Programação Estatística

Introdução ao R - Estrutura de Dados

Rachid Muleia, PhD in Statistics

Universidade Eduardo Mondlane
Faculdade de Ciências
Departamento de Matemática e Informática

2023-03-02

Objectos

Criação de objectos

Forma geral

- `variavel <- valor`

Nomes permitidos

- `meu.objecto`, `meu_objecto`, `meuObjecto`, `a`, `b`, `x1`, `x2`, `data1`, `1data`

Nomes não permitidos

- `.` seguido de um número no início: `.4you`
- Não se pode usar palavras reservadas: `if`, `else`, `repeat`, `for`, `while`, `function`, `TRUE`, ‘`FALSE`, etc.
- Mais descrições sobre palavras reservadas podem ser encontradas executando o código: `help("reserved")`

Tipos de dados

Tipo de dados

- Logical
- Numeric
- Integer
- Character
- Complex

Estruturas de dados

- Vectores
- Factor
- Listas -Matrix
- Data frame

Tipo de dados

Tipo de dados	Descrição	Exemplop
Logical	Verdadeiro ou Falso	TRUE ou FALSE
Numeric	Números reais	2.3, pi, `sqrt(2)`
Integer	Números inteiros	`5L, 4L,-1L`
Character	Sequência de caracteres	`"maria","UEM"`
Complex	Números complexos	2.1+3i, 1+0i

Vectores

Vectores

- Estrutura unidimensional
- Coleccção de valores onde todos têm o mesmo tipo de dados
- Exemplo: `(-2,3.1,2.4,5)`, `(TRUE, FALSE,TRUE,FALSE)`, `("Maria", "Joao","Augusto", "Antonio")`

Vectores

Vectores podem ser criados usando as seguintes funções:

- `c()`: função para combinar valores individuais
- `seq()`: função para criar uma sequência de valores
- `rep()`: função para criar réplica de valores

Vectores

Exemplo de criação de fectores usando a função c()

```
> c(1,2,3,4,5,6,7)
## [1] 1 2 3 4 5 6 7
> c(2:5,11:6)
##  [1]  2  3  4  5 11 10  9  8  7  6
```

```
> x<-c(2,7,8,12,3,25)
> x
## [1]  2  7  8 12  3 25
```

```
y<-c(red="Bob",blue="Dave",green="Jenny")
y
```

```
##     red    blue   green
##   "Bob"  "Dave" "Jenny"
```

Vectores

Exemplo de criação de vectores usando a função `seq()`

```
> seq(from=1,to=8)
## [1] 1 2 3 4 5 6 7 8
> seq(from=4,to=10,by=2)
## [1]  4  6  8 10
> seq(from=1,to=10,length=4)
## [1]  1  4  7 10
```

Vectores

Exemplo de criação de vectores usando a função `seq()`

```
> seq(from=1,to=8)
## [1] 1 2 3 4 5 6 7 8
> seq(from=4,to=10,by=2)
## [1]  4  6  8 10
> seq(from=1,to=10,length=4)
## [1]  1  4  7 10
```

```
> seq(1,8)
## [1] 1 2 3 4 5 6 7 8
> seq(4,10,2)
## [1]  4  6  8 10
> seq(1,10,,4)
## [1]  1  4  7 10
```

