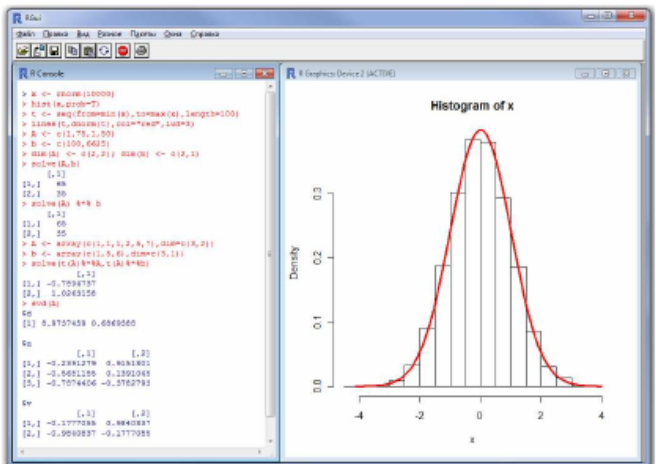
МУЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ

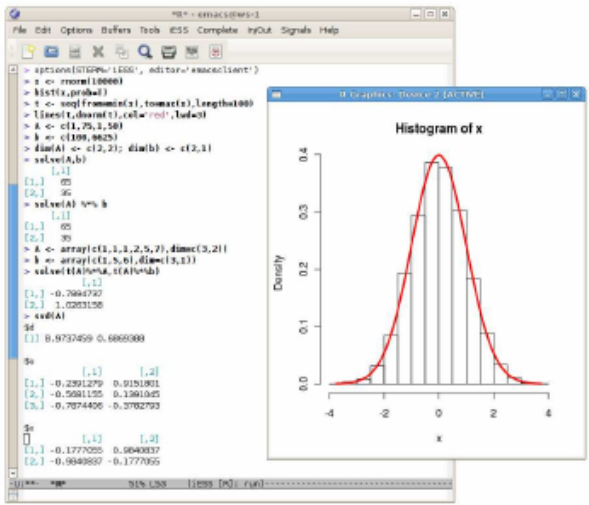
Лекція 12

Мультипарадигменна мова ймовірнісного програмування R

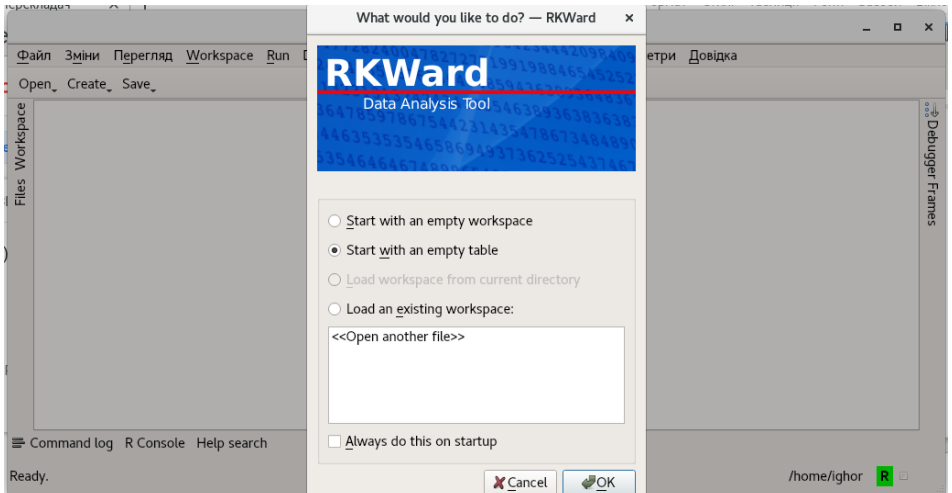
R - реалізація мови програмування **S**, призначеної для виконання статистичних обчислень і, при необхідності, графічного відображення результату. **R** синтаксично і функціонально дуже схожий на комерційний статистичний пакет **S-Plus**, хоча і не ідентичний йому.



