# Curso de Formação Executiva em Big Data e Data - Science - Introdução ao R

Fundação Getúlio Vargas

*Anna Carolina dos Santos Barros*

# Contents


**1) Welcome** **4**
1.1) O R . . . 4
1.2) O R Studio . . . 4

**2) Bases** **4**

**3) Primeiros passos com R e RStudio** **5**
3.1) Instalação do R . . . 5
3.2) Instalação do RSudio . . . 5
3.3) Conhecendo o RStudio . . . 5
3.4) Scripts . . . 7
3.4.1) Criando um Script . . . 7
3.4.2) Salvando um Script Existente . . . 8
3.4.3) Executando um Script . . . 8

**4) Primeiros Passos com o R** **8**
4.1) Operações Matemáticas . . . 8
4.2) Operações Lógicas . . . 9

**5) Objetos** **10**
5.1) Classes de objetos . . . 10
5.1.1) Númerico . . . 10
5.1.2) Character . . . 11
5.1.3) Data . . . 11
5.1.4) Lógica . . . 12
5.2) Conhecendo a classe de um objeto . . . 12

**6) Vetores** **12**
6.1) Indexação de Vetores . . . 13
6.2) Tamanho do Vetor . . . 13
6.3) Concatenando vetores . . . 13
Funções úteis . . . 14
6.4) Exercícios . . . 14

**7) Data Frames** **15**
7.1) Indexação de Data Frames . . . 15
7.2) Acessando colunas em data frame . . . 16
7.3) Criando uma nova coluna em uma data frame . . . 16
7.4) Conhecendo seu Data Frame . . . 17
Funções Úteis . . . 18
7.5) EXERCÍCIOS . . . 18

**8) Listas** **19**
8.1) Acessando os elementos em uma lista . . . 20
8.3) Adicionando Objetos em uma Lista . . . 20
8.4) Conhecendo a sua lista . . . 21
8.5) Exercícios . . . 21

**9) Trabalhando com Diretórios e Arquivos Externos** **21**
9.1) Conhecendo seu diretório de arquivos . . . 21
9.2) Mudando seu diretório de trabalho . . . 22
9.3) Leitura de Arquivos Externos . . . 22
9.4) Exportação de Arquivos . . . 24

9.5) Leitura de arquivos usando pacotes externos . . . 25
9.5.1) Leitura de arquivos no formato .xlsx . . . 25
9.5.2) Exportação de arquivos no formato .xlsx . . . 26

**10) Estruturas de Condição e Repetição** **26**
10.1) Estruturas de Condição . . . 26
10.1.1) Estrutura IF . . . 27
10.1.2) Estrutura If e Else . . . 27
10.1.3) Estrutura Ifelse . . . 27
10.2) Estrutura de Repetição . . . 28
10.2.1) Estrutura While . . . 28
10.2.2) Estrutura For . . . 29

**11) Funções** **31**
11.1) Exercícios . . . 32

**12) A função Which** **32**

**13) Exercícios** **34**
Base train.csv . . . 34
Base Human development index (HDI).csv . . . 34
Base dados_anp2.csv . . . 35

**References** **35**

# 1) Welcome

## 1.1) O R

O R não é apenas uma linguagem de programação, é também um ambiente integrado de desenvolvimento. É um projetoGNU,que foi desenvolvido em Bell Laboratories (antiga AT & T, agora Lucent Technologies) por John Chambers e seus colegas.(Team 2017)

De acordo co seus criadores (Team 2017) o R é um conjunto integrado de instalações de software para manipulação de dados, cálculo e exibição gráfica, e esse ambiente inclui:

- Uma instalação eficaz de tratamento e armazenamento de dados,
- Um conjunto de operadores para cálculos em matrizes, em particular matrizes,
- Uma grande coleção coerente e integrada de ferramentas intermediárias para análise de dados,
- Instalações gráficas para análise de dados e exibição na tela ou em
- Uma linguagem de programação bem desenvolvida, simples e eficaz que inclui condicionais, loops, funções recursivas definidas pelo usuário e instalações de entrada e saída.

Além disso o R conta com uma ampla comunidade acadêmica que oferece suporte e ajuda aoas usuários (CRAN) além de oferecer um extensão às funções usuais do R, com os pacotes.

Isso faz do R é um projeto "aberto",o que signica ser continuamente melhorado,atualizado e expandido pela comunidade global de desenvolvedor e seus uários incrivelmente apaixonados.(Bowles 2015)

## 1.2) O R Studio

O RStudio é um ambiente de desenvolvimento integrado (IDE) para R. Nele estão inclusos ferramentas que facilitam o uso da linguagem R, como um console e uma janela que suporta a execução do código direto, e instrumentos que auxiliam na análise histórica do script e debugação de possíveis erros.

O RStudio está disponível em open source e edições comerciais e é executado no desktop (Windows, Mac e Linux) ou em um navegador conectado ao RStudio Server ou ao RStudio Server Pro (Debian / Ubuntu, RedHat / CentOS e SUSE Linux).

# 2) Bases

Neste primeiro módulo (Introdução ao R) faremos uso de alguns arquivos externos que estão disponíveis na pasta *datasets* no nosso diretório no dropbox.

- `populacao.txt`,`populacao.csv` e `populacao.xlsx`: São todos a mesma base, porém em formatos diferentes. Trata-se dados simulados de uma amostra de pessoas de cinco regiões diferentes. Com informações sobre idade, gênero e região.
- `iris:` (inclída no R) refere-se a informaçõs sobre o comprimento e a largura de pétalas e sépalas (em centímetros) de 50 flores.
- `dados_anp2.csv`: Com o objetivo de evitar possíveis irregularidades, desde 02 de maio de 2004, a ANP publica o Levantamento de Preços e de Margens de Comercialização de Combustíveis, contemplando a ampliação do universo de municípios pesquisados, abrangendo todos os estados brasileiros. A divulgação da pesquisa encontra-se no site http://www.anp.gov.br/preco/ Os combustíveis incluídos nessa pesquisa são: Gasolina, Etanol, Diesel, GNV, Diesel S10. Essa base contém dados dos preços de venda e compra desses combustíveis no ano de 2016. Esses dados foram capturados usando a técnica de wescraping com R.

- `train.csv`: Essa base foi extraída do desafio da rede Kaggle que promove competições globais para cientistas de dados. O desafio do Titanic é o mais usado e o mais indicado para iniciantes em ciência de dados, trata-se de uma competição para prever se um passageiro sobreviveu ou não no navio. A base em questão fornece as informações:
  - survival: Status se sobreviveu ou não
  - pclass: Em qual classe pertencia
  - name: Nome do passageiro
  - sex: genero do passageiro
  - age: Idade
  - sibsp: Número de esposos e\ou imrmãos à bordo
  - parch: Número de pais\filhos à bordo
  - ticket: Número da passagem
  - fare: Tarifa
  - cabin: Número da cabine
  - embarked: Porta em que embarcou

-`Human development index (HDI).csv`: base com a evolução do Índice de desenvolvimento Humano (IDH) nos anos de 2013 e 2014.

# 3) Primeiros passos com R e RStudio

## 3.1) Instalação do R

Neste curso faremos uso da versão 3.3.2 do R, para instalá-lo siga os passos:

1. Vá no site cran.r-project.org
2. Faça download do arquivo
3. Clique em em executar

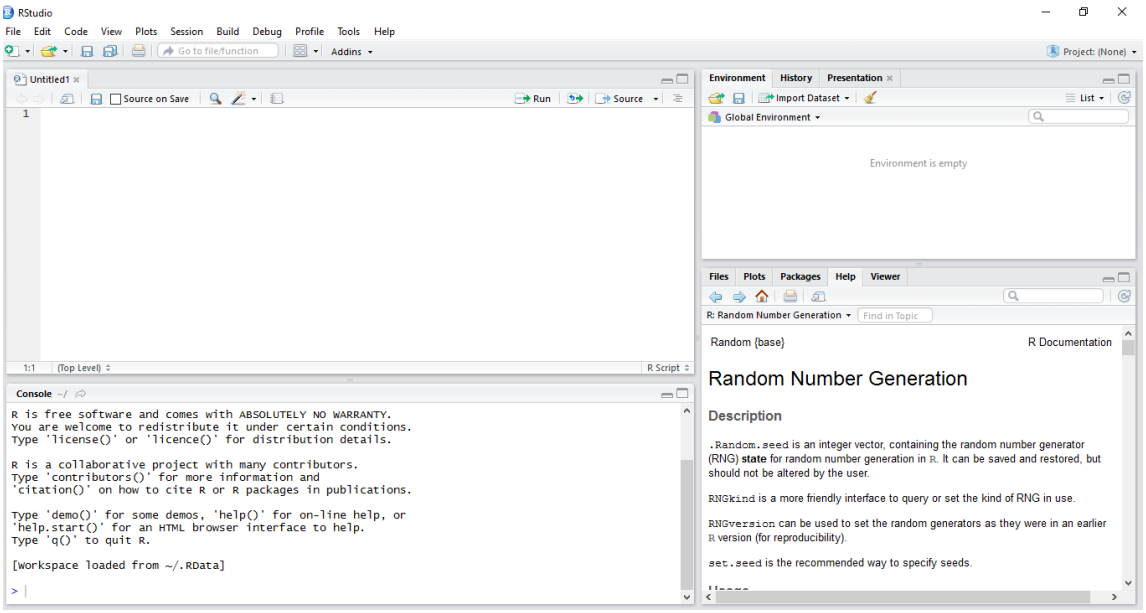
## 3.2) Instalação do RSudio

Neste curso além do R faremos uso do RStudio, para instalalo basta:

