Universidade Federal do Oeste do Pará – UFOPA
Campus Monte Alegre

Projeto de Monitoria de Ciência de Dados para a Aquicultura 4.0

Introdução ao Software R

Monitor: Luan Patrick

Orientador: Carlos Zarzar

O que é o R?

- O R é uma linguagem de programação com foco em análise de dados;
- Criado na Nova Zelândia por dois estatísticos: Ross Ihaka e Robert Gentleman;
  - Baseado na linguagem S, desenvolvida por John Chambers (e colegas) na Bell Laboratories;
- É uma linguagem de programação interpretada, voltada à interação dinâmica com os dados e modelos.

Por que o R?

- O R é gratuito e de código aberto;
- Compatível com todas as plataformas (Windows, Mac, Linux);
- É mais do que um software estatístico:
  - Ambiente que permite explorar dados interativamente; mas, à medida que a análise evolui, é uma linguagem de programação completa para desenvolver e automatizar soluções, desenvolver software (pacotes);
  - Ferramenta poderosa para manipulação, processamento, visualização e análise de dados, bem como simulações e modelagem estatísticas.
- Comunidade grande que contribui ativamente, com pessoas tanto do mercado (Google, AT&T, empresas de investimento e finanças) quanto da academia (professores de diversas universidades):
  - Mais de 8000 pacotes gratuitos e abertos para download.

Estatística

- Análise de Séries Temporais
- Plotagem de Gráficos Interpretativos
- Testes Computacionais

