos anteriormente aqui:

sprintf('Idade mínima: %d', min(df$idade))  
sprintf('Idade média: %.2f', mean(df$idade))  
  
print('Antes: ')  
print(df)  
  
# Somando 1 na idade  
df$idade = df$idade + 1  
  
print('Depois: ')  
print(df)

[1] "Idade mínima: 15"

[1] "Idade média: 16.67"

[1] "Antes: "  
 nr nome estudante idade  
1 1 Ana TRUE 15  
2 2 Adriana FALSE 18  
3 3 Daniel TRUE 17  
[1] "Depois: "  
 nr nome estudante idade  
1 1 Ana TRUE 16  
2 2 Adriana FALSE 19  
3 3 Daniel TRUE 18

Ou ainda operações lógicas:

print(df$idade >= 18)

[1] FALSE TRUE TRUE

e você verá como empregar isso abaixo para a seleção de linhas de uma tabela.

Esse é foi exemplo bastante simples. Na maior parte dos casos aqui não iremos construir conjuntos de dados, mas estaremos interessados em acessar um conjunto de dados, por exemplo os dados sobre casos de Tuberculose nos países ou dados com características e preço de Veículos, sobre os quais estaremos interessados na análise.

## Exploração Inicial da Estrutura dos Dados

Nos exemplos a seguir vamos empregar o mtcars um dos [datasets built-in](http://stat.ethz.ch/R-manual/R-patched/library/datasets/html/00Index.html) (<http://stat.ethz.ch/R-manual/R-patched/library/datasets/html/00Index.html>) do R. Mais adiante você aprenderá a importar dados de fontes próprias.

Antes de você começar a trabalhar qualquer conjunto de dados é importante que você conheça os dados que tem em mãos.

Os comandos abaixo permitem explorar a estrutura básica de um dataset, como o número de linhas, colunas, tipos de dados etc. e você provavelmente começará uma exploração de dados por esses comandos para entender a estrutura dos dados em mãos antes de selecionar e analisar de fato os dados.

Você pode executar cada um dos comandos em separado para exibir suas saídas.

# execute cada um dos comandos isoladamente  
head(mtcars) # cabeçalho do dataframe  
tail(mtcars) # final do dataframe  
dim(mtcars) # Dimensões  
ncol(mtcars) # nr de colunas  
nrow(mtcars) # nr de linhas  
names(mtcars) # nome das colunas  
rownames(mtcars) # nome das linhas  
str(mtcars) # Estrutura dos dados, como exibido no RStudio na janela superior direita  
summary(mtcars) # Summary statistics

mpg cyl disp hp drat wt qsec vs am gear carb  
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4   
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4   
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1   
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1   
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2   
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

mpg cyl disp hp drat wt qsec vs am gear carb  
Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.7 0 1 5 2   
Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.9 1 1 5 2   
Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.5 0 1 5 4   
Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.5 0 1 5 6   
Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.6 0 1 5 8   
Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.6 1 1 4 2

[1] 32 11

[1] 11

[1] 32

[1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"  
[11] "carb"

[1] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710"   
 [4] "Hornet 4 Drive" "Hornet Sportabout" "Valiant"   
 [7] "Duster 360" "Merc 240D" "Merc 230"   
[10] "Merc 280" "Merc 280C" "Merc 450SE"   
[13] "Merc 450SL" "Merc 450SLC" "Cadillac Fleetwood"   
[16] "Lincoln Continental" "Chrysler Imperial" "Fiat 128"   
[19] "Honda Civic" "Toyota Corolla" "Toyota Corona"   
[22] "Dodge Challenger" "AMC Javelin" "Camaro Z28"   
[25] "Pontiac Firebird" "Fiat X1-9" "Porsche 914-2"   
[28] "Lotus Europa" "Ford Pantera L" "Ferrari Dino"   
[31] "Maserati Bora" "Volvo 142E"

'data.frame': 32 obs. of 11 variables:  
 $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
 $ disp: num 160 160 108 258 360 ...  
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
 $ qsec: num 16.5 17 18.6 19.4 17 ...  
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
 $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...

mpg cyl disp hp   
 Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0   
 1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5   
 Median :19.20 Median :6.000 Median :196.3 Median :123.0   
 Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7   
 3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0   
 Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0   
 drat wt qsec vs   
 Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000   
 1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000   
 Median :3.695 Median :3.325 Median :17.71 Median :0.0000   
 Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375   
 3rd Qu.:3.920 3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000   
 Max. :4.930 Max. :5.424 Max. :22.90 Max. :1.0000   
 am gear carb   
 Min. :0.0000 Min. :3.000 Min. :1.000   
 1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:2.000   
 Median :0.0000 Median :4.000 Median :2.000   
 Mean :0.4062 Mean :3.688 Mean :2.812   
 3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:4.000   
 Max. :1.0000 Max. :5.000 Max. :8.000

Depois de analisar essas informações você pode então prosseguir com outras análises de mais refinadas e de interesse.