Vectores

Criação de vectores usando a função `rep()`

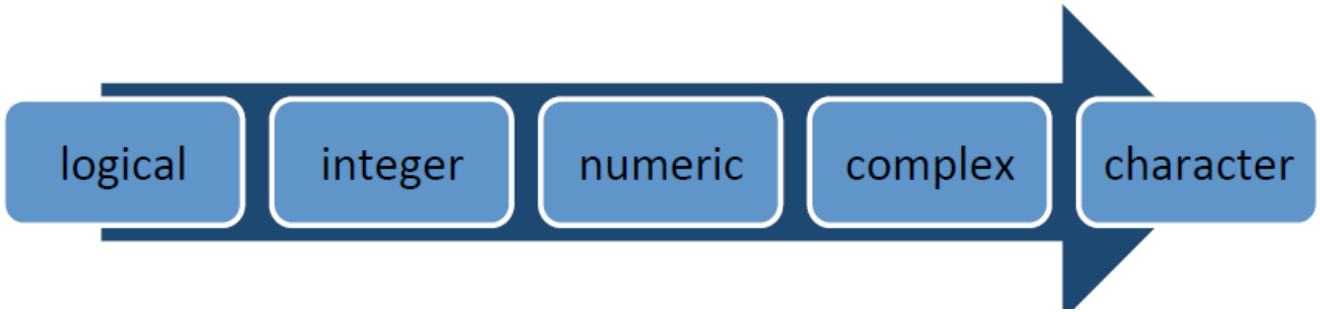
```
> rep(1,4)
## [1] 1 1 1 1
> rep(4:5,3)
## [1] 4 5 4 5 4 5
> rep(1:4,each=2)
## [1] 1 1 2 2 3 3 4 4
> rep(1:4,times=2,each=2)
##  [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

Vectores

Conversão do tipo de dados

- Vectores levam apenas um único tipo de dados
- Ao combinar diferentes tipos de dados, o R irá fazer a coerção para o tipo de dados mais flexível

Regra para coerção:



Vectores

Coerção de dados- exemplo

```
> x<-c(5,'b')    # será convertido para character
> x              # " " indica tipo character
## [1] "5" "b"
> y<-c(FALSE, 3) # será convertido para número
> y
## [1] 0 3
```

Vectores

Coerção de dados

A coerção de dados também pode ser feita usando as funções `as.class_name`.

- Exemplo
  - `as.numeric(x)`
  - `as.logical(x)`
  - `as.character(x)`
  - `as.integer(x)`
  - `as.factor(x)`
  - `as.complex(x)`

O tipo de dados pode ser verificado usando a função `typeof(x)` ou `class(x)`.

Operações com vectores

O acesso aos elementos de um vector pode ser feito usando o operador []

```
> x<-c(2,4,6,8,10)
> x[4]
## [1] 8
> x[3:5]
## [1]  6  8 10
> x[-2]
## [1]  2  6  8 10
```

Operações com vectores

Operações padrão em vectores são feitas elemento a elemento:

```
> c(2,5,3)+c(4,2,7)
## [1]  6  7 10
> c(2,5,3)+2
## [1] 4 7 5
> c(2,5,3)^2
## [1]  4 25  9
```

Operações com vectores

Algumas funções importantes

Operação	Descrição
`class(nome_vector)`	devolve o tipo do vector
`length(nome_vector)`	devolve o número total de elementos
`x[length(x)]`	último elemento do vector
`rev(nome_vector)`	devolve o vector invertido
`sort(nome_vector)`	devolve o vector ordenado
`unique(vector_nome)`	retorna um vector com valores únicos

Factores

- Um Fator é um vetor que representa dados categóricos
- Apenas pode conter categorias predefinidas
- Podem ser ordenados, assim como não

Exemplo: `("sim", "nao", "sim", "nao", "sim")`
`("masculino", "feminino", "masculino", "feminino")`
`("grande", "pequeno", "grande","pequeno")`

Factores

A criação de um factor pode ser feita usando a função `factor()`

```
> sexo <- c("masculino", "feminino", "masculino", "feminino")
> factor(sexo)
## [1] masculino feminino  masculino feminino
## Levels: feminino masculino
```

Factores

Ordenação de factores

- Existem várias formas para ordenar um factor.

