



## ROTEIRO 1

### Instalando o R/Rstudio e Noções Básicas

#### 1. Objetivo

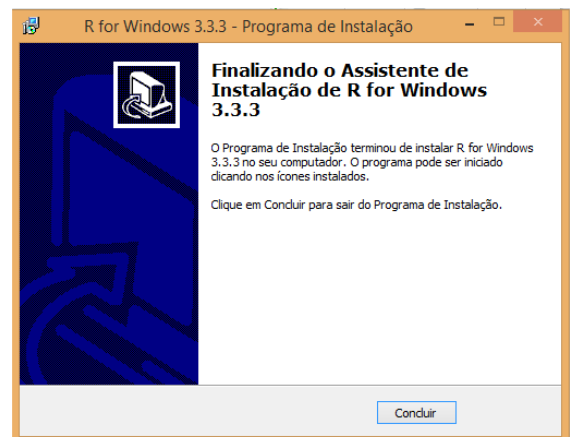
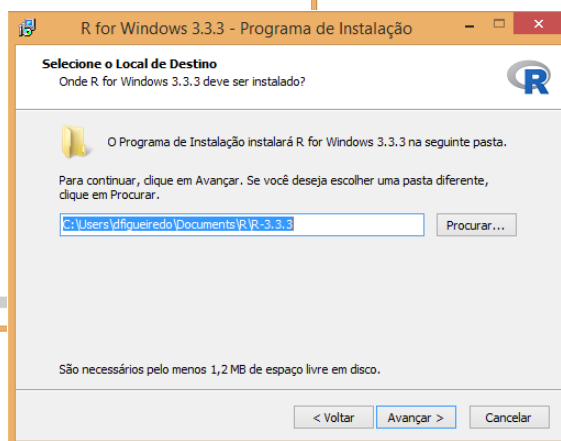
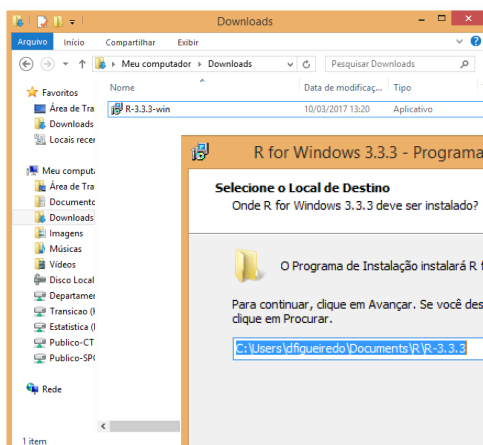
Guiar o aluno no processo de instalação do R/Rstudio e auxiliar na compreensão das noções básicas para a sua utilização.

#### 2. Expectativa

Apresentar os passos básicos para a instalação do R/Rstudio, para que o aluno possa construir mais um ambiente de estudo, em âmbito particular e com isso poder treinar e construir conceitos de análise estatística de dados bem como suas operações e aplicação.

#### 3. Passos para a instalação

- Para a instalação no Windows:
  1. Acesse o CRAN-R no site: <https://cran.r-project.org/bin/windows/base/> e procure por Download R.3.6.1 for Windows na página.
  2. Esperar a conclusão do arquivo executável.
  3. Procure o local onde o arquivo foi salvo (em geral, na pasta Downloads), e execute o arquivo. O Programa de Instalação abrirá uma janela para que você Selecione o Local de Destino. Uma vez selecionado, clique em Avançar.