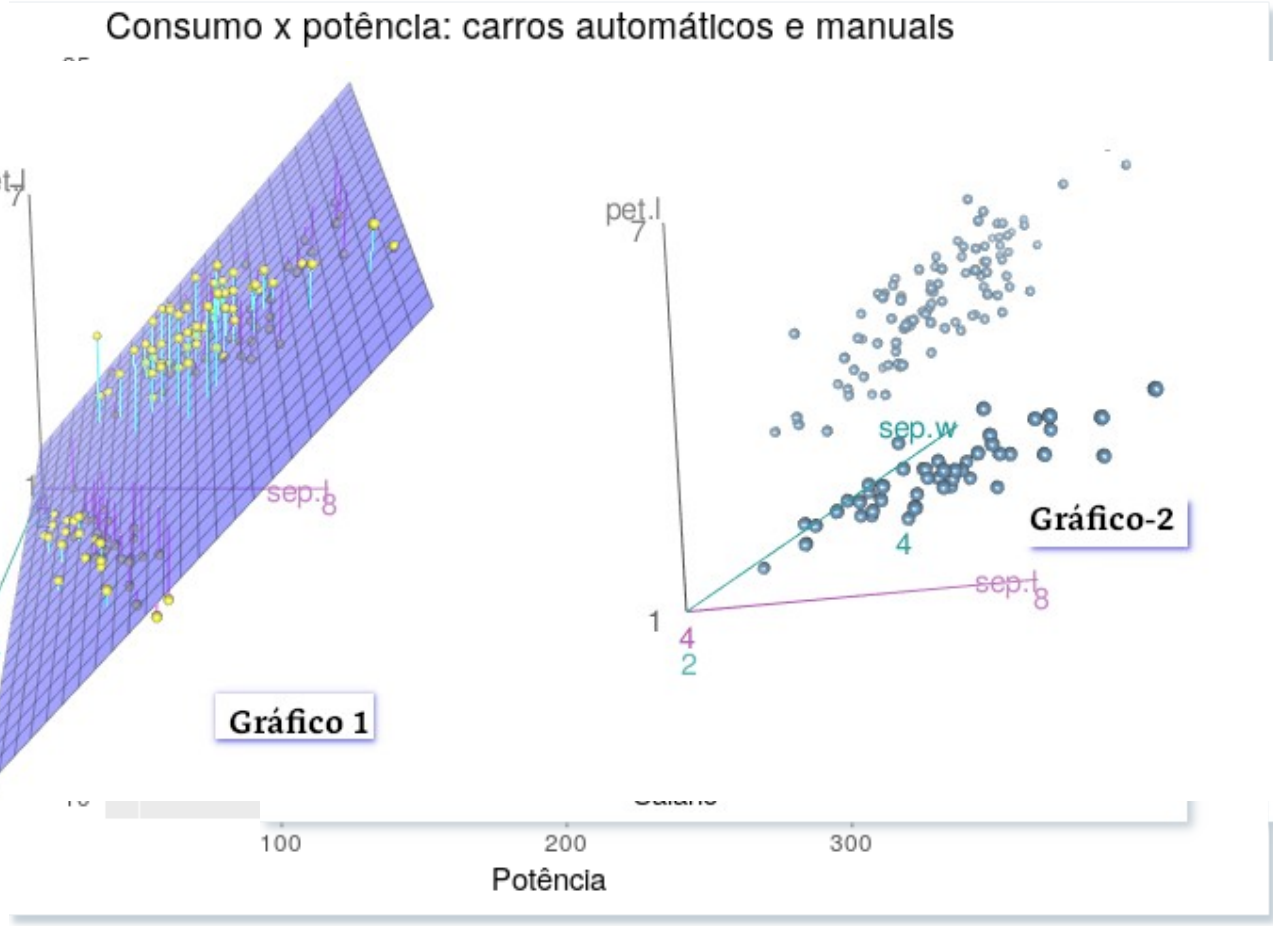
Ciência de Dados

- Suporte a Mineração de Dados e Big Data
- Importação/ Exportação de Dados
- Algoritmos para Aprendizagem de Máquina

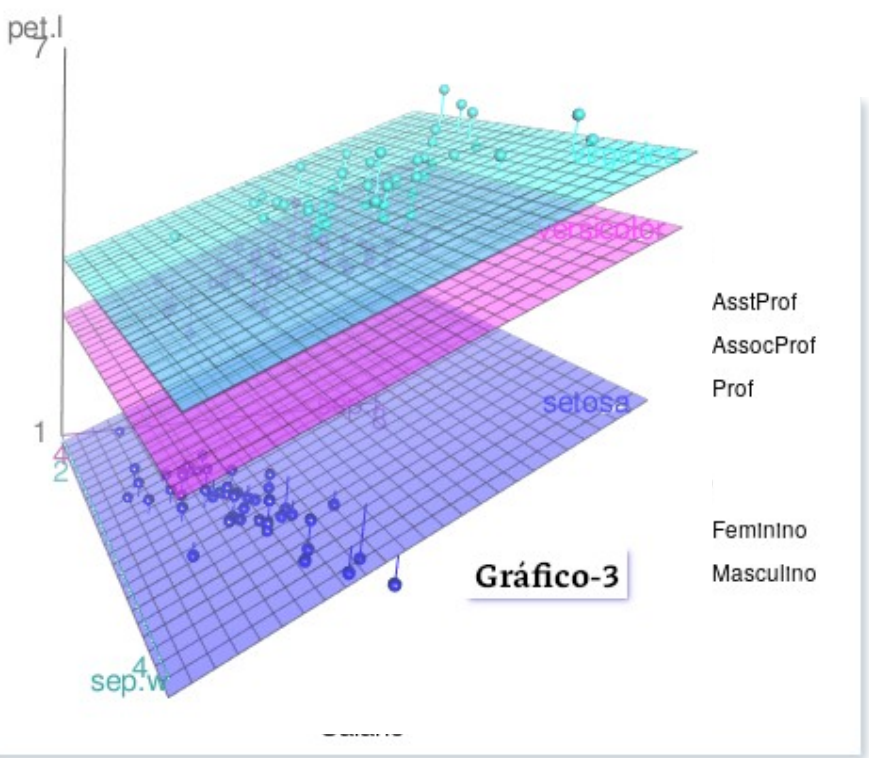
Outros Nichos Específicos

- Redes Sociais
- Desenvolvimento de Softwares
- Pesquisa Científica

Exemplos


Consumo x potência: carros automáticos e manuais
pet.l
sep.w
sep.l
Gráfico 1
100
200
300
Potência