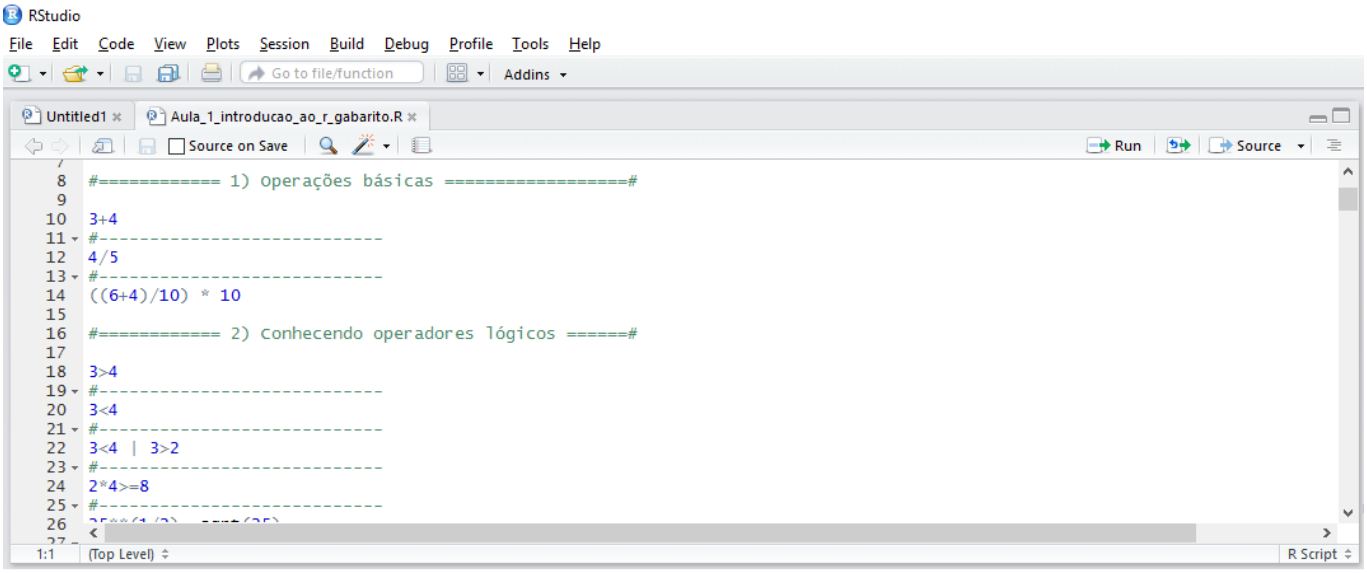
1. Vá no site https://www.rstudio.com/home/
2. Clique em dowload
3. Escolha o free
4. Faça o download do Arquivo
5. Clique em executar

## 3.3) Conhecendo o RStudio

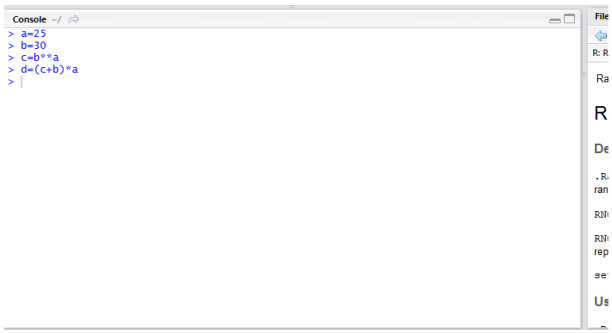
Ao abrir o RStudio será aberto uma tela grande com quatro outras telas:



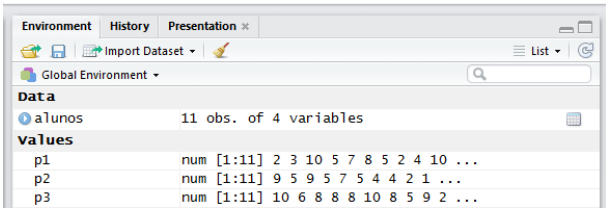

1. A primeira tela é destinada à escrita dos seus scripts, onde ficarão armazendos seus códigos.



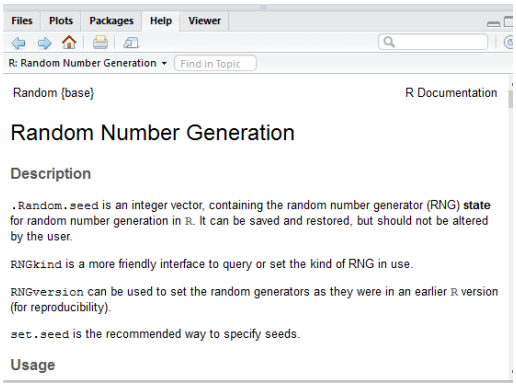

2. A segunda tela é onde são compilados os códigos escritos na primeira tela. Você também pode digitar os comnados no console, mas não serão aramzenados.



3. A terceira tela, destina-se principalmente à aramzenar os objetos criados nos scripts



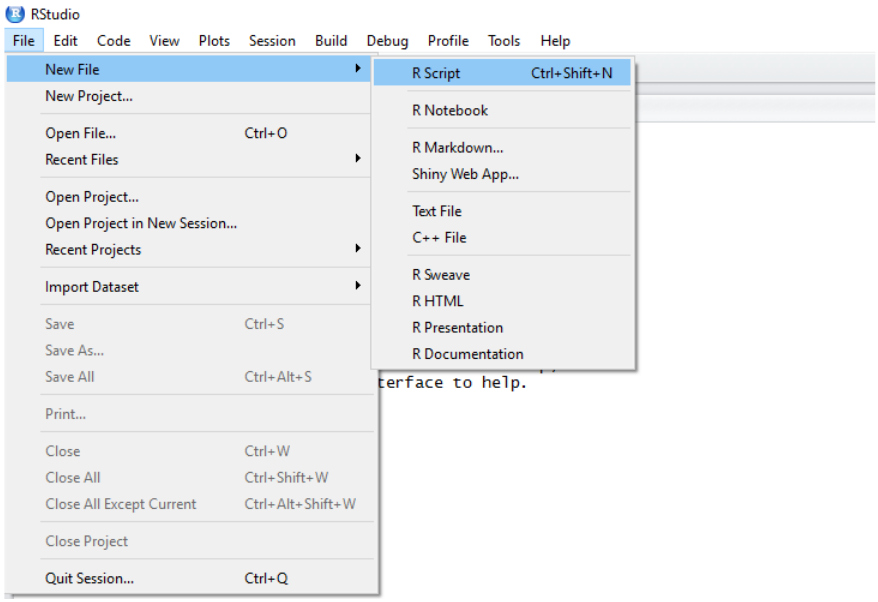

4. A quarta tela tem quatro diferente objetivos. O Primeiro é mostrar os plos (gráficos) que serão feitos em um script. o Segundo Objetivo é mostrar os arquivos dentro de um diretório.O quarto objetivo é responder os questionamentos de qualquer objeto fazendo uso da função `help()`.




## 3.4) Scripts

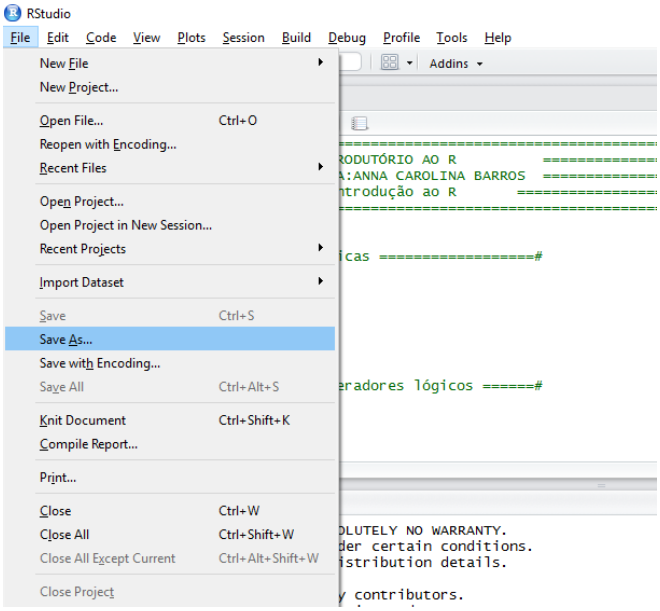
### 3.4.1) Criando um Script

Scripts são um arquivo em texto onde são armazenados os seus comandos em R, os scripts são escritos na primeira tela do RStudio. Para Criar um novo Script basta seguir o caminho: `File >> New File >> RScript`:



