Laborarator 6 – Probabilități și Statistică Matematică

FAMILIA DE FUNCTII *apply*

Pe lângă buclele `for` și `while`, în R există și un set de funcții care permit scrierea și rularea într-o manieră mai compactă a codului dar și aplicarea de funcții unor grupuri de date.

- `lapply()`: Evaluează o funcție pentru fiecare element al unei liste
- `sapply()`: La fel ca `lapply` numai că încearcă să simplifice rezultatul
- `apply()`: Aplică o funcție după fiecare dimensiune a unui `array`
- `tapply()`: Aplică o funcție pe submulțimi ale unui vector
- `mapply()`: Varianta multivariată a funcției `lapply`
- `split`: Împarte un vector în grupuri definite de o variabilă de tip factor.

6.1 `lapply()`

Funcția `lapply()` efectuează următoarele operații:

1. buclează după o listă, iterând după fiecare element din acea listă
2. aplică o *funcție* fiecărui element al listei (o funcție pe care o specificăm)
3. întoarce ca rezultat tot o listă (prefixul `l` vine de la listă).

Această funcție primește următoarele trei argument: (1) o listă `X`; (2) o funcție `FUN`; (3) alte argumente via `...`. Dacă `X` nu este o listă atunci aceasta va fi transformată într-una folosind comanda `as.list()`.

Considerăm următorul exemplu în care vrem să aplicăm funcția `mean()` tuturor elementelor unei liste

```
set.seed(222)
```

```
x <- list(a = 1:5, b = rnorm(10), c = rnorm(20, 1), d = rnorm(10
0, 5))
lapply(x, mean)
$a
[1] 3

$b
[1] 0.1996044

$c
[1] 0.7881026

$d
[1] 5.064188
```

Putem să folosim funcția `lapply()` pentru a evalua o funcție în moduri repetate. Mai jos avem un exemplu în care folosim funcția `runif()` (permite generarea observațiilor uniform repartizate) de patru ori, de fiecare dată generăm un număr diferit de valori aleatoare. Mai mult, argumentele $\min=0$min=0și $\max=3$max=3 sunt atribuite, prin intermediul argumentului `...`, funcției `runif`.

```
x <- 1:4
lapply(x, runif, min = 0, max = 3)
[[1]]
[1] 0.03443616

[[2]]
[1] 1.267361 1.365441

[[3]]
[1] 1.8084700 2.1902665 0.4139585
```

```
[[4]]
[1] 1.5924650 0.7355067 2.1483841 1.6082945
```

6.2 `sapply()`

Funcția `sapply()` are un comportament similar cu `lapply()` prin faptul că funcția `sapply()` apelează intern `lapply()` pentru valorile de input, după care evaluează:

- dacă rezultatul este o listă în care fiecare element este de lungime 1, atunci întoarce un vector
- dacă rezultatul este o listă în care fiecare element este un vector de aceeași lungime (>1), se întoarce o matrice
- în caz contrar se întoarce o listă.

Considerăm exemplul de mai sus

```
set.seed(222)
x <- list(a = 1:4, b = rnorm(10), c = rnorm(20, 1), d = rnorm(10
0, 5))
sapply(x, mean)
        a         b         c         d
2.5000000 0.1996044 0.7881026 5.0641876
```

6.3 `split()`

Funcția `split()` primește ca argument un vector sau o listă (sau un data.frame) și împarte datele în grupuri determinate de o variabilă de tip factor (sau o listă de factor).