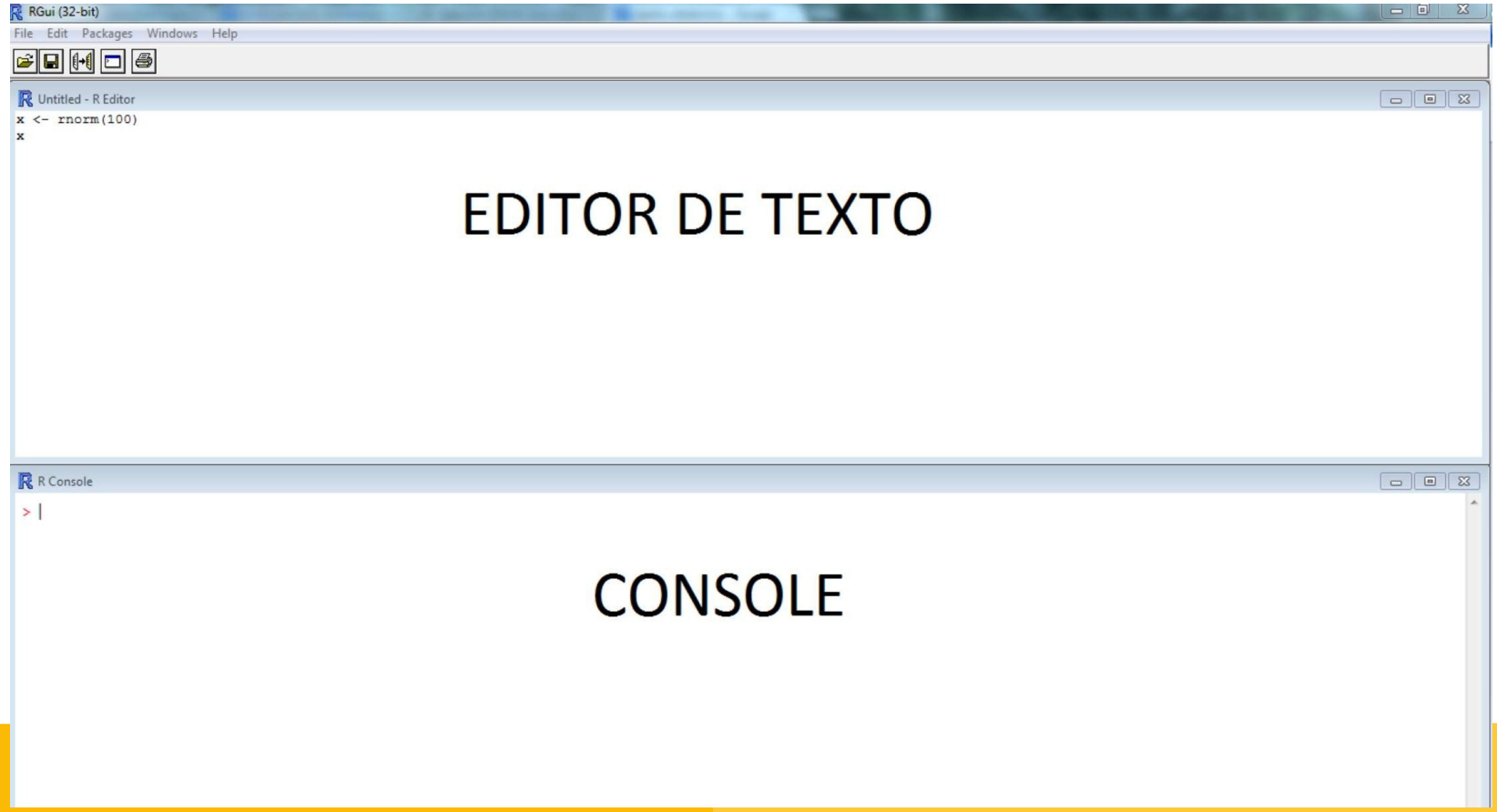
pet.l
sep.w
sep.l
Gráfico-2


pet.l
versicolor
setosa
sep.w
Gráfico-3
AsstProf
AssocProf
Prof
Feminino
Masculino

R e Rstudio

R Gui

Ao instalar o R, automaticamente também é instalada uma interface gráfica chamada R Gui. Abra o R no seu computador. A primeira tela que você verá é a do console. Abra também uma tela de editor de texto em "file" -> "new script".



