Família apply

(apply, lapply, sapply, mapply, tapply)

Carlos Roberto de Nazaré Carvalho Junior
Fernando José Pessoa Andrade
Ivanildo Batista da Silva Júnior
Vanessa Karoline Inacio Gomes
Vaniele da Silva Barros

07 de Junho de 2021

Universidade Federal Rural de Pernambuco
Programa de Pós-Graduação em Biometria e Estatística Aplicada (PPGBEA)
Uso de Software na Análise de Dados Biométricos
Professor : Antonio Samuel Alves da Silva

Sumário

O que é a família apply ?

Família apply

A família de funções *apply* é considerada como funções de *loop* que funcionam em tarefas repetidas e reduz a redundância que aparece devido ao *loop*. Essas funções estão embutidas no *R* e não há a necessidade de instalação separadamente. Elas permitem a manipulação *frames* de dados, *arrays*, matrizes, vetores; sendo alternativas aos *loops*, porém são mais eficientes pela rapidez no nível de execução. As principais funções são

- apply()
- lapply()
- sapply()
- mapply()
- tapply()

Família apply

Vantagens de usar a funções da família *apply*:

- Otimização do código;
- São mais rápidas que os *loops* convencionais;
- O processo torna-se menos complexo e mais compreensível;
- Permite a criação de objetos;
- Não há a necessidade de configurar o armazenamento e o gerenciamento das saídas, esse processo é feito de forma automática;
- Funções nativas, sem necessidade de instalação.

Função apply

Função apply

A função *apply* permite aplicar uma função em linhas ou colunas (*MARGINS*) de uma matriz ou em *dataframes*. Abaixo podemos ver a sintaxe da função:

```
apply(X, MARGIN, FUN, ...)
```

- *X* é um *array*, incluindo matrizes;
- *MARGIN* é um vetor que define em qual parte do *array* será aplicada a função;
- *FUN* é a função que será aplicada;
- As **reticências** são argumentos da função, que podem ser inseridos opcionalmente.

A função retorna um vetor ou matriz ou lista de valores obtidos pela aplicação de uma função às margens de um *array* ou matriz.

Exemplo I - função apply

Criando uma matriz

```
vetor1 <- 1:25
matriz = matrix(vetor1, 5,5)
matriz
```

```
##      [,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
```

Somando as linhas da matriz

```
apply(matriz, 1, sum)
```

```
## [1] 55 60 65 70 75
```

Somando as colunas da matriz:

```
apply(matriz, 2, sum)
```

```
## [1]  15  40  65  90 115
```

Exemplo II - função apply

Média das colunas do *dataset mtcars*.

```
apply(mtcars,2,mean)
```

```
##        mpg        cyl       disp         hp       drat         wt       qsec
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250  17.848750
##         vs         am       gear       carb
##   0.437500   0.406250   3.687500   2.812500
```

Aplicando a função de desvio padrão para as colunas *Sepal.Length* e *Petal.Length* do *dataset iris*.

```
apply(iris[c('Sepal.Length','Petal.Length')],2,sd)
```

```
## Sepal.Length Petal.Length
##    0.8280661    1.7652982
```

Exemplo III - função apply

Aplicando a função de soma para um *array*.

```
rnames <- c("l1","l2","l3")
cnames <- c("c1","c2","c3")
mnames <- c("m1","m2")

ex_array <- array(1:18,dim=c(3,3,2),
                  dimnames=list(rnames,
                                cnames,
                                mnames))

apply(ex_array,1,sum)
```

```
## l1 l2 l3
## 51 57 63
```

Exemplo IV

Criando duas variáveis com uma distribuição uniforme e combinando-as em outra variável como um *dataframe*.

```
x1 <- runif(10000)
x2 <- runif(10000)
d <- as.data.frame(cbind(x1, x2))
```

Loop VS função *apply*:

```
system.time(
  for (loop in c(1:length(d[, 1]))) {
    d$mean2[loop] <- mean(c(d[loop, 1],
                            d[loop, 2]))
})
```

```
##    user  system elapsed
##    0.57    0.00    0.58
```

```
system.time(
  d$mean1 <- apply(d, 1, mean))
```

```
##    user  system elapsed
##    0.07    0.00    0.08
```

O desempenho do da função *apply* foi melhor que do *loop*, pois foi mais rápido e custou menos do sistema do usuário.

