Laborarator 3 – Probabilități și Statistică Matematică

TIPURI SI STRUCTURI DE DATE (2)

3.3 Liste

Spre deosebire de vectori în care toate elementele trebuie să aibă același tip de dată, structura de dată din R de tip listă (*list*) permite combinarea obiectelor de mai multe tipuri. Cu alte cuvinte, o listă poate avea primul element un scalar, al doilea un vector, al treilea o matrice iar cel de-al patrulea element poate fi o altă listă. Tehnic listele sunt tot vectori, vectorii pe care i-am văzut anterior se numesc *vectori atomici*, deoarece elementele lor nu se pot diviza, pe când listele se numesc *vectori recursivi*.

Ca un prim exemplu să considerăm cazul unei baze de date de angajați. Pentru fiecare angajat, ne dorim să stocăm numele angajatului (șir de caractere), salariul (valoare numerică) și o valoare de tip logic care poate reprezenta apartenența într-o asociație. Pentru crearea listei folosim funcția `list()`:

```
a = list(nume = "Ionel", salariu = 1500, apartenenta = T)
a
$nume
[1] "Ionel"
$salariu
[1] 1500
$apartenenta
[1] TRUE
str(a) # structura listei
List of 3
 $ nume       : chr "Ionel"
 $ salariu    : num 1500
 $ apartenenta: logi TRUE
names(a) # numele listei
[1] "nume"        "salariu"     "apartenenta"
```

Numele componentelor listei `a` (nume, salariu, apartenenta) nu sunt obligatorii dar cu toate acestea pentru claritate sunt indicate:

```
a2 = list("Ionel", 1500, T)
a2
[[1]]
[1] "Ionel"

[[2]]
[1] 1500

[[3]]
[1] TRUE
```

Deoarece listele sunt vectori ele pot fi create și prin intermediul funcției `vector()`:

```
z <- vector(mode="list")
z
list()
z[["a"]] = 3
z
$a
[1] 3
```

3.3.1 Indexarea listelor

Elementele unei liste pot fi accesate în diferite moduri. Dacă dorim să extragem primul element al listei atunci vom folosi indexarea care folosește o singură pereche de paranteze pătrate `[]`

```
a[1]
$nume
[1] "Ionel"
```

```
a[2]
$salariu
[1] 1500

# ce obtinem cand extragem un element al listei a ?
str(a[1])
List of 1
 $ nume: chr "Ionel"
```

În cazul în care vrem să accesăm structura de date corespunzătoare elementului i al listei vom folosi două perechi de paranteze pătrate `[[]]` sau în cazul în care lista are nume operatorul `$` urmat de numele elementului i.

```
a[[1]]
[1] "Ionel"
a[[2]]
[1] 1500

a$nume
[1] "Ionel"
a[["nume"]]
[1] "Ionel"
```

Operațiile de adăugare, respectiv ștergere, a elementelor unei liste sunt des întâlnite.

Putem adăuga elemente după ce o listă a fost creată folosind numele componentei

```
z = list(a = "abc", b = 111, c = c(TRUE, FALSE))
z
$a
[1] "abc"
```

```
$b
[1] 111
$c
[1]  TRUE FALSE
z$d = "un nou element"
z
$a
[1] "abc"
$b
[1] 111
$c
[1]  TRUE FALSE
$d
[1] "un nou element"
```

sau indexare vectorială

```
z[[5]] = 200
z[6:7] = c("unu", "doi")
z
$a
[1] "abc"
$b
[1] 111
$c
[1]  TRUE FALSE
$d
[1] "un nou element"
[[5]]
[1] 200
[[6]]
```

```
[1] "unu"
[[7]]
[1] "doi"
```

Putem șterge o componentă a listei atribuindu-i valoarea `NULL`:

```
z[4] = NULL
z
$a
[1] "abc"
$b
[1] 111
$c
[1]  TRUE FALSE
[[4]]
[1] 200
[[5]]
[1] "unu"
[[6]]
[1] "doi"
```

Putem de asemenea să concatenăm două liste folosind funcția `c()` și să determinăm lungimea noii liste cu funcția `length()`.

```
l1 = list(1:10, matrix(1:6, ncol = 3), c(T, F))
l2 = list(c("Ionel", "Maria"), seq(1,10,2))
l3 = c(l1, l2)
length(l3)
[1] 5
str(l3)
List of 5
 $ : int [1:10] 1 2 3 4 5 6 7 8 9 10
```

```
 $ : int [1:2, 1:3] 1 2 3 4 5 6
 $ : logi [1:2] TRUE FALSE
 $ : chr [1:2] "Ionel" "Maria"
 $ : num [1:5] 1 3 5 7 9
```

3.4 Data frame-uri

La nivel intuitiv, o structură de date de tip *data frame* este ca o matrice, având o structură bidimensională cu linii și coloane. Cu toate acestea ea diferă de structura de date de tip matrice prin faptul că fiecare coloană poate avea tipuri de date diferite. Spre exemplu, o coloană poate să conțină valori numerice pe când o alta, valori de tip caracter sau logic. Din punct de vedere tehnic, o structură de tip data frame este o listă a cărei componente sunt vectori (atomici) de lungimi egale.

