1.4.3 Calculul cu vectori

1.4.3.1 Operaţii artimetice simple

Calculele pe vectori se efectuează pe fiecare componentă a vectorului în parte. Aplicarea de funcții simple asupra unu vector se face tot respectând același principiu.

Explicaţi exemplul de mai jos:

```
> x<-1:4
> y<-rep(2,4)
> x
[1] 1 2 3 4
> y
[1] 2 2 2 2
> x+y
[1] 3 4 5 6
> x*y
[1] 2 4 6 8
> sqrt(x)
[1] 1.000000 1.414214 1.732051 2.000000
> x^2
[1]  1  4  9 16
```

În cazul în care vectorii sunt de lungimi diferite se repetă valorile din vectorul mai "scurt" de atâtea ori până când ajunge la lungimea celui mai lung. Apoi se efectuează operațiile aritmetice de bază.

De exemplu:

```
> x<-1:3
> y<-rep(10,5)
> x
[1] 1 2 3
> y
[1] 10 10 10 10 10
> x+y
[1] 11 12 13 11 12
Warning message:
In x + y : longer object length is not a multiple of shorter object length
> y+x
[1] 11 12 13 11 12
Warning message:
In y + x : longer object length is not a multiple of shorter object length
```

Se observă faptul că valorile din vectorul x au fost repetate până când a ajuns la lungimea 5, apoi a fost efectuată operația de adunare componentă cu componentă. Evident ca adunarea este coutativă.

Exercițiu

1. Atribuiți în vectorul x, 5 valori, și în vectorul z 4 valori.

Calculați:

x*z; x+z; x^z

2. Atribuiți atâtea valori lui x și y astfel încât să nu obținem mesajul de avertizare la operațiile pe vectori.

1.4.3.2 Aplicarea de funcții pe vectori

`length()` – determină lungimea vectorului

`t()` – calculează vectorul transpus

sort() – sortează un vector crescător

sum() – însumează valorile unui vector

min() – valoarea minimă a vectorului

max() – valoarea maximă a vectorului

```
> x
[1] 2 3 4 5
> y
[1] 1 2 3 4 5 6 7 8
> length(x)
[1] 4
> length(y)
[1] 8
> t(x)
     [,1] [,2] [,3] [,4]
[1,]    2    3    4    5
> t(y)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    2    3    4    5    6    7    8
```

Observație: rezultatul transpunerii unui vector este o matrice. A se vedea secțiunea 1.4.4

```
> c<-c(2, -1,3.4,-34,11,45,-2.4)
> c
[1]   2.0  -1.0   3.4 -34.0  11.0  45.0  -2.4
> sort(c)
[1] -34.0  -2.4  -1.0   2.0   3.4  11.0  45.0
> sort (c, decreasing = TRUE)
[1]  45.0  11.0   3.4   2.0  -1.0  -2.4 -34.0


> sum(c)
[1] 24


> min(c)
```

```
[1] -34
> max(c)
[1] 45
```

1.4.3.3 Extragerea de subvectori

Pentru a extrage valorile unui subvector dintr-un vector dat se pot alege diverse modalităţi pentru a alege indecşii din vectorul curent

a. stabilirea unui interval

```
> c<-c(2, -1,3.4,-34,11,45,-2.4)
> c
[1]   2.0  -1.0   3.4 -34.0  11.0  45.0  -2.4
```

Pentru alegerea primelor 3 valori stabilim intervalul 1:3

```
> c[1:3]
[1]  2.0 -1.0  3.4
```

b. alegerea unor indecşi cu funcţia c()

```
> c[c(1,3,5)]
[1]  2.0  3.4 11.0
```

c. alegerea unor valori din vectori prin utilizarea unor expresii logice

```
> c[c>2]
[1]  3.4 11.0 45.0
```

Sau utilizând expresii compuse

```
> c[c>2 & c^2<100]
[1] 3.4
```

Operatorii logici care pot fi utilizați:

<	mai mic ca
>	mai mare ca
<=	mai mic sau egal ca
>=	mai mare sau egal ca

==	egal
!=	diferit
|	sau pe fiecare element
||	sau
!	not
&	şi pe fiecare element
&&	şi
xor(a,b)	sau exclusiv

Atenţie:

```
> c[c>2 && c^2<100]
numeric(0)
```

Există și funcția subset() care în principiu face același lucru ea putând fi utilizată și la alte tipuri de obiecte.

```
> subset(c, c>2 & c^2<100)
[1] 3.4
```

1.4.3.4 *Vectori nenumerici*

În afară de vectorii numerici există şi alte tipuri de vectori cum ar fi vectorii de tip string şi vectori care conţin expresii logice.

Aceşti vectori se declară şi se accesează exact ca si vectorii numerici:

```
> mama<-c("Cretulescu", "Elena")
> tata<-c("cretulescu","vasile", "gheorghe")
> mama
[1] "Cretulescu" "Elena"
> tata
[1] "cretulescu" "vasile"     "gheorghe"
```

```
> L = c(FALSE, TRUE, FALSE, TRUE, FALSE)
```

Dacă dorim să alegem acum valori din vectorul tata bazându-ne pe vectorul de valori logice L

```
> L=c(TRUE,FALSE,TRUE)
> tata
[1] "cretulescu" "vasile"     "gheorghe"
> tata[L]
[1] "cretulescu" "gheorghe"
```

Denumirea valorilor vecorului;

```
> names(tata)<-c("Nume","Prenume0","Prenume 1")
> tata
        Nume     Prenume0    Prenume 1
"cretulescu"     "vasile"   "gheorghe"
```

Evident acum putem accesa valoarea vectorului prin numele cheii

```
> tata["Nume"]
        Nume
"cretulescu"
```

1.4.4 Array-uri

Array-urile sunt vectori cu dimensiune dată. Atribuirea valorilor într-un array se face apelând funcţia `array(data=, dim=)`

```
> s<-array(c(1:24),24)
> s
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24
```

Exerciţiu:

Generaţi doua array-uri de dimensiune 10 şi 12 şi adunaţi aceste două array-uri.

1.4.5 Matrici

Sinatxa:

matrix(data, ncol=, byrow=FALSE)

Exerciţii:

1. Să se creeze următorii vectori:

a.) $s = (0.1^3 \cdot 0.2^1, 0.1^6 \cdot 0.2^4, ..., 0.1^{36} \cdot 0.2^{34})$

b.) $m = \left(2, \frac{2^2}{2}, \frac{2^3}{3}, ..., \frac{2^{25}}{25}\right)$

2. Să se calculeze:

a.) $\sum_{i=10}^{100}\left(i^3+4i^2\right)$

b.) $\sum_{i=1}^{25}\left(\frac{2^i}{i}+\frac{3^i}{i^2}\right)$