Roteiro do Estudo Dirigido I de Tópicos Avançados I (EC020)

Este Estudo Dirigido também está disponível no Portal Acadêmico.

Faremos mais atividades com vetores, matrizes, listas e dataframes, para consolidarmos os conceitos destes tipos avançados de dados.

Caros(as) Alunos(as), SIGAM O ROTEIRO, sequencialmente!

Os itens de exercícios (de 1 a 7) desse Estudo deverão ser desenvolvidos em dupla ou individual, em um Script.R (nomealuno1_nomealuno2.R) **e enviado por e-mail para rosannas@inatel.br, até o dia 30/março/20, segunda-feira, 10h.**

---

1) Pesquise e coloque no Script, qual a **função** que:
   a) permite listar todos os arquivos que estão no diretório de trabalho
   b) permite listar os tipos de dados criados (as variáveis de ambiente criadas)
   c) permite listar os pacotes que já estão carregados na memória para utilização

---

Fatores são variáveis (tipos de dados estatísticos) que são muito úteis em sumarização de estatísticas, plots e regressões.

Eles representam uma maneira muito eficiente para armazenar valores de caracteres, porque cada caracter único é armazenado apenas uma vez e os dados são armazenados como um vetor de inteiros.

É uma maneira de otimizar o armazenamento e o processamento de textos, palavras e caracteres.

Exemplo:
"Vamos criar um vetor de palavras"

```
vec1 <- c("Masculino","Feminino","Feminino","Masculino","Masculino")
vec1
[1] "Masculino" "Feminino"  "Feminino"  "Masculino"
[5] "Masculino"
```

*"Vamos converter esse vetor em fatores. Use o função **factor**()"*

```
fac_vec1 <- factor(vec1)
fac_vec1
[1] Masculino Feminino  Feminino  Masculino Masculino
Levels: Feminino Masculino
```

"Ele dividiu o vetor em 2 níveis: Masculino e Feminino, e em ordem alfabética dos níveis"
"Em fac_vec1 tem-se os elementos do vetor, mas internamente ele grava apenas os 2 níveis e um vetor de inteiros apontando para cada um dos elementos. Isso economiza espaço de armazenamento, espaço de memória e consequentemente agiliza e aumenta a performance"

"Vamos verificar as classes de vec1 e de fac_vec1"

```
class(vec1)
[1] "character"
```

class(fac_vec1)
```
"factor"
```

"Vamos imprimir os níveis de fac_vec1"
levels(fac_vec1)
```
[1] "Feminino"  "Masculino"
```

"Vamos ter uma visão geral sobre o conteúdo da variável: 2 elementos com Feminino e 3 com Masculino. A linguagem R divide o vetor em níveis (em ordem alfabética crescente) e cria um vetor de inteiros apontando para a quantidade de cada um dos níveis"
summary(fac_vec1)
```
Feminino Masculino
        2         3
```

"Podemos mudar os nomes dos níveis, se for necessário. Veja o resultado!"
levels(fac_vec1) <- **c**("Femea", "Macho")

---

2) Crie um vetor de strings com nome de 4 animais, repita o nome de um dos animais, portanto um vetor com 5 palavras:
   a) imprima esse vetor
   b) converta este vetor para fator e imprima na tela
   c) apresente os níveis deste fator e uma visão geral dos conteúdos dele

---

Fator ordenado: quando quiser definir uma ordem decrescente para o fator, utilize outros parâmetros na função **factor**.

form <- **c**("Fundamental", "Ensino Médio", "Maternal", "Fundamental", "Fundamental")
form
```
[1] "Fundamental"  "Ensino Médio" "Maternal"
[4] "Fundamental"  "Fundamental"
```

fac_form <- **factor**(form, order = TRUE, levels = **c**("Ensino Médio", "Fundamental", "Maternal")) # coloca os níveis desejados em ordem decrescente alfabética
fac_form
```
[1] Fundamental  Ensino Médio Maternal
[4] Fundamental  Fundamental
3 Levels: Ensino Médio < Fundamental < Maternal
```

summary(form) # vetor original, classe e modo: character e comprimento de 5
```
Length     Class      Mode
        5 character character
```

summary(fac_form) # fator, com a quantidade em cada nível ordenado
```
Ensino Médio  Fundamental     Maternal
           1            3            1
```