Pentru a crea un dataframe din vectori putem folosi funcția `data.frame()`. Această funcție funcționează similar cu funcția `list()` sau `cbind()`, diferența față de `cbind()` este că avem posibilitatea să dăm nume coloanelor atunci când le unim. Dată fiind flexibilitatea acestei structuri de date, majoritatea seturilor de date din R sunt stocate sub formă de dataframe (această structură de date este și cea mai des întâlnită în analiza statistică).

Să creăm un dataframe simplu numit `survey` folosind funcția `data.frame()`:

```
survey <- data.frame("index" = c(1, 2, 3, 4, 5),
                     "sex" = c("m", "m", "m", "f", "f"),
                     "age" = c(99, 46, 23, 54, 23))
survey
  index sex age
1     1   m  99
2     2   m  46
3     3   m  23
4     4   f  54
```

```
5     5   f  23
```

Funcția `data.frame()` prezintă un argument specific numit `stringsAsFactors` care permite convertirea coloanelor ce conțin elemente de tip caracter într-un tip de obiect numit **factor**. Un factor este o variabilă nominală care poate lua un număr bine definit de valori. De exemplu, putem crea o variabilă de tip factor *sex* care poate lua doar două valori: *masculin* și *feminin*. Comportamentul implicit al funcției `data.frame()` (`stringAsFactors = TRUE`) transformă automat coloanele de tip caracter în factor, motiv pentru care trebuie să includem argumentul `stringsAsFactors = FALSE`.

```
# Structura initiala
str(survey)
'data.frame':   5 obs. of  3 variables:
 $ index: num  1 2 3 4 5
 $ sex  : Factor w/ 2 levels "f","m": 2 2 2 1 1
 $ age  : num  99 46 23 54 23
survey <- data.frame("index" = c(1, 2, 3, 4, 5),
                     "sex" = c("m", "m", "m", "f", "f"),
                     "age" = c(99, 46, 23, 54, 23),
                     stringsAsFactors = FALSE)

# Structura de dupa
str(survey)
'data.frame':   5 obs. of  3 variables:
 $ index: num  1 2 3 4 5
 $ sex  : chr  "m" "m" "m" "f" ...
 $ age  : num  99 46 23 54 23
```

R are mai multe funcții care permit vizualizarea structurilor de tip dataframe. Tabelul de mai jos include câteva astfel de funcții:

Tabelul 4. Exemple de functii necesare pentru intelegerea structurii dataframe-ului

Funcție	**Descriere**
`head(x), tail(x)`	Printarea primelor linii (sau ultimelor linii).
`View(x)`	Vizualizarea obiectului într-o fereastră nouă, tabelară.
`nrow(x), ncol(x), dim(x)`	Numărul de linii și de coloane.
`rownames(), colnames(), names()`	Numele liniilor sau coloanelor.
`str(x)`	Structura dataframe-ului

```
data() # vedem ce seturi de date exista
# Alegem setul de date mtcars
?mtcars
starting httpd help server ... done
str(mtcars) # structura setului de date
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ..
.
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
```

```
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear
carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4
4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4
4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4
1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3
1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3
2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3
1
tail(mtcars)
                 mpg cyl  disp  hp drat    wt qsec vs am gear car
b
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5
2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5
2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5
4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5
6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5
8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4
2
rownames(mtcars)
 [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
 [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
 [7] "Duster 360"          "Merc 240D"           "Merc 230"
```

```
[10] "Merc 280"            "Merc 280C"           "Merc 450SE"
[13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleet
wood"
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
[31] "Maserati Bora"       "Volvo 142E"
names(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am
"   "gear"
[11] "carb"
View(mtcars)
```

3.4.1 Metode de indexare

Indexarea structurilor de tip dataframe se face la fel ca și indexarea listelor.

```
mtcars[1,1:4]
          mpg cyl disp  hp
Mazda RX4  21   6  160 110
mtcars[c(1,2),2]
[1] 6 6
mtcars$mpg
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4
17.3 15.2
[15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3
26.0 30.4
[29] 15.8 19.7 15.0 21.4
```

La fel ca vectorii, dataframe-urile (dar și listele) pot fi indexate logic

```
mtcars[mtcars$mpg > 25, ]
```

```
                mpg cyl  disp  hp drat    wt  qsec vs am gear ca
rb
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4
1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4
2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4
1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4
1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5
2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5
2
mtcars[(mtcars$mpg > 25) & (mtcars$wt < 1.8), ]
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Honda Civic  30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
Lotus Europa 30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2
```

O altă metodă de indexare este prin folosirea funcției `subset()`.