(А) Графічний інтерфейс **R (Rgui)**



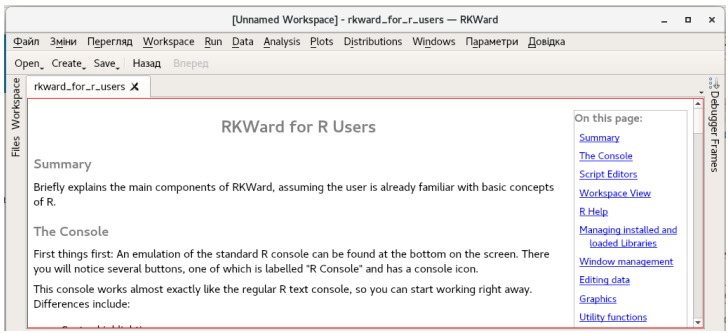
(В) **R** в **Emacs** (розширення **ESS**)


What would you like to do? — RKWard
RKWard
Data Analysis Tool
Start with an empty workspace
Start with an empty table
Load workspace from current directory
Load an existing workspace:
<<Open another file>>
Always do this on startup
Cancel
OK
Файл Зміни Перегляд Workspace Run
Open Create Save
Files Workspace
Debugger Frames
Command log R Console Help search
Ready.
/home/ighor

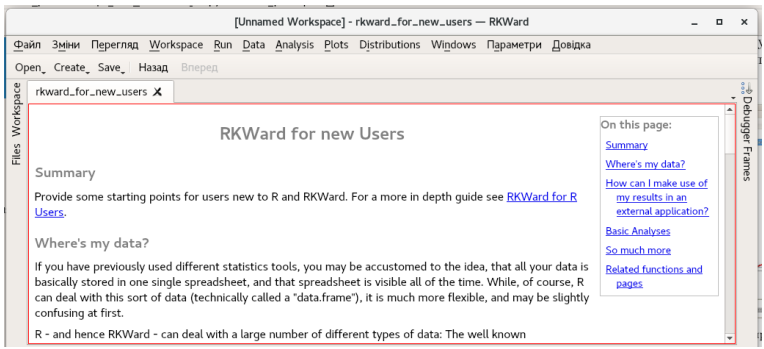
(C) R в RKWard

RKWard - це інтерфейс графічного інтерфейсу та **IDE** до **R**, потужна мова сценаріїв для статистичних обчислень. Він спрямований на надання корисних функцій як досвідченим користувачам **R**, які бажають скористатися можливостями сценаріями **R**, так і новим користувачам **R**, шукаючим простий спосіб виконання завдань статистичного обчислення.

Через великі відмінності в попередніх знаннях **R**, ми пропонуємо два різні посібники для початку роботи з **RKWard**. Перший орієнтований на користувачів, які володіють певними знаннями мови **R**, і зосереджений на впровадженні функцій **IDE RKWard**: *RKWard для користувачів R*.



Другий дає більше можливостей для ознайомлення з початком роботи з **RKWard**, але не так глибоко. Якщо ви ніколи раніше не користувались **R** або **RKWard**, вам слід розпочати з цього посібника, а потім прочитати посилання на сторінку вище: *RKWard для нових користувачів*.



Зазвичай прості обчислення в **R** виконуються інтерактивно, з використанням інтерфейсу командного рядка. Якщо ж потрібно виконати послідовність команд (сценарій), то її можна зберегти в окремому файлі, зазвичай має розширення **R**.

Команда **source** використовується для завантаження і виконання такого файлу:

```
source (filename)
```

Параметр **echo** дозволяє збільшити обсяг виведеної інформації:

```
source (filename, echo = TRUE)
```

Якщо не вказано шлях до файлу сценарію, то за замовчуванням використовується поточний каталог.

Команди **getwd** і **setwd** відповідно показують і змінюють робочий каталог:

```
getwd ()
setwd ("d: /")
```

Для завантаження в робочу пам'ять додаткових бібліотек використовується команда **library**

```
library:
library (lib _ name)
```

ПРОСТІ ОБЧИСЛЕННЯ

Нескладно обчислити значення скалярного виразу:

$$1 + 2 - \frac{3 \times 4}{5^6}$$

```
1 + 2 -3 * 4/5 ^ 6
```

Так само просто можна працювати в **R** з векторними величинами. Для створення вектора можна використовувати функцію конкатенації **c**, що створює вектор-стовпець, або функцію **seq**.