4. Pronto! Clique em Concluir e o R estará instalado no seu computador!

- Para a instalação no Ubuntu:

5. Abrir o terminal: **Ctrl+Alt+t**

6. Instalar o R no Ubuntu:

```
$ sudo apt-get install r-base r-base-dev
```

7. Adicionar o repositório:

```
$ sudo echo deb http://cran.rstudio.com/bin/linux/ubuntu xenial/ | sudo  
tee -a /etc/apt/sources.list
```

8. Atualizar:

```
$ sudo apt-get update
```

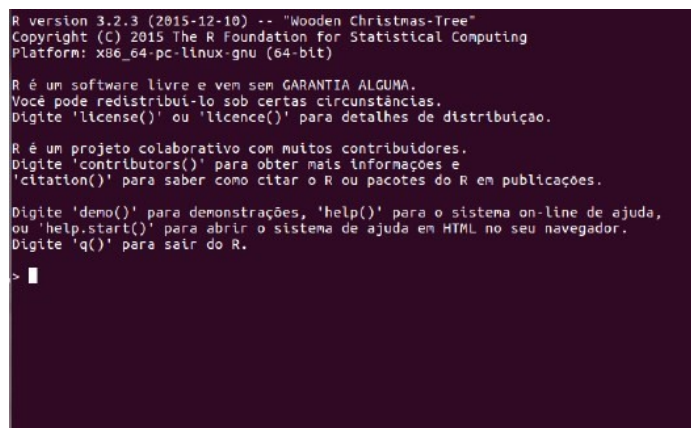
9. Instalar:

```
$ sudo apt-get install r-base r-base-dev
```

10. Para execução do software, ainda no terminal, digitar:

```
$ R
```

E uma nova janela abrirá com a execução do programa.



```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"  
Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu (64-bit)  
  
R é um software livre e vem sem GARANTIA ALGUMA.  
Você pode redistribuí-lo sob certas circunstâncias.  
Digite 'license()' ou 'licence()' para detalhes de distribuição.  
  
R é um projeto colaborativo com muitos contribuidores.  
Digite 'contributors()' para obter mais informações e  
'citation()' para saber como citar o R ou pacotes do R em publicações.  
  
Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,  
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.  
Digite 'q()' para sair do R.  
> █
```


## Instalação do RStudio

11. Acesse o RStudio no site: <https://www.rstudio.com/products/rstudio/download/> e escolha a versão RStudio Desktop Free, clicando em Download.

12. Escolha a plataforma que melhor se aplica ao seu Sistema Operacional

13. Nessa página você tem duas opções:

- Se você tiver acesso do tipo administrador, baixe a versão que está na lista de *Installers for Supported Platforms*. Em seguida a instalação será bem simples: fazer o download, abrir o instalador e seguir as instruções, clicando no botão “Avançar”.
- Se você não tiver acesso de administrador, faça o download da versão que está na lista Zip/Tarballs. (veja imagem acima)



[rstudio::conf](#)
[Products](#)
[Resources](#)
[Pricing](#)
[About Us](#)
[Blogs](#)
[Q](#)

**RStudio Desktop 1.0.136** — Release Notes

RStudio requires R 2.11.1+. If you don't already have R, download it [here](#).

### Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.0.136 - Windows Vista/7/8/10	81.9 MB	2016-12-21	93b3f307f567c33f7a4db4c114099b3e
RStudio 1.0.136 - Mac OS X 10.6+ (64-bit)	71.2 MB	2016-12-21	12d6d6ade0203a2fcef6fe3dea65c1ae
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (32-bit)	85.5 MB	2016-12-21	0a20fb89d8aeb39b329a640ddadd2c5
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (64-bit)	92.1 MB	2016-12-21	2a73b88a12a9fba9f6251cec8b41340
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	84.7 MB	2016-12-21	fa6179a7855bf0f939a34c169da45fd
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	85.7 MB	2016-12-21	2b3a148ded380b704e58496befb55545