```
> sizes <- factor(c("small", "large", "large", "small", "medium"))
> sizes
## [1] small  large  large  small  medium
## Levels: large medium small
```

- Pode-se ordenar um factor especificando as ordem das categorias

```
> sizes <- factor(sizes, levels=c('small', 'medium', 'large'))
> sizes
## [1] small  large  large  small  medium
## Levels: small medium large
> sizes<-factor(sizes,levels=c('large', 'medium', 'small'))
> sizes
## [1] small  large  large  small  medium
## Levels: large medium small
```

Factor

- Uma outra forma de ordenar é usando a função `relevel()`

```
> sizes <- factor(sizes, levels=c('small', 'medium', 'large'))
> sizes <- relevel(sizes, ref = 'medium')
> sizes
## [1] small  large  large  small  medium
## Levels: medium small large
```

- A função `relevel()` é muito importante para definir a categoria de referência

Listas

Listas

- Lista é uma colecção de estruturas de dados
- Uma lista pode armazenar qualquer tipo de dados, incluido lista
- Uma lista pode ser criada usando a função `list()`
- Maior parte das funções em R produzem outputs armazenados numa lista

Lista

Exemplo de criação de lista

```
> minha_Lista<-list(1:3,c("a","b"),c(TRUE,FALSE,TRUE))
>
> str(minha_Lista)
## List of 3
##  $ : int [1:3] 1 2 3
##  $ : chr [1:2] "a" "b"
##  $ : logi [1:3] TRUE FALSE TRUE
```

Assim como vimos na criação de vectores, os elementos da lista podem ter nome

```
> minha_Lista<-list(vector1=1:3,vector2=c("a","b"),vector3=c(TRUE,FALSE,TRUE))
>
> minha_Lista
## $vector1
## [1] 1 2 3
##
## $vector2
## [1] "a" "b"
##
## $vector3
## [1]  TRUE FALSE  TRUE
```

Lista

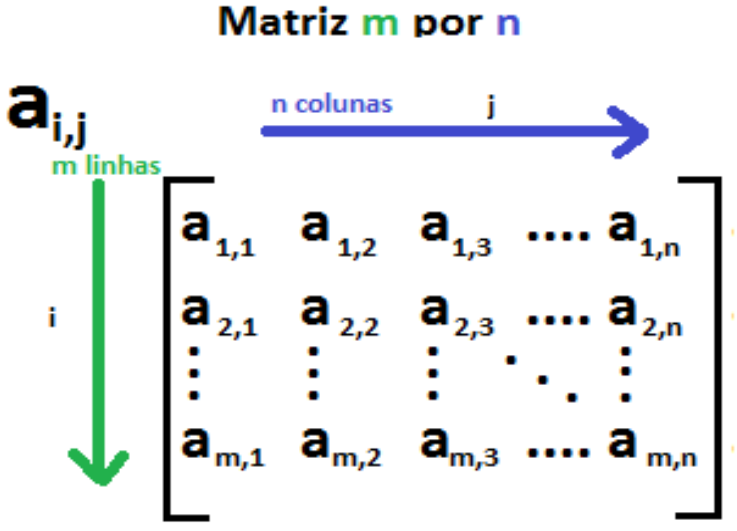
Uma outra forma de criar uma lista é usando a função `vector()`

```
> lista_vazia<-vector(mode='list', length=5)
>
> lista_vazia
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

Matriz

Matriz/ Matrix

- Uma colecção de valores com mesmo tipo de dados
- Estrutura bidimensional disposta em linhas e colunas.



Matriz

Uma matriz pode ser criada usando as seguintes funções:

- `matrix()` cria uma matrix especificando as linhas e as colunas
- `dim()` cria uma matrix definindo a dimensão do vector
- `cbind()` ou `rbind` cria uma matriz combinando colunas ou linhas, respectivamente

Matriz

```
> matrix(data=1:6,nrow=2,ncol=3,byrow=FALSE)
> matrix(data=1:6, # dados para  popular a matriz
+        nrow=2,   # numero de linhas
+        ncol=3,   # numero de colunas
+        byrow=FALSE) # preencher a matriz por linha
```

Exemplo de criação de matriz

```
> matrix(data=1:6,nrow=2,ncol=3,byrow=FALSE)
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Matriz

Criação de matriz usando a função `dim()`