Tabelul 5. Principalele argumente ale functiei subset()

Argument	Descriere
`x`	Un dataframe
`subset`	Un vector logic care indică liniile pe care le vrem
`select`	Coloanele pe care vrem să le păstrăm

```
subset(x = mtcars,
       subset = mpg < 12 &
                cyl > 6,
       select = c(disp, wt))
                    disp    wt
```

```
Cadillac Fleetwood   472 5.250
Lincoln Continental  460 5.424
```

3.4.2 Metode de manipulare

În această secțiune vom prezenta câteva metode mai avansate de manipulare a seturilor de date (a data frame-urilor).

Vom începe prin introducerea comenzii `order()` care permite sortarea liniilor unui data.frame în funcție de valorile coloanei de interes. De exemplu să considerăm cazul setului de date `mtcars`. Vrem să afisăm primele 10 mașini în funcție de greutatea lor (crescător și descrescător):

```
cars_increasing = rownames(mtcars[order(mtcars$wt),])
# afisarea celor mai usoare 10 masini
cars_increasing[1:10]
 [1] "Lotus Europa"   "Honda Civic"    "Toyota Corolla" "Fiat X1
-9"
 [5] "Porsche 914-2"  "Fiat 128"       "Datsun 710"     "Toyota
Corona"
 [9] "Mazda RX4"      "Ferrari Dino"
cars_decreasing = rownames(mtcars[order(mtcars$wt, decreasing =
TRUE),])
# afisarea celor mai grele 10 masini
cars_decreasing[1:10]
 [1] "Lincoln Continental" "Chrysler Imperial"   "Cadillac Fleet
wood"
 [4] "Merc 450SE"          "Pontiac Firebird"    "Camaro Z28"
 [7] "Merc 450SLC"         "Merc 450SL"          "Duster 360"
[10] "Maserati Bora"
```

Funcția `order()` permite ordonarea după mai mult de o coloană, de exemplu dacă vrem să ordonăm mașinile după numărul de cilindrii și după greutate atunci apelăm

```
mtcars[order(mtcars$cyl, mtcars$wt), 1:6]
                     mpg cyl  disp  hp drat    wt
Lotus Europa        30.4   4  95.1 113 3.77 1.513
Honda Civic         30.4   4  75.7  52 4.93 1.615
Toyota Corolla      33.9   4  71.1  65 4.22 1.835
Fiat X1-9           27.3   4  79.0  66 4.08 1.935
Porsche 914-2       26.0   4 120.3  91 4.43 2.140
Fiat 128            32.4   4  78.7  66 4.08 2.200
Datsun 710          22.8   4 108.0  93 3.85 2.320
Toyota Corona       21.5   4 120.1  97 3.70 2.465
Volvo 142E          21.4   4 121.0 109 4.11 2.780
Merc 230            22.8   4 140.8  95 3.92 3.150
Merc 240D           24.4   4 146.7  62 3.69 3.190
Mazda RX4           21.0   6 160.0 110 3.90 2.620
Ferrari Dino        19.7   6 145.0 175 3.62 2.770
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215
Merc 280            19.2   6 167.6 123 3.92 3.440
Merc 280C           17.8   6 167.6 123 3.92 3.440
Valiant             18.1   6 225.0 105 2.76 3.460
Ford Pantera L      15.8   8 351.0 264 4.22 3.170
AMC Javelin         15.2   8 304.0 150 3.15 3.435
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440
Dodge Challenger    15.5   8 318.0 150 2.76 3.520
Duster 360          14.3   8 360.0 245 3.21 3.570
Maserati Bora       15.0   8 301.0 335 3.54 3.570
Merc 450SL          17.3   8 275.8 180 3.07 3.730
Merc 450SLC         15.2   8 275.8 180 3.07 3.780
Camaro Z28          13.3   8 350.0 245 3.73 3.840
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845
```

```
Merc 450SE          16.4   8 275.8 180 3.07 4.070
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345
Lincoln Continental 10.4   8 460.0 215 3.00 5.424
```

Sunt multe situațiile în care avem la dispoziție două sau mai multe seturi de date și am vrea să construim un nou set de date care să combine informațiile din acestea. Pentru aceasta vom folosi funcția `merge()`. Principalele argumente ale acestei funcții se regăsesc în tabelul de mai jos:

Tabelul 6. Argumentele functiei merge

Argument	Descriere
`x, y`	Două data frame-uri ce urmează a fi unite
`by`	Un vector de caractere ce reprezintă una sau mai multe coloane după care se va face lipirea. De exemplu `by = "id"` va combina coloanele care au valori care se potrivesc într-o coloană care se numește `"id"`. `by = c("last.name", "first.name")` va combina coloanele care au valori care se potrivesc în ambele coloane `"last.name"` și `"first.name"`
`all`	Un vector logic care indică dacă vrem să includem sau nu liniile care nu se potrivesc conform argumentului `by`.