### Zip/Tarballs

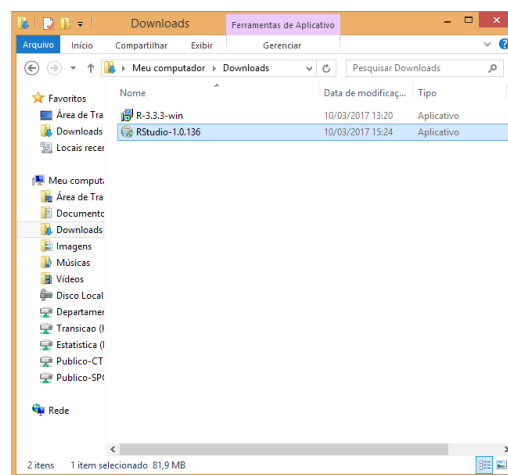
Zip/tar archives	Size	Date	MD5
RStudio 1.0.136 - Windows Vista/7/8/10	117.5 MB	2016-12-21	f415939bf5012c0ab127c7c7c9600be
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (32-bit)	86.2 MB	2016-12-21	fca75f953dd425694b7fd4335bd29165
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (64-bit)	93.2 MB	2016-12-21	7cf0092653aa44fc76325a8f1325fb1f
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	85.4 MB	2016-12-21	30c89299d30ec03b38098e51e9bf49b8
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	86.6 MB	2016-12-21	ea2a262f650e92f568f48edc1c093902

### Source Code

A tarball containing source code for RStudio v1.0.136 can be downloaded from [here](#)

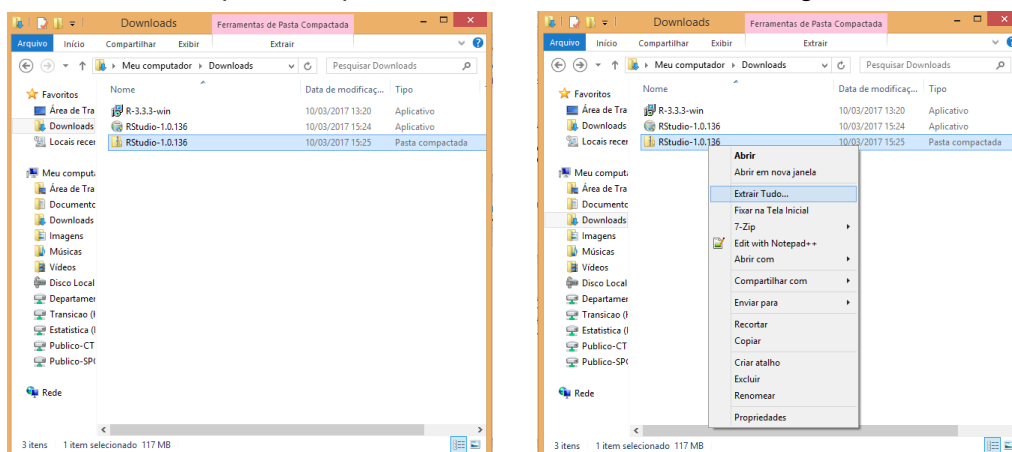
## Instalando no Windows se você for administrador

- Clique duas vezes no arquivo que você baixou da página do RStudio (ver imagem ao lado) e siga as instruções de instalação.

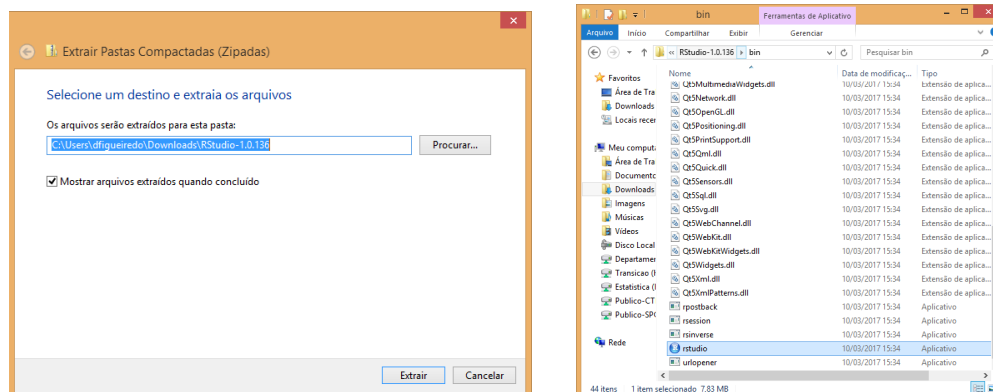


## Instalando no Windows se você não for administrador

- Se você não for administrador, você deve ter feito o download de um arquivo do tipo `.zip`, que contém o código do RStudio. É o arquivo selecionado na imagem da esquerda. Clique com o botão direito neste arquivo e depois em *Extrair Tudo* conforme a imagem da direita.



