Introdução a Linguagem R

Tipos de dados e operaç~~oes básicas

Delermando Branquinho Filho

Subsetting

Existem vários operadores que podem ser usados para extrair subconjuntos de objetos R. - `[` sempre retorna um objeto da mesma classe que o original; Pode ser usado para selecionar mais de um elemento (há uma exceção)

- `[[` é usado para extrair elementos de uma lista ou de um data frame; Ele só pode ser usado para extrair um único elemento e a classe do objeto retornado não será necessariamente uma lista ou um data frame.
- `$` é usado para extrair elementos de uma lista ou quadro de dados pelo nome; Semântica são semelhantes às de '[[.

Subsetting

```
> x <- c("a", "b", "c", "c", "d", "a")
> x[1]
[1] "a"
> x[2]
[1] "b"
> x[1:4]
[1] "a" "b" "c" "c"
> x[x > "a"]
[1] "b" "c" "c" "d"
> u <- x > "a"
> u
[1] FALSE TRUE TRUE TRUE TRUE FALSE
> x[u]
[1] "b" "c" "c" "d"
```

Subsetting de uma Matriz

Matrizes podem ser subdivididas da maneira usual com índices de tipo (i, j).

```
x <- matrix(1:6, 2, 3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
x[1, 2]
```

```
## [1] 3
```

```
x[2, 1]
```

```
## [1] 2
```

Os índices também podem estar ausentes.

```
x[1, ]
```

```
## [1] 1 3 5
```

```
x[, 2]
```

```
## [1] 3 4
```

Por padrão, quando um único elemento de uma matriz é recuperado, ele é retornado como um vetor de comprimento 1 ao invés de uma matriz 1×1. Esse comportamento pode ser desativado definindo `drop = FALSE`.

```
x <- matrix(1:6, 2, 3)
x[1, 2]
```

```
## [1] 3
```

```
x[1, 2, drop = FALSE]
```

```
##      [,1]
## [1,]    3
```

Da mesma forma, subconjunto de uma única coluna ou uma única linha lhe dará um vetor, não uma matriz (por padrão).

```
x <- matrix(1:6, 2, 3)
print("x[1,]")
```

```
## [1] "x[1,]"
```

```
x[1, ]
```

```
## [1] 1 3 5
```

```
print("imprimindo x",quote = TRUE)
```

```
## [1] "imprimindo x"
```

```
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
print("Com drop")
```

```
## [1] "Com drop"
```

```
x[1, , drop = FALSE]
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
```

Subsetting Lists

```
x <- list(foo = 1:4, bar = 0.6)
print("---")
```

```
## [1] "---"
```

```
x[1]
```

```
## $foo
## [1] 1 2 3 4
```

```
print("---")
```

```
## [1] "---"
```

```
x$foo
```

```
## [1] 1 2 3 4
```

```
print("---")
```

```
## [1] "---"
```

```
x[[1]]
```

```
## [1] 1 2 3 4
```

```
print("---")
```

```
## [1] "---"
```

```
x$bar
```

```
## [1] 0.6
```

```
print("---")
```

```
## [1] "---"
```

```
x[["bar"]]
```

```
## [1] 0.6
```

```
print("---")
```

```
## [1] "---"
```

```
x["bar"]
```

```
## $bar
## [1] 0.6
```

Subsetting Lists

```
x <- list(foo = 1:4, bar = 0.6, baz = "hello")
x
```

```
## $foo
## [1] 1 2 3 4
##
## $bar
## [1] 0.6
##
```

```
## $baz
## [1] "hello"
```

```
x[c(1, 3)]
```

```
## $foo
## [1] 1 2 3 4
##
## $baz
## [1] "hello"
```

```
x$foo
```

```
## [1] 1 2 3 4
```

```
x$baz
```

```
## [1] "hello"
```

Subsetting Lists

O operador `[[` pode ser usado com índices computados, ou calculados; `$` Só pode ser usado com nomes literais.

```
x <- list(foo = 1:4, bar = 0.6, baz = "hello")
name <- "foo"
x[[name]]  ## Índice computado para 'foo'
```

```
## [1] 1 2 3 4
```

```
x$name     ## Elemento 'nome' não existe!
```

```
## NULL
```

```
x$foo
```

```
## [1] 1 2 3 4
```

Subsetting elementos aninhados em uma lista

O `[[` pode ter uma sequência inteira.

```
x <- list(a = list(10, 12, 14), b = c(3.14, 2.81))
x
```

```
## $a
## $a[[1]]
## [1] 10
##
## $a[[2]]
## [1] 12
##
## $a[[3]]
## [1] 14
##
```

```
##
## $b
## [1] 3.14 2.81
```

```
x[[c(1, 3)]]
```

```
## [1] 14
```

```
x[[1]][[3]]
```

```
## [1] 14
```

```
x[[c(2, 1)]]
```

```
## [1] 3.14
```

Correspondência parcial

A correspondência parcial de nomes é permitida com `[[` e `$`.

```
x <- list(aardvark = 1:5)
x
```

```
## $aardvark
## [1] 1 2 3 4 5
```

```
x$a
```

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

```
x[["a"]]
```

```
## NULL
```

```
x[["a", exact = FALSE]]
```

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

Removendo Valores NA (not available)

Uma tarefa comum é remover valores em falta (`NA`s).

```
x <- c(1, 2, NA, 4, NA, 5)
x
```

```
## [1]  1  2 NA  4 NA  5
```

```
bad <- is.na(x)
bad
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE FALSE
```

```
x[!bad]
```

```
## [1] 1 2 4 5
```

```
x
```

```
## [1]  1  2 NA  4 NA  5
```

E se houver várias coisas a fazer com subconjuntos sem NA?

```
x <- c(1, 2, NA, 4, NA, 5)
x
```

```
## [1]  1  2 NA  4 NA  5
```

```
y <- c("a", "b", NA, "d", NA, "f")
y
```

```
## [1] "a" "b" NA  "d" NA  "f"
```

```
good <- complete.cases(x, y)
good
```

```
## [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE
```

```
x[good]
```

```
## [1] 1 2 4 5
```

```
y[good]
```

```
## [1] "a" "b" "d" "f"
```

Removendo valores NA

```
data("airquality")
airquality[1:6, ]
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
good <- complete.cases(airquality)
airquality[good, ][1:6, ]
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 7    23     299  8.6   65     5   7
## 8    19      99 13.8   59     5   8
```