Disciplina

R para Ciência de Dados – Aula 5

Manipulação de Objetos no R

Alan Rodrigo Panosso
Ciências Exatas (UNESP/Jaboticabal)

alan.panosso@unesp.br

Objetos no R

Objetos

no R são todas as ***estruturas de dados*** construídas por meio de atribuição

<-

- Recordando as regras para os nomes de objetos:

  - 1) Devem ***iniciar*** com um ***caractere alfabético***.
  - 2) Podem ser seguidos por mais caracteres alfabéticos e/ou numéricos.
  - 3) ***Não é permitido*** o uso de espaço em branco ou de caracteres especiais, como: @, #, &, *, +, ?,$ (***exceto*** o "_" e o "***.***").
  - 4) ***Não poderá ser uma palavra reservada*** a uma instrução do algoritmo (`if, for, while, repeat` ou nomes de funções).
  - 5) Devem ***ser significativos***.
  - 6) Apesar da linguagem ***aceitar acentuação***, a sua utilização ***não é recomendada***.

Objetos no R

Construção de objetos

Um objeto pode ser criado com a operação de "***atribuição***".

`<-` Forma pela qual os objetos recebem os dados no R, composta pelos símbolos "menor que" e "subtração" , ou "subtração" e "maior que" **SEM O ESPAÇO ENTRE ELES**.

```
x <- 5  #o objeto x recebe o valor 5
```

ou

```
15 -> X #o valor 15 é armazenado em no objeto X
```

`=` A atribuição também pode ser feita com o "sinal de igual", frequentemente utilizado na definição de ***argumentos*** das funções

```
y = 9 #o objeto y recebe o valor 9
9 = y #Erro, lado da atribuição inválido
```

Objetos no R

As principais ***classes*** de ***objetos*** do R são: ***escalar***, ***vetor***, ***lista***, ***fator***, ***matriz*** e ***data.frame***.

Objeto	Tipos	Suporta tipos Diferentes
escalar	apenas um elemento numérico, caractere, complexo ou lógico.	não
vetor	numérico, caractere, complexo ou lógico.	não
fator	numérico ou caractere.	não
matriz	numérico, caractere, complexo ou lógico.	não
data.frame	numérico, caractere, complexo ou lógico.	sim
lista	numérico, caractere, complexo, lógico função, expressão, entre outros.	sim

Objetos no R

Variável **=** *Um elemento*

Estrutura de dados **=** *Um conjunto*

Quando uma ***estrutura de dados*** é composta de variáveis ***com o mesmo tipo primitivo***, temos um conjunto ***homogêneo*** de dados (vetores, fatores e matrizes, por exemplo).

Escalar

É um objeto ao qual é atribuído um valor que pode ter qualquer um dos tipos, numérico, caractere, complexo ou lógico.

```
> x <- 10; x
[1] 10
> "Maria" -> nome; nome
[1] "Maria"
> y <- 3 - 2i; y
[1] 3-2i
> res  <- (x==20); res
[1] FALSE
```

Objetos no R

Vetor

É considerada ***a forma mais simples de armazenamento de dados*** em um objeto, são classificados como variáveis com um ou mais valores do mesmo tipo. Na verdade, um ***escalar*** para o R é definido como ***um vetor com um único elemento***. Essa estrutura de dados tem o 1 como índice inicial.

```
> c(3,2,7,4,5)
[1] 3 2 7 4 5
> 1:10
[1]  1  2  3  4  5  6  7  8  9 10
> seq(0,100,25)
[1]   0  25  50  75 100
> rep(1,10)
[1] 1 1 1 1 1 1 1 1 1 1
> rep(1:3,4)
[1] 1 2 3 1 2 3 1 2 3 1 2 3
> rep("testemunha",4)
[1] "testemunha" "testemunha" "testemunha" "testemunha"
> x<-1:100
> is.vector(x)
[1] TRUE
> is.vector(z)
[1] TRUE
> length(z)
[1] 0
```