Команда-стрілка **->** або **<-** застосовується для присвоювання значення змінної (в більшості випадків також можна використовувати **=**).

```
x <- c (1, 2, 3, 4, 5)
c (1, 2, 3, 4, 5) -> x
x <- 1: 5
x <- seq (from = 1, to = 5, by = 1)# Вектор чисел від 1 до 5 з кроком 1
x <- seq (from = 1, length = 5)# Вектор чисел від 1 довжини 5 (за замовчуванням крок дорівнює 1)
```

Вектор з декількох однакових елементів можна отримати за допомогою функції **rep**:

```
x <- c(rep(0,5),rep(1,5))  # x = (0,...,0,1,...,1)
```

$x = (\underbrace{0,\ldots,0}_{5},\underbrace{1,\ldots,1}_{5})$

До вектору можна застосовувати ті ж операції, що і до скалярної величини:

```
x <- 1:5
sin(x); x^2
```

```
[1]  0.8414710  0.9092974  0.1411200 -0.7568025 -0.9589243
[1]  1  4  9 16 25
```

Операції додавання і множення

```
x <- 1:10
sum(x); prod(x)                    # ∑_{i=1}^n x_i , ∏_{i=1}^n x_i


[1] 55
[1] 3628800
```

можна використовувати, наприклад, для обчислення вибіркових оцінок математичного очікування та дисперсії (функція **`length`** повертає кількість елементів в об'єкті):

```
m <- sum(x)/length(x)              # x̄ = (1/n) ∑_{i=1}^n x_i
s2 <- sum((x-m)^2)/(length(x)-1)   # s^2 = (1/(n-1)) ∑_{i=1}^n (x_i - x̄)^2
m; s2
[1] 5.5
[1] 9.166667
```

Звичайно, для обчислення цих значень можна просто використовувати вбудовані функції:

```
mean (x) # середнє арифметичне
var (x)  # вибіркова дисперсія
```

Якщо потрібно виконати обчислення з комплексними числами, то для завдання уявної частини комплексного числа використовується буква **i**:

```
1+2i              # 1 + 2i
1i^2              # i^2 = -1
(-1+0i)^0.5       # √-1 = i
asin(2+0i)
```

```
[1] 1+2i
[1] -1+0i
[1] 0+1i
[1] 1.570796-1.316958i
```

Нехай елементи вектору **x** відповідають різним множинам, причому для класифікації використовується вектор **y**. Функція **tapply (x, y, f)** дозволяє застосувати до елементів **x** операцію **f** відповідно до класифікації **y**:

```
x <- rnorm(500)                          # x_i ~ N(0,1)
y <- as.factor(rep(1:5,100))             # y ∈ {1,2,3,4,5} - качественная (номинальная) переменная
tapply(x,y,mean); tapply(x,y,sd)         # средние значения и стандартные отклонения для каждого класса
```

```
          1           2           3           4           5
-0.03774697  0.15088773 -0.01903427  0.08695536 -0.07665831
          1           2           3           4           5
0.9214526   0.9556831   0.9316465   0.9402901   1.0715009
```

Для виведення об'єктів на екран можна використовувати команду **print**:

```
for (i in 1: 5) print (i ^ 2)
```

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

а для створення форматованих рядків - **sprintf**:

```
for (i in 1: 5) print (sprintf ("sin (% d ^
2) ␣ = ␣% 6.3 f", i, sin (i ^ 2)))
```

```
[1] "sin(1^2) =  0.841"
[1] "sin(2^2) = -0.757"
[1] "sin(3^2) =  0.412"
[1] "sin(4^2) = -0.288"
[1] "sin(5^2) = -0.132"
```

Команда **format** перетворює об'єкт в рядок, використовуючи заданий формат.