```
> x<-1:6
> dim(x)<-c(2,3)
> x
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
> y<-1:6
> dim(y)<-c(3,2)
> y
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Matriz

Criação de matriz usando as funções `cbind()` e `rbind()`

```
> x<-1:3
> y<-10:12
>
> cbind(x,y)
##       x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
> rbind(x,y)
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

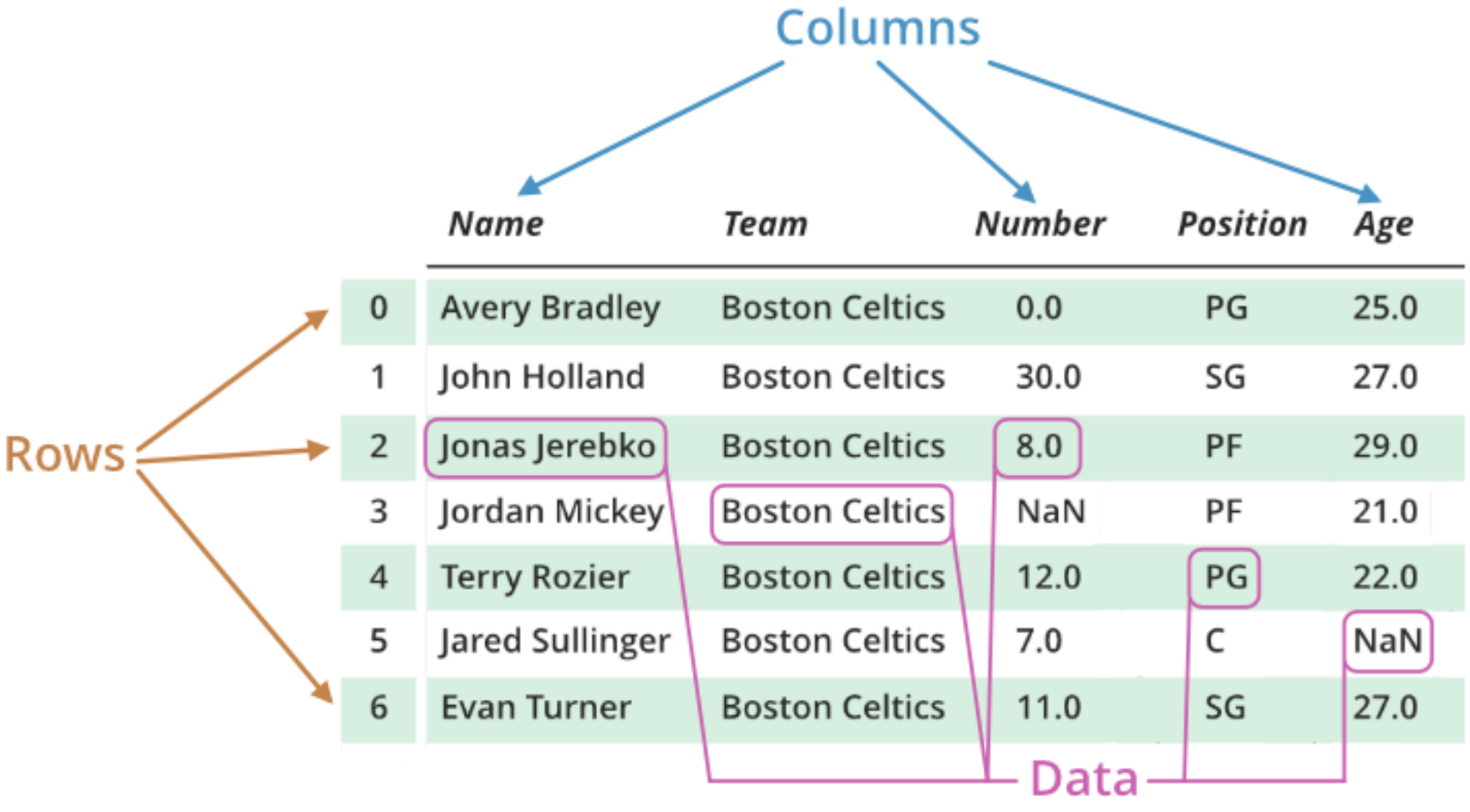
Data frame

Data frame

Colecção de vectores com igual tamanho

- Estrutura bidimensional disposta em linhas e colunas
  - **MAS**: DENTRO de uma coluna, cada célula deve ter o mesmo tipo de dados!
- `data frames` são usadas para representar todo conjunto de dados
- As colunas contêm vectores com diferentes tipos de dados

Data frame-layout



Uma data frame pode ser criada usando a função `data.frame()`.

Data frame

Exemplo de criação de data frame