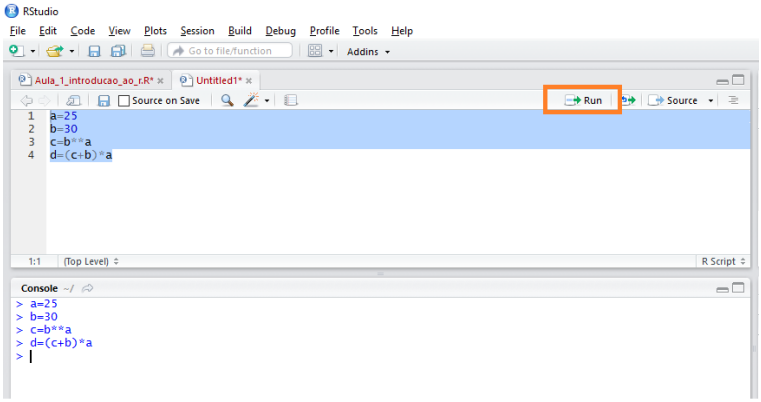
### 3.4.2) Salvando um Script Existente

Para salvar um script escrito, basta seguir o caminho:`File >> Save As`:




### 3.4.3) Executando um Script

Para fazer com que o R entenda que você está executando uma linha de comando existem duas maneiras. A primeira é fortemente desencorajada, que é colocar o cursor ao final da linha ou selecionar as linhas que deseja e excutar e clicar no botão:




A segunda forma vai facilitar a sua vida durante o curso que é:

$$\text{CTRL} + \text{R}$$

# 4) Primeiros Passos com o R

## 4.1) Operações Matemáticas

São operações que resultam em um número, essas operações são:

| Operação | Código em R |
|---|---|
| Soma | + |
| Subtração | - |
| Multiplicação | * |
| Divisão | / |
| Potência | a**b |
| Raiz Quadrada | sqrt() |
| Módulo (Resto) | %% |
| Quociente da divisão | %/% |
| Logarítimo | log(x,base) |
| Combinção | choose(x,n) |
| Fatorial | factorial(x) |

**Exemplo 4.1**: Qual é o resultado de $\sqrt[2]{25}$?

```
sqrt(25)
```

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

**Exemplo 4.2**: Qual é o resto da divisão de $\frac{5}{2}$?

```
5%%2
```

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

## 4.2) Operações Lógicas

São operações que testam se uma situação é verdadeira ou falsa, os principais operadores são:

| Operação | Código em R |
|---|---|
| Negação | ! |
| Contém | %in% |
| Maior | > |
| Menor | < |
| Igual | == |
| Diferença | != |

**Exemplo 4-** Tome $a = 25$ e $b = 45$ analise se a é maior:

```
a=25
b=45

a>b
```

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

Os operadores lógicos acima analisam se uma situação é verdadeira ou falsa de maneira pontual. Existem operadores lógicos que analisam de maneira mais ampla se uma situção é verdadeira ou não:

- Operador **E** - Retorna verdadeiro se toda a expressão for verdadeira. No R fazemos uso do operador `&` ou `&&`.
- Operador **OU** - Retorna verdadeiro se ao menos uma das partes da expressão for verdadeira. No R fazemos uso do operador `|` ou `||`.

- OPerador **OU Exclusivo -** Retorna verdadeiro, somente se uma das partes for verdadeira. No R para ou exclusivo fazemos uso do `xor()`

**Exemplo 4.4-** Analise se $25^2 = \sqrt{625}$ e se $Bigdata = bigData$:

```
25**2==sqrt(625) & "Bigdata" == "bigData"
```

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

**Exemplo 5.5-** Analise se $25^2 = \sqrt{625}$ ou se $Bigdata = bigData$:

```
25**2==sqrt(625) | "Bigdata" == "bigData"
```

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

# 5) Objetos

São objetos destinados a guardar temporariamente os valores calculados. Para atribuir uma valor à um Objeto basta digitar o comando `<-` ou `=`.

**OBS:** No R você pode criar objetos,e visualizados de duas maneiras distintas:

- A primeira é inserindo "(" antes do objeto e ")" após o objeto atribuído

**Exemplo 5.1**

```
(a=3)
```

```
## [1] 3
```

- A segunda e chamar o objeto pelo nome atribuído:

**Exemplo 5.2**

```
a=3
a
```

```
## [1] 3
```

## 5.1) Classes de objetos

### 5.1.1) Númerico

Como o próprio nome indica, são variáveis em que são atribuídas números.

**Exemplo 5.3-** Tome $a = 2$ e $b = 5$ faça $c = a + b$

```
a=2
b=5

c=a+b
c
```

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

**Exemplo 5.4 -** Tome $a = 15$ e $b = 25$ faça $c = a + b$ e $d = (\sqrt[3]{c})$

```
a=15
b=25

(c=a+b)
```

```
## [1] 40
```

```
d=c^(1/3)
d
```

```
## [1] 3.419952
```

### 5.1.2) Character

São palavras, textos, letras, entre outros. Para atribuir a uma variável tipo character é necessário colocar o valor entre "" (aspas duplas) ou '' (aspas simples).

**Exemplo 5.5. -** Tome $nome_1 = Big$ e $nome_2 = Data$:

```
nome1="big"
nome1
```

```
## [1] "big"
```

```
nome2='data'
nome2
```

```
## [1] "data"
```

**Exemplo 5.6 -** Faça $nomecompleto = nome_1 + nome_2$:

Para resolver essa questão é necessário fazer uso da função `paste()`, essa função server para concatenar duas ou mais variáveis character.

```
nomecompleto=paste(nome1,nome2)
nomecompleto
```

```
## [1] "big data"
```

Caso não seja útil o espaço entre as palavras, basta usar a função `paste0()`:

```
nomecompleto=paste0(nome1,nome2)
nomecompleto
```

```
## [1] "bigdata"
```

### 5.1.3) Data

Dados tipo data podem ser importados de diversos tipos em especial character. É muito útil transformar characteres em datas, principalmente quando só arquivos são importados de fontes externas.

**Exemplo 5.7-** Transforme 06/02/2017 em data:

```
data="06/02/2017"

formatodata=as.Date(x = data,format="%d/%m/%Y")

formatodata
```

```
## [1] "2017-02-06"
```

No exemplo acima foi feito o uso da função `as.Date()`, que tem como inputs:

- `x`= Objeto a ser transformado
- `format`, como esta divido o texto a ser transformado em data (no exemplo acima estava divido por "/"):
  - `%d`= representa o dia

- `%m`= representa o mês
- `%Y`= representa o ano

#### 5.1.4) Lógica

É uma variável booleana que poder receber como valor as situações `TRUE`(verdadeira) ou `FALSE`(Falso).

### 5.2) Conhecendo a classe de um objeto

Para saber qual é a classe de um objeto basta usar a função `class()` ou `str()`:

**Exemplo 5.8 -* Tome $a = 3$,descubra a classe de $a$*.

```
a=3

str(a)
```

```
##  num 3
```

**Exemplo 5.9 -** Tome $a = 3$,descubra a classe de $a$*.

```
a=3
class(a)
```

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

## 6) Vetores

Vetores são objetos que armazenam unidimensionalmente várias variáveis da mesma classe.

Imagine que você é um diretor de uma empresa que vende um determinado produto em todos os estados brasileiros. Assim, se você quisesse ter o conjunto de preços por estado de cada produtos você deveria criar 27 objetos numéricos diferentes.

$$\begin{aligned} \text{preco1} &= 1.5 \\ \text{preco2} &= 2.5 \\ \text{preco3} &= 4.5 \\ &\vdots \\ \text{preco27} &= 5.5 \end{aligned}$$

Para evitar esse trabalho desnecessário o correto é usar vetores, que listará todos os preços:

$$\mathbf{precos = [1.5, 2.5, 4.5, \ldots, 5.5]}$$

No R para criar um vetor basta usar a função `c()`:

**Exemplo 12** Crie um vetor chamado nomes que contenha:

```
nomes=c("Anna","Paula","Roberta","Ingrid","Fernanda","João")
nomes
```

```
## [1] "Anna"     "Paula"    "Roberta"  "Ingrid"   "Fernanda" "João"
```

| Anna |
|---|
| Paula |
| Roberta |
| Ingrid |
| Fernanda |
| João |

## 6.1) Indexação de Vetores

É comum, trabalhar com um ou mais elementos específicos de um vetor, para acessa-los é necessário conhecer a indexação:

$$elemento_i = V[i]$$

onde: $i$= é a posição do elemento solicitado $V$= Vetor em questão

No R, segue essa mesma lógica basta digitar `nome_vetor[posição desejada]`.
**Exemplo 6.1-** tome o vetor $v = [1, 10, 15, 5, 30, 2]$, qual é o primeiro elemento do vetor?

```
v=c(1,10,15,5,30,2)

v[1]
```

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

Caso queira mais de uma posição digite `nome_vetor[posicao_inicial:posicao_final]`

**Exemplo 6.2-** Ainda com o vetor $v = [1, 10, 15, 5, 30, 2]$, quais são os três primeiros elementos do vetor?

```
v[1:3]
```

```
## [1]  1 10 15
```

## 6.2) Tamanho do Vetor

Para saber o tamanho de um vetor, basta usar a função `length()`:

**Exemplo 6.3-** Qual é o tamanho do vetor $v = [1, 10, 15, 5, 30, 2]$?

```
v=c(1,10,15,5,30,2)
length(v)
```

```
## [1] 6
```

## 6.3) Concatenando vetores

Agora vamos supor que você é um professor e no primeiro dia de aula tem a lista com o nome dos alunos:

Mas no segundo dia surgiram mais 5 alunos:

Como colocar os novos nomes na antiga lista?