Um aspecto importante dos dados é sobre seu significado, a *semântica* dos dados. Nos dados acima o significado de cada atributo parece bastante óbvio para maior parte dos atributos. Mas o que seria o atributo wt? Em que unidade é medido? Você pode então buscar na documentação <https://stat.ethz.ch/R-manual/R-patched/library/datasets/html/mtcars.html> a descrição de cada atributo e verificar que wt é peso do veículo (Weight) e encontra-se em número de (1000 lbs) ou ainda descobrir que am refere-se ao veículo possuir ou não transmissão automática (0 = automatic, 1 = manual). Esses dados precisam em geral ser fornecidos para você, não podem ser obtidos por análise ou inspeção (exceto para casos muito óbvios) e por isso sempre vamos assumir que temos acesso a essas informações para procedermos as análises. De qualquer modo, em casos reais de Análise de Dados, é importante que você começe entendendo o significado de cada dado ou informação.

## Selecionando Dados

Seleções dos dados são bastante importantes. Você nem sempre estará interessado em todos os dados. Por exemplo você pode ter dados de produção de várias unidades de uma fábrica, mas estar interessado somente em dados das unidades de São Paulo (seleção de linhas ou casos). Ou você pode ter dados de vendas com diversas informações dos produtos (cor, modelo etc.) e dos clientes (nome, CPF etc.) e querer apenas dados de peso e dimensões do produto, e da origem e destino da compra para analisar os preços de frete (seleção de colunas ou atributos). Mais frequentemente ainda você vai realizar as duas seleções criando *slices* dos dados.

De um modo geral estaremos interessados em fazer 3 tipos de seleções de dados:

**Seleção de Colunas, Seleção de Linhas e de Linhas e Colunas**



Sendo uma estrutura de duas dimensões os dados de um dataframe podem ser referenciados do seguinte modo:

**dataframe [ linhas , colunas ]**

Onde linhas e colunas podem conter uma ou mais posições (uma seleção de dados). É comum também nos referirmos à seleção de dados como 'slice' (fatias) de dados.

### Seleção de Colunas ou Atributos



A seleção de colunas pode ser feita referenciando o nome ou a posição dos atributos. E você deve notar a vírgula indicando que, neste caso, não há nenhum filtro para linhas. Você deve executar isoladamente cada um dos comandos abaixo para exibir as saídas. As 3 seleções de mtcars produzem a mesma seleção de atributos.

# execute cada um dos comandos isoladamente  
print(names(mtcars))  
mtcars[ , c('cyl','disp','hp')]  
mtcars[ , c(2,3,4)]  
mtcars[ , c(2:4)]

[1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"  
[11] "carb"

cyl disp hp   
Mazda RX4 6 160.0 110  
Mazda RX4 Wag 6 160.0 110  
Datsun 710 4 108.0 93  
...

Ferrari Dino 6 145.0 175  
Maserati Bora 8 301.0 335  
Volvo 142E 4 121.0 109

cyl disp hp   
Mazda RX4 6 160.0 110  
Mazda RX4 Wag 6 160.0 110  
Datsun 710 4 108.0 93  
...

Ferrari Dino 6 145.0 175  
Maserati Bora 8 301.0 335  
Volvo 142E 4 121.0 109

cyl disp hp   
Mazda RX4 6 160.0 110  
Mazda RX4 Wag 6 160.0 110  
Datsun 710 4 108.0 93  
...

Ferrari Dino 6 145.0 175  
Maserati Bora 8 301.0 335  
Volvo 142E 4 121.0 109

Um caso especial de bastante interesse é a seleção de um único atributo. Neste caso você pode preferir empregar a seguinte notação:

mtcars$mpg

[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4  
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7  
[31] 15.0 21.4

O que apresenta o mesmo resultado dos comandos:

mtcars[,1]  
mtcars[,c(1)]  
mtcars[,c('mpg')]

[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4  
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7  
[31] 15.0 21.4

