RUDN university

5100

Программирование в среде R

Шевцов Василий Викторович,
директор ДИТ РУДН, shevtsov_vv@rudn.university

Функции семейства apply

Назначение

- apply-функции предназначены для организации векторизованных вычислений над объектами
- apply-функции выполняют параллельные вычисления, при которых программа обрабатывает одновременно весь массив (вектор) целиком или по несколько элементов вектора в каждый момент времени

apply()

- **apply(x, MARGIN, FUN, ...)**
  - где x – это преобразуемый объект,
  - MARGIN – индекс, обозначающий направление процесса вычислений (по столбцам или строкам),
  - FUN – применяемая для вычислений функция, в т.ч. function(),
  - a ... – это любые другие параметры применяемой функции.
  - Для матрицы или таблицы данных MARGIN = 1 обозначает строки, а MARGIN = 2 – столбцы.

apply()

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53

Задача – найти минимальное значение для каждой строки среди x,y,z

apply()

```
df <- diamonds

min_size <- c()
for(i in 1:nrow(df)){
  min_size <- c(min_size,min(df[i,8:10]))
}
```

>10 сек

```
min_size <- numeric(nrow(diamonds))
for(i in 1:nrow(df)){
  min_size[i] <- min(diamonds[i,8:10])
}
```

>10 сек

```
apply(diamonds[,8:10],1,min)
```

<1 сек

apply()

```
> m1 <- matrix(rnorm(30),nrow=5)
> apply(m1,MARGIN = 1,FUN=sd)
[1] 0.6677368 0.8644811 0.9966222 1.1406084 1.1558188
```

```
> apply(m1,MARGIN = 2,FUN=sd)
[1] 0.6296139 1.0550289 0.7463939 0.8915535 1.3299000
[6] 1.2490636
```

```
> apply(mtcars,2,sd)
       mpg        cyl       disp         hp
 6.0269481  1.7859216 123.9386938 68.5628685
      drat         wt       qsec         vs
 0.5346787  0.9784574  1.7869432  0.5040161
        am       gear       carb
 0.4989909  0.7378041  1.6152000
```

apply()

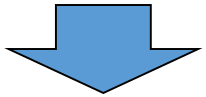
```
> range(1:10)
[1]  1 10
> m2 <- apply(m1,MARGIN = 2,FUN=range)
> is.matrix(m2)
[1] TRUE
```

```
> m2
           [,1]      [,2]       [,3]       [,4]
[1,] -0.1543624 -0.890092 -1.3927006 -0.6309224
[2,]  1.3140250  1.379002  0.5511297  1.5520951
          [,5]      [,6]
[1,] -1.691313 -1.305731
[2,]  1.403971  1.576996
```

В зависимости от выходных данных **apply** возвращает вектор или матрицу или список

apply() Обработка пропущенных значений

apply(data_frame,2,mean(na.rm=TRUE))



apply(data_frame,2,mean, na.rm=TRUE)

colMeans(apply(data_frame,na.rm=TRUE)
colSums()
rowMeans()
rowSums()

lapply()

используется в случаях, когда необходимо применить какую-либо функцию к каждому компоненту списка и получить результат также в виде списка (буква "l" в названии lapply() означает list – "список").

```
x <- list(a = 1, b = 1:3, c = 10:100)
> x $a [1] 1 $b [1] 1 2 3
$c [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
[31] 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
[61] 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
[91] 100
> lapply(x, FUN = length)
$a [1] 1 $b [1] 3 $c [1] 91
> lapply(x, FUN = sum)
$a [1] 1 $b [1] 6 $c [1] 5005
```

Длина ответа равна длине входных данных

sapply()

- используется в случаях, когда необходимо применить какую-либо функцию к каждому компоненту списка, но результат вывести в виде вектора (буква "s" в названии sapply() означает simplify – "упростить").

```
> x <- list(a = 1, b = 1:3, c = 10:100)
> x
$a
[1] 1

$b
[1] 1 2 3

$c
 [1]  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
[31]  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69
[61]  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
[91] 100

> sapply(x, FUN = length)
 a  b  c
 1  3 91
> sapply(x, FUN = sum)
   a    b    c
   1    6 5005
```

tapply()

- используется в случаях, когда необходимо применить какую-либо функцию fun к отдельным группам элементов вектора x, заданным в соответствии с уровнями какого-либо фактора group:
- tapply(x, group, fun, ...)
- является аналогом функции aggregate(), в качестве групирующей переменной выступает group

rapply()

- используется в случаях, когда необходимо применить какую-либо функцию к компонентам вложенного списка (буква "r" в названии rapply() означает recursively – "рекурсивно").

```
> x1 <- list(1:100)
> x <- list(a = 1, b = 1:3, c = 10:100, l = x1)
> lapply(x,sum)
Error in FUN(X[[i]], ...) : invalid 'type' (list) of argument
> rapply(x,sum)
   a    b    c    l
   1    6 5005 5050
```

mapply()