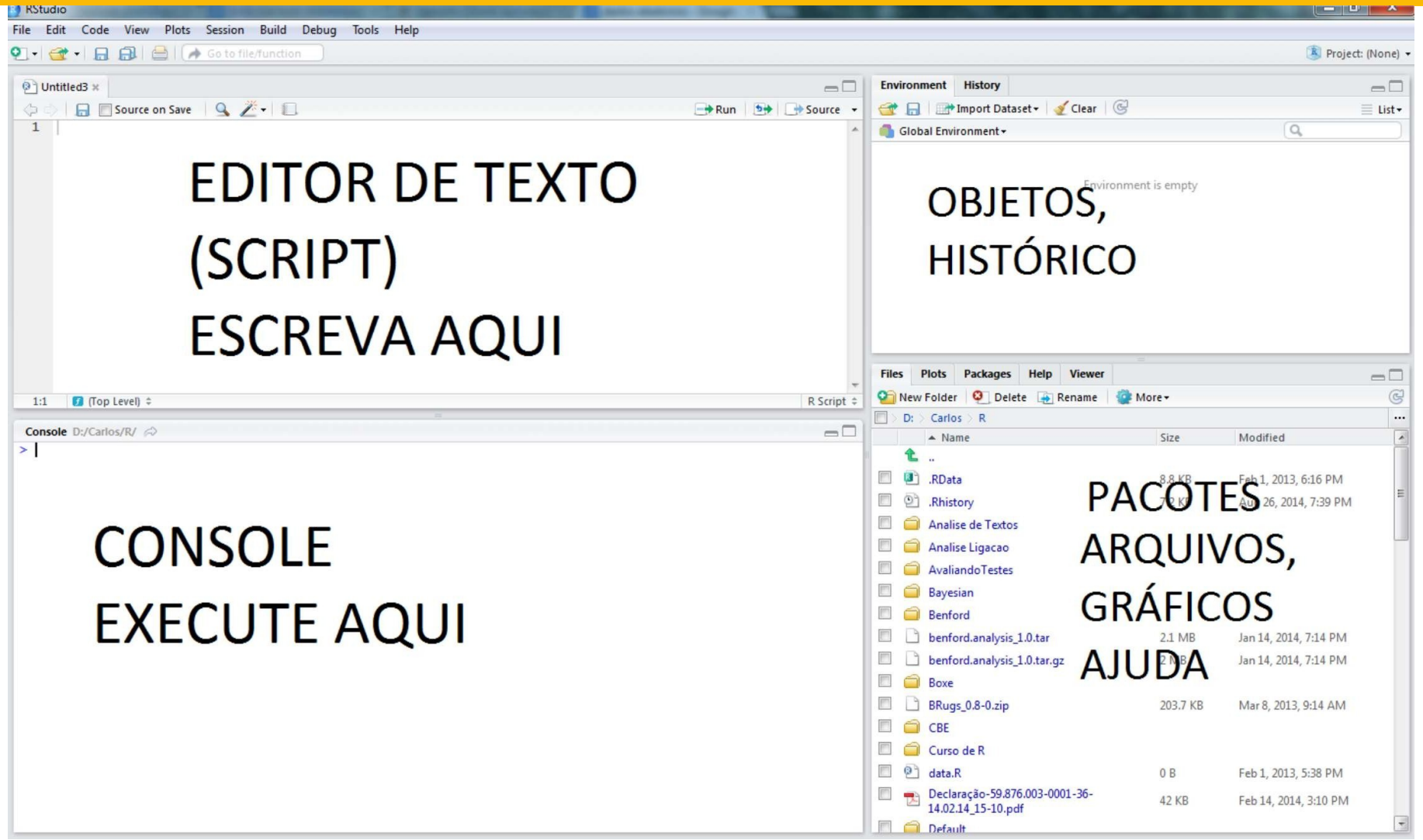
RStudio

(Integrated Development Environment - Ambiente de Desenvolvimento Integrado) chamado RStudio, com várias funcionalidades e gratuito. No decorrer de nossas aulas iremos utilizar somente o RStudio.

O RStudio tem algumas vantagens em relação ao R Gui:

- Highlight do código;
- Autocomplete;
- Match automático de parenteses e chaves;
- Interface intuitiva para objetos, gráficos, script;
- Projetos (com controle de versão);
- Interação com HTML, entre outras.

RStudio


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Untitled3
Source on Save
Run
Source
EDITOR DE TEXTO
(SCRIPT)
ESCREVA AQUI
Environment
History
Import Dataset
Clear
Global Environment
Environment is empty
OBJETOS,
HISTÓRICO
Console D:/Carlos/R/
CONSOLE
EXECUTE AQUI
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
D: Carlos R
Name
Size
Modified
.RData
.Rhistory
Analise de Textos
Analise Ligacao
AvaliandoTestes
Bayesian
Benford
benford.analysis_1.0.tar
benford.analysis_1.0.tar.gz
Boxe
BRugs_0.8-0.zip
CBE
Curso de R
data.R
Declaração-59.876.003-0001-36-14.02.14_15-10.pdf
Default
PACOTES
ARQUIVOS,
GRÁFICOS
AJUDA

Aritmética básica e vetorização

O R tem uma série de operadores de aritmética básica, entre eles:

Operador	Descrição
`x + y`	Soma (elemento a elemento)
`x - y`	Subtração (elemento a elemento)
`x * y`	Multiplicação (elemento a elemento)
`x / y`	Divisão (elemento a elemento)
`x ^ y`	Exponenciação (elemento a elemento)