```
> df <- data.frame(ID = 1:3, Sexo = c("F", "F", "M"),
+ Peso = c(71, 60, 65))
>
> df <- data.frame(
+ ID = 1:3,                    # elementos da primeira coluna
+ Sexo = c("F", "F", "M"), # elementos da segunda coluna
+ Peso = c(71, 60, 65))     # elementos da terceira coluna
```

Data frame

Exemplo de criação de data frame

```
> df <- data.frame(ID = 1:3, Sexo = c("F", "F", "M"),
+ Peso = c(71, 60, 65))
>
> df
  ID Sexo Peso
1  1    F   71
2  2    F   60
3  3    M   65
```

Data frame

Nota: Data frame, automaticamente, converte strings em factores.

```
> df_2<-data.frame(x=1:3,y=c("a","b","c"))
>
>
> str(df_2)  #  exibe a extrutura interna dos dados
'data.frame':   3 obs. of  2 variables:
 $ x: int  1 2 3
 $ y: chr  "a" "b" "c"
```

O argumento `stringsAsFactors = FALSE` impede esse comportamento

Data frame

Nota: Data frame, automaticamente, converte strings em factores.

```
> df_2<-data.frame(x=1:3,y=c("a","b","c"),
+                  stringsAsFactors = FALSE)
>
>
> str(df_2)
'data.frame':   3 obs. of  2 variables:
 $ x: int  1 2 3
 $ y: chr  "a" "b" "c"
```

Data frame

- Criar uma `data frame` manualmente leva muito tempo
- Além disso, a digitação convida erros
- Deve evitar digitar grandes conjuntos de dados no R manualmente

Data frame

- Criar uma `data frame` manualmente leva muito tempo
- Além disso, a digitação convida erros
- Deve evitar digitar grandes conjuntos de dados no R manualmente

As `data frames`, geralmente, são importadas usando as funções `read.csv()` e `read.table()`

Indexação

Indexação/subset- vectores

Para se aceder aos elementos de um vector pode-se usar seguinte notação `x[]`

```
> x<-10:22
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20 21 22
> x[7]
[1] 16
> x[1:4]
[1] 10 11 12 13
> x[c(1,4,6,5)]
[1] 10 13 15 14
```

Para excluir alguns numeros do vector usamos números negativos

```
> x<-10:22
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20 21 22
> x[-2:-7]  # excluir do segundo até ao sétimo elemento
[1] 10 17 18 19 20 21 22
```

Selecção condicional- vectores

- Serve para extrair elementos que satisfaçam uma determinada condição

```
> dados <- c(5, 15, 42, 28, 79, 4, 7, 14)
>
> # seleccione elementos maiores do que  15
> dados[dados>15]
[1] 42 28 79
> # Seleccione os valores maiores do que 15
> # menores ou iguais a 35:
>
> dados[dados > 15 & dados <= 35]
[1] 28
```

Indexação- função `which()`

- Muitas vezes estamos interessados em saber a posição do resultado de uma expressão condicional, ao invés do resultado em si
- A fução `which()` retorna as posições dos elementos que retornarem `TRUE` em uma expressão condicional

