ME115 - Linguagem R

Atividade Prática 04 - Gabarito

1º semestre de 2023

Introdução

Nessa atividade, exploraremos:

1. Objetos do tipo `factor`;
2. Funções da família *apply*;
3. A função `replicate()`.

Exercícios inspirados em http://r-tutorials.com/.

Atividade

1. Crie uma matriz como indicado abaixo e a seguir use a função da família *apply* que seja mais apropriada.

```
my.matrix <- matrix(data = c(6, 34, 923, 5, 0, 112:116, 5, 9, 34, 76, 2, 545:549),
                    nrow = 5)
my.matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    6  112    5  545
## [2,]   34  113    9  546
## [3,]  923  114   34  547
## [4,]    5  115   76  548
## [5,]    0  116    2  549
```

   a. Obtenha a média das linhas de `my.matrix` tal como o resultado abaixo.

```
## [1] 167.00 175.50 404.50 186.00 166.75
```

   Solução:

```
apply(my.matrix, MARGIN = 1, FUN = mean)
```

```
## [1] 167.00 175.50 404.50 186.00 166.75
```

   b. Obtenha a média das colunas de `my.matrix` tal como o resultado abaixo.

```
## [1] 193.6 114.0  25.2 547.0
```

   Solução:

```
apply(my.matrix, MARGIN = 2, FUN = mean)
```

```
## [1] 193.6 114.0  25.2 547.0
```

   c. Ordene as colunas de `my.matrix` em ordem crescente tal como o resultado abaixo.

```
##      [,1] [,2] [,3] [,4]
## [1,]    0  112    2  545
## [2,]    5  113    5  546
## [3,]    6  114    9  547
```

```
## [4,]   34  115   34  548
## [5,]  923  116   76  549
```

Solução:

```
apply(my.matrix, MARGIN = 2, FUN = sort)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0  112    2  545
## [2,]    5  113    5  546
## [3,]    6  114    9  547
## [4,]   34  115   34  548
## [5,]  923  116   76  549
```

2. Considere o conjunto de dados `mtcars` disponível no R.

   a. Usando as funções `lapply()`, `sapply()` e `mapply()` da família *apply*, obtenha os valores mínimos de cada coluna do banco de dados `mtcars` e guarde os valores nas variáveis `l`, `s`, `m`, respectivamente.

   Solução:

```
l <- lapply(mtcars, FUN = min)
l
```

```
## $mpg
## [1] 10.4
##
## $cyl
## [1] 4
##
## $disp
## [1] 71.1
##
## $hp
## [1] 52
##
## $drat
## [1] 2.76
##
## $wt
## [1] 1.513
##
## $qsec
## [1] 14.5
##
## $vs
## [1] 0
##
## $am
## [1] 0
##
## $gear
## [1] 3
##
## $carb
## [1] 1
```

```r
s <- sapply(mtcars, FUN = min)
s
```

```
##    mpg    cyl   disp     hp   drat     wt   qsec     vs     am   gear   carb
## 10.400  4.000 71.100 52.000  2.760  1.513 14.500  0.000  0.000  3.000  1.000
```

```r
m <- mapply(mtcars, FUN = min)
m
```

```
##    mpg    cyl   disp     hp   drat     wt   qsec     vs     am   gear   carb
## 10.400  4.000 71.100 52.000  2.760  1.513 14.500  0.000  0.000  3.000  1.000
```

b. Armazene as três saídas `l`, `s`, `m` em uma lista chamada `list.objects`.

Solução:

```r
listobjects = list(l, s, m)
```

c. Use uma função do tipo *apply* adequada para obter a classe de cada um dos três elementos da lista em `listobjects`.

Solução:

```r
sapply(FUN = class, X = listobjects)
```

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

3. A seguir, use `mapply()`.

a. Obtenha uma lista de 8 elementos que deve alternar entre "ME" and "115". Os comprimentos desses 8 elementos alternados diminuem passo a passo de 8 para 1.

Solução:

```r
mapply(rep, c("ME", "115"), 8:1)
```

```
## $ME
## [1] "ME" "ME" "ME" "ME" "ME" "ME" "ME" "ME"
##
## $`115`
## [1] "115" "115" "115" "115" "115" "115" "115"
##
## $<NA>
## [1] "ME" "ME" "ME" "ME" "ME" "ME"
##
## $<NA>
## [1] "115" "115" "115" "115" "115"
##
## $<NA>
## [1] "ME" "ME" "ME" "ME"
##
## $<NA>
## [1] "115" "115" "115"
##
## $<NA>
## [1] "ME" "ME"
##
## $<NA>
## [1] "115"
```

b. Ajuste a função do item (a) para obter os números de elemento adequados (1:8) para os 8 elementos da lista. Dica: argumento `USE.NAMES`.

