

Big Data Analytics

Introdução ao uso do R / RStudio



Agenda

- Histórico e motivação;
- Instalar a linguagem R e o RStudio;
- Compreender conceitualmente a linguagem R;
- Identificar os objetos do R.



Histórico e motivação



Um pouco de história

- O principal objetivo da linguagem R é analisar dados.
- Criadores: Ross Ihaka e Robert Gentleman, membros do departamento de Estatística da Universidade de Auckland, Nova Zelândia.



Um pouco de história

- A linguagem R recebeu este nome por ser baseada na linguagem S, desenvolvida no AT&T Bell Labs, e como homenagem aos dois criadores, ambos com nomes que iniciam pela mesma letra.
- É uma linguagem de programação interpretada, voltada para a manipulação de dados e apresentação gráfica de resultados.



Linguagem S

- Linguagem de programação estatística criada por John M. Chambers com participação de Rick Becker e Allan Wilks em suas versões iniciais. Todos trabalhavam na Bell Laboratories.
- Criada entre 1975 e 1976.
- As implementações modernas são:
 - R;
 - S-Plus.



A linguagem R

- A linguagem R é *open source*. Seu código fonte está sob a *Free Software Foundation's GNU General Public License*.
- Criada entre o final dos anos 1990 e o início dos anos 2000.
- Versões para diversos sistemas operacionais, sem alterações significativas:
 - Windows;
 - MacOS;
 - Linux.



A linguagem R

- É mais do que um software estatístico:
 - Ambiente que permite explorar dados interativamente;
 - À medida que a análise evolui, é uma linguagem de programação completa para desenvolver e automatizar soluções e/ou desenvolver pacotes de expansão que abordem necessidades específicas;
 - Ferramenta poderosa para manipulação, processamento, visualização e análise de dados, bem como simulações e modelagem estatísticas.

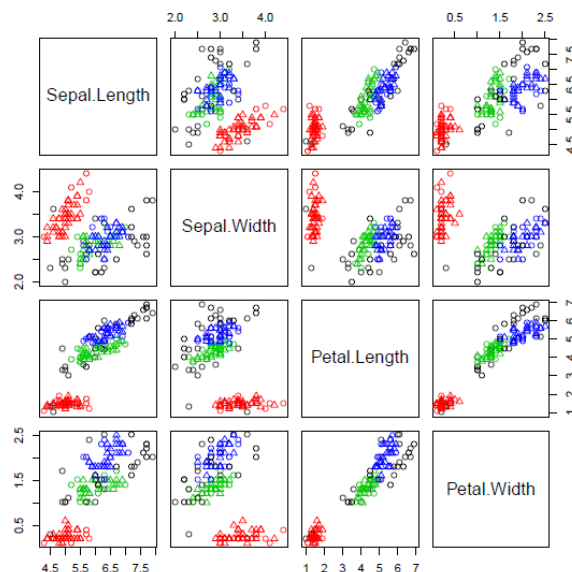


A linguagem R

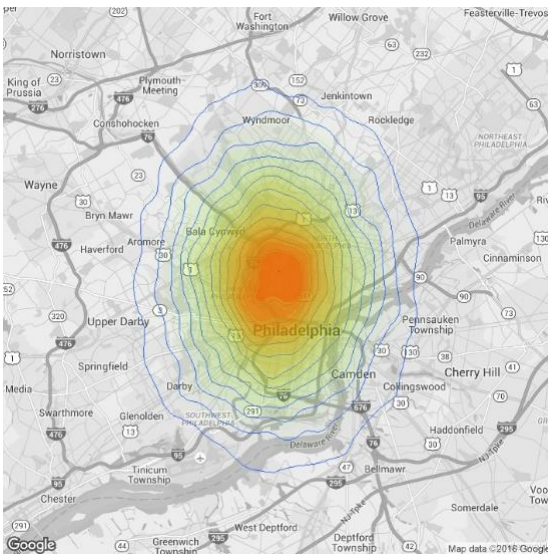
- Grande e ativa comunidade de usuários;
 - Empresas:
 - Google;
 - AT&T;
 - Empresas da área de investimento e finanças.
 - Academia:
 - Laboratórios;
 - Centros de pesquisa;
 - Professores de diversas universidades espalhadas pelo mundo.
- Mais de 4.000 pacotes gratuitos e abertos para download.