`x %/% y`	Divisão por inteiro (elemento a elemento)
`x %% y`	Resto da divisão (elemento a elemento)

Vetorização!

Muitos operadores e funções do R são vetorizados, isto é, os cálculos são realizados elemento a elemento. Durante o curso esteja sempre atento a isso e tente explorar a vetorização do R para facilitar seus cáculos.

Outras funções matemáticas

Além dos operadores básicos, há uma série de funções matemáticas comumente utilizadas, tais como:

Função	Descrição
`abs(x)`	Valor absoluto.
`log(x)`	Logaritmo natural.
`exp(x)`	Exponencial.
`sqrt(x)`	Raiz quadrada.
`factorial(x)`	Fatorial.

Todas essas funções **são vetorizadas**.

Outras funções matemáticas

Funções que calculam estatísticas descritivas dos vetores:

Função	Descrição
`sum(x)` e `cumsum(x)`	Soma e soma acumulada.
`prod(x)` e `cumprod(x)`	Produtório e produtório acumulado.
`min(x)`, `cummin(x)` e `pmin(x, y)`	Mínimo, mínimo acumulado e mínimo par a par.
`max(x)`, `cummax(x)` e `pmax(x, y)`	Máximo, máximo acumulado e máximo par a par.
`mean(x)`	Média.
`var(x)` e `sd(x)`	Variância e desvio-padrão.
`cov(x, y)` e `cor(x, y)`	Covariância e correlação.
`diff(x)`	Primeira diferença.

Operadores Relacionais

Vejamos agora alguns operadores relacionais. Estes operadores são importantes para determinar a relação entre dois vetores e, como seu resultado é um vetor lógico, são muito utilizados para fazer `subset`. Da mesma forma que os operadores matemáticos, operadores relacionais também são vetorizados.