Solução:

```
mapply(rep, c("ME", "115"), 8:1, USE.NAMES = F)
```

```
## [[1]]
## [1] "ME" "ME" "ME" "ME" "ME" "ME" "ME" "ME"
##
## [[2]]
## [1] "115" "115" "115" "115" "115" "115" "115"
##
## [[3]]
## [1] "ME" "ME" "ME" "ME" "ME" "ME"
##
## [[4]]
## [1] "115" "115" "115" "115" "115"
##
## [[5]]
## [1] "ME" "ME" "ME" "ME"
##
## [[6]]
## [1] "115" "115" "115"
##
## [[7]]
## [1] "ME" "ME"
##
## [[8]]
## [1] "115"
```

4. Considere o conjunto de dados `Titanic` disponível no R.

Solução:

```
str(Titanic)
```

```
##  'table' num [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
##  - attr(*, "dimnames")=List of 4
##   ..$ Class   : chr [1:4] "1st" "2nd" "3rd" "Crew"
##   ..$ Sex     : chr [1:2] "Male" "Female"
##   ..$ Age     : chr [1:2] "Child" "Adult"
##   ..$ Survived: chr [1:2] "No" "Yes"
```

a. Use uma função da família *apply* apropriada para obter a soma de homens e mulheres a bordo.

Solução:

```
apply(Titanic, 2, sum)
```

```
##   Male Female
##   1731    470
```

```
apply(Titanic, "Sex", sum, na.rm = TRUE)
```

```
##   Male Female
##   1731    470
```

b. Faça uma tabela com a soma dos sobreviventes, segundo sexo.

Solução:

```
apply(Titanic, c(2, 4), sum)
```

```
##         Survived
## Sex        No Yes
##   Male   1364 367
##   Female  126 344
```

```
apply(Titanic, c("Sex", "Survived"), sum)
```

```
##         Survived
## Sex        No Yes
##   Male   1364 367
##   Female  126 344
```

c. Faça uma tabela com a soma de passageiros, segundo sexo e idade.

Solução:

```
apply(Titanic, c(2, 3), sum)
```

```
##         Age
## Sex      Child Adult
##   Male      64  1667
##   Female    45   425
```

5. Considere a função `lapply()`.

a. Crie `listobj` uma lista de quatro matrizes, tal que:

```
first <- matrix(38:67, 3)
second <- matrix(56:91, 3)
third <- matrix(82:144, 3)
fourth <- matrix(46:93, 6)
```

Solução:

```
listobj <- list(first, second, third, fourth)
listobj
```

```
## [[1]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]   38   41   44   47   50   53   56   59   62    65
## [2,]   39   42   45   48   51   54   57   60   63    66
## [3,]   40   43   46   49   52   55   58   61   64    67
##
## [[2]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]   56   59   62   65   68   71   74   77   80    83    86    89
## [2,]   57   60   63   66   69   72   75   78   81    84    87    90
## [3,]   58   61   64   67   70   73   76   79   82    85    88    91
##
## [[3]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]   82   85   88   91   94   97  100  103  106   109   112   115   118   121
## [2,]   83   86   89   92   95   98  101  104  107   110   113   116   119   122
## [3,]   84   87   90   93   96   99  102  105  108   111   114   117   120   123
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## [1,]   124   127   130   133   136   139   142
```

```
## [2,]   125   128   131   134   137   140   143
## [3,]   126   129   132   135   138   141   144
##
## [[4]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]   46   52   58   64   70   76   82   88
## [2,]   47   53   59   65   71   77   83   89
## [3,]   48   54   60   66   72   78   84   90
## [4,]   49   55   61   67   73   79   85   91
## [5,]   50   56   62   68   74   80   86   92
## [6,]   51   57   63   69   75   81   87   93
```

b. Extraia a segunda coluna de cada matriz da lista de matrizes.

Solução:

```
lapply(listobj, "[", , 2)
```

```
## [[1]]
## [1] 41 42 43
##
## [[2]]
## [1] 59 60 61
##
## [[3]]
## [1] 85 86 87
##
## [[4]]
## [1] 52 53 54 55 56 57
```

c. Extraia a terceira linha de cada matriz da lista de matrizes.

Solução:

```
lapply(listobj, "[", 3 , )
```

```
## [[1]]
##  [1] 40 43 46 49 52 55 58 61 64 67
##
## [[2]]
##  [1] 58 61 64 67 70 73 76 79 82 85 88 91
##
## [[3]]
##  [1]  84  87  90  93  96  99 102 105 108 111 114 117 120 123 126 129 132 135 138
## [20] 141 144
##
## [[4]]
## [1] 48 54 60 66 72 78 84 90
```

6. Usando a família *apply* para trabalhar com classes de *data frames* (usaremos o conjunto de dados `iris` visto em aula).

a. Descubra qual coluna em `iris` não é numérica.

Solução:

```
which(!sapply(iris, is.numeric))
```

```
## Species
##       5
```

```
str(iris)
```

```
## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Observação: Cuidado quando forem usar `class` para verificar se algum objeto é númerico. Temos que `integer` é um valor númerico, mas note que

```
a <- as.integer(2)
class(a)
```

```
## [1] "integer"
```

```
class(a) == "numeric"
```

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

```
is.numeric(a)
```

```
## [1] TRUE
```

Ou seja, caso você queira selecionar todas as colunas com valores númericos, por exemplo, esse código (usando `class`) não selecionaria aquelas da classe integer, o que estaria errado. O melhor jeito nesse caso seria usar a função `is.numeric`, pois ela já leva em consideração todas as classe de valores numéricos.

b. Identifique os níveis da coluna não numérica. Dica: use função `levels()`.

Solução:

```
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

c. Considere a função `unique()` e compare os resultados.