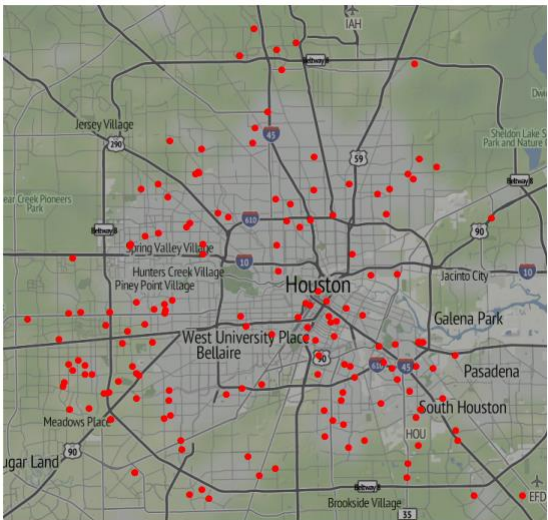
Exemplos



Exemplos



Exemplos



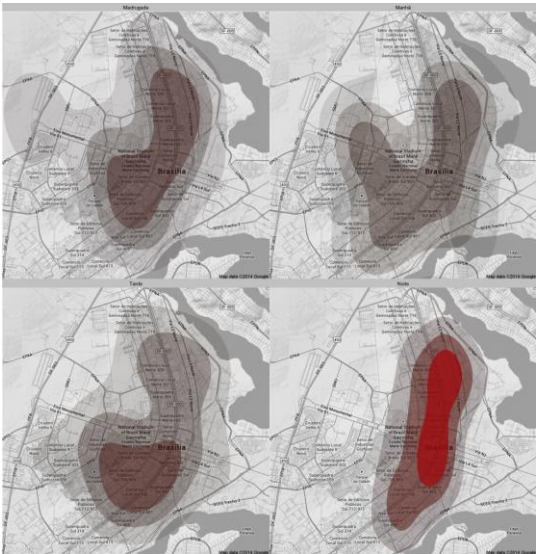
Exemplos



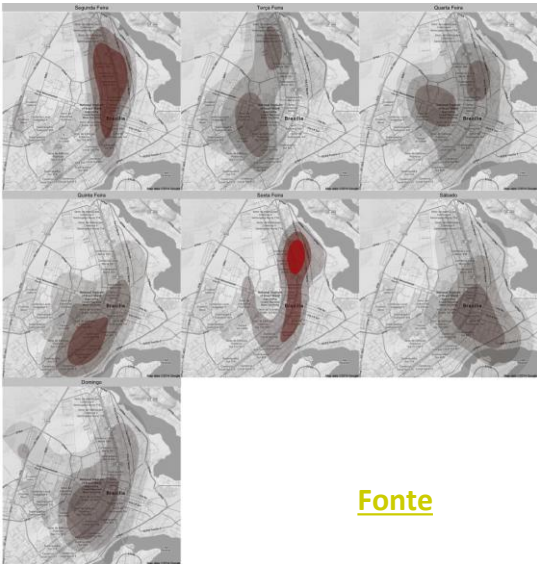
Exemplos



Exemplos



Exemplos



Fonte



Instalar a linguagem R e o RStudio



Instalação

- Os sites principais relacionados à linguagem R são os seguintes:
 - [The R Project for Statistical Computing](#)
 - Site oficial do projeto R.
 - [The Comprehensive R Archive Network](#)
 - (CRAN) Repositório de arquivos para instalação da ferramenta e bibliotecas de apoio.



Instalação

- Faça o download da instalação principal do *mirror* (espelho) do CRAN, localizado na Fundação Oswaldo Cruz.
 - [Download and Install R](#)
- Uma outra opção é o Microsoft R, que traz algumas melhorias em relação à *engine* R padrão (pacotes *multithread* nativos, dentre outras), sendo totalmente compatível com a *engine* R padrão.



