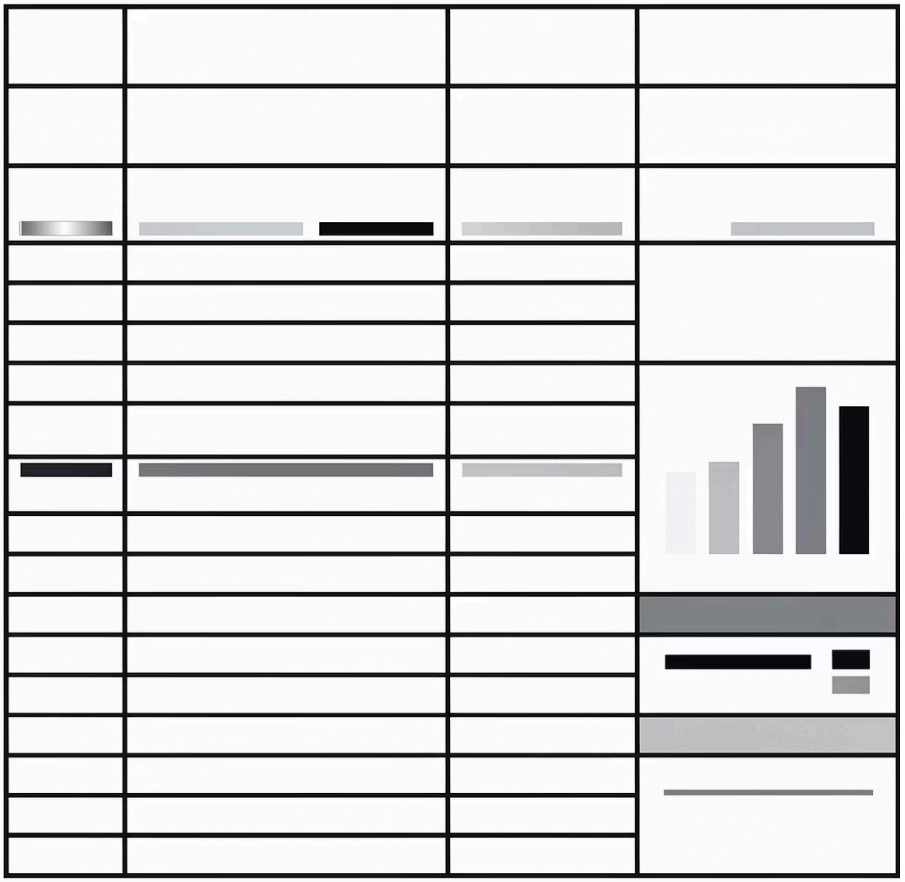


CEFET/RJ

ESTRUTURAS DE DADOS EM R

Data Frame

Data frames são a estrutura fundamental para análise de dados em R. Eles organizam informações em formato tabular, combinando a flexibilidade de listas com a estrutura de matrizes, permitindo diferentes tipos de dados em cada coluna.

Eduardo Ogasawara
eduardo.ogasawara@cefet-rj.br
https://eic.cefet-rj.br/~eogasawara

Criando Vetores Básicos

Exemplo Prático com Dados Antropométricos

```
weight <- c(60, 72, 57, 90, 95, 72)
height <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
subject <- c("A", "B", "C", "D", "E", "F")
```

A função `c()` combina valores em um vetor. Aqui criamos três vetores: peso em kg, altura em metros e identificadores dos participantes.

Esses vetores serão a base para construir nosso data frame completo.

Construindo a Estrutura

Criando uma Tabela Básica

01

Criar o Data Frame

Use `data.frame()` para combinar vetores em uma tabela estruturada

02

Visualizar os Dados

A função `head()` exibe as primeiras linhas da tabela

```
d <- data.frame(weight=weight, height=height, subject=subject)
head(d)

##   weight height subject
## 1     60   1.75       A
## 2     72   1.80       B
## 3     57   1.65       C
## 4     90   1.90       D
## 5     95   1.74       E
## 6     72   1.91       F
```

Cada coluna mantém seu nome e tipo de dados original, criando uma estrutura organizada e fácil de manipular.

Expandindo a Análise

Adicionando uma Coluna Calculada

```
d$bmi <- d$weight/d$height^2
head(d)
```

```
##   weight height subject      bmi
## 1     60   1.75       A 19.59184
## 2     72   1.80       B 22.22222
## 3     57   1.65       C 20.93664
## 4     90   1.90       D 24.93075
## 5     95   1.74       E 31.37799
## 6     72   1.91       F 19.73630
```

O operador `$` permite acessar e criar colunas. Aqui calculamos o IMC (Índice de Massa Corporal) usando a fórmula: peso / altura².

