Funções especiais na linguagem R

Ivanildo Batista

16 de março de 2021

FUNÇÕES ESPECIAIS

Função unlist(): reverte uma lista em um vetor

```
list1 = list(6,"b",15)#criando uma lista
list1
```

```
## [[1]]
## [1] 6
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] 15
```

```
class(list1)
```

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

```
unlist(list1)
```

```
## [1] "6"  "b"  "15"
```

```
vec1 = unlist(list1)
class(vec1)
```

```
## [1] "character"
```

```
list2 = list(v1 = 6, v2 = list(381,2190),v3 = c(30,217))
list2
```

```
## $v1
## [1] 6
##
## $v2
## $v2[[1]]
## [1] 381
##
## $v2[[2]]
## [1] 2190
##
##
## $v3
## [1]  30 217
```

```
mean(unlist(list2)) #retirando a média
```

```
## [1] 564.8
```

```
round(mean(unlist(list2))) #arredondando o resultado anterior
```

```
## [1] 565
```

Função DO.CALL(): aplica uma função a todos os elementos

```
data=list()
N=100

for (n in 1:N){
  data[[n]]=data.frame(index = n, char = sample(letters,1),z=rnorm(1))
}

head(data)
```

```
## [[1]]
##   index char          z
## 1     1    j -0.8326628
##
## [[2]]
##   index char          z
## 1     2    t -0.1594371
##
## [[3]]
##   index char          z
## 1     3    i -0.7306702
##
## [[4]]
##   index char          z
## 1     4    n -0.2614029
##
## [[5]]
##   index char        z
## 1     5    f 1.056918
##
## [[6]]
##   index char          z
## 1     6    m -0.7750788
```

```
do.call(rbind,data)
```

```
##     index char           z
## 1       1    j -0.832662761
## 2       2    t -0.159437149
## 3       3    i -0.730670182
## 4       4    n -0.261402901
## 5       5    f  1.056917965
## 6       6    m -0.775078800
## 7       7    n -0.482589472
## 8       8    l -2.062536044
## 9       9    z  1.091472952
## 10     10    o -0.325873107
## 11     11    q  1.654559729
## 12     12    a -0.463737326
## 13     13    c -0.093597027
## 14     14    e  0.543770551
## 15     15    d -0.835982514
## 16     16    s  0.417630429
## 17     17    r  0.761608004
## 18     18    b -1.551949435
## 19     19    m -1.376856126
## 20     20    z  1.813326511
## 21     21    i -0.645033617
## 22     22    v  0.591338424
## 23     23    l  0.738040048
## 24     24    m -0.526753990
## 25     25    o  0.399012660
## 26     26    e  1.870324826
## 27     27    y -1.118776682
## 28     28    o  0.228517255
## 29     29    p  0.792120034
## 30     30    r  1.147955167
## 31     31    l -0.416774115
## 32     32    h -0.380967977
## 33     33    b  0.363675182
## 34     34    i  0.681924439
## 35     35    f  2.435892782
## 36     36    q -0.874431650
## 37     37    q  0.909351927
## 38     38    x  0.840149968
## 39     39    r -0.558800417
## 40     40    d  0.583109245
## 41     41    o  1.496794996
## 42     42    z  1.088113298
## 43     43    k  0.160453320
## 44     44    a -0.886397877
## 45     45    l  0.623745276
## 46     46    x  1.491269677
## 47     47    k  1.060376015
## 48     48    e -1.592336760
## 49     49    s  0.680748808
## 50     50    k  0.692514305
## 51     51    h  0.316422557
## 52     52    g  1.314010688
```

```
## 53     53    t  0.318311117
## 54     54    n -0.394256407
## 55     55    l  0.720257985
## 56     56    h  0.458888191
## 57     57    o -0.109636621
## 58     58    s -1.073873318
## 59     59    a -1.027861011
## 60     60    x  1.346244405
## 61     61    a  1.472428415
## 62     62    c -0.564724273
## 63     63    b  0.077577226
## 64     64    d -0.321620642
## 65     65    e -0.652743425
## 66     66    o -0.007469217
## 67     67    e  0.891288456
## 68     68    t  0.228807498
## 69     69    y -0.369373923
## 70     70    z -0.436956538
## 71     71    t  0.135080167
## 72     72    i  0.441016618
## 73     73    d -0.784348029
## 74     74    d  1.568288641
## 75     75    g -0.488932318
## 76     76    u -0.873903421
## 77     77    d -0.109325907
## 78     78    j -0.259910094
## 79     79    g  0.240616775
## 80     80    o  0.950110117
## 81     81    e -0.010318239
## 82     82    g -2.223003035
## 83     83    s -0.227003545
## 84     84    c  0.428313755
## 85     85    h -1.023117061
## 86     86    s -0.056378648
## 87     87    k -0.951165661
## 88     88    f  0.497841778
## 89     89    p  0.170933953
## 90     90    o  0.294816268
## 91     91    v  0.128422582
## 92     92    t  0.313386942
## 93     93    k  0.926554068
## 94     94    b  0.734579082
## 95     95    c -0.343822511
## 96     96    o -0.074815680
## 97     97    p  1.723238532
## 98     98    i -0.973888574
## 99     99    p -0.065040783
## 100   100    g -0.735821450
```

```
class(do.call(rbind,data)) #gera um dataframe
```

```
## [1] "data.frame"
```