16. Você verá uma tela como a imagem a seguir. Não mude nada e clique em extrair. Espere o Windows completar a extração.



17. Agora, na pasta Downloads, a pasta que deixamos como local de extração (imagem acima), você terá uma pasta chamada: **Rstudio-1.0.136**.

18. Abra essa pasta e entre na subpasta com nome **bin**. Em seguida, procure pelo arquivo chamado **rstudio** e clique duas vezes. Isso abrirá o RStudio. Recomendo fixar o programa na barra de tarefas para não ter que ficar procurando nessa pasta sempre que quiser abri-la.

Observação: se você excluir a pasta que extraímos, o RStudio parará de funcionar.

## Instalando no Ubuntu

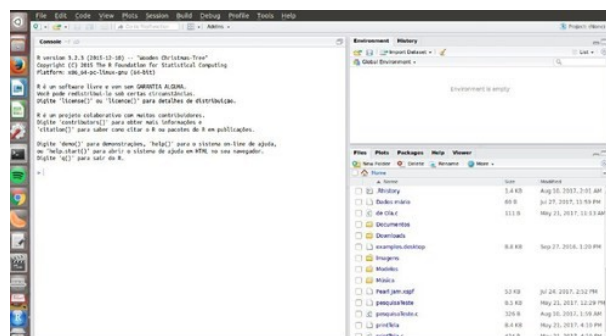
19. Instalar o gdebi:

```
$ sudo apt install gdebi
```

20. Abrir pelo terminal a pasta onde o arquivo foi baixado. Instalando o Rstudio (o arquivo foi salvo na pasta Downloads da máquina). O resultado esperado se encontra na abaixo. Na figura da esquerda, o processo de instalação. Na figura da direita, a execução do programa.

```
$ sudo gdebi -n rstudio-xenial-1.0.153-amd64.deb
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading state information... Done
(Lendo banco de dados ... 330095 ficheiros e directórios actualmente instalados.)
A preparar para descompactar rstudio-xenial-1.0.153-amd64.deb ...
A descompactar rstudio (1.0.153) sobre (1.0.153) ...
Configurando rstudio (1.0.153) ...
A processar 'triggers' para shared-mime-info (1.5-2ubuntu0.1) ...
A processar 'triggers' para hicolor-icon-theme (0.15-0ubuntu1) ...
A processar 'triggers' para desktop-file-utils (0.22-1ubuntu5.1) ...
A processar 'triggers' para bamfdaemon (0.5.3-bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
A processar 'triggers' para gnome-menus (3.13.3-6ubuntu3.1) ...
A processar 'triggers' para mime-support (3.59ubuntu1) ...
```