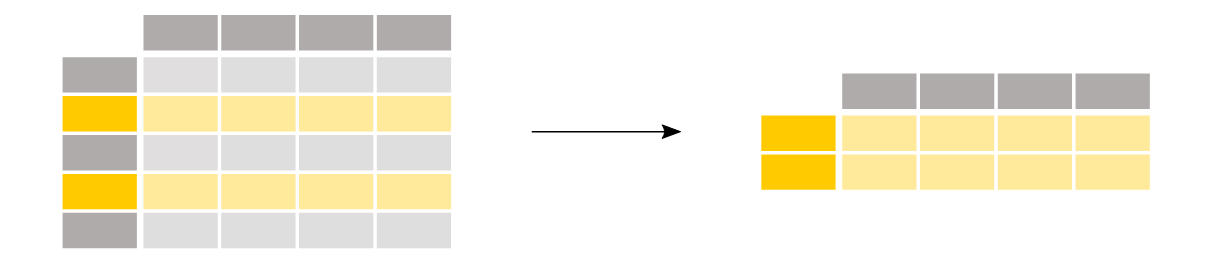
mas a notação com '$' é bastante útil pode ser empregada mais facilmente com inúmeras funções:

mean(mtcars$mpg)  
summary(mtcars$mpg)

[1] 20.09062

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 10.40 15.43 19.20 20.09 22.80 33.90

### Seleção de Linhas ou Casos



A seleção de linhas é mais interessante por que podemos especificar condições para os valores que buscamos. Por exemplo, você poderia indicar o tipo de peças que deseja ver em um DataFrame com dados de vários componentes, ou indicar a cidade das unidades de fábrica que você tem os dados de produção.

A seleção de linhas pode ser feita indicando-se a posição da linha ou, o que é mais interessante, um predicado de seleção.

# execute cada um dos comandos isoladamente  
mtcars[ 1:6, ] # equivalente a head(mtcars)  
mtcars[ mtcars$cyl > 6, ] # exibe somente veículos com mais de 6 cilindros

mpg cyl disp hp drat wt qsec vs am gear carb  
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4   
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4   
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1   
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1   
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2   
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

mpg cyl disp hp drat wt qsec vs am gear carb  
Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2   
Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4   
Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3   
Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3   
Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3   
Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4   
Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4   
Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4   
Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2   
AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2   
Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4   
Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2   
Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4   
Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8

e os predicados ainda podem ser combinados com operadores lógicos como & (e) e | (ou):

mtcars[ mtcars$cyl <= 4 & mtcars$hp > 100, ]

mpg cyl disp hp drat wt qsec vs am gear carb  
Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.9 1 1 5 2   
Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.6 1 1 4 2

A operação acima também pode ser reescrita empregando-se um vetor lógico:

L = mtcars$cyl <= 4 & mtcars$hp > 100  
L  
mtcars[ L , ]

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[25] FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE

mpg cyl disp hp drat wt qsec vs am gear carb  
Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.9 1 1 5 2   
Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.6 1 1 4 2

sendo exibidos somente as linhas com valores TRUE (surpreenda-se e experimente também executar mtcars[ -L , ]).

### Seleção de Linhas e Colunas



O uso mais geral é quando fazemos seleções de linhas e colunas dos dados, e às vezes nos referimos a esse subconjunto dos Dados de *Slice* (Fatia dos Dados).

Para isso basta combinar a seleção de linhas e colunas para trazer quaisquer 'fatias' de seu interesse dos dados.

# execute cada um dos comandos isoladamente  
mtcars[ mtcars$cyl > 6 & mtcars$am == 1, c('cyl','mpg','am') ] # exibe somente veículos com mais de 6 cilindros e transmissão am

cyl mpg am  
Ford Pantera L 8 15.8 1   
Maserati Bora 8 15.0 1

Ou ainda selecionar um único vetor de dados desse modo:

# execute cada um dos comandos isoladamente  
mtcars[ mtcars$cyl > 6 & mtcars$am == 1, ]$hp # exibe somente o vetor de valores hp para veículos com mais de 6 cilindros e transmissão am

[1] 264 335

Como a seleção de dados de um dataframe retorna também um dataframe, você pode empregar a seleção para produzir dataframes próprios com os seus dados de interesse.

# execute cada um dos comandos isoladamente  
  
my\_mtcars = mtcars[ mtcars$cyl > 6 & mtcars$am == 1, c('cyl','mpg','am') ]  
  
head(my\_mtcars)  
nrow(my\_mtcars)  
summary(my\_mtcars)  
  
print( my\_mtcars[ my\_mtcars$mpg > 15, ]) # todos os comandos aplicáveis a um dataframe são aplicáveis em my\_mtcars

cyl mpg am  
Ford Pantera L 8 15.8 1   
Maserati Bora 8 15.0 1

[1] 2

cyl mpg am   
 Min. :8 Min. :15.0 Min. :1   
 1st Qu.:8 1st Qu.:15.2 1st Qu.:1   
 Median :8 Median :15.4 Median :1   
 Mean :8 Mean :15.4 Mean :1   
 3rd Qu.:8 3rd Qu.:15.6 3rd Qu.:1   
 Max. :8 Max. :15.8 Max. :1

cyl mpg am  
Ford Pantera L 8 15.8 1

e as duas operações abaixo trazem os mesmos resultados.

my\_mtcars = mtcars[ mtcars$cyl > 6 & mtcars$am == 1, c('cyl','mpg','am') ]

## Importando um dataframe

O R permite importar dataframes de diferentes fontes de dados como bases SQL, arquivos .xlsx, .csv, .json etc. sendo necessário para alguns tipos de dados pacotes específicos para as funções e por hora vamos nos limitar a empregar arquivos com extensão .csv.

