Laborator 02

Petculescu Mihai-Silviu

Laborator 02
- Petculescu Mihai-Silviu
  - Funcţii
  - Scalari şi Vectori
  - Operaţii cu vectori
  - Funcţii elementare
  - Alte funcţii utile în manipularea vectorilor
  - Metode de indexare a vectorilor
  - Operatori logici
  - Matrice

Funcţii

```
# Variabile
> a = 5
> b = "abc"
> c = 4L
> d = FALSE
> e = 1 + 4i
> y = 1:10

# Functii
# class() # Ce tip de obiect este
> class(a) # [1] "numeric"
> class(b) # [1] "character"
> class(c) # [1] "integer"
> class(d) # [1] "logical"
> class(e) # [1] "complex"
> class(y) # [1] "integer"

typeof() # Care este tipul de date al obiectului
> typeof(a) # [1] "double"
> typeof(b) # [1] "character"
> typeof(c) # [1] "integer"
> typeof(d) # [1] "logical"
> typeof(e) # [1] "complex"
> typeof(y) # [1] "integer"

length() # Care este lungimea obiectului
> y
[1]  1  2  3  4  5  6  7  8  9 10
> length(y)
[1] 10

as.numeric() # Conversie de tipuri
> z = as.numeric(y)
> z
[1]  1  2  3  4  5  6  7  8  9 10
```

```
> typeof(z)
[1] "double"

# attributes() - care sunt atributele obiectului (metadata)
> attributes(a) # NULL
> attributes(y) # NULL
```

Scalari şi Vectori

```
> vector() # vector logic gol
logical(0)
> character(5) # vector("character", length = 5)
[1] "" "" "" "" ""
> numeric(5) # vector("numeric", 5)
[1] 0 0 0 0 0
> logical(5) # vector("logical", 5)
[1] FALSE FALSE FALSE FALSE FALSE

# Functia c() de concatenare
> x = c(0.5, 0.6) # numeric
> x = c(1+0i, 2+4i) # complex
> x = c(TRUE, FALSE) # [sau] x = c(T, F) # logical
> x = c("a", "b", "c") # character
> x = 9:29 # integer

# Concatenare cu vectori
> z = c("Sandra", "Traian", "Ionel")
> z = c(z, "Ana")
> z = c("George", z)
> z
[1] "George" "Sandra" "Traian" "Ionel" "Ana"

# Functia de repetitie rep(0, 5)
> rep(0, 5)
[1] 0 0 0 0 0
> rep(c(1,2,3), 5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
#  |__1__|__2__|__3__|__4__|__5__|
> rep(c(1,2,3), each = 5)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
#  |____1____|____2____|____3____|

# Functia de generare a unui vector cu elemente egal departate unele de altele
seq()
> seq(1, 10, 1) # [sau] 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1, 10, length.out = 5) # selectam numarul elementelor
[1] 1.00 3.25 5.50 7.75 10.00
```

Operaţii cu vectori

```
> a = 1:4
> b = c(5,5,6,7)

> a + b # adunare vectori
[1] 6 7 9 11
```

```
> a + 10 # adunare scalari
[1] 11 12 13 14
> a - b # scadere vectori
[1] -4 -3 -3 -3
> a-15 # scadere cu scalari
[1] -14 -13 -12 -11
> a * b # inmultire vectori
[1] 5 10 18 28
> a * 3 # inmultire cu scalari
[1] 3 6 9 12
> a / b # impartire vectori
[1] 0.2000000 0.4000000 0.5000000 0.5714286
> a / 100 # impartire cu scalari
[1] 0.01 0.02 0.03 0.04
> a ^ b # ridicare la putere vectori
[1] 1 32 729 16384
> a ^ 7 # ridicarea la putere cu scalari
[1] 1 128 2187 16384
```

Funcţii elementare

```
# exp(), log(), [a]sin(), [a]cos(), [a]tan()
> z = c(10, 25, 100)
> sin(z)
[1] -0.5440211 -0.1323518 -0.5063656
> tan(z)
[1]  0.6483608 -0.1335264 -0.5872139
> atan(z)
[1] 1.471128 1.530818 1.560797
```

Alte funcţii utile în manipularea vectorilor

```
# min(), max(), sum(), mean(), sd(), length(), round(), ceiling(), floor(), %%
(modulo), %/% (div), table(), unique().
> length(z)
[1] 3
> y = c("M","F","M","M","F")
> unique(y)
[1] "M" "F"
> table(y)
y
F M
2 3
```

Metode de indexare a vectorilor

```
> x = seq(1, 10, length.out = 21) # vectorul initial
> x
[1]  1.00 1.45 1.90 2.35 2.80 3.25 3.70 4.15 4.60 5.05 5.50 5.95
[13] 6.40 6.85 7.30 7.75 8.20 8.65 9.10 9.55 10.00

> x[1] # accesam primul element
[1] 1
> x[4:10] # accesam toate elementele deintre pozitiile 4 si 9
```

```
[1] 2.35 2.80 3.25 3.70 4.15 4.60 5.05
> x[-5] # toate elementele mai putin cel de pe pozitia 5
...
> x[-(1:3)] # toate elementele mai putin primele 3
...
> x > 3 # vector logic care ne arata care elemente sunt >= decat 3
[1]  FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[13] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> x[x > 3] # elementele din x care sunt mai mari decat 3
[1]  3.25 3.70 4.15 4.60 5.05 5.50 5.95 6.40 6.85 7.30 7.75 8.20
[13] 8.65 9.10 9.55 10.00
```

Operatori logici

```
# Egal: ==
# Diferit: !=
# Mai mic [sau egal]: < [sau] <=
# Mai mare [sau egal]: > [sau] >=
# Sau : | (vectorial) [sau] || (scalare)
# Si: & (vecorial) [sau] && (scalare)
# Negare: !
# In multime: %in%
```

Matrice

```
cbind(a, b, c) # combina vectorii ca si coloane
rbind(a, b, c) # combina vectorii ca si linii
matrix(x, nrow, ncol, byrow) # creaza o matrice dintr-un vector x

> x = 1:5
> y = 6:10
> z = 11:15
> cbind(x, y, z)
     x  y  z
[1,] 1  6 11
[2,] 2  7 12
[3,] 3  8 13
[4,] 4  9 14
[5,] 5 10 15
> rbind(x, y, z)
  [,1] [,2] [,3] [,4] [,5]
x    1    2    3    4    5
y    6    7    8    9   10
z   11   12   13   14   15
> m = matrix(data = 1:10, nrow = 2, ncol = 5)
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> m = matrix(data = 1:10, nrow = 2, ncol = 5, byrow = TRUE)
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
> dim(m) # dimensiunile matricei [nrow(m) = 2] [ncol(m) = 5]
[1] 2 5
```

```
> tpm = t(m) # transpusa
> tpm
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> m %*% tpm # inmultirea matricelor
     [,1] [,2]
[1,]   55  130
[2,]  130  330
> m[1, ]
[1] 1 2 3 4 5
> m[ , 5]
[1] 5 10
```