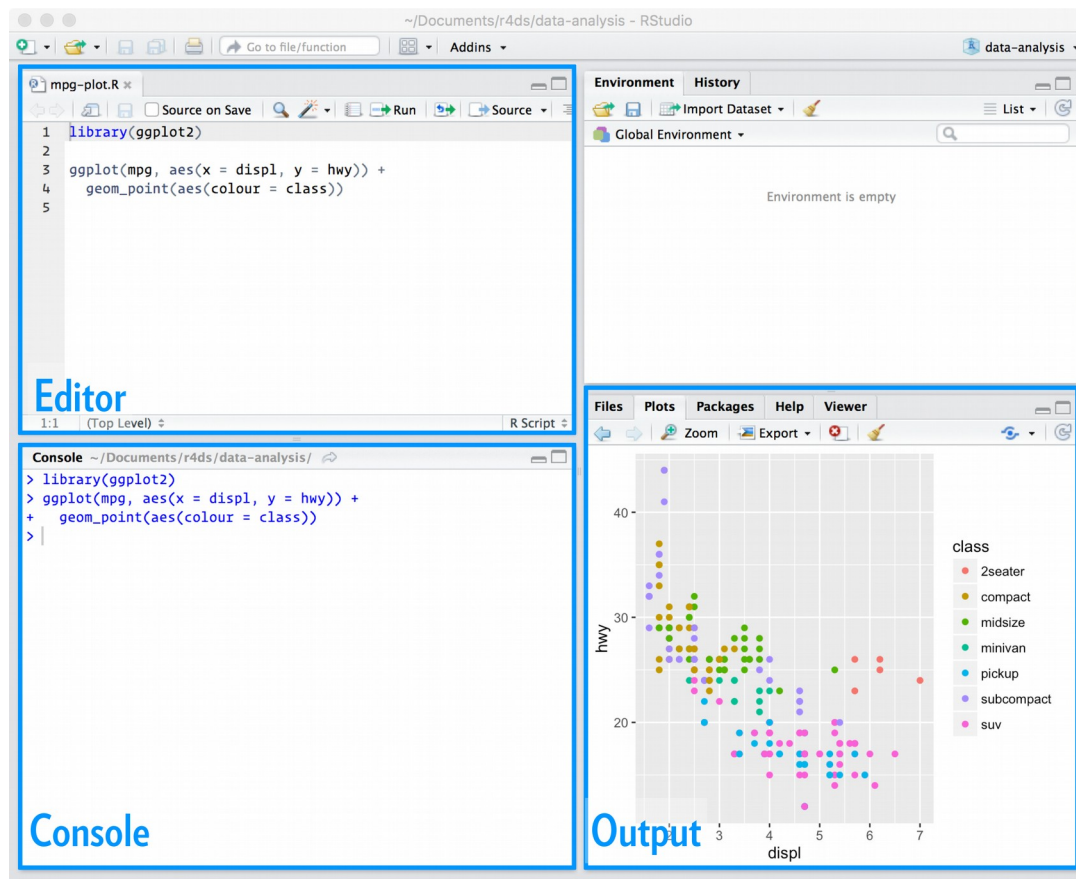
- Para outros sistemas operacionais, acesse o CRAN-R no site: <https://cran.r-project.org/> e procure pelo arquivo de Instruções (*Installation and other instructions*).

## 4. Noções Básicas

O R é uma linguagem de programação orientada a objetos bem intuitiva e a ideia do uso do RStudio é para facilitar o seu uso. O ambiente contém quatro janelas que se dividem da seguinte forma: o editor, o console, o environment e o output. Eles vêm nesta ordem, e depois você pode organizá-los da forma que preferir.

Listamos abaixo as funções dos principais painéis:

- Editor/Scripts: é onde escrevemos nossos códigos.
- Console: é onde rodamos o código e recebemos as saídas. O R vive aqui!
- Environment: painel com todos os objetos criados na sessão.
- Files: mostra os arquivos no diretório de trabalho. É possível navegar entre diretórios.
- Plots: painel onde os gráficos serão apresentados.
- Help: janela onde a documentação das funções serão apresentadas.
- History: painel com um histórico dos comandos rodados.



## R como calculadora

Pelo console, é possível executar qualquer comando do R.

```
1:30
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30
```

Esse comando é uma forma simplificada de criar um vetor de inteiros de 1 a 30. Os números que aparecem entre colchetes ([1] e [24]) indicam o índice do primeiro elemento impresso em cada linha.

Tente jogar no console  $2 * 2 - (4 + 4) / 2$ . Pronto! Com essa simples expressão você já é capaz de pedir ao R para fazer qualquer uma das quatro operações aritméticas básicas. A seguir, apresentamos uma lista resumindo como fazer as principais operações no R.

