

**UNIVERSIDADE DA REGIÃO DE JOINVILLE - UNIVILLE**

Bacharelado em Engenharia de Software (BES)

## **Estatística para computação**

**Professora Priscila Ferraz Franczak**

Engenheira Ambiental - UNIVILLE

Mestre em Ciência e Engenharia de Materiais - UDESC

Doutoranda em Ciência e Engenharia de Materiais - UDESC

[priscila.franczak@gmail.com](mailto:priscila.franczak@gmail.com)

# Plano de Aula

## 1. Vetores

1.1 Sequência

1.2 Repetições

1.3 Índice de vetores

## 2. Matrizes

2.1 Comandos `cbind( )` e `rbind( )`

2.2 Índices de matrizes

## 3. Arrays

3.1 Criando arrays

3.2 Índices em arrays

## 4. Exercícios



Todas as variáveis criadas pelo R são chamadas de objetos.

A tabela abaixo sintetiza os objetos e seus possíveis atributos:

<b>Objeto</b>	<b>Tipos</b>	<b>Suporta tipos diferentes</b>
Vetor	Numérico, caractere, complexo ou lógico	Não
Fator	Numérico ou caractere	Não
Matriz	Numérico, caractere, complexo ou lógico	Não
Array	Numérico, caractere, complexo ou lógico	Sim
Data.frame	Numérico, caractere, complexo ou lógico	Sim
ts	Numérico, caractere, complexo ou lógico	Sim
lista	Numérico, caractere, complexo, lógico, função, expressão, etc.	Sim

# 1. Vetores

- Uma das principais estruturas de dados do R é o vetor (**vector**).
- Pode armazenar mais de um valor.
- A maneira mais simples de criar um vetor é usando o comando `c( )`, que concatena elementos em um mesmo objeto.
- Concatenar: colocar em ordem, ligar ideias, argumentos.



Para declarar mais de um elemento dentro de um vetor, utilizaremos a seguinte sintaxe:

$$x=c(a_1, a_2, a_3, a_4, \dots, a_n)$$

o "c" significa combinação, porque vários elementos estão sendo combinados em um vetor.

## Exemplo:

```
>x<-c(2,3,5,7,11)    #os cinco primeiros  
números primos  
>x    #exibe o conteúdo do objeto x  
[1] 2 3 5 7 11
```

- Os argumentos de `c()` podem ser tanto elementos únicos quanto outros objetos com um ou mais elementos:

```
>y<-c(x,13,17,19)    #adicionando três  
números primos  
>y #exibindo y  
[1] 2 3 5 7 11 13 17 19
```

## 1.1 Sequências

- Uma das maneiras mais fáceis de escrever uma sequência numérica de valores é usar o comando `seq()`.
- Tem como argumentos: Início (from), Fim (to), Incremento (by).

`seq(início, fim, incremento)`



```
>seq(1,10,1) #sequência de inteiros de 1 a 10
```

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

```
>seq(1,10,2) #não termina em 10
```

```
[1] 1 3 5 7 9
```

```
>seq(10,1,3) #ordem inversa, equivocada
```

```
Error in seq.default(10, 1, 3) : sinal errado  
no argumento 'by'
```

```
>seq(10,1,-3) #usando incremento negativo
```

```
[1] 10 7 4 1
```



- Para sequências de números inteiros, em que o incremento é 1 (positivo ou negativo), pode-se usar “dois pontos”:

```
>a<-1:10 #cria sequência de inteiros de  
1 a 10
```

```
>a #exibe o conteúdo do objeto a  
[1] 1 2 3 4 5 6 7 8 9 10
```

```
>b<-10:1 #cria sequência de inteiros de  
10 a 1
```

```
>b #exibe o conteúdo do objeto b  
[1] 10 9 8 7 6 5 4 3 2 1
```

- Se o **vetor é muito longo** e não cabe em uma linha, o R vai usar as linhas seguintes para continuar imprimindo o vetor:

```
>longo<-100:50                                #sequência  
decrecente de 100 a 50  
>longo #exibe o conteúdo do objeto  
[1] 100 99 98 97 96 95 94 93 92 91  
[11] 90 89 88 87 86 85 84 83 82 81  
[21] 80 79 78 77 76 75 74 73 72 71  
[31] 70 69 68 67 66 65 64 63 62 61  
[41] 60 59 58 57 56 55 54 53 52 51  
[51] 50
```

## 1.2 Repetições

- Outro comando útil para produzir vetores é o `rep()`.
- Retorna o primeiro argumento repetido pelo número de vezes indicado pelo segundo argumento.

```
>rep(1,10) #cria uma repetição
```

```
>[1] 1 1 1 1 1 1 1 1 1 1
```

```
>rep(c(1,2),5) #repete o vetor 5 vezes
```

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

```
>c(rep(0,10), rep(1,5)) #concatenando repetições
```

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



- Pode-se ainda utilizar objetos como argumentos:

```
>x<-10
```

```
>rep(c(1,2),x) #repetição de "c(1,2)" x  
vezes
```