Você poderá fazer acesso a eles indicando uma url de origem dos dados um diretório local na sua máquina. Se estiver empregando o RStudio.Cloud, lembre-se que sua máquina R encontra-se na nuvem e para o acesso 'local' é necessário o upload do arquivo.

surveys = read.csv("https://github.com/csc-training/da-with-r/raw/master/DataFiles/portal\_data\_joined.csv")  
  
head(surveys)  
summary(surveys)

record\_id month day year plot\_id species\_id sex hindfoot\_length weight  
1 1 7 16 1977 2 NL M 32 NA   
2 72 8 19 1977 2 NL M 31 NA   
3 224 9 13 1977 2 NL NA NA   
4 266 10 16 1977 2 NL NA NA   
5 349 11 12 1977 2 NL NA NA   
6 363 11 12 1977 2 NL NA NA   
 genus species taxa plot\_type  
1 Neotoma albigula Rodent Control   
2 Neotoma albigula Rodent Control   
3 Neotoma albigula Rodent Control   
4 Neotoma albigula Rodent Control   
5 Neotoma albigula Rodent Control   
6 Neotoma albigula Rodent Control

record\_id month day year plot\_id   
 Min. : 1 Min. : 1.000 Min. : 1.0 Min. :1977 Min. : 1.00   
 1st Qu.: 8964 1st Qu.: 4.000 1st Qu.: 9.0 1st Qu.:1984 1st Qu.: 5.00   
 Median :17762 Median : 6.000 Median :16.0 Median :1990 Median :11.00   
 Mean :17804 Mean : 6.474 Mean :16.1 Mean :1990 Mean :11.34   
 3rd Qu.:26655 3rd Qu.:10.000 3rd Qu.:23.0 3rd Qu.:1997 3rd Qu.:17.00   
 Max. :35548 Max. :12.000 Max. :31.0 Max. :2002 Max. :24.00   
   
 species\_id sex hindfoot\_length weight   
 Length:34786 Length:34786 Min. : 2.00 Min. : 4.00   
 Class :character Class :character 1st Qu.:21.00 1st Qu.: 20.00   
 Mode :character Mode :character Median :32.00 Median : 37.00   
 Mean :29.29 Mean : 42.67   
 3rd Qu.:36.00 3rd Qu.: 48.00   
 Max. :70.00 Max. :280.00   
 NA's :3348 NA's :2503   
 genus species taxa plot\_type   
 Length:34786 Length:34786 Length:34786 Length:34786   
 Class :character Class :character Class :character Class :character   
 Mode :character Mode :character Mode :character Mode :character

# Para Saber Mais

1. Leia o artigo Wickham, H. (2014). **Tidy Data**. Journal of Statistical Software, 59(10), 1 - 23. e entenda mais sobre a organização dos dados. O texto completo está disponível em: <https://www.jstatsoft.org/article/view/v059i10>.
2. Acesse a página [datasets built-in](http://stat.ethz.ch/R-manual/R-patched/library/datasets/html/00Index.html) (<http://stat.ethz.ch/R-manual/R-patched/library/datasets/html/00Index.html>) e explore outros conjuntos de dados nativos do pacote R.
3. Não deixe de explorar a documentação do R em <https://cran.r-project.org/manuals.html> e a parte de Introdução do R Tutorial em <http://www.r-tutor.com/r-introduction>.

# Referências

1. Hadley Wickham, Garrett Grolemund. **R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.** O'Reilly Media Inc. (2017). (alternativamente disponível em: <https://r4ds.had.co.nz/>).
2. Wickham, H. (2014). **Tidy Data**. Journal of Statistical Software, 59(10), 1 - 23. <doi:http://dx.doi.org/10.18637/jss.v059.i10> Disponível em: <https://www.jstatsoft.org/article/view/v059i10> Acesso em: 29 de Agosto de 2021.
3. \_\_. **The R Manuals.** Disponível em: <https://cran.r-project.org/manuals.html> Acesso em: 29 de Agosto de 2021.