No R, para concatenar vetores basta usar a função `c(primeiro_objeto,segundo_objeto,...)`

**Exemplo 6.4-** Concatene as listas com os nomes doa alunos:

| Anna |
|---|
| Pedro |
| Carlos |
| Bruno |
| Vanessa |
| Paula |
| Italo |

| Jorge |
|---|
| Davi |
| Mariana |
| Carolina |
| Alice |

```
nomes1<-c("Anna","Pedro","Carlos","Bruno","Vanessa","Paula","Italo")
nomes2<-c("Jorge","Davi","Mariana","Carolina","Alice")

nomes_completo<-c(nomes1,nomes2)
nomes_completo
```

```
##  [1] "Anna"     "Pedro"    "Carlos"   "Bruno"    "Vanessa"  "Paula"
##  [7] "Italo"    "Jorge"    "Davi"     "Mariana"  "Carolina" "Alice"
```

**Funções úteis**

| Função | Saída |
|---|---|
| mean() | Médias dos valores em um vetor |
| max() | Retorna o valor Máximo |
| min () | Retorna o Valor Mínimo |
| sum() | Retorna a soma dos elementos |
| unique() | Remoção de duplicatas |

## 6.4) Exercícios

1. Crie um vetor $V = [105, 106, 150, 135, 120, 147]$:
   (a) Concatene com o vetor $v2 = [105, 250, 300, 175, 157, 147, 134]$.
   (b) Remova as duplicatas.
   (c) Qual é o terceiro elemento?
   (d) Qual é o o tamanho do vetor?
   (e) Qual é a média dos valores?
   (f) Qual é o maior elemento?

# 7) Data Frames

No mundo real as informações não estão contidas em uma unica dimensão (vetor), na maioria dos casos são necessárias mais informações. Por exemplo caso na lista dos alunos, é importante as datas das aulas para saber o nível de presença, o nome da matéria, entre outras informações.

Neste tipo de cenário o adequado é o uso de tabelas. No r as tabelas são chamadas de Data Frame, e são os objetos mais usados na linguagem.

No R para criar um data frame basta usar a função `data.frame()`.

**Exemplo 7.1-** Vamos supor que você é dono de uma loja e tenha 5 tipos de produtos diferentes, com diferentes preços:

| Produto | Preço |
|---|---|
| Produto A | R$ 5,00 |
| Produto B | R$ 15,00 |
| Produto C | R$ 4,00 |
| Produto D | R$ 6,00 |
| Produto E | R$ 8,00 |

Crie esta tabela no R:

```
Produto<-c("Produto A"," Produto B","Produto C", "Produto D","Produto E")
Preco<-c(5,15,4,6,8)

tabela_preco_produto<-data.frame(Produto,Preco)

tabela_preco_produto
```

```
##      Produto Preco
## 1  Produto A     5
## 2  Produto B    15
## 3  Produto C     4
## 4  Produto D     6
## 5  Produto E     8
```

## 7.1) Indexação de Data Frames

A indexação dos valores de uma data frame é parecida com a indexação dos vetores, sendo que com duas dimensões:

$$elemento_{i,j} = df[i, j]$$

onde: $df$= data frame criado $i$= é posição da linha onde está o objeto desejado $j$= é a posição da coluna do objeto desejado

No R, segue essa mesma lógica basta digitar `data_frame[posição linha, posição coluna]`.

**Exemplo 7.2-** Acesse o preço do *Produto D* na tabela produtos:

```
tabela_preco_produto[4,2]
```

```
## [1] 6
```

**Exemplo 7.3-** Acesse os preço do *Produto D* e *Produto E* na tabela produtos:

```
tabela_preco_produto[4:5,2]
```

```
## [1] 6 8
```

## 7.2) Acessando colunas em data frame

Além da indexação tradicional pelo número da coluna, existem outras duas formas muito úteis para acessar uma coluna de um data frame.

A primeira é `data_frame$nome_coluna`:

**Exemplo 7.4-** Acesse a coluna produtos na tabela_produto:

```
tabela_preco_produto$Produto
```

```
## [1] Produto A   Produto B Produto C  Produto D  Produto E
## Levels:  Produto B Produto A Produto C Produto D Produto E
```

A segunda forma é usando colchetes, `data_frame[,"nome_coluna"]`

**Exemplo 7.5-** Acesse a coluna produtos na tabela_produto:

```
tabela_preco_produto[,"Produto"]
```

```
## [1] Produto A   Produto B Produto C  Produto D  Produto E
## Levels:  Produto B Produto A Produto C Produto D Produto E
```

## 7.3) Criando uma nova coluna em uma data frame

A criação de um a nova coluna em um data frame é parecida com o acesso á colunas.

A primeira forma de criar uma nova coluna é `data_frame$nome_nova_coluna` .

**Exemplo 7.6-** Crie uma coluna quantidade, na tabela preços, que receba os valores:

$quantidade < -c(50, 100, 120, 150, 200)$

```
tabela_preco_produto$quantidade<-c(50,100,120,150,200)
tabela_preco_produto
```

```
##      Produto Preco quantidade
## 1  Produto A     5         50
## 2  Produto B    15        100
## 3  Produto C     4        120
## 4  Produto D     6        150
## 5  Produto E     8        200
```

A segunda forma de criar uma nova coluna é `data_frame[,"nome_nova_coluna"]`

**Exemplo 7.7-** Crie uma coluna, na tabela preços com os custos de cada produto:

$custos < -c(2, 12, 3, 5, 6)$

```
tabela_preco_produto[,"Custos"]<-c(2,12,3,5,6)
tabela_preco_produto
```

```
##      Produto Preco quantidade Custos
## 1  Produto A     5         50      2
## 2  Produto B    15        100     12
## 3  Produto C     4        120      3
```

```
## 4  Produto D     6        150      5
## 5  Produto E     8        200      6
```

## 7.4) Conhecendo seu Data Frame

É aconselhável conhecer seu objeto, data frame, com o qual está trabalhando. Até o momento trabalhamos com uma tabela pequena, a tabela preços, a qual era composta por 5 linhas(observações) e 4 colunas(variáveis). No mundo real tabelas como estas são exceção, neste sentido não é possível visualizar e extrair informações analogamente olhando a olho nu o data frame como feito na tabela preços.

O R oferece algumas funções que auxiliam a entender seu data frame. Para isso, vamos usar outra base de dados a `iris`(vide seção 2 bases).

A primeira função é a `str()` que ajuda a conhecer a classe de cada coluna (variável) no data frame.

```
base_flores<-iris

str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

A primeira saída dessa função é o numero de observações (linhas) e o número de colunas, depois saem as informações sobre cada coluna que exceto pela última são todas numéricas.

A segunda função é a `head()` que retorna as seis primeiras observações de seu data frame:

```
head(base_flores)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

A terceira função é a `summary()` que faz um "resumo" e todos os dados dentro de um data frame:

```
summary(base_flores)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
```

```
##
```

Perceba que para as colunas numéricas a função `summary` calculou as estatísticas descritivas e para as colunas de texto foram contabilizadas as quantidades.

A quarta função é a `names()` que retorna os nomes das colunas de uma data frame.

```
names(base_flores)
```

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

### Funções Úteis

Algumas funções úteis:

| Função | Saída |
|---|---|
| nrow() | Número de linhas (observações) em um data frame |
| ncol() | Número de colunas (varáveis) em um data frame |
| dim() | Retorna a dimensão do data frame. Número de linhas e úmero de colunas. |
| unique() | Remoção de duplicatas |
| tail() | Retorna as 6 últimas observações em um data frame |
| cbind() | Concatena dois data frames pelas colunas |
| rbind() | Concatena dois data frames pelas linhas |

## 7.5) EXERCÍCIOS

Crie uma tabela, chamada de `tabela_pessoas` e responda as questões:

| Nome | Gênero | Região | Idade |
|---|---|---|---|
| Paulo | 1 | Região 2 | 36 |
| Anna | 2 | Região 3 | 32 |
| Pedro | 1 | Região 3 | 30 |
| Viviane | 2 | Região 2 | 32 |
| Ricardo | 1 | Região 3 | 31 |
| Diego | 1 | Região 5 | 29 |
| Marcos | 1 | Região 2 | 35 |
| Renata | 2 | Região 5 | 33 |
| Victor | 1 | Região 5 | 25 |
| Bruno | 1 | Região 1 | 27 |
| Juliana | 2 | Região 1 | 27 |
| Adriana | 2 | Região 1 | 19 |
| Juliana | 2 | Região 2 | 31 |
| Beatriz | 2 | Região 1 | 22 |
| Vanessa | 2 | Região 2 | 36 |
| Ingrid | 1 | Região 3 | 34 |
| Mariana | 2 | Região 3 | 33 |

1. Quantas observações tem a tabela?
2. Quais são as médias das idades?
3. Quais são as classes de cada uma das colunas?
4. Faça um resumo da tabela.