Operador	Descrição
`x == y`	x é igual a y? Faz coerção.
`x != y`	x é diferente de y?
`x > y`	x é maior do que y?
`x >= y`	x é maior ou igual a y?
`x < y`	x é menor do que y?
`x <= y`	x é menor ou igual a y?

Criando matrizes: função `matrix()`

É possível criar matrizes no R com a função `matrix()`.

```
matrix(data = dados, ncol = numero_de_colunas, nrow = numero_de_linhas)
```

- No argumento `data` passamos o vetor que desejamos transformar em matriz;
- `ncol` especifica o número de colunas da matriz; e,
- `nrow` especifica o número de linhas da matriz.

Operações com matrizes

Operador	Descrição
`== != > >= < <=`	Operadores relacionais elemento a elemento
`+ - / *`	Soma, subtração, divisão e multiplicação elemento a elemento
`%*%`	Multiplicação de matriz
`%x%`	Produto de Kronecker
`t()`	Transposição de matriz
`solve()`	Inversão de matriz (ver também `qr()`, `svd()`, `chol()`)
`det()`	Determinante de uma matriz
`diag()`	Diagonal de uma matriz

Arrays

O que são um Arrays

Os arrays são estruturas de dados multidimensionais no R.

Permitem armazenar e manipular dados em várias dimensões.

No R, matrizes têm apenas duas dimensões.

Arrays podem ter mais de duas dimensões.

Arrays são mais flexíveis para lidar com dados complexos.

- Sintaxe: **array[índice]**.
- Exemplo: **my_array[1, 2]** retorna o elemento na primeira linha e segunda coluna.

Criando um Arrays

Uso da função ***array().***
Sintaxe: *array(data, dim, dimnames).*

Exemplo:
my_array <- array(c(1, 2, 3, 4), dim = c(2, 2)).

Usar índices para acessar elementos.
Sintaxe: **array[índice].**

Exemplo: my_array[1, 2] #retorna o elemento na primeira linha e segunda coluna.

Funções Úteis para Arrays

Operações aritméticas funcionam elemento por elemento.
Funções como **sum(), mean(), max(), min()** podem ser aplicadas.

Exemplo: **sum(my_array)** **#**retorna a soma de todos os elementos.

dimnames(): Define nomes para as dimensões.
dim(): Retorna as dimensões do array.
length(): Retorna o número total de elementos.
str(): Mostra a estrutura do array.

Exercício: Criando um Array e Realizando Operações

```
# Criando um array bidimensional
matriz <- array(c(1, 2, 3, 4, 5, 6), dim = c(2, 3))

# Calculando a soma de todos os elementos da matriz
soma_total <- sum(matriz)

# Exibindo a matriz e a soma
print("Matriz:")
print(matriz)
print(paste("Soma total dos elementos da matriz:", soma_total))
```

```
# Criando um array tridimensional
dados <- array(1:24, dim = c(2, 3, 4))

# Acessando elementos
elemento1 <- dados[1, 2, 3]
elemento2 <- dados[2, 3, 4]

# Exibindo os elementos
print("Array de dados:")
print(dados)
print(paste("Elemento na primeira dimensão (1, 2, 3):",
elemento1))
print(paste("Elemento na primeira dimensão (2, 3, 4):",
elemento2))
```

```
# Criando arrays unidimensionais
array1 <- c(1, 2, 3, 4, 5)
array2 <- c(5, 4, 3, 2, 1)

# Realizando operações algébricas
soma_arrays <- array1 + array2
dif_arrays <- array1 - array2
produto_arrays <- array1 * array2
divisao_arrays <- array1 / array2

print(produto_arrays)
print("Divisão de arrays:")
print(divisao_arrays)
```

Por que um `data.frame`?

A solução para isso é o `data.frame`. O `data.frame` é talvez o formato de dados mais importante do R. No `data.frame` cada coluna representa uma variável e cada linha uma observação. Essa é a estrutura ideal para quando você tem várias variáveis de classes diferentes em um banco de dados.