32 ou 64 Bits?

- Para a maioria dos usuários, a instalação em modo nativo é a recomendada (de acordo com o sistema operacional).
- Informações interessantes em [R for Windows FAQ](#).



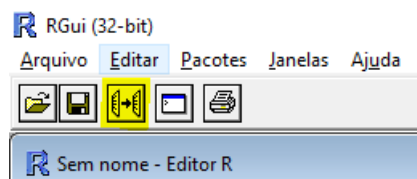
RGui

- Ao instalar o R, automaticamente também é instalada uma interface gráfica chamada RGui.
- Ao abrirmos o R, a primeira tela que veremos é a do console.
- Vamos abrir uma tela de editor de texto em “Arquivo” -> “Novo script”.
- Podemos organizar as janelas no menu “Janelas”.



RGui

- Escreveremos nossas instruções na janela do Editor.
- Podemos então executar uma linha ou um conjunto de linhas previamente selecionadas com o comando botão “Executar linha ou seleção”.



Executando algumas instruções

- Vamos digitar na janela do editor algumas instruções.
- Após cada linha, apertar o botão mencionado acima para executar o comando.
- O resultado aparecerá na janela Console.



Executando algumas instruções

- `5+5`
- `10-5`
- `10-20`
- `3*15`
- `43*12.5` (Atenção com o separador decimal!)
- `14/7`
- `21/45`
- `34/0`
- `plot(1:10)`
- Agora, digite `q()` na janela Console e tecle ENTER.

Auxiliando a programação

- Esta interface vem com o R porém é bastante limitada
- Para ganhar produtividade e facilitar a tarefa de programação, vamos utilizar uma IDE (*Integrated Development Environment* , ou Ambiente de Desenvolvimento Integrado) mais poderosa, chamada **RStudio**.



Auxiliando a programação

- O RStudio possui várias facilidades, tais como:
 - Realce contextual da linguagem;
 - Ferramentas mais avançadas de edição;
 - Recurso de *Autocomplete*, como apoio à programação;
 - Preenchimento automático de parênteses e chaves;
 - Interface intuitiva para objetos, gráficos, script, entre outras.

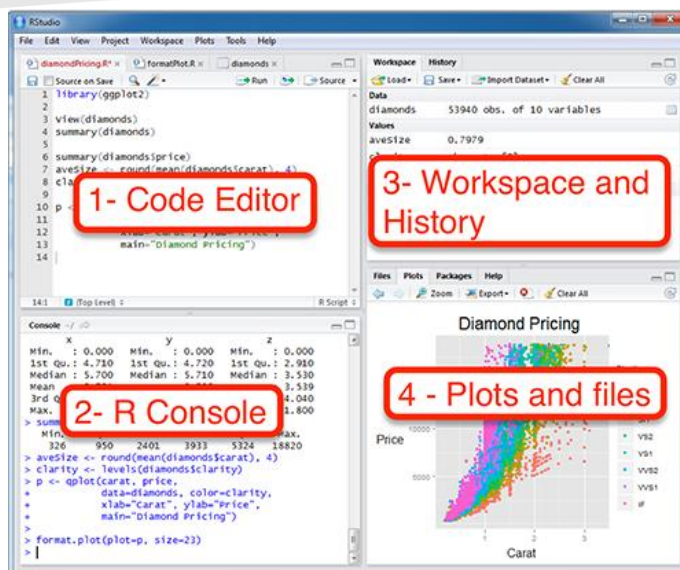


Auxiliando a programação

- A página principal do RStudio fica hospedada em <https://www.rstudio.com/>.
- A página do produto RStudio fica em <https://www.rstudio.com/products/rstudio/>.
- Vídeo [RStudio Overview](#).
- Verifique as características do produto Desktop.
- Faça o download da versão gratuita.
- Finalmente, instale o produto!



RStudio



Auxiliando a programação

- Script:
 - A tela superior esquerda do RStudio é o editor de texto onde você vai escrever seus Scripts.
 - Possui, por exemplo, realce de código para facilitar a programação.
- Console:
 - No canto inferior esquerdo fica o console.
 - Nada mais é do que uma seção aberta do R, em que os comandos são executados.
 - Devemos nos acostumar a escrever o código no Script ao invés de ficar escrevendo diretamente no console.