# 8) Listas

Listas podem ser consideradas uma espécie de contêiner que armazena diferentes tipos de objetos em diferentes dimensões, não importando a sua classe.

Para criar uma lista basta usar a função `list()`.

**Exemplo 8.1-** Crie uma lista com os parâmetros:

- Nome do Curso: Curso de Formação Executiva em Big Data e Data Science
- Turma: 3
- Aula: Introdução ao R

```
minha_lista<-list("Curso de Formação Executiva em Big Data e Data Science",3, "Introdução ao R")
minha_lista
```

```
## [[1]]
## [1] "Curso de Formação Executiva em Big Data e Data Science"
##
## [[2]]
## [1] 3
##
## [[3]]
## [1] "Introdução ao R"
```

A lista `minha_lista` foi criada de maneira correta, porém os seus elementos não possuem nomes(chaves), vamos recriá-las com seus respectivos nomes:

```
minha_lista2<-list(nome_curso="Curso de Formação Executiva em Big Data e Data Science",
                   turma=3, Aula="Introdução ao R")
minha_lista2
```

```
## $nome_curso
## [1] "Curso de Formação Executiva em Big Data e Data Science"
##
## $turma
## [1] 3
##
## $Aula
## [1] "Introdução ao R"
```

## 8.1) Acessando os elementos em uma lista

Existem duas formas distintas de acessar os elementos em uma lista. A primeira forma é por meio do uso de `nome_lista[[posição desejada]]`

**Exemplo 8.2-** Acesse o terceiro elemento da lista com as informações do curso.

```
minha_lista2[[3]]
```

```
## [1] "Introdução ao R"
```

A segunda forma é por meio do uso de `nome_lista$nome_objeto`

**Exemplo 8.3-** Em `minha_lista2`, acesse o objeto com o nome do curso:

```
minha_lista2$nome_curso
```

```
## [1] "Curso de Formação Executiva em Big Data e Data Science"
```

## 8.3) Adicionando Objetos em uma Lista

Para adicionar objetos à uma lista basta chamar o nome de sua lista com o conector `$` e nome do objeto a ser criado: `nome_lista$nome_elemento`.

**Exemplo 8.4-** Em `minha_lista2`, crie um objeto que receba a lista com nomes e outro objeto que receba as idades:

- nomes: Anna, Marco, Fernando, Juliana, Diego, Rafael
- idades: 29, 31, 22, 20, 27, 28

```
minha_lista2$alunos<-c("Anna", "Marco", "Fernando", "Juliana", "Diego","Rafael")
minha_lista2$idades<-c(29, 31, 22, 20, 27, 28)
minha_lista2
```

```
## $nome_curso
## [1] "Curso de Formação Executiva em Big Data e Data Science"
##
## $turma
## [1] 3
##
## $Aula
## [1] "Introdução ao R"
##
```

```
## $alunos
## [1] "Anna"     "Marco"    "Fernando" "Juliana"  "Diego"    "Rafael"
##
## $idades
## [1] 29 31 22 20 27 28
```

### 8.4) Conhecendo a sua lista

De maneira análoga com o data frame, também é possível conhecer e analisar as listas, sem ler á olho nu elemento a elemento.

A primeira função é o `str()` que retorna a classe do objeto (que é uma lista) e a classe deu seus elementos:

**Exemplo 8.5-** Aplique a função `str()` em `minha_lista2`.

```
str(minha_lista2)
```

```
## List of 5
##  $ nome_curso: chr "Curso de Formação Executiva em Big Data e Data Science"
##  $ turma     : num 3
##  $ Aula      : chr "Introdução ao R"
##  $ alunos    : chr [1:6] "Anna" "Marco" "Fernando" "Juliana" ...
##  $ idades    : num [1:6] 29 31 22 20 27 28
```

A segunda função é `summary()` que retorna um resumo da lista e de seus objetos:

**Exemplo 8.6-** Aplique a função `summary()` em `minha_lista2`.

```
summary(minha_lista2$idades)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   23.25   27.50   26.17   28.75   31.00
```

### 8.5) Exercícios

Tome os vetores: $nomes = [Anna, Paula, Roberta, , Ingrid, fernanda, João]$,$pesos = [52, 65, 70, 58, 48, 70]$ e $alturas = [1.54, 1.76, 1.65, 1.60, 1.68, 1.70]$. Criem uma lista com esses vetores chamado lista_pessoas.Depois crie um quarto objeto chamado IMC de acordo coma equação, depois responda as questões:

$$IMC = \frac{Peso}{(Altura)^2}$$

1. Qual é o primeiro elemento da lista?
2. Quais são as classes dos objetos na lista?
3. Qual é o maior IMC?

## 9) Trabalhando com Diretórios e Arquivos Externos

### 9.1) Conhecendo seu diretório de arquivos

Para saber em qual diretório(pasta) o seu script em R está apontando basta fazer uso da função `getwd()`.

```
getwd()
```

```
## [1] "C:/Users/ac_ba/Dropbox/Apostilia R"
```

Também é muito útil saber quais arquivos estão contidos no diretório. Para isso o R e R Studio apresentam as duas soluções a primeira é por meio da seta no canto superior do console conforme a figura abaixo:

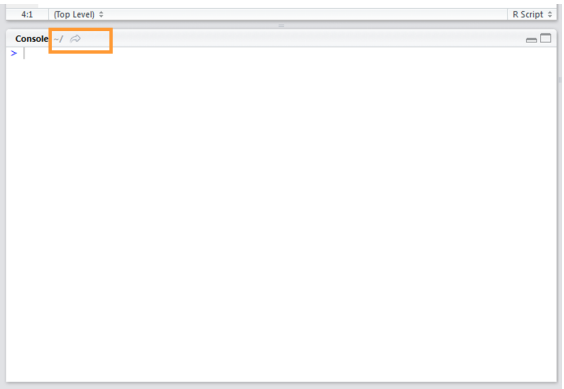



Figure 1: Console

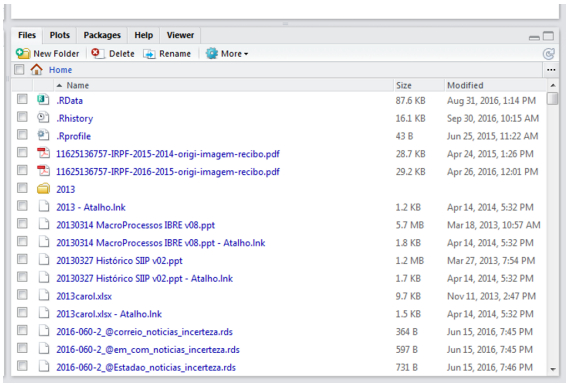



Figure 2: Files

Como pode ser visualizado os arquivos aparecerão na tela dos files;

A outra maneira de visualizar seus arquivos (e a mais recomendada!) é por meio da função `list.files()`

## 9.2) Mudando seu diretório de trabalho

Em muitos casos, não trabalhamos com arquivos somente em um diretório, é preciso altera-lo. Para alter um diretório de trabalho basta usar a função `setwd()`.

**Atenção!!** Em ambiente windows `\` não é reconhecido e deverá ser trocado por `\\` ou `/`.

**Exemplo 9.1-** Altere seu diretório de trabalho para a pasta da FGV do Curso no Dropbox:

```
setwd("C:\\Users\\ac_ba\\Dropbox\\FGV - Big Data & Data Science - alunos\\01 Big Data & Data Science + I
```

## 9.3) Leitura de Arquivos Externos

Nem todos os objetos que usamos no R são calculados dentro de um Script. Assim, é indispensável a leitura de arquivos externos. O R oferece uma gama de funções e pacotes para a leitura de arquivos externos nos formatos, .txt,.csv, dentre outros. Estas funções lêem os arquivos externos e transforma automáticamente em data frames.