Função lapply

Função lapply

A função *lapply* é útil para realizar operações em objetos de lista. Retorna um objeto do tipo lista do mesmo comprimento do conjunto original.

Possui sintaxe conforme abaixo:

```
lapply(X, FUN, ...)
```

- *X* pode ser um vetor, lista, variáveis em *dataframes* ou matrizes;
- *FUN* é a função que será aplicada;
- As **reticências** são argumentos da função, que podem ser inseridos opcionalmente.

Diferente da função *apply*, não há a necessidade de especificar a *MARGIN*.

Exemplo I -função lapply

Criando uma lista com três elementos:

```
listal = list(a = 1:30, b=c(23,56,89,96,36,25,74), c=rnorm(10))
listal
```

```
## $a
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30
##
## $b
## [1] 23 56 89 96 36 25 74
##
## $c
##  [1] -0.2813623  0.6198714 -0.1548512 -1.0946567 -0.9147516 -0.1356464
##  [7]  0.3793756 -0.5464622  0.1270922  1.3742296
```

Exemplo I - função lapply

Somatório

```
lapply(listal, sum)
```

```
## $a
## [1] 465
##
## $b
## [1] 399
##
## $c
## [1] -0.6271615
```

Média

```
lapply(listal, mean)
```

```
## $a
## [1] 15.5
##
## $b
## [1] 57
##
## $c
## [1] -0.06271615
```

Variância

```
lapply(listal, var)
```

```
## $a
## [1] 77.5
##
## $b
## [1] 909.3333
##
## $c
## [1] 0.5387446
```

Desvio padrão

```
lapply(listal, sd)
```

```
## $a
## [1] 8.803408
##
## $b
## [1] 30.15515
##
## $c
## [1] 0.7339922
```

Exemplo II - função lapply

```
filmes <- c("SPYDERMAN","STAR WARS",
            "THE AVENGERS","JUSTICE LEAGUE")

filmesl <-lapply(filmes, tolower)
filmesl
```

```
## [[1]]
## [1] "spyderman"
##
## [[2]]
## [1] "star wars"
##
## [[3]]
## [1] "the avengers"
##
## [[4]]
## [1] "justice league"
```

```
filmesl <-unlist(lapply(filmes, tolower))
filmesl
```

```
## [1] "spyderman"      "star wars"      "the avengers"   "justice league"
```

Exemplo III - função lapply

Aplicando a função *sum* a duas colunas da base de dados *murders* do pacote *dslabs*.

```
soma <- lapply(murders[c('population',
                         'total')],sum)
```

```
print(soma)
```

```
## $population
## [1] 309864228
##
## $total
## [1] 9403
```

```
class(soma)
```

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

Função sapply

Função sapply

Essa função recebe uma lista, vetor ou *dataframe* como entrada e dá uma saída em vetor ou matriz. É útil para operações em objetos de lista e retorna um objeto de lista do mesmo comprimento do conjunto original. Abaixo pode-se ver a sua sintaxe:

```
sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

- *X* é uma lista, vetor ou *dataframe*;
- *FUN* é a função que será aplicada;
- As **reticências** são argumentos da função, que podem ser inseridos opcionalmente.
- *simplify* é um argumento lógico. argumento que especifica se queremos simplificar os resultados ou não.
- *USE.NAMES* usado para acessar valores na saída (vetores de caracteres)

O objetivo dessa função é a simplicação da saída.

Exemplo I - função sapply

Criando uma lista de valores numéricos:

```
listal<- list(a = 1:30, b=c(23,56,89,96,36,25,74), c=rnorm(10))
listal
```

Com Simplificação: retorna uma vetor com valores numéricos.

```
x1<- sapply(listal, sum, simplify = TRUE)

print(x1)
```

```
##           a           b           c
## 465.000000 399.000000   4.748674
```

```
print(class(x1))
```

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

Sem simplificação: A saída retorna um lista.