Auxiliando a programação

- Área de trabalho e histórico:
 - Ficam no canto superior direito. Os objetos criados na sessão e o histórico dos comandos podem ser acessados ali.



Auxiliando a programação

- Arquivos, Gráficos, Pacotes, Ajuda:
 - Ficam no canto inferior direito. Podemos explorar pastas e arquivos diretamente do RStudio na aba “Files”;
 - Os gráficos que forem feitos apareceram na aba “Plots”.
 - Os pacotes instalados em sua máquina estão listados em “Packages”.
 - As ajudas das funções aparecem em “Help”;
 - Finalmente, o “Viewer” serve para visualização de imagens/páginas em HTML e JavaScript.

Executando algumas instruções

- Vamos agora replicar o que fizemos no RGui, agora no RStudio, e verificar o comportamento da IDE.
- Após cada linha, apertar a combinação <CTRL + ENTER> para executar o comando digitado.
 - OBS.: acionar a tecla <ENTER> apenas adiciona a linha digitada ao script, sem executá-la.
- O resultado aparecerá na janela Console e/ou na janela Plot.



Executando algumas instruções

- 5+5
- 10-5
- 10-20
- 3*15
- 43*12.5 (Atenção com o separador decimal!)
- 14/7
- 21/45
- 34/0
- `plot(1:10)`
- Agora, digite `q()` na janela Console e tecle **ENTER**.

Executando algumas instruções

- Vamos agora fazer uma atribuição simples de um valor a uma variável e fazer uso desta variável recém criada.
- Vamos limpar a área de script e digitar as instruções abaixo (não executá-las ainda):

```
val_x <- 15
plot(1:val_x)
```

- Agora vamos selecionar as duas instruções e executá-las em sequência.
- Como vimos, podemos executar instruções ou conjunto de instruções com **CTRL+ENTER**.



Autocomplete, ajuda no contexto e ajuda geral

- O RStudio oferece os recursos de *Autocomplete* para funções e ajuda no contexto.
- Além disso, existe um sistema de ajuda bem desenvolvido dentro da IDE.
- Vamos ver como funciona!
 - `plo + TAB`
 - `plot(+ TAB`
 - `?plot`
 - `help(plot)`



Atalhos

- O uso da IDE pode se tornar muito eficiente com o conhecimento dos atalhos oferecidos pela ferramenta.
- Seguem alguns dos mais úteis:



Atalhos

- **CTRL + 1**: passa o cursor para o script;
- **CTRL + 2**: passa o cursor para o console;
- **SETA PARA CIMA** (no console): acessa o histórico de comandos anteriores.
- Ao abrir um novo script:
 - **CTRL + ALT + SETA PARA ESQUERDA OU DIREITA**: Navega entre as abas de script abertas.



Atalhos

- **CTRL + SHIFT + P**: “Previous command”, roda o último comando executado;
- **CTRL + SHIFT + ENTER**: “Source”, executa o script inteiro;
- **CTRL + S**: salva o script;
- **CTRL + L**: limpa o console;
- **CTRL + F**: busca (e substituição). Aceita REGEX;
- **ALT + SHIFT + K**: lista de atalhos.



Mais ajuda

- O RStudio oferece ainda buscas dentro do sistema de Ajuda.
 - `help.search("normal")`
- Outra fonte importante de apoio é o conhecido site *Stack Overflow*.
 - [Tópicos em português sobre o R](#);
 - [Tópicos em inglês sobre o R](#).



Mais ajuda

- Um grande auxílio à busca na web por tópicos relacionados à linguagem R é o site [Rseek](#).
- É uma ferramenta de busca no Google específica para a linguagem R devido à ambiguidade do nome da mesma.



Usando o R na AWS

- Uma boa opção para uso do R é a infraestrutura oferecida pela AWS.
- [Louis Aslett](#) oferece em seu site ótimos AMIs prontos para o uso de ambientes completos com R e RStudio na AWS.
- Até acesso ao Dropbox já vem configurado. Vale a pena utilizá-los.
- O site inclui instruções completas para a instalação.



