Introdução ao R

Willber Nascimento

2022-06-19

Aula 2 - Estruturas de dados em R : vetores, listas e data.frames

Vetores e Listas

Um vetor (`vector`) é uma estrutura de dados que aloca elementos dentro dele. Eles são dois tipos principais: vetores atômicos e listas. A principal diferença entre esses dois tipos de vetores é que os primeiros possuem apenas elementos de um único tipo, enquanto as listas podem possuir elementos de vários.

O exemplo de um vetor no R

```
meu_vetor <- 1
```

Vetores com mais de um elemento

```
meu_vetor <- c(1,2,3,4)
meu_vetor
```

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

Um vetor de texto

```
texto <- c("Eu", "sou", "um vetor de ", "texto")
```

Verificação de tipos

```
typeof(meu_vetor)
```

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

```
typeof(texto)
```

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

Agora um vetor que combina números e textos

```
vetor <-  c(1,2, '3')
is.numeric(vetor)
```

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

```
typeof(vetor)
```

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

Podemos acessar o tamanho de um vetor com a função `length`

```
length(vetor)
```

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

```
str(vetor)
```

```
##  chr [1:3] "1" "2" "3"
```

acessando elementos de um vetor

Podemos acessar os elementos do vetor `numeros` usando os colchetes `[]` indicando o **índice** ou a posição desejada.

```
set.seed(12345)
numeros <- sample(1:100, 20, replace=T)
numeros
```

```
##  [1] 14 51 80 90 92 24 58 93 75 96 88  2 86 75 38 94 10 81 32 40
```

```
numeros[1]        # acessa o elemento no index 1
```

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

```
numeros[10]       # acessa o elemento no index 10
```

```
## [1] 96
```

```
numeros[13:17]    # acessa os elementos nos indexes 13 até o 17
```

```
## [1] 86 75 38 94 10
```

```
numeros[-1]       # acessa os elementos do vetor excluindo o da posicao 1.
```

```
##  [1] 51 80 90 92 24 58 93 75 96 88  2 86 75 38 94 10 81 32 40
```

```
numeros[c(2,5)]   # retorna os elementos na posicao 2 e na posicao 5
```

```
## [1] 51 92
```

operações com vetores

Operações matemáticas

```r
c(1,2,3) + 2
```

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

Por padrão, a operação atua sobre cada elemento do vetor e retorna um resultado para um deles.

Resumos estatísticos agregam os valores dentro do vetor.

```r
mean(numeros)
```

```
## [1] 60.95
```

```r
sd(numeros)
```

```
## [1] 31.81108
```

```r
median(numeros)
```

```
## [1] 75
```

Operações com caracteres

```r
texto <- c('a', 'b', 'c')
toupper(texto)
```

```
## [1] "A" "B" "C"
```

Listas

Listas são vetores mais complexos e expandem as funcionalidades que temos com vetores atômicos uma vez que podemos alocar tipos de vetores diferentes dentro das listas. A forma comum de usar uma lista pode ser:

```r
x <- list(1, 2, 3)
x
```

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

Podemos averiguar o tipo do objeto, como padrão

```
typeof(x)
```

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

Adicionalmente, os elementos das listas podem ser nomeados

```
minha_lista <- list(x = 123, b = c("verde","branco","vermelho"), c = c(TRUE, FALSE))
minha_lista
```

```
## $x
## [1] 123
##
## $b
## [1] "verde"    "branco"   "vermelho"
##
## $c
## [1]  TRUE FALSE
```

Podemos usar a função `str` para visualizar toda estrutura da lista:

```
str(minha_lista)
```

```
## List of 3
##  $ x: num 123
##  $ b: chr [1:3] "verde" "branco" "vermelho"
##  $ c: logi [1:2] TRUE FALSE
```

Podemos inclusive adicionar listas dentro de listas:

```
pessoa1 <- list(produtos=c("a", "b", "c"), quantidade=c(3,2,7), avaliacao=5)

pessoa2 <- list(produtos=c("d", "e", "f"), quantidade=c(1,4,6),avaliacao=3)


lista_pessoas <- list(pessoa1, pessoa2)
lista_pessoas
```

```
## [[1]]
## [[1]]$produtos
## [1] "a" "b" "c"
##
## [[1]]$quantidade
## [1] 3 2 7
##
## [[1]]$avaliacao
## [1] 5
##
##
## [[2]]
## [[2]]$produtos
```

```
## [1] "d" "e" "f"
##
## [[2]]$quantidade
## [1] 1 4 6
##
## [[2]]$avaliacao
## [1] 3
```

Acessando elementos de uma lista

Como nos vetores, podemos selecionar apenas alguns elementos de uma lista usando uma notação bastante semelhante. Você precisará usar `[]` ou `[[]]` a depender do que você deseja acessar.