Operações vetorizadas aplicam o cálculo automaticamente a todas as linhas.

Gerenciamento de Colunas

Removendo uma Coluna

1

Data Frame Original

Contém weight, height, subject e bmi

2

Atribuir NULL

`d$subject <- NULL`

3

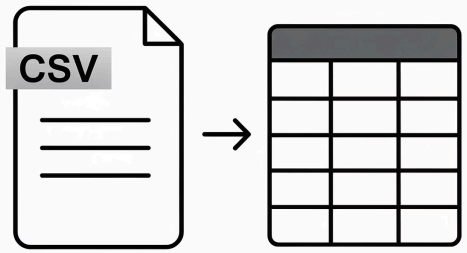
Coluna Removida

Tabela agora sem a coluna subject

```
d$subject <- NULL
head(d)

##   weight height      bmi
## 1     60   1.75 19.59184
## 2     72   1.80 22.22222
## 3     57   1.65 20.93664
## 4     90   1.90 24.93075
## 5     95   1.74 31.37799
## 6     72   1.91 19.73630
```

Atribuir `NULL` a uma coluna a remove permanentemente do data frame, simplificando a estrutura quando necessário.



Importação de Dados

Carregando uma Tabela a Partir de um CSV

Os arquivos CSV podem ser lidos local ou remotamente usando a função `read.table`. Esta é uma das formas mais comuns de importar dados externos para o R.

```
wine = read.table("http://archive.ics.uci.edu/ml/machine-learning-
databases/wine/wine.data",
header = TRUE, sep = ",")
head(wine)

##   X1 X14.23 X1.71 X2.43 X15.6 X127 X2.8 X3.06 X.28 X2.29 X5.64 X1.04 X3.92 X1065
## 1  1  13.20  1.78  2.14  11.2  100 2.65  2.76 0.26  1.28  4.38  1.05  3.40  1050
```

Especifique `sep=","` para arquivos CSV e `header=TRUE` se a primeira linha contiver nomes de colunas.

Persistência de Dados

Salvando, Apagando e Carregando uma Tabela

Salvar

`save(wine, file="wine.RData", compress=TRUE)` salva o objeto em formato binário comprimido

Remover

`rm(wine)` apaga o objeto da memória para liberar recursos

Carregar

`load("wine.RData")` restaura o objeto salvo de volta para a memória

```
save(wine, file="wine.RData", compress=TRUE)
rm(wine)
load("wine.RData")
head(wine, 3)

##   X1 X14.23 X1.71 X2.43 X15.6 X127 X2.8 X3.06 X.28 X2.29 X5.64 X1.04 X3.92 X1065
## 1  1  13.20  1.78  2.14  11.2  100 2.65  2.76 0.26  1.28  4.38  1.05  3.40  1050
## 2  1  13.16  2.36  2.67  18.6  101 2.80  3.24 0.30  2.81  5.68  1.03  3.17  1185
```

O formato .RData preserva tipos de dados e estruturas complexas, sendo ideal para armazenamento intermediário.

Exportação Universal

Salvando em CSV

```
write.table(wine, file="wine.csv",
row.names=FALSE, quote=FALSE, sep=",")
```

A função `write.table` exporta data frames para arquivos CSV, compatíveis com Excel e outros softwares.

- `row.names=FALSE`: omite números de linha
- `quote=FALSE`: remove aspas dos textos
- `sep=","`: define vírgula como separador

CSV é o formato universal para compartilhamento de dados tabulares entre diferentes plataformas e linguagens.

Seleção de Dados

Filtrando Tabela

A filtragem de data frames é feita através do índice de linhas, utilizando condições lógicas que retornam vetores de TRUE/FALSE.