```
x2<- sapply(listal, sum, simplify = FALSE)

print(x2)
```

```
## $a
## [1] 465
##
## $b
## [1] 399
##
## $c
## [1] 4.748674
```

```
print(class(x2))
```

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

Exemplo II - função sapply

Usando a base de dados *mtcars*:

Com Simplificação

```
y1<- sapply(mtcars[c('mpg','hp')],
            range, simplify = TRUE)

print(y1)
```

```
##       mpg  hp
## [1,] 10.4  52
## [2,] 33.9 335
```

```
print(class(y1))
```

```
## [1] "matrix" "array"
```

Sem simplificação

```
y2<- sapply(mtcars[c('mpg','hp')],
            range, simplify = FALSE)

print(y2)
```

```
## $mpg
## [1] 10.4 33.9
##
## $hp
## [1]  52 335
```

```
print(class(y2))
```

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

Exemplo III - função sapply

Aplicando a função *toupper* a um vetor de *strings*.

```
Vingadores<- c("homem de ferro", "capitão américa",
               "thor", "hulk")

Vingadores1 <- sapply(Vingadores,toupper,simplify=TRUE,USE.NAMES=TRUE)

print(Vingadores1)
```

```
##    homem de ferro   capitão américa             thor             hulk
##  "HOMEM DE FERRO" "CAPITÃO AMÉRICA"           "THOR"           "HULK"
```

```
Vingadores<- c("homem de ferro","capitão américa","thor","hulk")

Vingadores1 <- sapply(Vingadores,toupper, simplify = TRUE, USE.NAMES = FALSE)

print(Vingadores1)
```

```
## [1] "HOMEM DE FERRO"  "CAPITÃO AMÉRICA" "THOR"            "HULK"
```

Função mapply

Função mapply

O *mapply()* é uma versão multivariada da função *sapply()*. O *mapply()* aplica uma função em paralelo ao conjunto de argumentos fornecido. Ele aplica a mesma função a cada argumento passado.

Quando existirem estruturas de dados diferentes (exemplo: vetores, listas) e houver a necessidade de aplicar a função para os primeiros elementos de cada, depois para os segundos e assim por diante, forçando o resultado em um vetor ou *array* como na função *sapply*.

Função mapply

A sintaxe para a função *mapply()* é mostrada abaixo:

```
mapply(FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE,
       USE.NAMES = TRUE)
```

- *FUN* é uma função a ser aplicada;
- As *reticências* contém objetos *R* para aplicar na função;
- *MoreArgs* é uma lista de outros argumentos para a função;
- *simplify* indica se o resultado deve ser simplificado;
- *USE.NAMES* é um argumento lógico (*TRUE* ou *FALSE*) para excluir ou não os rótulos da saída, quando pertinente.

Exemplo I - função mapply

Produto entre os valores das sequências.

```
mapply(prod, 1:2, 1:2, 1:2)
```

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

Somatório entre os valores das sequências.

```
mapply(sum, 1:3,10:12,2:4,7:9)
```

```
## [1] 20 24 28
```

rep dos valores de uma sequência com base em outra sequência.

```
mapply(rep, 5:8, 4:1)
```

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

Exemplo II - função mapply

Aplicando um vetor a outro vetor.

```
mapply(function(x, y) {x^y},
       x = c(2,3), y = c(4))
```

```
## [1] 16 81
```

Quando os vetor são múltiplos.

```
mapply(function(x, y) {x^y},
       c(2,3), c(3,4,5,6))
```

```
## [1]   8  81  32 729
```

Exemplo III - função mapply

Quando os vetor não são múltiplos.

```
mapply(function(x, y) {x^y},
       c(2,3), c(3,4,5))
```

```
## Warning in mapply(function(x, y) {: argumento longo não é múltiplo do
## comprimento do curto
```

```
## [1]  8 81 32
```

Inserindo rótulos aos resultados.

```
mapply(function(x, y) {x^y},
       c(a = 2, b = 3), c(A = 3, B = 4))
```

```
##  a  b
##  8 81
```

Exemplo IV - função mapply

Excluindo rótulos.

```
mapply(function(x, y) {x^y},
       c(a = 2, b = 3), c(A = 3, B = 4),
       USE.NAMES = FALSE)
```

```
## [1]  8 81
```

Inserindo outros argumentos.