A primeira função é a `read.table()`, que é considederada a raiz, muitas funções são derivadas desta. A função `read.table()` é composta pelos campos:

- **file-** nome do arquivo / caminho e nome do arquivo;
- **header-** determina se o arquivo tem cabecalho ou nao, default TRUE; no caso de FALSE a funcao cria nomes para as colunas;
- **sep-** separador da coluna (default tab);
- **dec-**separador decimal (default ponto);

**Exemplo 9.2-** Leia o arquivo `teste.txt` no diretório *datasets*:

```
setwd("C:\\Users\\ac_ba\\Dropbox\\FGV - Big Data & Data Science - alunos\\01 Big Data & Data Science + I
pessoas<-read.table(file="populacao.txt",header = TRUE,sep = "\t",dec=".")
head(pessoas)
```

```
##                Nome Sexo   Região Idade
## 1       ABEL MATTOS    1 Região 2    36
## 2  ADAILTON CARDOSO    1 Região 3    32
## 3   ADALBERTO NEUSA    1 Região 3    30
## 4   ADALBERTO THOME    1 Região 2    32
## 5 ADALTINO HICKMANN    1 Região 3    31
## 6      ADELMO CRIPA    1 Região 5    29
```

A função `read.table()` não é tão eficiente por isso existem variações destas função. A primeira é a `read.csv()`, que possui parametros parecidos com a anterior:

- **file-** nome do arquivo / caminho e nome do arquivo;
- **header-** determina se o arquivo tem cabecalho ou nao, default TRUE; no caso de FALSE a funcao cria nomes para as colunas;
- **sep-** separador da coluna (default ",");
- **dec-**separador decimal (default ponto);

**Exemplo 9.3-** Na pasta *datasets* leia o arquivo `populacao.csv`:

```
setwd("C:\\Users\\ac_ba\\Dropbox\\FGV - Big Data & Data Science - alunos\\01 Big Data & Data Science + R
pessoas_csv<-read.csv(file="populacao.csv")
head(pessoas_csv)
```

```
##                Nome Sexo   Região Idade
## 1       ABEL MATTOS    1 Região 2    36
## 2  ADAILTON CARDOSO    1 Região 3    32
## 3   ADALBERTO NEUSA    1 Região 3    30
## 4   ADALBERTO THOME    1 Região 2    32
## 5 ADALTINO HICKMANN    1 Região 3    31
## 6      ADELMO CRIPA    1 Região 5    29
```

O padrão que a função `read.csv()` lê é o arquivo .csv padrão americano em que o separado é o ",". No Brasil o padrão é o separador ";" e dec ",". Por isso, existe a função `read.csv2()`, que te como parâmetros:

- **file-** nome do arquivo / caminho e nome do arquivo;
- **header-** determina se o arquivo tem cabecalho ou nao, default TRUE; no caso de FALSE a funcao cria nomes para as colunas;
- **sep-** separador da coluna (default ";");
- **dec-**separador decimal (default virgula)

**Exemplo 9.4-** Na pasta *datasets* leia o arquivo `dados_anp2.csv`:

```
setwd("C:\\Users\\ac_ba\\Dropbox\\FGV - Big Data & Data Science - alunos\\01 Big Data & Data Science + R
combustivel<-read.csv2(file="dados_anp2.csv")
head(combustivel)
```

```
##                                 RAZAO_SOCIAL
## 1                    Auto Posto Lacasema Ltda
## 2               Imperial Angra Auto Posto Ltda
## 3                  Posto Praia da Ribeira Ltda
## 4                     Auto Posto Japuíba Ltda.
## 5                   Posto dos Pescadores Ltda.
## 6 Golfinho Comercio Derivados de Petroleo Ltda
##                                          ENDERECO               BAIRRO
## 1 Avenida Almte Jair Carneiro Toscano de Brito, 417 Parque das Palmeiras
## 2                           Rua das Palmeiras, 210 Parque das Palmeiras
## 3      Rodovia Br 101, S/n Km 487,8 - Lado Esquerdo Belem Tq. da Japuiba
## 4   Rodovia Governador Mário Covas, S/n Km 96 + 400              Japuiba
```

```
## 5                           Alameda Industria, 309              Centro
## 6                     Rua Doutor Coutinho, 8 Loja A  Cais de Santa Luzia
##                       BANDEIRA PRECO_VENDA PRECO_COMPRA
## 1                       RAIZEN       4.198          NaN
## 2                       RAIZEN       4.220          NaN
## 3 PETROBRAS DISTRIBUIDORA S.A.       4.278         3.58
## 4                     IPIRANGA       4.279         3.58
## 5 PETROBRAS DISTRIBUIDORA S.A.       4.329        3.468
## 6                       RAIZEN       4.400        3.598
##   MODALIDADE_DE_COMPRA FORNECEDOR DATA_COLETA         CIDADE UF
## 1                    -          -  13/09/2016 ANGRA DOS REIS RJ
## 2                    -          -  13/09/2016 ANGRA DOS REIS RJ
## 3                  CIF          -  13/09/2016 ANGRA DOS REIS RJ
## 4                  CIF          -  13/09/2016 ANGRA DOS REIS RJ
## 5                  FOB          -  13/09/2016 ANGRA DOS REIS RJ
## 6                  CIF          -  13/09/2016 ANGRA DOS REIS RJ
##   COMBUSTIVEL
## 1    Gasolina
## 2    Gasolina
## 3    Gasolina
## 4    Gasolina
## 5    Gasolina
## 6    Gasolina
```

## 9.4) Exportação de Arquivos

Nem todo mundo usa o R. Em muitos casos seus resultados serão utilizados por outros softwares. Neste sentido, o R permite a exportação dos resultados para outras extensões no formato .txt, .csv, dentre outros.

A exportação de arquivos ocorre de maneira análoga ao que é feito na leitura de dados, a primeira função é a `write.table()` e existem outras derivadas desta. A função `write.table()` tem como parâmetros:

- `x`- nome do data frame a ser salvo
- `file` - diretório + nome do arquivo a ser salvo
- `quotes` - Booleano, decide se as colunas charcteres ou fatores serão salvos com aspas duplas.
- `row.names`- Booleano, `TRUE` ou `FALSE`, significa se o arquivo salvo terá ou não nome nas linhas
- `col.names`- Booleano, `TRUE` ou `FALSE`, significa se o arquivo salvo terá ou não nome nas colunas
- `append` - Booleano, `TRUE` ou `FALSE`, decide se a tabela será anexada a um arquivo já existente ou não
- `sep` - Character, informa qual será o separador das colunas -‘`dec` - Character, decide qual será o separador número decimal

**Exemplo 9.5-** Tome a tabela abaixo com nomes e idades, exporte ela para o formato .txt, como o nome `idades.txt`

Table 1: My caption

| Nomes | Idades |
|---|---|
| Marco | 36 |
| Carol | 29 |
| João | 28 |
| Caio | 8 |
| Vinicius | 10 |

```
Nomes<-c("Marco","Carol","João","Caio","Vinicius")
Idades=c(36,29,28,8,10)

frame_idades<-data.frame(Nomes,Idades)

write.table(frame_idades,"idades.txt")
```

A função `write.table()` também não é muito indicada dado que não apresenta um bom desempenho. Assim, existe outras funções que são variações da função `write.table()`

A primeira função é a `write.csv()` que apresenta os mesmos parâmetros que a `write.table()`, sendo que considera como default o `sep=","` e o `dec=.`

**Exemplo 9.6-** Faço mesmo que foi feito no exemplo 31 sendo que no formato .csv.

```
Nomes<-c("Marco","Carol","João","Caio","Vinicius")
Idades=c(36,29,28,8,10)

frame_idades<-data.frame(Nomes,Idades)

write.csv(frame_idades,"idades.csv")
```

A segunda função é a`write.csv2()`, voltada para arquivoc .csv brasileiros, que apresenta os mesmos parâmetros que a `write.table()`, sendo que considera como default o `sep=";"` e o `dec=.`

**Exemplo 9.7-** Faço mesmo que foi feito no exemplo 31 sendo que no formato .csv.

```
Nomes<-c("Marco","Carol","João","Caio","Vinicius")
Idades=c(36,29,28,8,10)

frame_idades<-data.frame(Nomes,Idades)

write.csv2(frame_idades,"idades.csv")
```

## 9.5) Leitura de arquivos usando pacotes externos

Existem alguns arquivos em que as funções padrão do R, não conseguem ler ou exportar. Para isso é necessário fazer uso dos pacotes. Para usar um pacote o primeiro passo é instala-lo com a função `install.packages("nome do pacote")`.

Vamos instalar o pacote `xlsx`para leitura ou exportação de arquivos no formato .xslx.

```
install.packages("xlsx")
```

### 9.5.1) Leitura de arquivos no formato .xlsx

Para ler arquivos no formato .xlsx, primeiro será necessário carregar o pacote `xlsx` usando o comando `library()`. O segundo passo é usar a função `read.xlsx`, com os parâmetros:

- `file`- Endereço + nome do arquivo a ser lido.
- `header`- Booleano, TRUE ou FALSE. Determina se o arquivo vai ter cabeçalho ou não. O Default é TRUE.
- `sheetIndex`- Índice da Aba a ser lida.
- `SheetName`- Nome da aba a ser lida (Default=NULL).
- `rowIndex` - Índices das linhas a serem lidas (Default=NULL).
- `colIndex` – índices das colunas que devem ser lidas (default NULL).