Compreender conceitualmente a linguagem R



A linguagem R

- A linguagem R é uma linguagem de programação 'dinamicamente tipada', ou seja, podemos modificar os tipos de dados contidos em variáveis em programas que já estejam em execução, com isso, elimina-se a necessidade de conversões relacionadas aos tipos de dados.
- Devemos apenas tomar cuidado com a conversão na importação de dados (mais detalhes à frente).



A linguagem R

- A linguagem R é usada para manipulação de conjuntos de dados em geral, análises estatísticas e produção de documentos e apresentações centradas em dados.
- Também nos fornece uma ampla variedade de modelos lineares e não lineares, testes estatísticos, análise de séries temporais, modelos de agrupamentos, além de ser altamente extensível por meio de pacotes criados pela comunidade de usuários.



A linguagem R

- Um dos pontos fortes do R é a qualidade na geração de gráficos e visualizações dos resultados, inclusão de símbolos matemáticos e de fórmulas, quando estas passam a ser necessárias.
- Facebook e FourSquare, apenas para citar dois exemplos, utilizam a linguagem R tanto para recomendações quanto para modelagem de comportamentos dos usuários.



Pacotes

- A distribuição de códigos criados pela comunidade de usuários do R é feita por meio de pacotes.
- Um pacote é um conjunto de códigos independente que adiciona funcionalidades à linguagem.
- Ao carregar um pacote, você está adicionando suas funções ao ambiente, permitindo que você chame estas funções diretamente.



Pacotes

- Vamos tentar usar a função **mvrnorm**, que gera números aleatórios de uma a partir de uma distribuição normal (esta função é definida no pacote chamado MASS, não vem instalada como padrão).

```
# Matriz Var-Covar
Sigma<-matrix(c(10,3,3,2), nrow = 2, ncol = 2)
# Médias
mu<-c(1, 10)
#Gera 100 observações
x<-mvrnorm(n = 100, mu, Sigma)
```



Pacotes

- Fazendo a correção:

```
# Carrega a biblioteca MASS
library(MASS)
# Matriz Var-Covar
Sigma<-matrix(c(10,3,3,2), nrow = 2, ncol = 2)
# Médias
mu<-c(1, 10)
#Gera 100 observações
x<-mvrnorm(n = 100, mu, Sigma)
```



Pacotes

- Para saber o que temos disponível no ambiente R no momento, usamos o comando **search()**.
- Para retirar um pacote do ambiente, usamos o comando **detach()**.

detach(package:MASS)



Pacotes

- No caso do pacote ter uma função com o mesmo nome de uma outra já carregada, a função que prevalece é a do pacote que foi carregado por último.
- Podemos usar o nome do pacote e o operador '::' antes de chamar a função. Neste caso não há ambiguidade.

x <- MASS::mvrnorm(n = 100, mu, Sigma)



Pacotes

- Também podemos carregar pacotes pela interface RGui ou do RStudio.
- No RGui basta usar a opção de menu chamada “Pacotes”.
- No RStudio, basta usar o menu do canto inferior direito, clicando na caixa ao lado do nome do pacote.



Pacotes

- Outra opção é usar a função **`install.packages()`**.
- Ex.: o pacote **`dplyr`** é muito utilizado para manipulação de dados. Para instalá-lo e desinstalá-lo, temos os seguintes comandos:

```
# Instala o pacote  
install.packages("dplyr")
```

```
# Desinstala o pacote  
remove.packages("dplyr")
```



Pacotes

- Podemos instalar uma série de pacotes ao mesmo tempo:

```
install.packages(c("ggplot2", "ggthemes",  
"tidyr", "reshape2", "stringr"))
```



Exemplo

- Vamos a um exemplo simples para termos uma ideia de como usar o R em análise de dados.
- Temos um conjunto de dados contendo cotações das ações de Petrobras e Vale.
- Este é apenas um exemplo motivacional, não se preocupe em saber neste momento como cada uma das instruções funcionam.



