R – na potrzeby laboratoriów MWAD

Krzysztof Jurczuk, ostatnia aktualizacja 2006-01-17

SPIS TREŚCI

I. Wektory

- tworzenie wektora

przykłady:

```
> vector1 = c(1, 2, 3)
> vector1
[1] 1 2 3
> vector2 <- c(1, 2, 3)
> vector 2
[1] 1 2 3
> vector3 <- c("1", "2", "3")
> vector3
[1] "1"  "2"  "3"
> vector4 = 1:20
> vector4
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

- operacje na wektorach

- max (maksymalna wartość)
- min (minimalna wartość)
- mean (wartość średnia)
- median (mediana)
- mad (medianowe odchylenie bezwzględne)
- quantile (dowolny kwanty)
- sd (odchylenie standardowe)
- var (wariancja)
- length (długość wektora)
- sum (suma elementów)
- prod (iloczyn elementów)
- sort (zwraca wektor posortowany rosnąco)

- which (zwraca wektor indeksów, przy których argument miał wartość TRUE)
- diff (zwraca wektor zawierający różnice pomiędzy sąsiadującymi elementami)

przykłady:

```
> quantile(vector1, 0.5)# kwantyl 0.5, czyli mediana
> sum( vector1>3 )     # ile jest elementów większych od  3
> sum( a[a>3] )           #suma elementów większych od 3
> which( a>3 )        # indeksy elementów większych od 3
```

II. Macierze

- tworzenie macierzy

przykłady:

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

> y <- array( c(1:3, 3:1), dim=c(3,2) )
> y
     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1

> tab1 = array(1, c(4,4))
> tab1
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    1    1    1
[3,]    1    1    1    1
[4,]    1    1    1    1
```

- operacje na macierzach

przykłady:

```
> x[y]                    #elementy macierzy y są indeksami
[1] 9 6 3
> x[y] <- -1
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5   -1   13   17
[2,]    2   -1   10   14   18
[3,]   -1    7   11   15   19
[4,]    4    8   12   16   20

> x[3,3]
[1] 11
> x[,3]                         #3 kolumna
[1] -1 10 11 12
```

```
> x[3,]                          #3 rząd
[1] -1  7 11 15 19

> matrix1 = array( 1:16, c(4,4))
> matrix2 = array( 2, c(4,4))
> matrix1
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> matrix2
     [,1] [,2] [,3] [,4]
[1,]    2    2    2    2
[2,]    2    2    2    2
[3,]    2    2    2    2
[4,]    2    2    2    2
> matrix1 * matrix2        # wymnożenie elementów o takich
                           # samych indeksach
     [,1] [,2] [,3] [,4]
[1,]    2   10   18   26
[2,]    4   12   20   28
[3,]    6   14   22   30
[4,]    8   16   24   32
> matrix1 %*% matrix2    #mnozenie macierzy
     [,1] [,2] [,3] [,4]
[1,]   56   56   56   56
[2,]   64   64   64   64
[3,]   72   72   72   72
[4,]   80   80   80   80
```

III. Pętle

- for

for(***name*** **in** *expr_1***)** *expr_2*

name – zmienna pętli

expr_1 - zakres pętli

expr_2 - powtarzane wyrażenie

przykład:

```
> tab1 <- 1:5
> for( i in 1:5 ) print( tab1[i] )
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

- while

while(*condition* **)** *expr*

```
> tab1 <- 1:5
> i = 1
> while ( i<3 ) { print(tab1[i]); i=i+1; }
[1] 1
```

```
[1] 2
```

Przy obu tych pętli iteracje mogą być przerwane przez użycie słowa **break.**

IV. Tworzenie własnych funkcji

name <- **function(** *arg_1, arg_2, arg_3, ...***)** *expr*

przyklady:

```
> matrix1 <- array(1:16, c(4,4))
> matrix2 <- array(16:1, c(4,4))
> f_corr <- function ( mat1, mat2 ) { mat3 = mat1 %*% mat2;
+ cor( mat3 );
+ }
> f_corr                  #wyświetlenie treści funkcji
function ( mat1, mat2 ) { mat3 = mat1 %*% mat2;
cor( mat3 );              #liczy macierz korelacji
}
> f_corr( matrix1, matrix2 )  #wywołanie funkcji
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    1    1    1
[3,]    1    1    1    1
[4,]    1    1    1    1
> f_covv <- function ( mat1, mat2 ) { mat3 = mat1 %*% mat2;
+ cov( mat3 );               #liczy macierz kowariancji
+ }
> f_covv
function ( mat1, mat2 ) { mat3 = mat1 %*% mat2;
cov( mat3 );
}
> f_covv( matrix1, matrix2 )
          [,1] [,2]      [,3]     [,4]
[1,] 5606.6667 4060 2513.3333 966.6667
[2,] 4060.0000 2940 1820.0000 700.0000
[3,] 2513.3333 1820 1126.6667 433.3333
[4,]  966.6667  700  433.3333 166.6667
```

V. Instrukcje warunkowe

if(condition) expr else expr

przyklady:

```
> if( 4 < 5 ) print("4<5") else print("4>5");
[1] "4<5"
```

VI. Operatory

- - (operator odejmowania)
- + (operator dodania)
- * (operator mnożenia)
- / (operator dzielenia rzeczywistego)
- %/% (operator dzielenia całkowitego)
- %*% (operator mnożenia macierzy)
- ^ (operator potęgowy)
- < (operator mniejszości)
- > (operator większości)
- == (operator równośći)
- >= (operator większości-równości)
- <= (operator mniejszości-równości)
- && (operator logiczny i)
- || (operator logiczny lub)