РОЗПОДІЛ ВИПАДКОВИХ ЧИСЕЛ

В **R** для роботи з розподілами випадкових чисел використовуються складові імена команд, що складаються з префікса, що визначає що буде результатом операції, і назви розподілу. Наприклад, для нормального розподілу **(norm)** команда **rnorm (n)** генерує **n** нормально розподілених випадкових чисел, **dnorm (x)** повертає щільність ймовірності в точці **x**, **pnorm (x)** - значення функції розподілу ймовірностей в точці **x**, а **qnorm (α)** - квантиль (значення зворотної функції розподілу) для ймовірності **α**.

```
rnorm(1000)              # 1000 случайных чисел со стандартным нормальным распределением N(0,1)
rnorm(1000,mean=1,sd=2)  # N(1,2^2)
dnorm(2)                 # плотность вероятности f(2) для N(0,1): (2π)^{-1/2} exp(-t^2/2) |_{t=2}
pnorm(2)                 # значение функции распределения F(2): ∫_{-∞}^{x} f(t) dt |_{x=2}
qnorm(0.2)               # значение обратной функции распределения F^{-1}(0.2): F(x) = 0.2
```

Додаткові параметри дозволяють більш точно задати вид розподілу.

Распределение	Название в R	Дополнительные аргументы
Биномиальное	binom	size, prob
Коши	cauchy	location $= 0$, scale $= 1$
Логистическое	logis	location $= 0$, scale $= 1$
Логнормальное	lnorm	meanlog $= 0$, sdlog $= 1$
Нормальное	norm	mean $= 0$, sd $= 1$
Пуассона	pois	lambda
Равномерное	unif	min $= 0$, max $= 1$
Стьюдента	t	df
Фишера	f	df1, df2
Хи-квадрат	chisq	df
Экспоненциальное	exp	rate $= 1$

ПРОСТІ ОПИСОВІ СТАТИСТИКИ

Вхідні дані:

```
x <- runif(100)      # вектор (неотрицательных) чисел
n <- length(x)       # длина вектора
w <- x/sum(x)        # вектор весов: ∑_{i=1}^{n} w_i = 1
```

Середні значення.
Середнє арифметичне:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

```
sum ( x ) / n
xm <- mean ( x );xm
```

Зважене середнє:

$$\bar{x}_w = \sum_{i=1}^{n} w_i x_i.$$

```
sum ( w * x )
weighted . mean (x , w )
```

Геометричне середнє:

$$g_x = \sqrt[n]{\prod_{i=1}^{n} x_i} = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\ln(x_i)\right).$$

```
( prod ( x ))^(1 / n )
exp (1 / n * sum ( log ( x )))
```

Гармонійне середнє:

$$h_x = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

```
n / sum (1 / x )
```

Медіана ($\tilde{\mathbf{x}}$ - відсортована по зростанню вибірка $\mathbf{x}$):

$$\begin{cases} \tilde{x}_{\lceil n/2 \rceil}, & n \text{ — нечётное}, \\ \frac{1}{2}\left(\tilde{x}_{n/2} + \tilde{x}_{n/2+1}\right), & n \text{ — чётное}. \end{cases}$$

```
X <- sort ( x )
if (n %% 2) X [ceiling (n / 2)]median ( x )
else ( X [ n / 2]+ X [ n / 2+1]) / 2
```

Міри розкиду

Вибіркова дисперсія:

$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

```
sum (( x - xm )^2) / (n -1)
s2 <- var ( x ); s2
cov (x , x )
```

Стандартне відхилення:

$$s_x = \sqrt{s_x^2}.$$

```
s2 ^0.5
s <- sd ( x ); s
```

Коефіцієнт варіації (s_x / $\bar{X}$):

```
s / xm
```

Розмах варіації:

$R = x_{max} - x_{min}$.

```
R <- max ( x ) - min ( x ); R
```

Відносний розмах варіації (R/$\bar{X}$):

```
R / xm
```

Середнє лінійне відхилення:

$$a = \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{x}|.$$

```
a <- sum ( abs (x - xm )) / n ; a
```

Відносне відхилення за модулем **(a/X̄):**

```
a / xm
```

Середньо квартильне відхилення:

$$q = \frac{Q_3 - Q_1}{2}$$

```
q <- (quantile (x,0.75)-quantile (x,0.25))/2
```