Exemplo

```
# Carregando os pacotes necessários
library(ggplot2)
library(reshape2)

# Configurando a coluna de datas corretamente
setAs("character", "myDate", function(from)
as.Date(from, format="%d/%m/%Y") )

# Carregando os dados
dados <- read.csv("PETR4_VALE5.csv",
colClasses=c('myDate', 'numeric', 'numeric'),
header = TRUE, sep = ";", dec = ",")

# Ajustando os dados para gráfico
meltdados <- melt(dados, id="Date")
```



Exemplo

```
# Criando o gráfico
ggplot(meltdados, aes(x=Date, y=value, colour=variable, group=variable)) + geom_line()
+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



Identificar os objetos do R



Objetos

- Temos diferentes tipos de objetos no R.
- Estes objetos podem ser vetores (chamados de “variáveis”, quando o vetor tem tamanho 1), matrizes, *arrays*, *data frames* e funções.
- No dia-a-dia do uso do R nomeamos objetos para uso posterior.



Variáveis

- A atribuição de valores às variáveis na linguagem R é feita com o operador '`<-`', conhecido como *"assignment operator"*.
 - Ex.: `x <- 20` # Atribui o valor 20 à variável `x`
- Em muitos casos o operador '`=`' é similar ao '`<-`', porém nem sempre este é o caso.
 - Ex.: `y = 25` # Atribui o valor 25 à variável `y`



Variáveis

- Recomenda-se no entanto o uso do operador '`<-`' sempre pois é exclusivo para atribuição de valores.
- O operador '`=`' tem um uso diferenciado em alguns casos e veremos isso mais adiante.



Variáveis

- O operador '`<-`' nada mais é que uma função de atribuição. Uma outra função que tem o mesmo objetivo é `assign("nome_Objeto", valor_Objeto)`.

– Ex.: `assign("z", 30)` # cria a variável
z com o valor 30



Variáveis

- Após a atribuição de valores podemos operar com as variáveis recém criadas.

`x + y + z` # soma x,y,z

`x/z` # divide x por z

`y * z` # multiplica y por z

`t <- x * y + z` # cria t com o resultado
de `(x*y+z)`



Variáveis

- A linguagem R é sensível ao uso de maiúsculas e minúsculas, portanto isso deve ser levado em consideração ao nomearmos variáveis.
- **Variáveis com os nomes `ab`, `Ab`, `aB` e `AB` são diferentes!**



Variáveis

- Nomes de variáveis no R podem conter combinações arbitrárias de números, textos, assim como ponto (.) e sublinhado (_).
- Porém, os nomes não podem começar com números ou sublinhado.

```
- xp_t2.F34 <- "variável válida"  
- _problema <- 2500  
- 5_outro_problema <- 5000
```



Tipos de variáveis

| Tipo de variável | Exemplos | Uso |
|------------------|--------------|------------------------------|
| Lógica | TRUE, FALSE | <code>v <- TRUE</code> |
| | | <code>print(class(v))</code> |
| | | Produz o seguinte resultado |
| | | <code>[1] "logical"</code> |
| Numérica | 12.3, 5, 999 | <code>v <- 23.5</code> |
| | | <code>print(class(v))</code> |
| | | Produz o seguinte resultado |
| | | <code>[1] "numeric"</code> |
| Inteira | 2L, 34L, 0L | <code>v <- 2L</code> |
| | | <code>print(class(v))</code> |
| | | Produz o seguinte resultado |
| | | <code>[1] "integer"</code> |



Tipos de variáveis

| Tipo de variável | Exemplos | Uso |
|------------------|-------------------------------------|---|
| Complexa | 3 + 2i | <code>v <- 2+5i</code> |
| | | <code>print(class(v))</code> |
| | | Produz o seguinte resultado |
| | | <code>[1] "complex"</code> |
| Texto | "bom", "TRUE", '23.4' | <code>v <- "TRUE"</code> |
| | | <code>print(class(v))</code> |
| | | Produz o seguinte resultado |
| | | <code>[1] "character"</code> |
| Raw | "Hello" is stored as 48 65 6c 6c 6f | <code>v <- charToRaw("Hello")</code> |
| | | <code>print(class(v))</code> |
| | | Produz o seguinte resultado |
| | | <code>[1] "raw"</code> |



Trabalhando com objetos

- Para listar todos os objetos que estão na área de trabalho, utilizamos a função `ls()`.
- Já a função `rm(objeto)`, remove um objeto da área de trabalho.
- Para salvar uma cópia de todos os objetos criados podemos utilizar a função `save.image()`.
- Para limparmos a área de trabalho usamos a instrução `rm(list = ls())`.
- Para carregarmos um arquivo com os objetos usamos a função `load()`.



Vetores

- Possui uma dimensão e um tipo de dado.
- Sequência de variáveis de diversos tipos. Não permite mistura de dados (coerção - próximo slide).