Acessando a primeira pessoa

```
lista_pessoas[[1]]
```

```
## $produtos
## [1] "a" "b" "c"
##
## $quantidade
## [1] 3 2 7
##
## $avaliacao
## [1] 5
```

Acessando os produtos da primeira pessoa

```
lista_pessoas[[1]][2]
```

```
## $quantidade
## [1] 3 2 7
```

Caso queira apenas os valores naquele vetor, sem o nome do elemento, use mais colchetes para desempacotar os itens.

```
lista_pessoas[[1]][[2]]
```

```
## [1] 3 2 7
```

Como você ver, o indexador `[]` retorna a lista na posição indicada. Isto é, ela indexa e retorna uma lista menor. Contudo, você pode estar interessado em acessar os elementos detalhados dentro de uma das listas em questão usando o indexador `[[]]`.

Finalmente, podemos usar o símbolo `$` para acessar elementos nomeados dentro das listas. Lembrando que quando temos multiplas listas aninhadas é necessário indexar a lista primeiro.

```
lista_pessoas[[1]]$produtos
```

```
## [1] "a" "b" "c"
```

Podemos chamar os itens pelo seus rótulos também

```r
lista_pessoas[[1]][["avaliacao"]]
```

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

Data Frames

Um data frame é uma estrutura de dados semelhante a uma planilha tradicional. Ele é uma combinação de vetores onde as linhas representam observações e as colunas representam características dessas observações.

Agora, vamos criar um data frame no R.

```r
estados <- c("AL", "PE", "DF")
idh <- c(0.641, 0.673, 0.863)

df <- data.frame(estados, idh)
df
```

```
##   estados   idh
## 1      AL 0.641
## 2      PE 0.673
## 3      DF 0.863
```

Vejamos a estrutura do objeto `df`

```r
str(df)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ estados: chr  "AL" "PE" "DF"
##  $ idh    : num  0.641 0.673 0.863
```

Podemos também usar a função `dim` para observamos a estrutura do banco de dados

```r
dim(df)
```

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

O resultado apresenta o número de linhas e colunas respectivamente.

Podemos acessar os seus elementos como as listas `[[]]` ou como matrizes `[,]`. Sendo esta última a forma mais recorrente.

```r
df[[1]]
```

```
## [1] "AL" "PE" "DF"
```

```r
df[,1]
```

```
## [1] "AL" "PE" "DF"
```

A forma de matriz [linha, coluna] é interessante já que podemos as linhas de forma intuitiva. Por exemplo:

```
df[1,1]
```

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

Podemos utilizar o operador $para buscar as colunas baseados no seu nome

```
df$idh
```

```
## [1] 0.641 0.673 0.863
```

```
df$idh[3]
```

```
## [1] 0.863
```

Como você nota, o RStudio já autocompleta para você a medida que você digita as letras que quer buscar. Além disso, o atalho `alt+tab` faz isso também.

Adiconando linhas e colunas

Podemos adicionar novas linhas a um data frame usando a função `rbind`

```
sp <- c("SP", 0.789)

df <- rbind(df, sp)

df
```

```
##   estados   idh
## 1      AL 0.641
## 2      PE 0.673
## 3      DF 0.863
## 4      SP 0.789
```

Veja que precisamos fornecer linhas com a mesma quantidade de elementos do data frame original. A função `cbind` pode ser usada para adicionar novas colunas

```
regiao <- c("NE", "NE","CO", "SE")

df <- cbind(df, regiao)

df
```

```
##   estados   idh regiao
## 1      AL 0.641     NE
## 2      PE 0.673     NE
## 3      DF 0.863     CO
## 4      SP 0.789     SE
```

