

Apply-ok

Csima Judit

BME, VIK,
Számítástudományi és Információelméleti Tanszék

2018. március 8. és 15.

apply-ok

Ciklusokat használhatunk interaktívan ill. egysoros parancsként is:
apply-ok

A leggyakoribbak:

- `lapply`: Egy lista minden elemére lefuttatja ugyanazt a függvényt
- `sapply`: Uaz, mint az `lapply`, csak próbálja egyszerűbben kiírni az eredményt
- `apply`: egy többdimenziós objektum (mátrix vagy data frame) soraira ill. oszlopaira futtatunk le vmit
- `mapply`: több argumentumú függvényt futtat le úgy, hogy az argumentumok több listából jönnek

Nagyon hasznos még (ált. `lapply` előtt): `split`

Vannak még más apply-ok is.

lapply

Két kötelező argumentuma van: X = milyen listára, FUN = milyen függvényt futtasson le.

```
> l <- list(1:5, rnorm(5))
> lapply(l, mean)
[[1]]
[1] 3
[[2]]
[1] -0.9900133
```

Lehet további argumentum, ha a FUN függvénynek szüksége van rá.

```
> lapply(1:3, rnorm, 10, 1)
[[1]]
[1] 12.67631

[[2]]
[1] 9.622761 10.824292

[[3]]
[1] 8.534720 9.355985 7.674501
```

```
> l <- lapply(1:3, rnorm, 10, 1)
> lapply(l, mean)
[[1]]
[1] 11.60651

[[2]]
[1] 9.510879

[[3]]
[1] 9.566127
```

Lehetne ezt egyszerűbben is írni....

sapply

az lapply mindig listát ad vissza, az sapply megpróbálja egyszerűsíteni az eredményt:

- ha az eredmény egy lista, aminek minden eleme 1 hosszúságú, akkor vektort ad vissza
- ha az eredmény egy lista, aminek minden eleme egy azonos hosszúságú (> 1) vektor, akkor mátrixot ad vissza
- egyébként uazt adja vissza, mint lapply (listát)

```
> sapply(1, mean)
[1] 11.60651 9.510879 9.566127
```

A sima mean nem megy:

```
> mean(1)
[1] NA
```

Warning message:

```
In mean.default(1) : argument is not numeric or logical:
returning NA
```

Lehet a függvényt az l/sapply-on belül definiálni:

```
> lapply(1:3, function(n) n*n)
```

```
[[1]]
```

```
[1] 1
```

```
[[2]]
```

```
[1] 4
```

```
[[3]]
```

```
[1] 9
```

```
> sapply(1:3, function(n) n*n)
```

```
[1] 1 4 9
```


Split data frame-re

Valamelyik oszlop értékei szerint szétvágja a data frame-et csoportokra, eredmény egy lista, melynek minden tagja egy data frame:

```
> l<- split(airquality, airquality$Month)
> class(l)
[1] "list"
> length(l)
[1] 5
```

```
> sapply(1, function(x) colMeans(x[,c(1,2)], na.rm=TRUE))
      5      6      7      8      9
Ozone 23.61538 29.44444 59.11538 59.96154 31.44828
Solar.R 181.29630 190.16667 216.48387 171.85714 167.43333
```

Ez egy sorban ugyanaz, mintha azt írtam volna:

```
> colm <- function(x){ colMeans(x[,c(1,2)], na.rm=TRUE)}
> sapply(1, colm)
```

apply

Kiszámol egy függvényt egy mátrix vagy data frame minden sorára ill. oszlopára

Argumentumai: X = többdimenziós objektum, MARGIN= melyik dimenzió szerint kell számolni (row = 1, column = 2), FUN= mit kell csinálni, és még esetleg egyéb argumentumok, ha az kell a FUN függvénynek.

```
> x <- matrix(rnorm(200), 20, 10)
> apply(x, 2, mean)
[1] 0.04868268 0.35743615 -0.09104379
[4] -0.05381370 -0.16552070 -0.18192493
[7] 0.10285727 0.36519270 0.14898850
[10] 0.26767260
```

a korábban már látott colMeans (és társai) igazából rövidítések:

```
rowSum = apply(x, 1, sum)
```

```
rowMeans = apply(x, 1, mean)
```

```
colSum = apply(x, 2, sum)
```

```
colMeans = apply(x, 2, mean)
```

mapply

Több argumentummal rendelkező függvényekre apply:

```
> mapply(function(x,y,z) rnorm(x,y,z), 1:3, 3:5, 1:3)
```

```
[[1]]
```

```
[1] 3.540355
```

```
[[2]]
```

```
[1] 2.931486 2.437551
```

```
[[3]]
```

```
[1] 6.3667617 7.0573508 -0.4421403
```

Ez uaz, mint az `mapply(rnorm, 1:3, 3:5, 1:3)`