1. Criar Condição

`i <- d$height > 1.7` gera vetor lógico

2. Verificar Resultado

Vetor `i` contém TRUE onde altura > 1.7

3. Aplicar Filtro

`d[i,]` retorna apenas linhas TRUE

```
i <- d$height > 1.7
i
## [1] TRUE TRUE FALSE TRUE TRUE TRUE

d[i,]
## weight height bmi
## 1 60 1.75 19.59184
## 2 72 1.80 22.22222
## 4 90 1.90 24.93075
## 5 95 1.74 31.37799
## 6 72 1.91 19.73630
```

Análise de Performance

Desempenho em Data Frame

A função `rnorm(n, m, s)` gera n números aleatórios seguindo uma distribuição normal com média m e desvio padrão s. `Sys.time()` é usado para medir o tempo decorrido das operações.

```
rheight <- rnorm(100000, 1.8, sd=0.2)
rweight <- rnorm(100000, 72, sd=15)

start_time <- Sys.time()
hw <- data.frame(height=rheight, weight=rweight)
hw$bmi <- hw$weight/hw$height^2
end_time <- Sys.time()
end_time - start_time

## Time difference of 0.001685143 secs
```

100K

Registros

Volume de dados processados

0.002s

Tempo

Operação vetorizada

Operações vetorizadas são extremamente eficientes, processando todos os elementos simultaneamente.

Antipadrão de Performance

Inclusão de Atributo a Partir de For Loop

```
start_time <- Sys.time()
hw <- data.frame(height=rheight, weight=rweight)
for (i in 1:nrow(hw)) {
  hw$bmi[i] <- hw$weight[i]/hw$height[i]^2
}
end_time <- Sys.time()
end_time - start_time

## Time difference of 8.374172 secs
```

Mais Lento

Comparado com operação vetorizada

Importante: A atribuição de valores elemento por elemento em data frames é extremamente demorada. Evite loops quando operações vetorizadas estão disponíveis.

Otimização Intermediária

Convertendo Tabela para Matriz, Processando e Voltando para Tabela

Data Frame

Estrutura original

Matriz

Conversão com `as.matrix()`

Processamento

Loop otimizado

Data Frame

Resultado final

```
start_time <- Sys.time()
hw <- data.frame(height=rheight, weight=rweight)
hwm <- as.matrix(hw)
bmi <- 0
for (i in 1:nrow(hwm)) {
  bmi[i] <- hwm[i,1]/hwm[i,2]^2
}
hw$bmi <- bmi
end_time <- Sys.time()
end_time - start_time

## Time difference of 0.2269666 secs
```

Matrizes têm acesso mais rápido que data frames. Esta abordagem é um meio-termo: mais rápida que loops em data frames, mas ainda mais lenta que operações totalmente vetorizadas.

Vetorizado

0.002s - Mais rápido

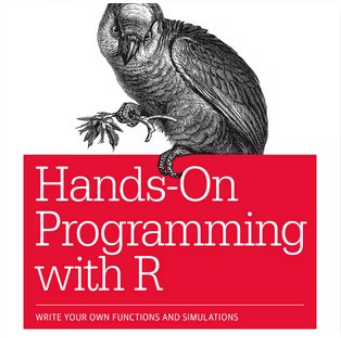
Matriz + Loop

0.227s - Intermediário

DataFrame + Loop

8.374s - Mais lento

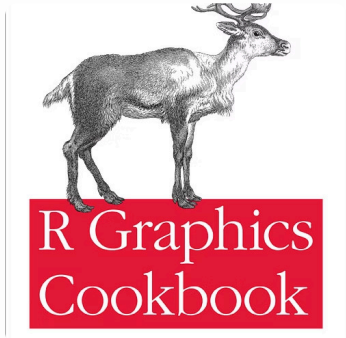
Referências



Hands-on Programming

Aprenda R criando suas próprias funções e simulações

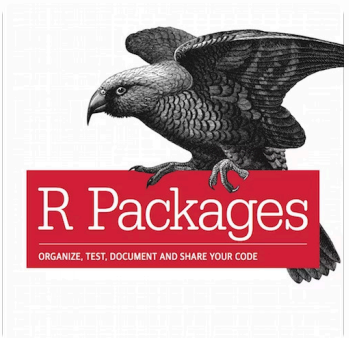
https://rstudio-education.github.io/hopr/index.html



R Graphics Cookbook

Domine visualizações de dados em R

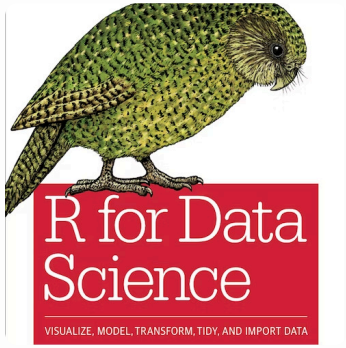
https://r-graphics.org



R Packages

Desenvolva seus próprios pacotes R

https://r-pkgs.org/index.html



R for Data Science

Guia completo para ciência de dados

https://r4ds.had.co.nz

G. Grolemund, 2014, *Hands-On Programming with R: Write Your Own Functions and Simulations*. O'Reilly Media, Inc.