1 + 1 # adição	## [1] 1.666667
## [1] 2	4 ^ 2 # potência
4 - 2 # subtração	## [1] 16
## [1] 2	5 %% 3 # resto da divisão de 5 por 3
2 * 3 # multiplicação	## [1] 2
## [1] 6	5 %/% 3 # parte inteira da divisão de 5 por 3
5 / 3 # divisão	## [1] 1

Além do mais, as operações e suas precedências são mantidas como na matemática, ou seja, divisão e multiplicação são calculadas antes da adição e subtração. E os parênteses nunca são demais!

Uma outra forma de executar uma expressão é escrever o código no editor e teclar **Ctrl + Enter** ou **Ctrl + R**. Assim, o comando é enviado para o console, onde é diretamente executado.

Se você digitar um comando incompleto, como `5 +`, e apertar **Enter**, o R mostrará um `|`, o que não tem nada a ver com somar alguma coisa. Isso significa que o R está esperando que você complete o seu comando. Termine o seu comando ou aperte **Esc** para recomeçar.

```
> 5 -  
+  
+ 5  
[1] 0
```

Se você digitar um comando que o R não reconhece, ele retornará uma mensagem de erro.

**NÃO ENTRE EM PÂNICO!**

Ele só está avisando que não conseguiu interpretar o comando. Você pode digitar outro comando normalmente em seguida.

```
> 5 % 5  
Error: unexpected input in "5 % 5"  
> 5 - 5  
[1] 0
```

## Objetos

O R te permite salvar dados dentro de um objeto. Para isso, utilizamos o operador `<-`.

No exemplo abaixo, salvamos o valor 1 em `a`. Sempre que o R encontrar o símbolo `a`, ele vai substituí-lo por 1.

```
a <- 1  
a  
## [1] 1
```

**Atenção!**

O R diferencia letras maiúsculas e minúsculas, isto é, `a` é considerado um objeto diferente de `A`.

## Objetos atômicos

Existem cinco classes básicas ou “atômicas” no R:

character	numeric	
integer	complex	logical

Veja alguns exemplos:

```
# characters  
"a"  
## [1] "a"  
"1"  
## [1] "1"  
"positivo"  
## [1] "positivo"  
"Error: objeto x não encontrado"  
## [1] "Error: objeto x não encontrado"  
# numeric  
1  
## [1] 1  
0.10  
## [1] 0.1  
0.95  
## [1] 0.95  
pi
```

```
## [1] 3.141593
# integer
1L
## [1] 1
5L
## [1] 5
10L
## [1] 10
# complex (raramente utilizado para análise de dados)
2 + 5i
## [1] 2+5i
# logical
TRUE
## [1] TRUE
FALSE
## [1] FALSE
```

Para saber a classe de um objetivo, você pode usar a função `class()`.

```
x <- 1
class(x)
## [1] "numeric"
y <- "a"
class(y)
## [1] "character"
z <- TRUE
class(z)
## [1] "logical"
```

## Vetores

Vetores no R são os objetos mais simples que podem guardar objetos atômicos.

```
vetor1 <- c(1, 2, 3, 4)
vetor2 <- c("a", "b", "c")
vetor1
## [1] 1 2 3 4
vetor2
## [1] "a" "b" "c"
```

Um vetor tem sempre a mesma classe dos objetos que guarda.

```
class(vetor1)
## [1] "numeric"
class(vetor2)
## [1] "character"
```

De forma bastante intuitiva, você pode fazer operações com vetores.

```
vetor1 - 1
## [1] 0 1 2 3
```

Quando você faz `vetor1 - 1`, o R subtrai `1` de cada um dos elementos do vetor. O mesmo acontece quando você faz qualquer operação aritmética com vetores no R.

```
vetor1 / 2
vetor1 * 10
```