---

3) Crie um vetor de strings com os nomes "Bacharelado" em 3 posições, "Especialização" em uma posição e "Mestrado" em 2 posições, portanto um vetor com 6 palavras:
   a) imprima esse vetor
   b) converta este vetor para um fator ordenado e o imprima na tela

c) apresente uma visão geral dos conteúdos dele
d) altere o nome dos níveis desse fator de Bacharelado para Graduação, Especialização para Lato Sensu e Mestrado para Stricto Sensu

4) Crie um dataframe a partir de 3 vetores de tamanho 4 cada:
um com 4 strings (contendo o código de algumas disciplinas do 9º. Período),
um com 4 valores lógicos (TRUE, para a disciplina fácil e FALSE para a que não é)
um com 4 valores floats (notas que você espera tirar na primeira prova dessas disciplinas)
Faça a nomeação das colunas deste dataframe com "disciplina" "dificuldade" "nota". Mostre este dataframe criado.
Use a função que apresenta os tipos dos dados deste dataframe. Fale sobre o tipo do dado da coluna dos códigos das disciplinas.

Estruturas de Controle em R: elas permitem que se faça validações, que se repita um bloco de código um número de vezes e que se realize mudanças no comportamento do Script de acordo com determinadas regras.

Condicional If-Else: sintaxes

if (condição){
conjunto de tarefas} # se condição verdadeira, executa este conjunto de tarefas
else
{outro conjunto de tarefas} # se condição falsa, executa este conj. de tarefas

ifelse (condição, tarefa1, tarefa2) # se condição verdadeira, executa-se
tarefa1, se não, executa-se tarefa2

Repetição Loop For: sintaxe

for (i in 1:N) { # para um determinado contador e dentro de um
conjunto de tarefas} # range de números (de 1 a N), se realiza o conjunto
de tarefas

Repetição Loop While: sintaxe

while (condição satisfeita) { # enquanto a condição for verdadeira, se
conjunto de tarefas} # realiza o conjunto de tarefas, quando a
condição deixa de ser verdadeira, o loop
é encerrado e a operação é concluída

Repetição Repetições: sintaxes

rep (x, y) # **rep** (repita x, y vezes)

repeat { }

Nos loops:
next pula para o próximo valor dentro do range estabelecido para o loop
break interrompe o loop

---

5) Crie um loop (use qualquer solução/estrutura de loop) que verifique se há números maiores que 10 e que são pares no vetor abaixo. Se houver, imprima esse(s) número(s). Se não houver, envie uma mensagem de que não há números com este adjetivo.

   Considere os vetores abaixo para fazer o teste do código acima:
   # Crie o Vetor vec1
   vec1 <- **c**(12, 3, 14, 19, 34, 2, 25)

   Para o vetor vec1 acima deverá sair:
   ```
   [1] 12
   [1] 14
   [1] 34
   ```

   # Crie o Vetor vec2
   vec2 <- **c**(21, 3, 15, 19, 37, 2, 25)

   Para o vetor vec2 acima deverá sair:
   ```
   [1] "Não tem nesse vetor números pares maiores que 10"
   ```

---

6) Considere a lista abaixo. Crie um loop que imprima cada elemento desta lista. Atenção à diseposição da dimensão desta lista.

   lst1 <- **list**(1, 1, 2, 3, 5, 8, 13, 21)

Matrizes

Soma por linha: função **rowSums**(matriz)

Soma por coluna: função **colSums**(matriz)

Ordenar os elementos de uma matriz: função **order**(matriz). Esta função retorna os índices dos elementos ordenados. Para listar os elementos ordenados, você precisa colocar matriz[**order**(matriz)]

