Tipos de dados em R

Wilson Freitas

Tudo é Objeto

Objetos

Tudo, absolutamente tudo, no R é objeto

Tipos de Objetos

Os objetos possuem tipos físicos e tipos abstratos:

- tipos físicos: tipo que indica como o objeto é armazenado na memória
  - *retornado pela função* `mode`
- tipos abstratos: tipo que define o comportamento do objeto
  - *retornado pela função* `class`

Tipos Atômicos

- São os tipos (`mode`) mais fundamentais mais fundamentais do R:
  - `numeric` *(ou* `double`*),* `complex`*,* `logical`*,* `character`*,* `integer`*,* `raw` *(raw bytes)*

Tipos Físicos

```
c(mode(0), mode(""), mode(list()), mode(c), mode(TRUE))
```

```
## [1] "numeric"   "character" "list"      "function"  "logical"
```

```
mode(data.frame(a=2, b=3))
```

```
## [1] "list"
```

```
mode(as.Date('2000-01-01'))
```

```
## [1] "numeric"
```

Os mais comuns são: `numeric`, `character`, `list`, `function`, `logical`, mas existem outros.

Tipos Abstratos

```
c(class(0), class(""), class(list()), class(c), class(TRUE))
```

```
## [1] "numeric"   "character" "list"      "function"  "logical"
```

```
class(data.frame(a=2, b=3))
```

```
## [1] "data.frame"
```

```
class(as.Date('2000-01-01'))
```

```
## [1] "Date"
```

São os tipos que podem ser criados pelos usuários para adicionar funcionalidades ao R.

Estruturas de dados: Vetores

Vetores

- É a estrutura de dados mais atômica no R

```
1:3
```

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

```
c(1, 2, 3) # função de concatenação **curinga**
```

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

- Não há escalares no R

```
is.vector(1)
```

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

Vetores

- A função `vector(mode="logical", length=0)` cria vetores
- `mode` pode ser um tipo atômico, `list` ou `expression`

```
vector('character', 10)
```

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

```
vector('numeric', 10)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```
vector('list', 2)
```

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

Vetores

- É possível criar vetores de comprimento 0 (`length=0`)

```
vector('integer')
```

```
## integer(0)
```

```
vector('logical')
```

```
## logical(0)
```

Vetores

- As funções dos tipos atômicos podem ser usadas na criação de vetores

```
numeric(5)
```

```
## [1] 0 0 0 0 0
```

```
integer(5)
```

```
## [1] 0 0 0 0 0
```

```
character(5)
```

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

Propriedades dos Vetores

Vetores são homogêneos

- Todos os elementos em um vetor possuem o mesmo `mode`

```
mode(1:4)
```

```
## [1] "numeric"
```

```
mode(5)
```

```
## [1] "numeric"
```

```
mode(integer(0))
```

```
## [1] "numeric"
```

Propriedades dos Vetores

Elemento `NA`

`NA` é `logical`, mas se adapta ao tipo do vetor a que pertence

```
mode(NA)
```

```
## [1] "logical"
```

```
v <- c(1, NA)
mode(v)
```

```
## [1] "numeric"
```

```
mode(v[2])
```

```
## [1] "numeric"
```

Propriedades dos Vetores

Vetores possuem comprimento (função `length`)

```
length(1:4)
```

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

```
length(5)
```

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

```
length(integer(0))
```

```
## [1] 0
```

Propriedades dos Vetores

Indexação

- Vetores podem ser indexados por posição

```
v <- c(21, 42, 63)
v[1]
```

```
## [1] 21
```

```
v[3]
```

```
## [1] 63
```

Propriedades dos Vetores

Indexação

- Vetores podem ser indexados por vetores de índices

```
v[c(2, 3)]
```

```
## [1] 42 63
```

```
v[c(3, 1, 2)]
```

```
## [1] 63 21 42
```

Propriedades dos Vetores

Indexação

- Vetores podem ser indexados por vetores de `logical`

```
v[c(TRUE, FALSE, TRUE)]
```

```
## [1] 21 63
```

Propriedades dos Vetores

Indexação

- Índices negativos retornam o vetor sem o elemento indexado

```
v[-2]
```

```
## [1] 21 63
```

```
v[c(-1, -3)]
```

```
## [1] 42
```

- Não pode misturar índices negativos e positivos

Propriedades dos Vetores

Vetores possuem nomes (função `names`)

```
(v <- c(21, 42, 63))
```

```
## [1] 21 42 63
```

```
names(v)
```

```
## NULL
```

```
names(v) <- c("1M", "2M", "3M")
v
```

```
##  1M  2M  3M
## 21 42 63
```