Você também pode fazer operações que envolvem mais de um vetor:

```
vetor1 * vetor1
## [1] 1 4 9 16
```

Neste caso, o R alinhará os dois vetores e multiplicar elemento por elemento. Isso pode ficar um pouco confuso quando os dois vetores não possuem o mesmo tamanho:

```
vetor2 <- 1:3
vetor * vetor2
## Error in eval(expr, envir, enclos): object 'vetor' not found
```

O R alinhou os dois vetores e, como eles não possuíam o mesmo tamanho, foi repetindo o vetor menor até completar o vetor maior. Esse comportamento é chamado de reciclagem e é útil para fazer operações elemento por elemento (vetorizadamente), mas às vezes pode ser confuso. Com o tempo, você aprenderá a se aproveitar dele.

## Misturando objetos

Vetores são homogêneos

Os elementos de um vetor são sempre da mesma classe. Ou todos são numéricos, ou são todos character, ou todos são lógicos etc. Não dá para ter um número e um character no mesmo vetor, por exemplo.

Se colocarmos duas ou mais classes diferentes dentro de um mesmo vetor, o R vai forçar que todos os elementos passem a pertencer à mesma classe. O número 1.7 viraria "1.7" se fosse colocado ao lado de um "a".

```
y <- c(1.7, "a") ## character
y <- c(TRUE, 2) ## numeric
y <- c(TRUE, "a") ## character
```

A ordem de precedência é:

DOMINANTE character > complex > numeric > integer > logical RECESSIVO

## Forçando classes explicitamente

Você pode coagir um objeto a ser de uma classe específica com as funções as.character(), as.numeric(), as.integer() e as.logical(). É equivalente à função convert() do SQL.

```
x <- 0:4
class(x)
## [1] "integer"
as.numeric(x)
## [1] 0 1 2 3 4
as.logical(x)
## [1] FALSE TRUE TRUE TRUE TRUE
as.character(x)
## [1] "0" "1" "2" "3" "4"
```

Se o R não entender como coagir uma classe na outra, ele soltará um warning informando que colocou NA no lugar.

```
x <- c("a", "b", "c")
as.numeric(x)
## Warning: NAs introduced by coercion
## [1] NA NA NA
```

Observação

O NA tem o mesmo papel que o null do SQL. Porém, há um NULL no R também, com diferenças sutis.



## Matrizes

Matrizes são vetores com duas dimensões (e por isso só possuem elementos de uma mesma classe).

```
m <- matrix(1:6, nrow = 2, ncol = 3)
m
##      [,1] [,2] [,3]
## [1,]  1  3  5
## [2,]  2  4  6
dim(m) # função dim() retorna a dimensão do objeto.
## [1] 2 3
```

Repare que os números de 1 a 6 foram dispostos na matriz coluna por coluna (column-wise), ou seja, preenchendo de cima para baixo e depois da esquerda para a direita.

### Operações úteis em matrizes

```
m[3, ] # seleciona a terceira linha
m[, 2] # seleciona a segunda coluna
m[1, 2] # seleciona o primeiro elemento da segunda coluna
t(m)    # matriz transposta
m %*% n # multiplicação matricial
solve(m) # matriz inversa de m
```

## Fatores

Fatores podem ser vistos como vetores de inteiros que possuem rótulos (levels).

```
sexo <- c("M", "H", "H", "H", "M", "M", "H")
fator <- as.factor(sexo)
fator
## [1] M H H H M M H
## Levels: H M
as.numeric(fator)
## [1] 2 1 1 1 2 2 1
```

Eles são úteis para representar uma variável categórica (nominal e ordinal).

Na modelagem, eles serão tratados de maneira especial em funções como `lm()` e `glm()`.

