Aula 2

Eduardo

September 28, 2016

Em nossa segunda aula vimos como manipular vetores, matrizes e data.frames

Vetores

A partir da criação de um vetor podemos ter interesse em acessar diferentes elementos dentro de um vetor e podemos realizar essa operação de diferentes formas:

```
vet1 <- (1:6)*100
vet1
```

```
## [1] 100 200 300 400 500 600
```

Suponha que temos interesse em acessar o terceiro elemento do vetor. Sempre que formos acessar determinados elementos de um objeto vamos usar os colchetes [] após o nome do elemento conforme o seguinte exemplo:

```
vet1
```

```
## [1] 100 200 300 400 500 600
```

```
vet1[3]
```

```
## [1] 300
```

Podemos também ter interesse em acessar um conjunto de elementos de um determinado vetor. Suponha que temos interesse nos três ultimos elementos do vetor *vet1*, podemos acessar esses elementos indicando em um vetor dentro dos colchetes a posição em que estão esses elementos.

```
vet1
```

```
## [1] 100 200 300 400 500 600
```

```
vet1[c(4,5,6)]
```

```
## [1] 400 500 600
```

Podemos tambem indicar utilizando elementos lógicos quais são os elementos que devem ser retornados utilizando *TRUE* e quais não devem ser retornados utilizando *FALSE*. Vejamos o exemplo de selecionar os três ultimos elementos como no exemplo anterior:

```
vet1
```

```
## [1] 100 200 300 400 500 600
```

```
vet1[c(F,F,F,T,T,T)]
```

```
## [1] 400 500 600
```

Pode-se utilizar o princípio de reciclagem na seleção de elementos utilizando os valores lógicos. Note que para o vetor de dimensão 6 foi necessário indicar *TRUE* e *FALSE* para as seis posições do vetor. Se utilizarmos o princípio de reciclagem podemos selecionar apenas os elementos que estiverem nas posições ímpares por exemplo fazendo:

```
vet1
```

```
## [1] 100 200 300 400 500 600
```

```
vet1[c(T,F)]
```

```
## [1] 100 300 500
```

E assim a primeira posição será selecionada e a segunda não. Como o vetor que indica as posições tem tamanho dois enquanto o vetor *vet1* tem tamanho 6, o vetor que indica as posições acaba, e ele é reciclado na definição se os próximos elementos devem ser exibidos ou não. Assim o terceiro elemento será exibido e o quarto elemento não será exibido pois o vetor *c(T,F)* será reciclado, e o mesmo princípio será utilizado na definição se o quinto e sexto elemento devem ser exibidos.

Na escolha de elementos de um vetor pode-se utilizar sinal negativo para indicar quais elementos não devem ser incluidos, conforme os exemplos:

```
vet1
```

```
## [1] 100 200 300 400 500 600
```

```
vet1[-3]
```

```
## [1] 100 200 400 500 600
```

```
vet1[-c(1,2,3)]
```

```
## [1] 400 500 600
```

E assim escolhemos, no primeiro caso, todos os elementos de *vet1* menos o terceiro, e no segundo caso, todos menos o primeiro, segundo e terceiro.

Ordenação

Vimos que podemos utilizar a seleção dos elementos de um vetor para ordenar um vetor segundo alguma ordenação de interesse. Considere o seguinte vetor:

```
vet2 <- c(9,3,5,2)
vet2
```

```
## [1] 9 3 5 2
```

Se tivermos interesse em exibir esse vetor de forma ordenada crescente devemos exibir primeiro o quarto elemento, depois o segundo elemento, depois o terceiro elemento e por último o primeiro elemento. Podemos fazer isso da seguinte forma:

```
vet2[c(4,2,3,1)]
```

```
## [1] 2 3 5 9
```

E o vetor está ordenado de forma crescente. Em casos de vetores maiores não seria possível definir a ordem de apresentação dos valores de forma manual e a função *order* define a posição dos vetores ordenados da mesma forma que foi estabelecida visualmente.

```
order(vet2)
```