Fator

É um tipo particular de ***vetor***, onde cada elemento repetido (uma ou mais vezes) é considerado ***um nível do fator***. ***Muito usado nas análises estatísticas*** – variáveis classificatórias, utilize a função `gl()` para criar os fatores. Já, a função `levels()` retorna os níveis de determinado fator.

```
> gl(2,3)
[1] 1 1 1 2 2 2
Levels: 1 2
> gl(2,3,24)
 [1] 1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2
Levels: 1 2
> gl(4,3,labels=c("Trat.1","Trat.2","Trat.3","testemunha"))
 [1] Trat.1     Trat.1     Trat.1     Trat.2     Trat.2     Trat.2
 [7] Trat.3     Trat.3     Trat.3     testemunha testemunha testemunha
Levels: Trat.1 Trat.2 Trat.3 testemunha
> as.factor(rep(1:3,5))
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3
> x<-gl(5,4)
> is.factor(x)
[1] TRUE
> levels(x)
[1] "1" "2" "3" "4" "5"
```

Matriz

Objeto com ***n*** × ***m*** componentes (***n*** linhas e ***m*** colunas), com o mesmo atributo, que são referenciados por dois índices da seguinte forma [n, m].

```
c1 <- c(1,1,1,2,2,2)
c2 <- c(1:3,1:3)
c3<-c(12,14,16,25,22,29)
c1; c2; c3

m1<-cbind(c1,c2,c3); m1     # cbind cria a matriz tendo os vetores como colunas
mode(m1)
class(m1)
length(m1) # retorna o número total de valores da matriz

m2<-rbind(c1,c2,c3) # rbind cria a matriz tendo os vetores como linhas
m2
mode(m2); class(m2)

x<- c(rep(1:2,c(3,3)),1:3,1:3,12,14,16,25,22,29) #criando o vetor x
m3<-matrix(x, ncol=3) # ncol é o numero de colunas da matriz
m3

m4<-matrix(x, ncol=3, byrow=T) # byrow = dados serão organizados por linhas.
m4
```

Objetos no R

```
m1[,3]  #especifica  a coluna 3 da matriz m1
m1[2,]  #especifica  a linha 2 da matriz m1
m1[2,3] #especifica  o elemento da linha 2 e coluna 3 da matriz m1
m1[c(1,3,5),] #extraindo 3 linhas (linha1, linha3 e linha5)
m1[,2:3]         #extraindo 2 colunas (col5, col6 e col7)
m1[c(2,3),c(1,3)] #extraindo uma sub-matriz 2x2
m1[c(4,5,6),2:3] #extraindo uma sub-matriz 3x2

A=matrix(c(1,3,2,8,9,11,0,4,3), ncol=3,byrow=T); A
B=matrix(c(rep(1,3),rep(2,3),rep(3,3)),ncol=3); B
S=A+B; S# soma de matrizes
D=A-B; D# subtração de matrizes
P=A %*% B; P     # produto de matrizes
T=t(A); T        # transposta de matriz A
det(A); det(B)   # determinante de matrizes A e B
IA=solve(A); IA  # inversa de matriz A
round(A%*%IA,0)  # matriz identidade
dim(A)  # dimensões da matriz (linhas colunas)
summary(A)       # medidas-resumo da matriz
summary(as.vector(A))
colSums(A)       #retorna a soma das colunas da matriz z
rowSums(B)       #retorna a soma das linhas da matriz z
```

Estrutura de dados

Um ***conjunto heterogêneo de dados*** é composto por elementos que ***não são do mesmo tipo primitivo***, como exemplo, temos os data.frames e as listas (ou registros).

data.frame

Os data.frames são considerados a melhor forma de armazenar dados, pois cada linha corresponde a uma unidade (amostra), indivíduo ou pessoa, e cada coluna representa uma medida realizada em cada unidade (variáveis).