A função `levels()` retorna os rótulos do fator:

```
levels(fator)
## [1] "H" "M"
```

## Listas

Listas são um tipo especial de vetor que aceita elementos de classes diferentes.

```
x <- list(1:5, "Z", TRUE, c("a", "b"))
x
## [[1]]
## [1] 1 2 3 4 5
##
## [[2]]
## [1] "Z"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] "a" "b"
```

É um dos objetos mais importantes para armazenar dados e vale a pena saber manuseá-los bem. Existem muitas funções que fazem das listas objetos incrivelmente úteis.

Criamos uma lista com a função `list()`, que aceita um número arbitrário de elementos. Listas aceitam QUALQUER tipo de objeto. Podemos ter listas dentro de listas, por exemplo.

Como para quase todas as classes de objetos no R, as funções `is.list()` e `as.list()` também existem. Na lista `pedido` abaixo, temos `numeric`, `Date`, `character`, vetor de `character` e `list` contida em uma lista:

```
pedido <- list(pedido_id = 8001406,
              pedido_registro = as.Date("2017-05-25"),
              nome = "Athos",
              sobrenome = "Petri Damiani",
              cpf = "12345678900",
              email = "athos.damiani@gmail.com",
              qualidades = c("incrível", "impressionante"),
              itens = list(
                list(descricao = "Ferrari",
                     frete = 0,
                     valor = 500000),
                list(descricao = "Dolly",
                     frete = 1.5,
                     valor = 3.90)
              ),
              endereco = list(entrega = list(logradouro = "Rua da Glória",
                                              numero = "123",
                                              complemento = "apto 71"),
                              cobranca = list(logradouro = "Rua Jose de Oliveira Coutinho",
                                              numero = "151",
                                              complemento = "5o andar")
              )
)
```

### Operações úteis

```
pedido$cpf      # elemento chamado 'cpf'
pedido[1]       # nova lista com apenas o primeiro elemento
pedido[[2]]     # segundo elemento
pedido["nome"]  # nova lista com apenas o elemento chamado 'nome'
```

Certamente você se deparará com listas quando for fazer análise de dados com o R. Nos tópicos mais aplicados, aprofundaremos sobre o tema.

### data.frame

Um `data.frame` é o mesmo que uma tabela do SQL ou um spreadsheet do Excel, por isso são objetos muito importantes.

Usualmente, seus dados serão importados para um objeto `data.frame`. Em grande parte do curso, eles serão o principal objeto de estudo.

Os `data.frame`'s são listas especiais em que todos os elementos possuem o mesmo comprimento. Cada elemento dessa lista pode ser pensado como uma coluna da tabela. Seu comprimento representa o número de linhas.

Já que são listas, essas colunas podem ser de classes diferentes. Essa é a grande diferença entre `data.frame`'s e matrizes.

Algumas funções úteis:

- `head()` - Mostra as primeiras 6 linhas.
- `tail()` - Mostra as últimas 6 linhas.
- `dim()` - Número de linhas e de colunas.
- `names()` - Os nomes das colunas (variáveis).
- `str()` - Estrutura do data.frame. Mostra, entre outras coisas, as classes de cada coluna.
- `cbind()` - Acopla duas tabelas, lado a lado.
- `rbind()` - Empilha duas tabelas.

O exemplo abaixo mostra que uma lista pode virar `data.frame` se todos os elementos tiverem o mesmo comprimento.

```
minha_lista <- list(x = c(1, 2, 3), y = c("a", "b"))
as.data.frame(minha_lista)
## Error in (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE, : arguments imply
differing number of rows: 3, 2
minha_lista <- list(x = c(1, 2, 3), y = c("a", "b", "c"))
as.data.frame(minha_lista)
##      x y
## 1 1 a
## 2 2 b
## 3 3 c
```

## 5. Referências

1. Comandos adaptados de <http://material.curso-r.com/instalacao/>
2. OLIVEIRA, Paulo Felipe de; GUERRA, Saulo; MCDONNEL, Robert. Ciência de Dados com R – Introdução. Brasília: Editora IBPAD, 2018.
3. Livro disponível em <https://cran.r-project.org/doc/contrib/Landeiro-Introducao.pdf>