Să presupunem că avem la dispoziție un set de date în care apar 5 studenți și notele pe care le-au obținut la examenul de statistică:

```
stat_course = data.frame(student = c("Ionel", "Maria", "Gigel",
"Vasile", "Ana"),
                         note_stat = c(9, 8, 5, 7, 9))
```

și să presupunem că avem notele acestor studenți la examenul de algebră

```
alg_course = data.frame(student = c("Maria", "Ana" , "Gigel", "I
onel", "Vasile"),
                         note_alg = c(10, 8, 9, 7, 9))
```

Scopul nostru este să creăm un singur tabel în care să regăsim notele la ambele materii:

```
combined_courses = merge(x = stat_course,
                         y = alg_course,
                         by = "student")
combined_courses
  student note_stat note_alg
1     Ana         9        8
2   Gigel         5        9
3   Ionel         9        7
4   Maria         8       10
5  Vasile         7        9
```

O a treia funcție care joacă un rol important în manipularea data frame-urilor este funcția `aggregate()` care, după cum îi spune și numele, permite calcularea de funcții pe grupe de date din setul initial. Argumentele principale ale acestei funcții sunt date în tabelul de mai jos:

Tabelul 7. Argumentele functiei aggregate

Argument	Descriere
`formula`	O formulă de tipul `y ~ x1 + x2 + ...` unde y este variabila dependentă iar x1, x2, … sunt variabilele independente. De exemplu, `salary ~ sex + age` va agrega o coloană `salary` la fiecare combinație unică de `sex` și `age`
`FUN`	O funcție pe care vrem să o aplicăm lui y la fiecare nivel al variabilelor independente. E.g. `mean` sau `max`.
`data`	Data frame-ul care conține variabilele din `formula`
`subset`	O submulțime din `data` pe care vrem să le analizăm. De exemplu, `subset(sex == "f" & age > 20)` va restrânge analiza la femei mai învârstă de 20 de ani.

Structura generală a funcției `aggregate()` este

```
aggregate(formula = dv ~ iv, # dv este data, iv este grupul
          FUN = fun, # Functia pe care vrem sa o aplicam
          data = df) # setul de date care contine coloanele dv s
i iv
```

Să considerăm setul de date `ChickWeight` și să ne propunem să calculăm pentru fiecare tip de dietă greutatea medie:

```
# Fara functia aggregate
mean(ChickWeight$weight[ChickWeight$Diet == 2])
[1] 122.6167
mean(ChickWeight$weight[ChickWeight$Diet == 3])
[1] 142.95
mean(ChickWeight$weight[ChickWeight$Diet == 4])
[1] 135.2627
# Cu ajutorul functiei aggregate
aggregate(formula = weight ~ Diet,   # DV este weight, IV este Di
et
          FUN = mean,                # calculeaza media pentru fi
ecare grup
          data = ChickWeight)        # dataframe este ChickWeight
  Diet   weight
1    1 102.6455
2    2 122.6167
3    3 142.9500
4    4 135.2627
```

Funcția `aggregate()` a întors un data.frame cu o coloană pentru variabila independentă `Diet` și o coloană pentru greutatea medie.

Dacă vrem să calculăm greutățile medii în funcție de dietă pentru găinile care au mai puțin de 10 săptămâni de viață atunci folosim opțiunea `subset`:

```
aggregate(formula = weight ~ Diet,   # DV este weight, IV este Di
et
          FUN = mean,                # calculeaza media pentru fi
ecare grup
          subset = Time < 10,        # gainile care au mai putin
de 10 saptamani
```

```
          data = ChickWeight)       # dataframe este ChickWeight
  Diet   weight
1    1 58.03093
2    2 63.40000
3    3 65.94000
4    4 69.36000
```

Putem să includem de asemenea mai multe variabile independente în formula funcției `aggregate()`. De exemplu putem să calculăm greutatea medie a găinilor atât pentru fiecare tip de dietă cât și pentru numărul de săptămâni de la naștere:

```
aggregate(formula = weight ~ Diet + Time,  # DV este weight, IV sunt Diet și Time
          FUN = mean,               # calculeaza media pentru fiecare grup
          data = ChickWeight)       # dataframe este ChickWeight
```

- Considerați setul de date `mtcars`. Calculați:

a. Greutatea medie în funcție de tipul de transmisie.

b. Greutatea medie în funcție de numărul de cilindrii.

c. Consumul mediu în funcție de numărul de cilindrii și tipul de transmisie.