Course: R Language in Computational Probability and Statistics

Lecture 2: Vectors. Vector functions.

Arithmetic operators and special values.

Lecturer: Nataliia Kruhlova

Типи даних

В мові програмування R найпростішою структурною одиницею є вектор. Скаляри вважаються векторами з одним елементом. Існують п'ять основних типів векторів, рис. 1:

Рис. 1. Типи векторів в R, https://www.canva.com/ [1].

Ці типи векторів використовуються для представлення різних видів даних. Наприклад, логічні вектори можна використовувати для булевих операцій, числові - для числових обчислень, цілі - для індексування, комплексні - для роботи з комплексними числами, а символьні - для роботи з текстовими даними.

Арифметичні операції.

У таблиці 1 наведено основні арифметичні операції над векторами.

Табл. 1

Арифметична операція	Оператор
Додавання	**+**
Віднімання	**-**
Множення	*****
Ділення	**/**
Піднесення до степеня	^

Приклади.

```
> 3*2
[1] 6
>1/4
[1] 0.25
> 4^3
[1] 64
```

Функції.

Деякі функції, що реалізовані в R:

sqrt()- корінь квадратний,

abs() - модуль,

sin(), cos(), tan(), asin(x), acos(x), atan(x), atan2(x) - тригонометричні функції,

pi #π = 3.1415926...

exp () – експоненціальна функція,

log(), log10(), **log2() –** логарифмічні функції,

factorial () – факторіал,

choose () - число комбінацій,

ceiling () єдиний числовий аргумент x округляється до найменшого цілого, не менше x.

floor () єдиний числовий аргумент x округляється до найбільшого цілого, не більше x.

round (x, digits = k) аргумент x округляється з точністю до k знаків після коми.

max(x), min(x) – пошук максимального/мінімального елемента.

sum(x) - сума елементів вектора.

prod(x) – добуток елементів вектора.

mean(x) – середнє арифметичне елементів вектора.

Приклади.

```
>sqrt(16)
[1] 4
>factorial(5)# 5!
[1] 120
>choose(4,3)# C_4^3
[1] 4
>round(15.845456244,digits=2)
[1] 15.85
>floor(13.74545462)
[1] 13
>ceiling(14.84545462)
[1] 15
```

Числовий вектор можна створити, якщо просто перерахувати його елементи:

```
> x<-c(2,7,9,11)
> x
[1]  2  7  9 11
```

Генерування послідовностей.

Таким чином задається послідовність впорядкованих цілих чисел від меншого до більшого з кроком 1:

```
> x<-c(1:6)
> x
[1] 1 2 3 4 5 6
```

Можна вектор задати і в зворотному порядку:

```
> y<-c(7:1)
> y
[1] 7 6 5 4 3 2 1
```

Функція **seq** дозволяє генерувати послідовності, вказавши мінімальне, максимальне значення, крок послідовності або кількість членів послідовності.

Приклади.

```
> seq(-2,4,by=0.5)
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0
> seq(-3,5,length.out = 8)
[1] -3.0000000 -1.8571429 -0.7142857  0.4285714  1.5714286  2.7142857
3.8571429
[8]  5.000000
```

Функція **rep()** повторює елемент послідовності необхідну кількість разів:

```
> x<-c(-1, 7,9)
> rep(x,3)
[1] -1  7  9 -1  7  9 -1  7  9
```

Функція **paste()** об'єднує символи в єдину послідовність символів.

Приклад.

```
> sequence<-paste("A", "B", 2:-1, sep="|")
> sequence
[1] "A|B|2"  "A|B|1"  "A|B|0"  "A|B|-1"
```

Вектори логічного типу

Компоненти логічного вектора приймають два значення: **TRUE** і **FALSE**. Логічні вектори можна створити або перерахуванням елементів, або за допомогою умов.

Приклад.

```
> x<-c(F,T,T,F)
> x
[1] FALSE  TRUE  TRUE FALSE
> y<-(7>9)
> y
[1] FALSE
```

Логічні операції

Логічні операції: <, <=, >, >=, == (знак рівності), ! = (нерівності).
Якщо **cl** і **c2** є логічними виразами, то **cl & c2** означає їх перетин («і»), а **cl | c2** означає їх об'єднання («або»), !**c1** означає заперечення **cl**, **c1==c2** - співпадіння об'єктів.
Знак пропущених значень даних - **NA**.
NaN *(Not a Number)* означає, що вираз не є числом.
Ділення будь-якого числа на 0 дає **Inf** *(infinity)*.

```
> 5/0
[1] Inf
> 0/0
[1] NaN
```

Дії над векторами

Операції над векторами виконуються так само, як із числами. Якщо вектори мають різну довжину, то коротший вектор повторюють циклічно потрібну кількість разів. Проте система видає повідомлення, що об'єкти мають різну довжину.