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

Note que o resultado da função order foi o mesmo que estabelecido manualmente na ordenação realizada anteriormente em que vimos que deveriamos apresentar primeiro o quarto elemento, depois o segundo elemento, depois o terceiro elemento e por último o primeiro elemento *4,2,3,1* Dessa forma, podemos utilizar o resultado da função *order* sem precisar estabelecer manualmente a ordenação dos elementos:

```
vet2[order(vet2)]
```

```
## [1] 2 3 5 9
```

Apesar de ser funamental entender o funcionamento e a utilização da função *order*, no caso de ordenação de vetores simples é mais fácil utilizar a função *sort* que já retorna o vetor ordenado.

```
sort(vet2)
```

```
## [1] 2 3 5 9
```

Caso tenhamos interesse numa ordenação decrescente ambas as funções *order* e *sort* possuem uma opção para essa forma.

```
sort(vet2,decreasing = T)
```

```
## [1] 9 5 3 2
```

```
order(vet2, decreasing = T)
```

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

```
vet2[order(vet2,decreasing = T)]
```

```
## [1] 9 5 3 2
```

Matrizes

Uma extensão do coneito do vetor é a utilização de matrizes. Para criar uma matriz vamos utilizar a função *matrix*, passando como argumentos os valores que irão preencher a matriz e o número de linhas ou colunas que formarão a matriz.

No primeiro exemplo vamos criar uma matriz quadrada de 3 linhas e 3 colunas com os números de 1 até 9:

```
mat1 <- matrix(1:9,nrow=3)
mat1
```

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

e podemos obter o mesmo resultado determinando o número de colunas.

```
mat2 <- matrix(1:9,ncol=3)
mat2
```

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

Note que por padrão o preenchimento dos elementos na matriz é feito por colunas, de forma que primeiro será preenchida a coluna um, posteriormente a coluna dois e as demais colunas. Para que os dados sejam preenchidos por linhas utilizaremos o argumento *byrow=TRUE* na definição da matriz:

```
mat3 <- matrix(1:9,ncol=3,byrow = TRUE)
mat3
```

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

Assim como visto para o caso dos vetores, para acessar determinados elementos de uma matriz deve-se utilizar os colchetes indicando os endereços de linha e coluna de interesse *[linha,coluna]*. Para acessar o elemento que se encontra na linha 2 e coluna 3 faremos:

```
mat1
```

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

```
mat1[2,3]
```

```
## [1] 8
```

Pode-se também acessar um conjunto de linhas ou conjunto de colunas. Para acessar toda a linha dois por exemplo utiliza-se:

```
mat1[2,]
```

```
## [1] 2 5 8
```

Para acessar as colunas dois e três:

```
mat1[,c(2,3)]
```

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

Pode-se utilizar esse recurso para ordenar por exemplo as linhas da matriz:

```
mat1
```

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

```
mat1[c(3,2,1),]
```

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

Podemos fazer operações de forma similar às que foram realizadas com vetores, lembrando que o princípio de reciclagem será utilizado sempre que as dimensões dos operandos for diferente. No exemplo seguinte o valor 1000 será reciclado para a soma com cada um dos elementos da matriz

```
mat1
```

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

```
mat1+1000
```

```
##      [,1] [,2] [,3]
## [1,] 1001 1004 1007
## [2,] 1002 1005 1008
## [3,] 1003 1006 1009
```

No exemplo seguinte o vetor será reciclado, de forma que as operações são realizadas inicialmente na primeira coluna da matriz e posteriormente na segunda coluna e de forma análoga para as demais colunas.

```r
vet <- c(100,200,300)
vet
```

```
## [1] 100 200 300
```

```r
mat1 + vet
```

```
##      [,1] [,2] [,3]
## [1,]  101  104  107
## [2,]  202  205  208
## [3,]  303  306  309
```