Filtros nos data.frames

Filtrar elementos de um data frame é uma tarefa muito corriqueira. Aqui só precisamos expandir a notação dos indices ou usar uma função para isso. No primeiro caso, vamos fitrar os estados da região nordeste

```
df[df$regiao=="NE", ]
```

```
##   estados   idh regiao
## 1      AL 0.641     NE
## 2      PE 0.673     NE
```

Note a importância da vírgula ao final. Sem ela, o retorno não seria um data frame filtrando as informações necessárias. Podemos ainda expandir essa notação inserindo mais critérios:

```
df[df$regiao=="NE" & df$idh > 0.65, ]
```

```
##   estados   idh regiao
## 2      PE 0.673     NE
```

Como você pode ver, quanto mais critérios de filtros, mais complexa se torna a notação. Para isso, já existe algumas funções disponíveis para facilitar o trabalho, abaixo vemos a utilização do `subset`

```
subset(df, regiao=="NE" & idh > 0.65)
```

```
##   estados   idh regiao
## 2      PE 0.673     NE
```

A função permite que mantenhamos o banco de dados estatístico sem a necessidade de repetição de seu nome ao longo da chamada de cada filtro.

Nomeando e renomeando variáveis nos data.frames

Frequentemente é necessário nomear ou renomear as variáveis de um banco de dados. Seja por que eles ainda não tem ou porque você deseja nomes mais adequados. Para isso, usamos a função `names`

```
names(df)
```

```
## [1] "estados" "idh"     "regiao"
```

```
names(df) <- c("sigla_uf", "indice.de.idh", "REGIÃO da UF")
df
```

```
##   sigla_uf indice.de.idh REGIÃO da UF
## 1       AL         0.641           NE
## 2       PE         0.673           NE
## 3       DF         0.863           CO
## 4       SP         0.789           SE
```

Se chamarmos apenas a função `names(objeto)` ela retornará os nomes atuais. Se atribuirmos `<-` um vetor de caracteres ao objeto seus nomes serão alterados.

Exercícios

1. Crie um data.frame chamado `maiores` com os seguintes dados: o banco terá possuirá duas colunas, time e títulos do brasileirão; o banco terá dez linhas com os nomes dos dez clubes com mais taças nacionais. Os dados são: Palmeiras 10 títulos, Santos 8, Flamengo 8,Corinthians 7, São Paulo 6, Cruzeiro 4, Fluminense 4, Vasco 4, Internacional 3, Bahia 2.

2. Renomei as colunas do banco de dados `maiores` para `clubes` e `tacas_BR`.

3. Faça uma seleção que retorna apenas a coluna de títulos.

4. Faça uma seleção que retorna apenas os times com mais de 8 títulos.

   4.1 Faça uma seleção que retorna os times com mais que dois títulos e menos que 6 títulos.

5. Crie uma coluna com as siglas dos estados de cada um dos times. Palmeiras, Santos, Corinthians, São Paulo deveriam receber o valor 'SP'; Flamengo, Fluminense e Vasco deveriam receber o valor de 'RJ'; Cruzeiro 'MG'; Internacional 'RS'; e Bahia 'BA'.

6. Qual o tipo de cada uma das colunas?

Como pedir ajuda?

Todo o ecossistema do R conta com uma ampla documentação. Você pode consultá-la direto do RStudio com `? alguma coisa`.

```
?is.logical
?as.numeric
```

Porém, nem sempre nossa dúvida será tirada consultando a documentação ofical. Muitas vezes vamos precisar consultar os universitários. Eles estão no StackOverflow. Lá, centenas de dúvidas e erros comuns já possuem resposta e você pode se juntar a comunidade para fazer consultas e para ajudar nas respostas. Normalmente, usamos o Google para encontrar as respostas que precisamos usando `como fazer isso aqui in r`. É importante usar a pesquisa em inglês visto que a grande maioria de soluções estará nessa linguagem.

Outros materiais

- Gerenciamento de projetos com R e Rstudio https://odysee.com/@willbernascimento:8/Git-Github-e-RStudio-Gerenciamento-de-projetos-de-analise-de-dados
- Como instalar o R e o RStudio no Windows
- Como gerar tabelas e gráficos com PSPP