**Exemplo 9.8-** Na pasta datasets leia o arquivo `populcao.xlsx`:

```
library(xlsx)
```

```
## Loading required package: rJava
```

```
## Loading required package: xlsxjars
```

```
setwd("C:\\Users\\ac_ba\\Dropbox\\FGV - Big Data & Data Science - alunos\\01 Big Data & Data Science + I
populacao<-read.xlsx("populacao.xlsx",sheetIndex = 1)
head(populacao)
```

```
##                Nome Sexo   RegiÃ.o Idade
## 1       ABEL MATTOS    1 RegiÃ£o 2    36
## 2  ADAILTON CARDOSO    1 RegiÃ£o 3    32
## 3   ADALBERTO NEUSA    1 RegiÃ£o 3    30
## 4   ADALBERTO THOME    1 RegiÃ£o 2    32
## 5 ADALTINO HICKMANN    1 RegiÃ£o 3    31
## 6      ADELMO CRIPA    1 RegiÃ£o 5    29
```

### 9.5.2) Exportação de arquivos no formato .xlsx

Para exportar arquivos no formato .xlsx também é necessário o pacote `xlsx`. A função utilizada para a exportação deste tipo de arquivo é a `write.xlsx()` que tem como parâmetros:

- `x` - data frame a ser salvo no formato.xlsx
- `file` - nome do arquivo
- `sheetName` - Nome da aba do arquivo a ser salvo
- `sheetName` - índice aba do arquivo a ser salvo
- `row.names`- Booleano, `TRUE` ou `FALSE`, significa se o arquivo salvo terá ou não nome nas linhas
- `col.names`- Booleano, `TRUE` ou `FALSE`, significa se o arquivo salvo terá ou não nome nas colunas

** Exemplo 9.9-** Crie um arquivo chamado funcionarios.xlsx com os dados:

| Nome | Salário |
|---|---|
| Anna | R$ 3000,00 |
| Bruno | R$ 5000,00 |
| Fernando | R$ 2500,00 |
| Viviane | R$ 1000,00 |
| Bernardo | R$ 500,00 |

```
Nome<-c("Anna","Bruno","Fernando","Viviane","Bernardo")
Salario<-c(3000,5000,2500,1000,500)

df_salarios<-data.frame(Nome,Salario)

write.xlsx(df_salarios,"funcionarios.xlsx",sheetName ="Salario",row.names = FALSE)
```

# 10) Estruturas de Condição e Repetição

## 10.1) Estruturas de Condição

Fazemos uso de estruturas de condição, quando em um código queremos executar um procedimento caso uma condição seja atendida.

### 10.1.1) Estrutura IF

Executa uma ação caso uma condição seja atendida, caso não seja prosegue com os outros comandos normalmente.

**Exemplo 10.1-** Tome $a = 31$ e $b = 30$, analise se $a$ é maior que b.

```
a=31
b=30

if(a<b){
  cat("a é maior que b")
}
```

### 10.1.2) Estrutura If e Else

Caso uma condição seja atendida o código executa uma condição caso não seja executa outra.

**Exemplo 10.2-** Tome $a = 31$ e $b = 30$, analise qual dois dois objetos é o maior.

```
a=31
b=30

if(a<b){
  cat("a é maior que b")
}else{
  cat("b é maior que a")
}
```

```
## b é maior que a
```

**Exemplo 10.3-** E se $a = 30$?

```
a=31
b=30

if(a<b){
  cat("a é maior que b")
}else if(b>a){
  cat("b é maior que a")
} else{
  cat("São iguais")
}
```

```
## São iguais
```

### 10.1.3) Estrutura Ifelse

O R apresenta uma versão mais compacta da estrutura `if` e `else` que é a função `ifelse()` que tem como parâmetros:

- teste lógico
- condição se everdadeira
- condição se falso

**Exemplo 10.4-** Repetindo o exemplo 32.Tome $a = 31$ e $b = 30$, analise qual dois dois objetos é o maior.

```r
a=30
b=30
ifelse(a>b,"a é maior","b é maior")
```

```
## [1] "b é maior"
```

O `ifelse()` não é apenas uma versão mais prática da estrutura `if` e `else`, ele também é uma analise do R vetorizavel. Falaremos disso mais a frente.

## 10.2) Estrutura de Repetição

Imagine que você tem um código em R, e precisa que uma mesma sequência de tarefas sejam executadas repetidamente diante de algum parâmetro, que pode ser convergência de um número, executar tarefas em um determinado objeto, dentre outras opções. Um exemplo claro é que você tem uma tabela com seus produtos e preços, e você deseja dobrar todos os preços.

| Produto | Preço | Novo Preço |
|---|---|---|
| Produto A | R$ 5,00 | |
| Produto B | R$ 15,00 | |
| Produto C | R$ 4,00 | |
| Produto D | R$ 6,00 | |
| Produto E | R$ 8,00 | |

Com o que foi aprendido até agora, você repetiria 5 vezes a ação de dobrar os preços até acabar a tabela. Improdutivo, não?

Para isso existem as estruturas de repetição, que servem para executar repetidamente uma série de ações.

### 10.2.1) Estrutura While

O while é um estrutura de repetição que executa uma ação n vezes enquanto uma determinada condição for atendida. Esta estrtura é largamente usada para convergências.

**Exemplo 10.5** Tome $a = 10$ faça uma estrutura imprimir o valor de $a$ e que seja iterado em uma unidade enquanto $a < 20$:

```r
a=10
while(a<20){
  print(a)
  a<-a+1
}
```

```
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
```

** Exemplo 10.6-** Vamos retomar o exemplo do ínicio do capítulo. Suponha que você é o dono de uma loja e você deseja dobrar o preço de todos os seus produtos:

| Produto | Preço | Novo Preço |
|---|---|---|
| Produto A | R$ 5,00 | |
| Produto B | R$ 15,00 | |
| Produto C | R$ 4,00 | |
| Produto D | R$ 6,00 | |
| Produto E | R$ 8,00 | |

```
Produto<-c("Produto A"," Produto B","Produto C", "Produto D","Produto E")
Preco<-c(5,15,4,6,8)
preco_novo<-NA

tabela_preco_produto<-data.frame(Produto,Preco,preco_novo)

produtos<-1

while(produtos<=5){

  tabela_preco_produto[produtos,"preco_novo"]<-2* tabela_preco_produto[produtos,"Preco"]
  produtos<-produtos+1
}

tabela_preco_produto
```

```
##      Produto Preco preco_novo
## 1  Produto A     5         10
## 2  Produto B    15         30
## 3  Produto C     4          8
## 4  Produto D     6         12
## 5  Produto E     8         16
```

### 10.2.2) Estrutura For

Para resolvermos o **Exemplo 10.6** tivemos que seguir os passos

- Atribuir um valor inicial à uma espécie de contador (produtos)
- Testar a condição
- Executar as ações
- Atualizar a variável contador

Um pouco trabalhoso, não? E este tipo de situação ilustra uma larga parcela do uso de estruturas de repetição. Assim para este tipo de situção é recomendado a estrutura `FOR`.

O `for` é uma estrutura que ja atualiza o valor da variável contadora com limites préviamente informados. No R, são considerados como limites um objeto (em geral vetores), em que `for` pecorrer.

**Exemplo 10.7-** Tome o vetor $v = [16, 18, 59, 35, 27, 37, 38]$ faça uma estrutura que imprima os valores do vetor:

```
v=c(16,18,59,35,27,37,38)

for (i in 1:length(v)){
  print(v[i])
}
```

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

```
## [1] 18
## [1] 59
## [1] 35
## [1] 27
## [1] 37
## [1] 38
```

**Exemplo 10.8-** Vamos retomar o exemplo 36 e refaze-lo com `for()`.Suponha que você é o dono de uma loja e você deseja dobrar o preço de todos os seus produtos:

| Produto | Preço | Novo Preço |
|---|---|---|
| Produto A | R$ 5,00 | |
| Produto B | R$ 15,00 | |
| Produto C | R$ 4,00 | |
| Produto D | R$ 6,00 | |
| Produto E | R$ 8,00 | |

```
Produto<-c("Produto A"," Produto B","Produto C", "Produto D","Produto E")
Preco<-c(5,15,4,6,8)
preco_novo<-NA

tabela_preco_produto<-data.frame(Produto,Preco,preco_novo)

for(i in nrow(tabela_preco_produto)){

  tabela_preco_produto[i,"preco_novo"]<-2*tabela_preco_produto[i,"Preco"]

}

tabela_preco_produto
```

```
##      Produto Preco preco_novo
## 1  Produto A     5         NA
## 2  Produto B    15         NA
## 3  Produto C     4         NA
## 4  Produto D     6         NA
## 5  Produto E     8         16
```