LAPPLY() X DO.CALL()

```
y = list(1:3,4:6,7:9)
y
```

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

```
lapply(y, sum) #aplicou para cada elemento da lista
```

```
## [[1]]
## [1] 6
##
## [[2]]
## [1] 15
##
## [[3]]
## [1] 24
```

```
do.call(sum,y)#aplicou para todos os elementos da lista
```

```
## [1] 45
```

Usando o pacote plyr: usando uma função desse pacote para fazer a mesma coisa que o DO.CALL()

```
library(plyr)

ldply(y,sum)
```

```
##   V1
## 1  6
## 2 15
## 3 24
```

Função STRPLIT(): divide uma string

```
texto = "Esta é uma string"

strsplit(texto, "") #divindindo por letra
```

```
## [[1]]
##  [1] "E" "s" "t" "a" " " " " "é" " " " " "u" "m" "a" " " " " "s" "t" "r" "i" "n" "g"
```

```
strsplit(texto, " ") #dividindo por palavra
```

```
## [[1]]
## [1] "Esta"   "é"      "uma"    "string"
```

```
dates = c('199-05-23','2001-12-30','2004-12-17')
temp = strsplit(dates, '-') #dividindo datas
temp
```

```
## [[1]]
## [1] "199" "05"  "23"
##
## [[2]]
## [1] "2001" "12"   "30"
##
## [[3]]
## [1] "2004" "12"   "17"
```

```
class(temp)
```