Argumentele aceste funcții sunt

```
str(split)
```

```
function (x, f, drop = FALSE, ...)
```

unde

- `x` este un vector, o listă sau un data.frame
- `f` este un factor sau o listă de factori

Considerăm următorul exemplu în care generăm un vector de date și îl împărțim după o variabilă de tip factor creată cu ajutorul funcției `gl()` (*generate levels*).

```
x <- c(rnorm(10), runif(10), rnorm(10, 1))
f <- gl(3, 10)
split(x, f)
$`1`
 [1] -2.27414224 -0.11266780  0.61308167  0.07733545  0.57137727
 [6]  0.11672493 -0.95685256 -1.90008460 -1.48972089  0.55925676

$`2`
 [1] 0.91159086 0.03291829 0.78368939 0.11852882 0.64443831 0.78
790988
 [7] 0.82451477 0.05642366 0.65075027 0.95426854

$`3`
 [1]  2.6666242  2.6634334  1.8106280 -0.7837308  1.6575684  0.1
546575
 [7]  0.4930056 -0.9031544  2.4042311  1.4106863
```

Putem folosi funcția `split` și în conjuncție cu funcția `lapply` (atunci când vrem să aplicăm o funcție `FUN` pe grupuri de date).

```
lapply(split(x, f), mean)
$`1`
[1] -0.4795692
```

```
$`2`
[1] 0.5765033


$`3`
[1] 1.157395
```

6.4 `tapply()`

Funcția `tapply()` este folosită pentru aplicarea unei funcții `FUN` pe submulțimile unui vector și poate fi văzută ca o combinație între `split()` și `sapply()`, dar doar pentru vectori.

```
str(tapply)
function (X, INDEX, FUN = NULL, ..., default = NA, simplify = TR
UE)
```

Argumentele acestei funcții sunt date de următorul tabel:

Tabelul 14. Argumentele functiei tapply

Argument	Descriere
`X`	un vector
`INDEX`	este o variabilă de tip factor sau o listă de factori
`FUN`	o funcție ce urmează să fie aplicată
`...`	argumente ce vor fi atribuite funcției `FUN`
`simplify`	dacă vrem să simplificăm rezultatul

Următorul exemplu calculează media după fiecare grupă determinată de o variabilă de tip factor a unui vector numeric.

```
x <- c(rnorm(10), runif(10), rnorm(10, 1))
f <- gl(3, 10)
f
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
Levels: 1 2 3
tapply(x, f, mean)
           1            2            3
-0.0007774025  0.3736457792  0.5789436983
```

Putem să aplicăm și funcții care întorc mai mult de un rezultat. În această situație rezultatul nu poate fi simplificat:

```
tapply(x, f, range)
$`1`
[1] -2.1904113  0.9249901
$`2`
[1] 0.004445296 0.998309704
$`3`
[1] -0.3379675  1.9327099
```

6.5 `apply()`

Funcția `apply()` este folosită cu precădere pentru a aplica o funcție liniilor și coloanelor unei matrice (care este un `array` bidimensional). Cu toate acestea poate fi folosită pe tablouri multidimensionale (`array`) în general. Folosirea funcției `apply()` nu este mai rapidă decât scrierea unei bucle `for`, dar este mai compactă.

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

Argumentele funcției `apply()` sunt

- `X` un tablou multidimensional

- `MARGIN` este un vector numeric care indică dimensiunea sau dimensiunile după care se va aplica funcția
- `FUN` este o funcție ce urmează să fie aplicată
- `...` alte argumente penru funcția `FUN`

Considerăm următorul exemplu în care calculăm media pe coloane într-o matrice

```
x <- matrix(rnorm(200), 20, 10)

apply(x, 2, mean)  ## media fiecarei coloane

 [1]  3.745002e-02  1.857656e-01 -2.413659e-01 -2.093141e-01 -2.
562272e-01

 [6]  8.986712e-05  7.444137e-02 -7.460941e-03  6.275282e-02  9.
801550e-02
```

precum și media după fiecare linie

```
apply(x, 1, sum)   ## media fiecarei linii

 [1]  2.76179139  2.53107681  0.87923177  1.80480589  0.98225832

 [6] -3.06148753 -1.40358820 -0.65969812 -1.63717046 -0.29330726

[11] -2.41486442 -3.15698523  2.27126822 -3.88290287 -3.15595194

[16]  5.41211963  2.32985530 -3.05330574 -0.02110926 -1.34909559
```