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

## 1.3 Índices de vetores

- Os elementos armazenados em um vetor podem ser acessados através dos chamados índices, com o uso de colchetes [ ].

```
>x<-5:1      #sequência de 5 a 1
```

```
>x          #exibindo x
```

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

```
>x[3]       #apenas o 3º elemento de x
```

```
[1] 3
```

```
>x[2:4]     #do 2º ao 4º elemento de x
```

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

```
>x[c(2,4)]  #2º e 4º elementos
```

```
[1] 4 2
```

- Podemos também construir expressões lógicas para acessar determinados elementos de um vetor:

```
>x<-(5:1)
```

```
>x
```

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

```
>x[x<4] #só os elementos menores que 4
```

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



- Índices precedidos de sinal negativo (como se fossem valores negativos) podem eliminar elementos de um vetor.

```
>>x<-(5:1)
```

```
>x
```

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

```
>x[-2]      #todos, menos o 2º elemento de x
```

```
[1] 5 3 2 1
```

## 2. Matrizes

- Há várias formas de criar uma matriz.
- O comando `matrix( )` recebe um vetor como argumento e o transforma em uma matriz de acordo com as dimensões especificadas:

```
x<-matrix(data=dados, nrow=m, ncol=n, byrow=Q)
```

m= número de linhas

n= número de colunas

Se `Q = 1`, ativa disposição por linhas (T)

Se `Q = 0`, mantém disposição por colunas (F)

# Exemplo

```
x<-matrix(data=dados,nrow=m,ncol=n,byrow=Q)
```

```
> A<-matrix(c(1:10),2,5,1)
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
> A<-matrix(c(1:10),2,5,T)
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
> |
```

OBS: por padrão, o **byrow** é definido como FALSE (0). Se não houver identificação, a orientação será por colunas.



# Exemplo

```
x<-matrix(data=dados,nrow=m,ncol=n,byrow=Q)
```

```
> x<-(1:12)
> x
[1]  1  2  3  4  5  6  7  8  9 10 11 12
> B<-matrix(x,ncol=3)
> B
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> |
```

Só informamos o vetor e o número de colunas aqui.

[3,2] representa o elemento localizado na primeira linha e na primeira coluna, que neste caso é o 7.

- O comando `summary( )` também pode ser usado para obter informações de qualquer objeto, inclusive de matrizes.
- Quando utilizado com matrizes, ele opera nas colunas da matriz como se cada uma delas fosse um vetor, calculando algumas medidas descritivas:

```
> x<-(1:12)
```

```
> x
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
> B<-matrix(x,ncol=3)
```

```
> B
```

```
      [,1] [,2] [,3]
[1,]     1     5     9
[2,]     2     6    10
[3,]     3     7    11
[4,]     4     8    12
```

```
>
```

```
> summary(B)
```

	v1	v2	v3
Min.	:1.00	Min. :5.00	Min. : 9.00
1st Qu.:	:1.75	1st Qu.:5.75	1st Qu.: 9.75
Median	:2.50	Median :6.50	Median :10.50
Mean	:2.50	Mean :6.50	Mean :10.50
3rd Qu.:	:3.25	3rd Qu.:7.25	3rd Qu.:11.25
Max.	:4.00	Max. :8.00	Max. :12.00



- Caso deseje um resumo de todos os elementos da matriz, transforme em vetor antes de usar o `summary( )`:

```
> summary(as.vector(B))
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.00   3.75   6.50   6.50   9.25  12.00

> summary(x)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.00   3.75   6.50   6.50   9.25  12.00
```

Transforma a matriz no vetor x, de partida.

## 2.1 Comandos `cbind( )` e `rbind( )`

`bind` = ligar, vincular.

`c` = columns

`r` = rows

- Esses comandos concatenam colunas ou linhas, respectivamente, na matriz (ou vetor) original.

- Adicionando columnas:

```
> x<-matrix(10:1,ncol = 2)
```

```
> x
```

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

```
> y<-cbind(x,1:5) #adicionando uma 3ª coluna
```

```
> y #exibindo y
```

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



- Adicionando linhas à matriz **y** criada anteriormente:

```
> y<-rbind(y,c(99,99,99)) #adicionando uma nova linha
> y #exibindo y
```

	[,1]	[,2]	[,3]
[1,]	10	5	1
[2,]	9	4	2
[3,]	8	3	3
[4,]	7	2	4
[5,]	6	1	5
[6,]	99	99	99

- Agora a matriz **y** possui uma linha a mais.