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

```
matrix(unlist(temp), ncol=3,byrow = TRUE) #criando uma matriz com as datas divididas
```

```
##      [,1]   [,2] [,3]
## [1,] "199"  "05" "23"
## [2,] "2001" "12" "30"
## [3,] "2004" "12" "17"
```

```
Names = c('Brin, Sergey','Page,Larry','Dorsey,Jack','Glass,Noah','Williams,Evan',
          'Stone,Riz')

cofounded = rep(c('Google','Twitter'),c(2,4)) #repetir uma quantidade de palavras de forma espec
ífica

cofounded
```

```
## [1] "Google"  "Google"  "Twitter" "Twitter" "Twitter" "Twitter"
```

```
temp = strsplit(Names,',')
temp
```

```
## [[1]]
## [1] "Brin"    " Sergey"
##
## [[2]]
## [1] "Page"  "Larry"
##
## [[3]]
## [1] "Dorsey" "Jack"
##
## [[4]]
## [1] "Glass" "Noah"
##
## [[5]]
## [1] "Williams" "Evan"
##
## [[6]]
## [1] "Stone" "Riz"
```

```
frase = 'Muitas vezes temos que repetir algo diversas vezes, mas é estranho
repetir palavras várias vezes'

palavras = strsplit(frase, " ")[[1]]
palavras
```

```
##  [1] "Muitas"    "vezes"     "temos"     "que"       "repetir"   "algo"
##  [7] "diversas"  "vezes,"    "mas"       "é"         "estranho"  "\nrepetir"
## [13] "palavras"  "várias"    "vezes"
```

```
unique(tolower(palavras)) #identificar palavras únicas
```

```
##  [1] "muitas"    "vezes"     "temos"     "que"       "repetir"   "algo"
##  [7] "diversas"  "vezes,"    "mas"       "é"         "estranho"  "\nrepetir"
## [13] "palavras"  "várias"
```

```
antes = data.frame(attr = c(1,30,4,6),tipo = c("pao_e_agua","pao_e_agua2"))
antes
```

```
##   attr        tipo
## 1    1  pao_e_agua
## 2   30 pao_e_agua2
## 3    4  pao_e_agua
## 4    6 pao_e_agua2
```

```
strsplit(as.character(antes$tipo),'_e_')
```

```
## [[1]]
## [1] "pao"  "agua"
##
## [[2]]
## [1] "pao"   "agua2"
##
## [[3]]
## [1] "pao"  "agua"
##
## [[4]]
## [1] "pao"   "agua2"
```

Usando a função STR_SPLIT_FIXED():

```
library(stringr) #chamando o pacote

str_split_fixed(antes$tipo,'_e_',2) #vai gerar uma matriz
```

```
##      [,1]  [,2]
## [1,] "pao" "agua"
## [2,] "pao" "agua2"
## [3,] "pao" "agua"
## [4,] "pao" "agua2"
```

```
depois = strsplit(as.character(antes$tipo),'_e_')
do.call(rbind,depois) #gera uma matriz
```

```
##      [,1]  [,2]
## [1,] "pao" "agua"
## [2,] "pao" "agua2"
## [3,] "pao" "agua"
## [4,] "pao" "agua2"
```

Usando a função SEPARATE():

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

antes %>%
  separate(tipo,c("pao","agua"),"_e_") #gera um dataframe
```

```
##   attr pao  agua
## 1    1 pao  agua
## 2   30 pao agua2
## 3    4 pao  agua
## 4    6 pao agua2
```

Operadores de atribuição

```
#vec = 1:4
vec2<-1:4 #tanto "=" quanto "<-" fazem a mesma coisa

#class(vec)
class(vec2)
```

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

```
#typeof(vec)
typeof(vec2)
```

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

```
#mean(x=1:10)
#x dá um erro

mean(x<-1:10)
```

```
## [1] 5.5
```

```
x #não dá um erro
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Criação de objetos

```
vetor1 = 1:4
vetor2 = c(1:4)
vetor3 = c(1,2,3,4)

class(vetor1)
```

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

```
class(vetor2)
```

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

```
class(vetor3) #a classe desse é diferentes dos anteriores
```

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

```
typeof(vetor1)
```

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

```
typeof(vetor2)
```

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

```
typeof(vetor3) #o typeof desse também é diferente
```

```
## [1] "double"
```

```
matriz1 = matrix(1:4,nr=2)
matriz2 = matrix(c(1:4),nr=2)
matriz3 = matrix(c(1,2,3,4),nr=2)

class(matriz1)
```

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

```
class(matriz2)
```

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

```
class(matriz3) #a classe de todos são iguais
```

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

```
typeof(matriz1)
```

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

```
typeof(matriz2)
```

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

```
typeof(matriz3) #para a terceira matriz, o typeof foi diferente
```

```
## [1] "double"
```