Trat.	Repetições				
	1	2	3	4	5
1	25,5	28,4	24,1	27,5	26,3
2	31,6	30,5	29,3	31,1	29,4

```
df1<-data.frame(Trat = gl(2,5),Rep = c(1:5,1:5),
y = c(25.5,  28.4, 24.1, 27.5, 26.3, 31.6, 30.5, 29.3, 31.1, 29.4))
length(df1)      # retorna o número de colunas do data.frame
names(df1)       # retorna o nome das colunas do data.frame
df1[1]           # retorna o primeiro elemento do data.frame (coluna tr)
df1$Trat         # retorna os componentes da coluna tr
df1[,1]          # semelhante a df1$tr
df1[3,]          # linha 3 do data.frame
```

Objetos no R

Listas

São usadas para combinar diferentes tipos de objetos (***vetores***, ***matrizes***, ***números*** ou ***caracteres***) em um mesmo objeto. Os valores de uma lista podem ser referenciados por um índice.

```
lista1 <- list(nome= "Maria", idade=18, notas=c(98,95,96)); lista1
names(lista1)      # retorna os nomes dos componentes de lista1
lista1$nome        # extraindo o componente "nome" da lista "lista1"
lista1$notas       # extraindo as notas da lista "lista1"
lista1$notas[2]    # segundo elemento do componente "notas" da lista "lista1"
lista1[1]          # primeiro elemento da lista "lista1"
lista1[2]          # segundo elemento da lista "lista1"
lista1[3]          # terceiro elemento da lista "lista1"
lista1[[1]]        # veja a diferença, semelhante a lista1$nome
lista1[[2]]        # veja a diferença, semelhante a lista1$idade
lista1[[3]]        # terceiro componente da lista "lista1"
lista1[[3]][2]     # segundo elemento do terceiro componente da lista "lista1"
lista1$notas[2]    # idêntico a "lista1"[[3]][2]
mode(lista1)
class(lista1)
mode(lista1$nome)  # semelhante mode(lista1[[1]])
mode(lista1$idade) # semelhante mode(lista1[[2]])
```

Até agora observamos que existem vário tipo de objetos no R

Coerção é a conversão entre os diferentes tipos de objetos, por meio de funções específicas

Testando os tipos de objetos

Utilize as funções `is.tipo` para testar os diferentes tipos de objetos construídos até agora. Para converter esses objetos em formas específicas, utilize as funções `as.tipo`.

Type	Testing	Coercing
Array	is.array	as.array
Character	is.character	as.character
Complex	is.complex	as.complex
Dataframe	is.data.frame	as.data.frame
Double	is.double	as.double
Factor	is.factor	as.factor
List	is.list	as.list
Logical	is.logical	as.logical
Matrix	is.matrix	as.matrix
Numeric	is.numeric	as.numeric
Raw	is.raw	as.raw
Time series (ts)	is.ts	as.ts
Vector	is.vector	as.vector

Objetos no R

Para criar uma função no R é necessário a seguinte atribuição:

```
nomefuncao <- function(argumento1, ..., argumento_n)
```

O uso da função criada é:

```
nomefuncao(argumento1, ..., argumento_n)
#criando função para calcular a média
media = function(dados)
        {
        valor = sum(dados)/length(dados)
        print(valor)
        }
x = seq(1, 10, 2)
media(x)
#vamos editar o vetor x e inserir um NA
x = edit(x)
media(x) #a nossa função se perde devido à presença do NA
mean(x, na.rm = T) #a função existente no R desconsidera o NA
```

Objetos no R

```
#criando função para calcular o desvio-padrão
meudp = function(x){
        xbarra = mean(x)
        n = length(x)
        somax = sum(x)
        somax2 = sum(x*x)
        numerador = somax2-((somax^2)/n)
        denominador = n-1
        variancia = numerador/denominador
        desviopadrao = sqrt(variancia)
        return(desviopadrao)
        }
qqcoisa = c(1, 3, 3, 2, 5, 5, 5, 6, 4)
meudp(qqcoisa)
sd(qqcoisa) #apenas para conferir!
```

Objetos no R

```
#criando função para calcular o desvio-padrão
meuresumo= function(x){
        xbarra = mean(x)
        n = length(x)
        somax = sum(x)
        somax2 = sum(x*x)
        numerador = somax2-((somax^2)/n)
        denominador = n-1
        variancia = numerador/denominador
        desviopadrao = sqrt(variancia)
        list(n_amostral = n, media = xbarra, sigma2 = variancia, sigma =
    desviopadrao)
                                    }
qqcoisa = c(1, 3, 3, 2, 5, 5, 5, 6, 4)
meuresumo(qqcoisa)
```

RESOLVER A LISTA 07