Propriedades dos Vetores

- Com elementos nomeados, o vetor pode ser indexado pelos nomes

```
v["2M"]
```

```
## 2M
## 42
```

```
v[c("1M", "3M")]
```

```
## 1M 3M
## 21 63
```

Propriedades dos Vetores

- Indexação de elementos que não existem retorna `NA`

```
(v <- 1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
v[11]
```

```
## [1] NA
```

- Atribuição de elementos fora do índices cria elementos `NA`

```
v[20] <- 2
v
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 NA NA NA NA NA NA NA NA NA  2
```

Propriedades dos Vetores

- Podemos concatenar elementos a um vetor com a função c

```
v <- 1:5
c(v, 1:3, -1, 15:22)
```

```
##  [1]  1  2  3  4  5  1  2  3 -1 15 16 17 18 19 20 21 22
```

Estruturas de dados: Listas

Listas

Listas são heterogêneas

- Listas podem acomodar objetos de diferentes tipos

```
(cid <- list("Wilson Freitas", 38, as.Date('1976-07-12')))
```

```
## [[1]]
## [1] "Wilson Freitas"
##
## [[2]]
## [1] 38
##
## [[3]]
## [1] "1976-07-12"
```

Listas

Listas também possuem nomes

```
names(cid)
```

```
## NULL
```

```
names(cid) <- c("nome", "idade", "nascimento")
cid
```

```
## $nome
## [1] "Wilson Freitas"
##
## $idade
## [1] 38
##
## $nascimento
## [1] "1976-07-12"
```

Listas

- Listas podem ser criadas diretamente com nomes

```
list(nome="Wilson Freitas", idade=38, nascimento=as.Date('1976-07-12'))
```

```
## $nome
## [1] "Wilson Freitas"
##
## $idade
## [1] 38
##
## $nascimento
## [1] "1976-07-12"
```

Listas

Indexação

- Listas podem ser indexadas por posição e vetores de índices

```
cid[1]
```

```
## $nome
## [1] "Wilson Freitas"
```

```
cid[c(2, 3)]
```

```
## $idade
## [1] 38
##
## $nascimento
## [1] "1976-07-12"
```

Listas

Indexação

- Listas podem ser indexadas por nomes

```
cid['nome']
```

```
## $nome
## [1] "Wilson Freitas"
```

```
cid[c('idade', 'nascimento')]
```

```
## $idade
## [1] 38
##
## $nascimento
## [1] "1976-07-12"
```

Listas

- O operador de indexação `[` de uma lista, retorna uma sublista

```
nome <- cid['nome']
is.list(nome)
```

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

- Para acessar o conteúdo do elemento de uma lista devemos usar o operador `[[` ou `$`

```
cid[['nome']]
```

```
## [1] "Wilson Freitas"
```

```
cid$nome
```

```
## [1] "Wilson Freitas"
```

- O operador `[[` não aceita múltiplos índices.

Listas

- Podemos adicionar elementos a uma lista

```
cid$sapato <- 42
cid[['casado']] <- TRUE
cid
```

```
## $nome
## [1] "Wilson Freitas"
##
## $idade
## [1] 38
##
## $nascimento
## [1] "1976-07-12"
##
## $sapato
## [1] 42
##
## $casado
## [1] TRUE
```

Listas

- Podemos concatenar uma elementos a uma lista sem alterá-la com a função c

```
c(cid, filhos=2, paciência=0)
```

```
## $nome
## [1] "Wilson Freitas"
##
## $idade
## [1] 38
##
## $nascimento
## [1] "1976-07-12"
##
## $sapato
## [1] 42
##
## $casado
## [1] TRUE
##
## $filhos
## [1] 2
##
## $paciência
## [1] 0
```

A função c

- Ela combina valores em um vetor ou uma lista
  - *Para valores homogêneos ela retorna um vetor*
  - *Para valores heterogêneos ela segue uma regra de precedência para coerção:*

```
NULL < raw < logical < integer < double < complex < character < list < expression
```

```
c(TRUE, FALSE, 10)
```

```
## [1]  1  0 10
```

```
c(TRUE, FALSE, 10, 1i)
```

```
## [1]  1+0i  0+0i 10+0i  0+1i
```

```
c(TRUE, FALSE, 10, 1i, "n")
```

```
## [1] "TRUE"  "FALSE" "10"    "0+1i"  "n"
```

A função c

```
c(TRUE, FALSE, 10, 1i, "n", list(5))
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] FALSE
##
## [[3]]
## [1] 10
##
## [[4]]
## [1] 0+1i
##
## [[5]]
## [1] "n"
##
## [[6]]
## [1] 5
```