Приклади.

```
> x<-c(1:4)
> y<-(-3:1)
> x+y
[1] -2  0  2  4  2
> z<-c(1,2,0,-5)
> x/z
[1]  1.0  1.0  Inf -0.8
> tan(x+z)
[1] -2.1850399  1.1578213 -0.1425465 -1.5574077
```

Інші поширені функції векторів:
length () - довжина вектора;
sort () - сортування компонент вектора.
diff () - різниці сусідніх компонент вектора;
var() - дисперсія компонент вектора;
sd() - стандартне відхилення;
range () - діапазон від min до max;

pmax () - вектор покомпонентного максимуму аргументів;
pmin () - вектор покомпонентного мінімуму аргументів;
order() - сортування елементів вектора, але виводяться номери елементів вектора у порядку зростання елементів.

Приклади.

```
> pmin(1:5,c(3,4,0,-2,1))
[1]  1  2  0 -2  1
> length(-5:5)
[1] 11
> sum(-3:3)
[1] 0
> prod(2^x)
[1] 1024
> z<-c(1,2,0,-5)
> order(z)
[1] 4 3 1 2
> sort(z)
[1] -5  0  1  2
```

Генерування випадкових послідовностей

sample – генерування вибірки.

Синтаксис функції:

sample(x, size, replace = TRUE / FALSE, prob = NULL)

Значення аргументів:

x - числовий, символьний або логічний вектор довжини більше одиниці,
size – додатне ціле число, об'єм вибірки,
replace – якщо TRUE - вибір з поверненням, якщо FALSE - вибір без повернення. Для вибірки з нерівними ймовірностями елементів аргумент **prob** - числовий вектор зі значеннями відповідних ймовірностей. Якщо **prob=NULL**, рівноймовірна вибірка (сумарна ймовірність може бути більше 1).

Приклад.

```
> x<-c(0,1)
> sample(x,size=5,replace=T,prob=c(2/3,1/2))
[1] 0 1 0 1 0

> sample(x,size=10,replace=T)
 [1] 0 1 0 0 1 0 1 0 0 0
```

Іменовані вектори

Елементи векторів можуть мати імена у мові R. Це називається іменованим вектором. Для того щоб створити іменований вектор, потрібно встановити атрибут **names**, який відповідає за присвоєння імен кожному елементу вектора.

Приклад.

```
.> x<-c(97,86,77,62)
> names(x)<-c("відмінно", "дуже добре", "добре", "задовільно")
> x
    відмінно дуже добре      добре задовільно
[1]       97        86        77        62
```

Виклик функції зазвичай у мові R повинен бути записаний праворуч від знаку привласнення <-. Хоча іноді дозволено використання виклику функції зліва від <-, але це вважається нестандартним і може спричинити помилку в більшості випадків.

Індексація векторів

У мові R існує гнучка система індексації, яка дозволяє обирати для обробки даних конкретні частини векторів (матриць, багатовимірних масивів). В цій лекції ми розглянемо приклади застосування індексації для векторів. Доступ до і-го елемента вектора можна отримати, використовуючи квадратні дужки.

Приклад.

```
> x<-c(97,86,77,62)
> names(x)<-c("відмінно", "дуже добре", "добре", "достатньо")
> x
відмінно дуже добре      добре достатньо

      97        86        77        62
> x[3]
добре
  77
```

У векторів нумерація елементів завжди починається з 1. Якщо елемент вектора має ім'я, то його можна викликати, безпосередньо вказавши ім’я елемента.

```
> x["відмінно"]
відмінно
      97
```

Якщо використати прямі дужки з вектором індексів всередині, то можна отримати підвектор, що складається з відповідних елементів у вказаному порядку:

```
> x[c(4,2)]
 достатньо дуже добре
       62        86
```

Якщо у квадратних дужках вказати від'ємні значення індексів, то відповідні елементи будуть видалені із вектора.

```
> x[-c(1,3)]
дуже добре  достатньо
       86        62
```

Для індексації також застостовуються логічні вектори, тоді вибиратись будуть лише елементи, яким відповідає значення **T (TRUE)**:

```
> x[c(T,F,F,T)]
 відмінно достатньо
       97        62
```

У квадратних дужках можна використовувати будь-який вираз, результат якого буде застосовано для індексації.

```
> x[x>85]
  відмінно дуже добре
       97        86
```

Приклад. Потрібно переназвати елементи вектору x<-c(97,86,77,62), які вище 85 балів, на «підходить», а інші- «перездати».

```
> names(x)[x<=85]<-"перездати"
> names(x)[x>85]<-"підходить"
> x
підходить підходить перездати перездати
       97        86        77        62
```

Список джерел

1) Canva. https://www.canva.com/