**Exemplo 10.9-** Vamos considerar quer você é o diretor de uma escola, nesta escola um aluno é aprovado se obtiver um rendimento maior ou igual à 70%. Esse rendimento é medido pela média aritimética das três provas p1,p2 e p3 respectivamente. Caso o aluno tenha média menor ou igual 5 ele está em recuperação, caso seja inferior à 5 este aluno é reprovado. Classifique o status de cada aluno.

```
alunos<-c("Anna","Paulo","Pedro","Serafin","Marco","Karina","Giuliana",
          "Diego","Natalia","Ingrid","Daiane")

p1<-c(2,3,10,5,7,8,5,2,4,10,8)
p2<-c(9,5,9,5,7,5,4,4,2,1,8)
p3<-c(10,6,8,8,8,10,8,5,9,2,9)

alunos<-data.frame(alunos,p1,p2,p3)

alunos$media<-(alunos$p1+alunos$p2+alunos$p3)/3
alunos$status<-NA
```

| Aluno | P1 | P2 | P3 |
|---|---|---|---|
| Anna | 2 | 9 | 10 |
| Paulo | 3 | 5 | 6 |
| Pedro | 10 | 9 | 8 |
| Serafin | 5 | 5 | 8 |
| Marco | 7 | 7 | 8 |
| Karina | 8 | 5 | 10 |
| Giuliana | 5 | 4 | 8 |
| Diego | 2 | 4 | 5 |
| Natalia | 4 | 2 | 9 |
| Ingrid | 10 | 1 | 2 |
| Daiane | 8 | 8 | 9 |

```
for (i in 1:nrow(alunos)){
  if (alunos[i,"media"]>=7){
    alunos[i,"status"]<-"Aprovado"
  } else if(alunos[i,"media"]<7 & alunos[i,"media"]>=5){
    alunos[i,"status"]<-"Recuperação"
  } else{
    alunos[i,"status"]<-"Reprovado"
  }
}
alunos
```

```
##      alunos p1 p2 p3    media      status
## 1      Anna  2  9 10 7.000000    Aprovado
## 2     Paulo  3  5  6 4.666667   Reprovado
## 3     Pedro 10  9  8 9.000000    Aprovado
## 4   Serafin  5  5  8 6.000000 Recuperação
## 5     Marco  7  7  8 7.333333    Aprovado
## 6    Karina  8  5 10 7.666667    Aprovado
## 7  Giuliana  5  4  8 5.666667 Recuperação
## 8     Diego  2  4  5 3.666667   Reprovado
## 9   Natalia  4  2  9 5.000000 Recuperação
## 10   Ingrid 10  1  2 4.333333   Reprovado
## 11   Daiane  8  8  9 8.333333    Aprovado
```

# 11) Funções

Funções são linhas de comando previamente escritas que são chamadas pelo seu nome seguido de `()`, são muito úteis para não termos que repetir inúmeras vezes alguns comandos.Até o momento usamos várias funções do R. Mas ele também permite você construa a sua própria função definindo chamando o comando `function()`. Que tem como estrutura básica:

```
function(input1,input2){

 output<-comandos

  return(output)
}
```

Onde:

- *input*<-São os parâmetros que entrarão na função e serão usados
- *output*<- É que será retornado pela função
- *return*()<- função que libera o output

**Exemplo 11.1-** Construa uma função que retorna a média de dois números:

```
media<-function(a,b){

  m<-(a+b)/2

  return(m)
}

media(3,5)
```

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

OK!! Uma função, um resultado! Mas e se quisermos que uma função retorne mais de um resultado? Faremos uso do que foi aprendido no capitulo 8: as listas.

**Exemplo 11.2-** tome duas variáveis $a$ e $b$, crie uma função que retorne a soma, a média e a multiplicação entre esses dois números.

```
calcula_tudo<-function(a,b){
  media<-(a+b)/2
  soma<-a+b
  multiplicacao<-a*b

  return(list(soma=soma,media=media,multiplicacao=multiplicacao))
}

calcula_tudo(4,10)
```

```
## $soma
## [1] 14
##
## $media
## [1] 7
##
## $multiplicacao
## [1] 40
```

### 11.1) Exercícios

1. Crie uma função que calcule a área e o perímetro de um triângulo retângulo, com os as entradas:

a. Cateto 1
b. Cateto 2

O output será uma lista com o primeiro ojeto sendo a àrea e o segundo sendo o perímetro.

## 12) A função Which

A função `which()` é uma função muito útil no R. Essa função faz uma análise com um unico comando se um objeto (Coluna ou linha de data frame, vetores) atende uma condição. o Output dessa função são os índices

dos objetos que atendem a condição solicitada.

**Exemplo 11.3-** Tome o vetor $v = [15, 20, 35, 38, 24, 50]$, analise os elementos se são múltiplos de 5.

```
v=c(15,20,35,38,24,50)

which(v %%5==0)
```

```
## [1] 1 2 3 6
```

Como o output da função são índices, podemos aplica-los para fazer filtros nos objetos.

**Exemplo 11.4-** Repita o exemplo acima e crie um vetor $v2$ que receba os valores de $v1$ que sejam múltiplos de 5.

```
v=c(15,20,35,38,24,50)

v2=v[which(v %%5==0)]
```

A função which também é facilmente aplicável à datas frames.

**Exemplo 11.5-** Tome a tabela com idades abaixo, retorne quem é a pessoa mais velha?

| Nomes | Idades |
|---|---|
| Claudia | 30 |
| Paulo | 28 |
| Fernando | 45 |
| Gisele | 36 |
| Anna | 27 |
| Pedro | 18 |
| João | 21 |
| Matheus | 25 |

```
nomes<-c("Claudia","Paulo","Fernando","Gisele","Anna","Pedro","João","Matheus")
idades<-c(30,28,45,36,27,18,21,25)

frame_pessoas<-data.frame(nomes,idades)

maior_idade<-max(idades)

pessoa_mais_velha<-frame_pessoas[which(frame_pessoas$idades==maior_idade),]

pessoa_mais_velha
```

```
##      nomes idades
## 3 Fernando     45
```

# 13) Exercícios

## Base train.csv

Leia o arquivo `train.csv`,na pasta *datasets*, com os dados de alguns passageiros do acidente do Titanic como informado no **Capítulo 2.** e responda as questões:

1. Quantas variáveis possui o arquivo? Quantas observações o arquivo tem?
2. Quais são as classes das variáveis?
3. Qual é a média das dos preços dos tickets?
4. Faça um filtro na tabela e crie dois outros data frames. Um para o genero masculino e o outro para o genero feminino.
5. Crie duas listas uma para informações do data frame do genero feminino e outro para o genero masculino. Cada lista deve ser composta:
   - Número total de Passageiros
   - Número de Sobreviventes
   - Numero de passageiros na primeira classe
   - preço do ticket
   - numero de parentes\filhos

Com base nas listas criadas, responda:

6. Qual genero teve o maior número de pessoas embarcadas?
7. Qual genero sobreviveu mais?
8. Qual genero teve a maior média do número de parentes?

## Base Human development index (HDI).csv

Leia a base `Human development index (HDI).csv` com os dados da evolução do IDH (Índice de Desenvolvimento Humano) dos países e responda os questionamentos.

1. Crie uma função que classifique os países (em uma coluna extra na tabela) em 2014 de acordo com a tabela:

Table 2: My caption

| Valor IDH | Classificação |
|---|---|
| IDH $\leq$ 0.534 | Baixo |
| 0.534 <IDH $\leq$ 0.710 | Médio |
| 0.710<IDH $\leq$ 0.796 | Alto |
| IDH >0.796 | Muito Alto |

2. Qual pais cresceu mais em relação à 2013?
3. Qual pais caiu mais em relação à 2013?
4. Quantos enstão com classificação baixa?
5. Qual é a posição do Brasil?

### Base dados_anp2.csv

Crie uma variavel chamada anp que receba a leitura de dados ***dados_anp2.csv***.

- Dica 1: na leitura use o argumento stringsAsFactors = FALSE
- Dica 2: transforme a coluna PRECO_COMPRA em númerica, com a função as.numeric()
- Dica 3: faça os valores nulos de PRECO_COMPRA receberem na.

1. Faça o summary para entender a sua base.
2. Quantos preços foram coletados?
3. Crie uma tabela com a frequência de postos por combústivel, atribua essa tabela à variável "quantidade_postos"
4. Qual combustível teve menos preços coletados? Isso faz sentido?
5. Qual é o posto com menor preço de venda? É confiável essa fonte (dica: olhe para o fornecedor e a bandeira.)
6. Crie a tabela dados_etanol, que é um filtro do data frame ***anp***. Sumarize dados_etanol por UF e média dos preços de venda do etanol.
7. Qual é o estado com a menor média de preços de venda do etanol. Isso faz sentido?
8. Exporte para o mesmo arquivo em excel os data frames:

- anp
- dados_etanol
- etanol_sumarizado

# References

Bowles, M. 2015. "Ensemble Packages in R. See: Http://Blog. Revolutionanalytics. Com/2014/04/Ensemble-Packages-in-R. Html." Accessed.

Team, R Core. 2017. "R: A Language and Environment for Statistical Computing. Vienna: R Foundation for Statistical Computing. Available Online at: H Ttp." *Www. R-Project. Org.*