```
vetor1 <- c(1:10)
vetor1
length(vetor1)
mode(vetor1)
class(vetor1)
```



Coerção

- Um vetor somente pode ter elementos de uma única classe.
- Não é possível, por exemplo, misturar textos com números.



Coerção

- Quando misturamos elementos de classes diferentes em um vetor, o R faz a coerção do objeto para uma das classes, obedecendo uma ordem de prioridade:

lógico < inteiro < numérico < complexo < texto



Coerção - Exemplos

```
x <- c(1, 2, 3L)
class(x)
[1] "numeric"
```

```
x <- c(1, 2, 3L, 4i)
class(x)
[1] "complex"
```

```
x <- c(1, 2, 3L, 4i, "5")
class(x)
[1] "character"
```



Coerção

- Podemos forçar a conversão de um vetor de uma classe para outra com as funções **as.xxx** (sendo “xxx” a classe).
- Porém, nem sempre essa conversão faz sentido, e pode resultar em erros ou Nas (“Not Applicable”).



Coerção - Exemplos

```
numero <- c(546.90, 10, 789)
as.character(numero) # Vira texto
[1] "546.9" "10" "789"

texto <- c("aspas duplas", 'aspas
simples', "aspas 'dentro' do texto")
as.numeric(texto) # Não faz sentido
[1] NA NA NA

logico <- c(TRUE, FALSE, TRUE)
as.numeric(logico) # TRUE->1, FALSE->0
[1] 1 0 1
```

Concatenação

- E o que é aquele 'c' antes dos parênteses em várias das atribuições nos exemplos anteriores?
- É uma função nativa do R usada para criar um vetor a partir de valores explícitos. Se quisermos criar um sequência de valores, utilizamos o operador ':', como nos exemplos da função `plot()` acima.



Matriz

- Possui duas dimensões e um tipo de dado.
- Coleção de vetores em linhas e colunas, sendo que todos os vetores devem ser do mesmo tipo de dado.

```
matriz1 <- matrix(1:10, nrow =2)
matriz1
length(matriz1)
mode(matriz1)
class(matriz1)
```



Array

- Possui duas ou mais dimensões e um tipo de dado.
- Similar às matrizes, porém permite mais dimensões.

```
array1 <- array(1:5, dim=c(3,3,3))
array1
length(array1)
mode(array1)
class(array1)
```



Data frames

- Permite dados de diferentes tipos.
- É uma matriz com diferentes tipos de dados.

```
View(iris)
length(iris)
mode(iris)
class(iris)
```



Listas

- Coleção de diferentes objetos.
- Diferentes tipos de dados são possíveis e comuns.

```
lista1 <- list(a=matriz1,
b=vetor1)
lista1
length(lista1)
mode(lista1)
class(lista1)
```



Funções

- Em R, também são vistas como objetos.

```
func1 <- function(x) {
  var1 <- x * x
  return(var1)
}
```

```
func1(5)
class(func1)
```



Resumo

| Objeto | Tipos | Permite mistura de tipos? |
|-------------------|-------------------------------------|---------------------------|
| Vetor | Numérico, texto, complexo ou lógico | Não |
| Matriz | Numérico, texto, complexo ou lógico | Não |
| Array | Numérico, texto, complexo ou lógico | Não |
| Data frame | Numérico, texto, complexo ou lógico | Sim |
| Lista | Numérico, texto, complexo ou lógico | Sim |
| Funções | Expressões | N/A |