```
> dados <- c(5, 15, 42, 28, 79, 4, 7, 14)
> dados[dados > 15]
[1] 42 28 79
> which(dados>15)
[1] 3 4 5
> dados[dados > 15 & dados <= 35]
[1] 28
> which(dados > 15 & dados <= 35)
[1] 4
```

Indexação de matrizes

Em estruturas de dados multidimensionais (por exemplo, matrizes e quadros de dados), um elemento na m-ésima linha, n-ésima coluna pode ser acessado pelo expressão `x[m, n]`

```
> mat <- matrix(1:9, nrow = 3,ncol=3)
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> #Acesse o valor que está na linha 2 da coluna 3
>
> mat[2,3]
[1] 8
```

Indexação de matrizes

A m-ésima linha inteira pode ser extraída pela expressão `x[m, ]`

```
> mat <- matrix(1:9, nrow = 3,ncol=3)
>
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> mat[2, ]
[1] 2 5 8
```

A n-ésima coluna inteira pode ser extraída pela expressão `x[,n]`

```
> mat <- matrix(1:9, nrow = 3,ncol=3)
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> mat[,3 ]
[1] 7 8 9
```

Indexação de matriz

Pode-se também aceder a uma determinada linha e coluna em simultâneo

```
> mat <- matrix(1:9, nrow = 3,ncol=3)
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> mat[1:3,2 ]
[1] 4 5 6
```

Indexação de listas

Considere a seguinte lista:

```
> lis <- list(c(3, 8, 7, 4), mat, 5:0)
> lis
[[1]]
[1] 3 8 7 4

[[2]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

[[3]]
[1] 5 4 3 2 1 0
```

Acessar o segundo elemento da lista

```
> lis[[2]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Indexação de listas

Aceder o segundo valor do primeiro componente da lista

```
> lis[[1]][2]
[1] 8
```

Listas nomeadas

- Componentes de uma lista nomeada podem ser acessados usando o operador $

```
> lis <- list(vetor1 = c(3, 8, 7, 4), mat = mat, vetor2 = 5:0)
> lis
$vetor1
[1] 3 8 7 4

$mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

$vetor2
[1] 5 4 3 2 1 0
> lis$mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Indexação de listas

Alternativamente, podemos aceder a lista da seguinte forma:

```
> lis[["mat"]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

O símbolo $ é utilizado para acessar componentes nomeados de listas ou data frames

Indexação de data frame

Considere a seguinte data frame

```
> df1 <- data.frame(A = 4:1, B = c(2, NA, 5, 8))
> df1
  A  B
1 4  2
2 3 NA
3 2  5
4 1  8
```

Para acessar o segundo elemento da primeira coluna (seguindo a mesma lógica das matrizes, viste que tem duas dimensões)

```
> df1[2,1]
[1] 3
> # acesse todas as linhas da coluna B
> df1[,2]
[1]  2 NA  5  8
> # ou
>
> df1[,'B']
[1]  2 NA  5  8
> # ou
>
> df1$B
[1]  2 NA  5  8
```

Indexação de data frame

Aceder todos elementos da primeira linha

```
> df1[1,]
  A B
1 4 2
```

Selecção condicional de data frame

```
> dados <- data.frame(ano = c(2001, 2002, 2003, 2004, 2005),
+                     captura = c(26, 18, 25, 32, NA),
+                     porto = c("SP", "RS", "SC", "SC", "RN"))
>
> #Extraia deste objeto apenas a linha correspondente ao ano 2004:
>
> dados[dados$ano == 2004, ]
   ano captura porto
4 2004      32    SC
> #Mostre as linhas apenas do porto "SC":
>
> dados[dados$porto == "SC", ]
   ano captura porto
3 2003      25    SC
4 2004      32    SC
> #Observe as linhas onde a captura seja maior que 20, selecionando apenas a coluna captura
>
> dados[dados$captura > 20, "captura"]
[1] 26 25 32 NA
```