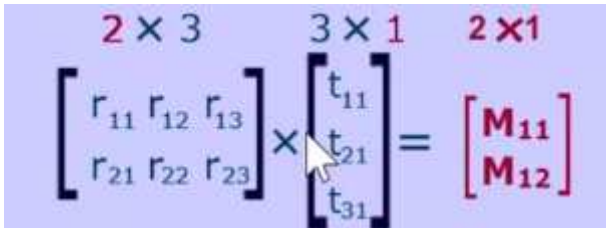
Multiplicação de Matrizes:

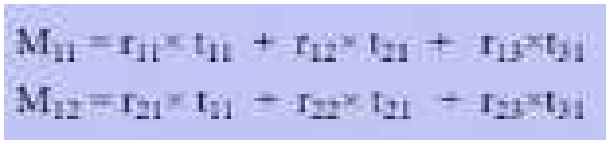
1) Multiplicação de um escalar por uma matriz:

$$3\begin{bmatrix}5 & 2 & 11\\ 9 & 4 & 14\end{bmatrix} = \begin{bmatrix}3\cdot 5 & 3\cdot 2 & 3\cdot 11\\ 3\cdot 9 & 3\cdot 4 & 3\cdot 14\end{bmatrix} = \begin{bmatrix}15 & 6 & 33\\ 27 & 12 & 42\end{bmatrix}$$

2) Multiplicação de matrizes:

A matriz resultante tem o número de linhas da primeira matriz e o número de colunas da segunda matriz. Se invertermos as matrizes, não será possível multiplicar.





Considere as duas matrizes abaixo e veja uma multiplicação **element-wise** e uma multiplicação **tradicional** entre elas.

```
mat1 <- matrix(c(1:50), nrow = 5, ncol = 5, byrow = T)
mat1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
[5,]   21   22   23   24   25

mat2 <- t(mat1)  # transposta de mat1
mat2
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25

# Multiplicação element-wise: multiplicação em nível de elemento, bastante utilizada
# em aprendizado de máquina

mat3 <- mat1 * mat2          # operador *
mat3

     [,1] [,2] [,3] [,4] [,5]
[1,]    1   12   33   64  105
[2,]   12   49   96  153  220
[3,]   33   96  169  252  345
[4,]   64  153  252  361  480
[5,]  105  220  345  480  625

# Multiplicação tradicional de matrizes
# multiplica primeira linha da mat1 com a primeira coluna de mat2
```

```
# Item [1,1] ==> (1 x 1) + (2 x 2) + (3 x 3) x (4 x 4) x (5 x 5) = 55
# Item [2,1] ==> (6 x 1) + (7 x 2) + (8 x 3) x (9 x 4) x (10 x 5) = 130
# Item [3,1] ==> (11 x 1) + (12) x 2) + (13 x 3) x (14 x 4) x (15 x 5) = 205

mat4 <- mat1 %*% mat2      # operador %*%
mat4
     [,1] [,2] [,3] [,4] [,5]
[1,]   55  130  205  280  355
[2,]  130  330  530  730  930
[3,]  205  530  855 1180 1505
[4,]  280  730 1180 1630 2080
[5,]  355  930 1505 2080 2655
```

Para imprimir elementos da matriz que estejam em um intervalo, basta apenas um for: por exemplo, imprima os números da matriz mat4 que estejam no intervalo [520,1520]

```
for (i in mat4){
  if (i >= 520 & i <= 1520) {
    print (i)
  }
}
```

---

7) Crie um vetor com os 5 primeiros números naturais (0 a 4). E usando a mat2 criada acima, faça a multiplicação tradicional do vetor por esta matriz mat2.
   Apresente a soma das linhas da matriz mat1 e a soma das colunas da matriz mat2.
   Apresente a ordenação da matriz mat4.
   Imprima os elementos da matriz mat1 que são superiores a 15.