- используется в случаях, когда необходимо поэлементно применить какую-либо функцию одновременно к нескольким объектам (например, получить сумму первых элементов векторов, затем сумму вторых элементов векторов, и т.д.). Результат возвращается в виде вектора или массива другой размерности
- Буква "m" в названии mapply() означает multivariate – "многомерный" (имеется в виду одновременное выполнение вычислений над элементами нескольких объектов).

replicate()

- **Функция позволяет провести серию вычислений с целью генерации набора чисел по заданному алгоритму.**
- **replicate(n, expr, simplify=TRUE)**
  - n – число повторов,
  - expr – функция или группа выражений, которые надо повторить n раз ,
  - simplify = TRUE – необязательный параметр, который пробует упростить результат и представить его в виде вектора или матрицы значений.

replicate()

```
> replicate(5,rnorm(n=10,mean = 0))
             [,1]        [,2]        [,3]        [,4]        [,5]
 [1,]  0.8794987 -1.25067392  0.22614817 -0.84589266  1.10927249
 [2,] -0.3949991  0.23498348 -0.02829349  0.00639850  1.55757712
 [3,] -0.8162822 -0.60144752  1.32132198 -0.35723071  1.14098093
 [4,]  0.3507224  0.20920440 -1.28503232  0.63951895 -0.28341435
 [5,]  1.3193412  1.08584741 -0.62092921  0.10058936 -0.56130703
 [6,] -1.0303396  1.23373439 -1.24558938  0.87106111  0.05652448
 [7,] -0.9571155 -1.03439197 -0.34114690  0.29526020  2.14332222
 [8,] -0.5799248 -0.71021435 -1.04370193 -1.67186428 -0.47783249
 [9,]  1.5172604 -1.75068189 -0.72183832 -1.35900662 -1.02466297
[10,] -0.4107434  0.01837745 -0.17946855  0.08499628  0.56741416
```

by()

- является своего рода аналогом функции tapply(), с той разницей, что она применяется для таблиц.
- Таблица data разделяется в соответствии с заданным столбцом-фактором group на подмножество подтаблиц и для обработки каждой такой части определяется функция fun:
- by(data, group, fun, ...)

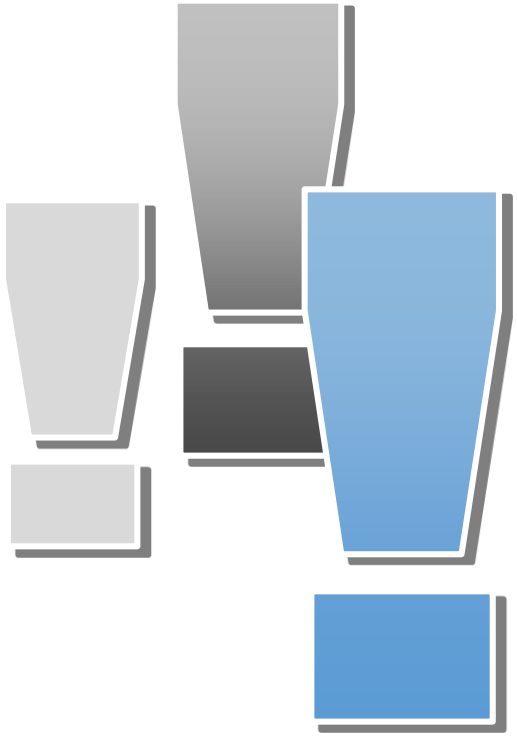
outer()

- позволяет выполнить комбинаторную операцию fun над элементами двух массивов или векторов x и y, не прибегая к явному использованию "двойного" цикла:
- outer(x, y, fun="*", ...)

```
> x <- 1:5 ; y <- 1:5
> outer(x,y,"*")
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    2    4    6    8   10
[3,]    3    6    9   12   15
[4,]    4    8   12   16   20
[5,]    5   10   15   20   25
```

```
> a <- letters[1:8]
> b <- 1:8
> outer(a, b, paste, sep="")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] "a1" "a2" "a3" "a4" "a5" "a6" "a7" "a8"
[2,] "b1" "b2" "b3" "b4" "b5" "b6" "b7" "b8"
[3,] "c1" "c2" "c3" "c4" "c5" "c6" "c7" "c8"
[4,] "d1" "d2" "d3" "d4" "d5" "d6" "d7" "d8"
[5,] "e1" "e2" "e3" "e4" "e5" "e6" "e7" "e8"
[6,] "f1" "f2" "f3" "f4" "f5" "f6" "f7" "f8"
[7,] "g1" "g2" "g3" "g4" "g5" "g6" "g7" "g8"
[8,] "h1" "h2" "h3" "h4" "h5" "h6" "h7" "h8"
```

Спасибо за внимание!



Шевцов Василий Викторович

shevtsov_vv@rudn.university
+7(903)144-53-57