```
mapply(function(x,y,z,k){(x+k)^(y+z)},
       c(a=2,b=3),c(A=3,B=4),MoreArgs=list(1,2))
```

```
##    a    b
##  256 3125
```

Função tapply

Função tapply

É usada para aplicar uma função em subconjuntos de um vetor. É usada principalmente quando um conjunto de dados que pode ser dividido em grupos (por meio de fatores) e queremos dividir o conjunto de dados em grupos; e dentro de cada grupo, queremos aplicar uma função. A sintaxe dessa função é

```
tapply(X, INDEX, FUN = NULL, ..., default = NA, simplify = TRUE)
```

- *X* é o vetor em que a função será aplicada;
- *INDEX* é o vetor de fatores;
- *FUN* é a função que será aplicada no subgrupo;
- *simplify* é o parâmetro de simplificação (*TRUE* ou *FALSE*).

Exemplo - função tapply

Exemplo

```
Ref_Laranjeira <- c(0.0561,0.0548,0.2061,0.1568,
                    0.1668,0.1685,0.0846,0.0681,
                    0.0645,0.0845,0.2141,0.2168,
                    0.0513,0.0566,0.0476,0.0684,
                    0.2168) #vetor de resposta

Bandas <- c("B1","B2","B7","B5","B5","B5","B4",
            "B3","B3","B4","B7","B7","B1","B2",
            "B1","B4","B7")

bandas_f <- factor(Bandas)  #vetor de fatores indicando os grupos experimentais
```

Média

```
Média <- tapply(Ref_Laranjeira,bandas_f,mean)
Média
```

```
##          B1          B2          B3          B4          B5          B7
## 0.05166667 0.05570000 0.06630000 0.07916667 0.16403333 0.21345000
```

Exemplo - função tapply

Desvio padrão

```
sd <- tapply(Ref_Laranjeira, bandas_f, sd)
sd
```

```
##           B1           B2           B3           B4           B5           B7
## 0.004261846 0.001272792 0.002545584 0.009324341 0.006321656 0.005062608
```

Variância

```
var <- tapply(Ref_Laranjeira,bandas_f,var)
var
```

```
##           B1           B2           B3           B4           B5           B7
## 1.816333e-05 1.620000e-06 6.480000e-06 8.694333e-05 3.996333e-05 2.563000e-05
```

Outras funções

Outras funções

- ***vapply( )*** : quando querer utilizar o *sapply()*, mas caso precise de um código mais rápido;
- ***rapply( )*** : aplicação de uma função para cada elemento de uma lista aninhada de forma recursiva;
- ***eapply( )*** : aplica uma função aos valores nomeados de um ambiente (*environment*) e retorna os resultados como uma lista;
- ***mclapply()*** : versão paralelizada da *lapply()*, que retorna uma lista do mesmo comprimento de *X*, cada elemento da qual é o resultado da aplicação da função ao elemento correspondente de *X*;

Funções relacionadas

- ***sweep( )***: é provavelmente a função mais próxima da família *apply*. Usada quando deseja replicar diferentes ações nos margens do elementos escolhido (limitado à matrizes);
- ***rep( )***: É frequentemente usada com as demais funções da família *apply*;
- ***aggregate( )***: Divide os dados em subconjuntos, calcula estatísticas resumidas para cada um e retorna o resultado em um formato conveniente;
- ***by( )***: é semelhante à função de aplicação, mas é usada para aplicar funções sobre o quadro de dados ou matriz.

Conclusão

Tabela resumo

Função	Tipos de dados de entrada	Tipos de dados de saída
apply( )	*Dataframes*, matrizes ou *array* (com margens)	Vetor, matriz, *array* ou lista
lapply( )	Vetor, lista, variáveis em *dataframes* ou matrizes	Lista
sapply( )	Vetor, lista, variáveis em *dataframes* ou matrizes	Matriz, vetor ou lista
mapply( )	Vetor, lista, variáveis em *dataframes* ou matrizes	Matriz, vetor ou lista
tapply( )	*Ragged array* (*array* irregular)	*Array*

Referências

Documentação das funções da família *apply* no *R documentation*:

apply
lapply
sapply
mapply
tapply
eapply
rapply

Obrigado !