Solução:

```
unique(iris$Species)
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

7. Considere o conjunto de dados `PlantGrowth` disponível no R. Use a função `tapply()` para calcular o desvio padrão dos valores de peso de cada grupo.

Solução:

```
tapply(PlantGrowth$weight, PlantGrowth$group, sd)
```

```
##      ctrl      trt1      trt2
## 0.5830914 0.7936757 0.4425733
```

Desafio

8. O código a seguir simula o desempenho de um teste t para dados não normais. Use `sapply()` e uma função construída por você para extrair o p-valor de cada uma das 100 tentativas armazenadas em `trials`.

```
trials <- replicate(100, t.test(rpois(10, 10), rpois(7, 10)), simplify = FALSE)
```

Solução:

```
class(trials)
```

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

```
names(trials[[1]])
```

```
##  [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
##  [6] "null.value"  "stderr"      "alternative" "method"      "data.name"
```

```
sapply(trials, function(x) x[["p.value"]])
```

```
##   [1] 0.138511505 0.546684426 0.020396891 0.338210517 0.130497331 0.648423802
##   [7] 0.437663129 0.943962522 0.757232681 0.232685290 0.217312744 0.799097778
##  [13] 0.106469293 0.939185269 0.201228233 0.743284595 0.887871299 0.453760579
##  [19] 0.785517674 0.403532292 0.623897132 0.574867921 0.867218567 0.836370536
##  [25] 0.596315923 0.868058892 0.836128058 0.020216864 0.068437529 0.708123450
##  [31] 0.791893481 0.574608686 0.651138684 0.001566456 0.591274891 0.603151162
##  [37] 0.303991054 0.267763655 0.125799558 0.564112685 0.980676090 0.371880246
##  [43] 0.443715765 0.389114257 0.928954994 0.823520182 0.246894935 0.548832518
##  [49] 0.514904195 0.926002673 0.316301722 0.615992999 0.403541600 0.005017778
##  [55] 0.892348842 0.004881811 0.604639916 0.051470328 0.558496009 0.408531017
##  [61] 0.423902974 0.233490180 0.024951436 0.158147458 0.859996888 0.851661569
##  [67] 0.219053773 0.151643583 0.247441995 0.649688687 0.611019410 0.171538007
##  [73] 0.627409495 0.519317533 0.025688329 0.617921146 0.177082890 0.989821968
##  [79] 0.590424626 0.525308634 0.959416936 0.492807594 0.266071620 0.350358705
##  [85] 0.272314021 0.468991401 0.151239325 0.686423684 0.005608815 0.250019448
##  [91] 0.631660778 0.041614875 0.505436566 0.963864946 0.339680585 0.172488475
##  [97] 0.834199370 0.821720903 0.023388723 0.234528800
```

9. Considere o exercício anterior. Produza o mesmo resultado livrando-se da função que você construiu e use "[[" diretamente.

```
sapply(trials, "[[", "p.value")
```

```
##   [1] 0.138511505 0.546684426 0.020396891 0.338210517 0.130497331 0.648423802
##   [7] 0.437663129 0.943962522 0.757232681 0.232685290 0.217312744 0.799097778
##  [13] 0.106469293 0.939185269 0.201228233 0.743284595 0.887871299 0.453760579
##  [19] 0.785517674 0.403532292 0.623897132 0.574867921 0.867218567 0.836370536
##  [25] 0.596315923 0.868058892 0.836128058 0.020216864 0.068437529 0.708123450
##  [31] 0.791893481 0.574608686 0.651138684 0.001566456 0.591274891 0.603151162
##  [37] 0.303991054 0.267763655 0.125799558 0.564112685 0.980676090 0.371880246
##  [43] 0.443715765 0.389114257 0.928954994 0.823520182 0.246894935 0.548832518
##  [49] 0.514904195 0.926002673 0.316301722 0.615992999 0.403541600 0.005017778
##  [55] 0.892348842 0.004881811 0.604639916 0.051470328 0.558496009 0.408531017
##  [61] 0.423902974 0.233490180 0.024951436 0.158147458 0.859996888 0.851661569
##  [67] 0.219053773 0.151643583 0.247441995 0.649688687 0.611019410 0.171538007
##  [73] 0.627409495 0.519317533 0.025688329 0.617921146 0.177082890 0.989821968
##  [79] 0.590424626 0.525308634 0.959416936 0.492807594 0.266071620 0.350358705
##  [85] 0.272314021 0.468991401 0.151239325 0.686423684 0.005608815 0.250019448
##  [91] 0.631660778 0.041614875 0.505436566 0.963864946 0.339680585 0.172488475
##  [97] 0.834199370 0.821720903 0.023388723 0.234528800
```

Agradecimento

O material foi produzido pela Profa. Tatiana Benaglia para o curso de ME115.