- Podemos usar os comandos para juntar matrizes:

quem      com o quê

↓

```
> z<-cbind(y,rep(88,6),y)
```

> z

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	10	5	1	88	10	5	1
[2,]	9	4	2	88	9	4	2
[3,]	8	3	3	88	8	3	3
[4,]	7	2	4	88	7	2	4
[5,]	6	1	5	88	6	1	5
[6,]	99	99	99	88	99	99	99

OBS: Para chegar na matriz z é necessário dar o comando “run” desde a criação da matriz x, pois um passo está ligado ao outro.

```
> k<-cbind(y,y)
```

```
> k
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	10	5	1	10	5	1
[2,]	9	4	2	9	4	2
[3,]	8	3	3	8	3	3
[4,]	7	2	4	7	2	4
[5,]	6	1	5	6	1	5
[6,]	99	99	99	99	99	99



## 2.2 Índices de matrizes

- Podemos extrair partes de uma matriz usando **colchetes**.
- O **primeiro número** do colchete indica o **número da linha** e o **segundo, o número da coluna** em que o elemento se encontra.

- ❖ Exemplo: extrair o elemento da segunda linha e quinta coluna da matriz k definida anteriormente:

```
> k<-cbind(y,y)
```

```
> k
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	10	5	1	10	5	1
[2,]	9	4	2	9	4	2
[3,]	8	3	3	8	3	3
[4,]	7	2	4	7	2	4
[5,]	6	1	5	6	1	5
[6,]	99	99	99	99	99	99

```
> k[2,5]
```

```
[1] 4
```

- Podemos também extrair uma linha ou coluna inteira:
- Exemplo:
- Extrair linha:  $k[3,]$
- Extrair coluna:  $k[:,4]$



```
>k[3,]
```

```
[1] 8 3 3 8 3 3
```

```
>k[,4]
```

```
[1] 10 9 8 7 6 99
```

```
> k
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	10	5	1	10	5	1
[2,]	9	4	2	9	4	2
[3,]	8	3	3	8	3	3
[4,]	7	2	4	7	2	4
[5,]	6	1	5	6	1	5
[6,]	99	99	99	99	99	99

- Pode-se extrair mais de uma linha ou coluna utilizando um vetor de índices.
- Neste caso, o objeto resultante é uma matriz:

```
> k[c(1,3,5),] #extraíndo as linhas 1,3 e 5
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	10	5	1	10	5	1
[2,]	8	3	3	8	3	3
[3,]	6	1	5	6	1	5

```
> |
```

### 3. Arrays

- São similares aos vetores e matrizes, mas podem ter qualquer número de dimensões.
- Para o R, um array é um simples vetor com o atributo de dimensões (`dim`).



## 3.1 Criando Arrays

- Pode acontecer de duas maneiras:

1. Atribuir dimensões a um vetor com o comando `dim()`

```
> x<-1:12    #cria vetor em x  
> dim(x)<-c(2,3,2) #atribuindo 3 dimensões a x  
> x          #exibe x
```

```
, , 1
```

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

```
, , 2
```

	[,1]	[,2]	[,3]
[1,]	7	9	11
[2,]	8	10	12

2. Usando o comando `array()`, em que o primeiro argumento é o vetor de dados e o segundo argumento são suas dimensões.

```
> y<-array(1:12,c(2,3,2))    #array com 3 dimensões  
> y                          #exibe y
```

```
, , 1
```

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

```
, , 2
```

	[,1]	[,2]	[,3]
[1,]	7	9	11
[2,]	8	10	12

2 linhas  
3 colunas  
2 matrizes

Podemos verificar as dimensões dos vetores  
x e y:

```
>dim(x)
[1] 2 3 2
>dim(y)
[1] 2 3 2
>
```



- Podemos atribuir nomes às dimensões do array usando o comando `dimnames( )`.
- OBS: é preciso que os nomes estejam em formato de lista (`list`). Veremos mais adiante os detalhes de “listas”.
- Vamos usar o vetor `x` visto anteriormente:

```
> dimnames(x) <- list(
+   lat=c("10°15'S", "10°20'S"),
+   long=c("22°15'W", "22°10'W", "22°05'W"),
+   alt=c("650m", "750m"))
```

```
>
> x
, , alt = 650m

      long
lat    22°15'W 22°10'W 22°05'W
10°15'S      1      3      5
10°20'S      2      4      6
```

```
, , alt = 750m

      long
lat    22°15'W 22°10'W 22°05'W
10°15'S      7      9     11
10°20'S      8     10     12
```

O símbolo **+** indica a continuação da linha anterior.

lat = linhas  
long = colunas  
alt = matrizes

- Podemos conferir os nomes dados:

```
> dimnames(x)
```

```
$lat
```

```
[1] "10°15'S" "10°20'S"
```

```
$long
```

```
[1] "22°15'W" "22°10'W" "22°05'W"
```

```
$alt
```

```
[1] "650m" "750m"
```



## 3.2 Índices em Arrays

- Funcionam de maneira semelhante aos índices de vetores e matrizes.
- Os elementos de cada dimensão do array são separados por **vírgula dentro de colchetes**.

linha  
coluna  
matriz  
 > x[1,2,1]  
 [1] 3  
 > x[1,2,]  
 650m 750m  
 3 9

```

> x
, , alt = 650m

      long
lat   22°15'W 22°10'W 22°05'W
  10°15'S      1      3      5
  10°20'S      2      4      6

, , alt = 750m

      long
lat   22°15'W 22°10'W 22°05'W
  10°15'S      7      9     11
  10°20'S      8     10     12
  
```



# **4. Exercícios**