As operações entre duas matrizes são realizadas elemento a elemento.

```r
mat4 <- matrix((1:9)*1000,nrow=3)
mat4
```

```
##      [,1] [,2] [,3]
## [1,] 1000 4000 7000
## [2,] 2000 5000 8000
## [3,] 3000 6000 9000
```

```r
mat1 + mat4
```

```
##      [,1] [,2] [,3]
## [1,] 1001 4004 7007
## [2,] 2002 5005 8008
## [3,] 3003 6006 9009
```

Para realizar multiplicação entre matrizes conforme as técnicas de álgebra linear deve-se utilizar o operador _%*%_ conforme o seguinte exemplo

```r
mat1 %*% mat4
```

```
##       [,1]  [,2]   [,3]
## [1,] 30000 66000 102000
## [2,] 36000 81000 126000
## [3,] 42000 96000 150000
```

Assim como os vetores, as matrizes são objetos de tipo homogêneo, de forma que todos os elementos devem ser do mesmo tipo(numérico, caracter, lógico). Caso elementos de tipos diferentes sejam inseridos em uma matriz, alguns elementos serão transformados para que todos tenham o mesmo tipo.

```r
matletras <- matrix(letters[1:9],nrow=3)
matletras
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "d"  "g"
## [2,] "b"  "e"  "h"
## [3,] "c"  "f"  "i"
```

```r
matmisturada <- matrix(c(TRUE,letters[1:7],1000),nrow=3)
matmisturada
```

```
##      [,1]   [,2] [,3]
## [1,] "TRUE" "c"  "f"
## [2,] "a"    "d"  "g"
## [3,] "b"    "e"  "1000"
```

É comum a necessidade de reunir em uma tabela diferentes tipos de variávies e para essa finalidade pode-se utilizar objetos do tipo *data.frame*

Data.frame

No próximo exemplo será criada uma tabela com informações de diferentes tipos:

```r
notas <- c(10,7,3,5)
nomes <- c("Ana", "bia", "edu", "joao")
cotista <- c(T,F,T,F)
sexo <- c("F","F","M","M")
alunos <- data.frame(nomes,notas,cotista,sexo)
alunos
```

```
##   nomes notas cotista sexo
## 1   Ana    10    TRUE    F
## 2   bia     7   FALSE    F
## 3   edu     3    TRUE    M
## 4  joao     5   FALSE    M
```

De forma análoga à matriz, os elementos podem ser acessados a partir dos endereços de *[linha,coluna]*. No caso das tabelas é muito comum que diferentes variáveis sejam nomeadas e pode-se também utilizar os nomes das variáveis para acessa-las. No próximo exemplo está ilustrada a possibilidade de acessar a variável *notas* de diferentes formas:

```r
alunos[,2]
```

```
## [1] 10  7  3  5
```

```r
alunos[,"notas"]
```

```
## [1] 10  7  3  5
```

```r
alunos$notas
```

```
## [1] 10  7  3  5
```

Partindo do acesso de dados no *data.frame* os próximos exemplos ilustram como ordenar uma tabela e selecionar determinadas observações segundo alguma condição de interesse.

Inicialmente ordenar a tabela pelas notas dos alunos de forma decrescente:

```r
alunos[order(alunos$notas,decreasing = T),]
```

```
##   nomes notas cotista sexo
## 1   Ana    10    TRUE    F
## 2   bia     7   FALSE    F
## 4  joao     5   FALSE    M
## 3   edu     3    TRUE    M
```

Selecionar apenas os alunos com notas maiores ou iguais a 6:

```r
alunos[alunos$notas >= 6, ]
```

```
##   nomes notas cotista sexo
## 1   Ana    10    TRUE    F
## 2   bia     7   FALSE    F
```

Selecionar apenas os alunos cotistas:

```r
alunos[alunos$cotista,]
```

```
##   nomes notas cotista sexo
## 1   Ana    10    TRUE    F
## 3   edu     3    TRUE    M
```