Відносне квартильне відхилення **(q/X̄):**

```
q / xm
```

ІНТЕРВАЛЬНІ ОЦІНКИ

Нехай $x \sim N(\mu, \sigma^2)$; $\bar{X} = xm$, $s^2_x = s2$ і $s_x = s$ – вибіркові оцінки математичного очікування, дисперсії і стандартного відхилення. Тоді для заданого рівня значущості α (ймовірність помилки першого роду):

$$\bar{x} + t_{\alpha/2,n-1}\frac{s_x}{\sqrt{n}} \leqslant \mu \leqslant \bar{x} + t_{1-\alpha/2,n-1}\frac{s_x}{\sqrt{n}}$$

```
alpha <- 0.05
c ( xm + qt ( alpha / 2 , df =n -1) * s / n
^0.5 , xm + qt (1 - alpha / 2 , df =n -1) * s
/ n ^0.5)
```

$$\frac{(n-1)s_x^2}{\chi^2_{1-\alpha/2,n-1}} \leqslant \sigma^2 \leqslant \frac{(n-1)s_x^2}{\chi^2_{\alpha/2,n-1}}$$

```
c (( n -1) * s2 / qchisq (1 - alpha / 2 , df
=n -1) , (n -1) * s2 / qchisq ( alpha / 2 ,
df =n -1))
```

МАТРИЦІ В МОВІ R

Якщо потрібно створити матрицю, то можна ввести список її елементів (по стовпцях) і задати розмірність за допомогою команди **dim:**

```
A <- c(1,75,1,50); dim(A) <- c(2,2)   # (1  1 / 75  50)
b <- c(100,6625);  dim(b) <- c(2,1)   # (100 / 6625)
```

Команди **matrix** або **array** дозволяють одночасно ввести дані і задати розмірність:

```
A <- matrix (c (1,75,1,50), nrow =2, ncol =2)
A <- array (c (1,75,1,50) , dim = c (2,2)); A
```

```
     [,1] [,2]
[1,]    1    1
[2,]   75   50
```

Зауважимо, що масив **(array)** може мати більше двох розмірностей:

```
array (1: 8, dim = c (2, 2, 2))
```

```
, , 1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
, , 2
     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Обчислення зворотної матриці здійснюється за допомогою команди **solve**. Використання цієї ж команди з двома аргументами дозволяє вирішити систему лінійних алгебраїчних рівнянь. Для множення матриць застосовується оператор **%*%**:

```
solve (A)%*% b
solve (A, b)
```

```
     [,1]
[1,]   65
[2,]   35
```

Для обчислення псевдооберненої матриці Мура-Пенроуза можна використовувати команду **ginv** з бібліотеки **MASS**:

```
A <- matrix (c(1,1,1,1,2,3),nrow=3,ncol=2); A
```

```
     [,1] [,2]
[1,]    1    1
[2,]    1    2
[3,]    1    3
```

```
MASS :: ginv ( A )
```

```
             [,1]          [,2]       [,3]
[1,]  1.333333  3.333333e-01 -0.6666667
[2,] -0.500000 -2.775558e-17  0.5000000
```

Оператор **`*`** виконує поелементне множення. Для зовнішнього множення використовується оператор **`%o%`**: якщо $x = (x_1, \ldots, X_m)$, $y = (y_1, \ldots, Y_n)$, то z – $(m \times n)$ – матриця вигляду

$x \otimes y = z$, $z_{ij} = x_i y_j$, $i = 1,2,\ldots,m$, $j = 1,2,\ldots,n$.

Нехай

```
x <- 1:3
y <- seq ( from =1 , by =3 , length =3)
x; y
```

```
[1] 1 2 3
[1] 1 4 7
```

Поелементне, скалярне та зовнішнє добутки:

```
x*y; x%*%y;  x%o%y
```

```
[1]  1  8 21
     [,1]
[1,]   30
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    8   14
[3,]    3   12   21
```

Власні значення і власні вектори симетричною матриці **S** обчислюються за допомогою функції eigen, що повертає список з двох компонент - **values** і **vectors:**

```
A <- array (runif (9), dim = c (3, 3)); A
# Матриця випадкових чисел, A_i, j ~ U (0, 1)
```

```
          [,1]      [,2]      [,3]
[1,] 0.90492571 0.3020807 0.9157105
[2,] 0.05258745 0.1213333 0.1022146
[3,] 0.72055611 0.8529154 0.4392352
```

```
S <- t(A)%*% A; S
# Симетрична матрица S = A^TA
```

```
          [,1]      [,2]     [,3]
[1,] 1.3408571 0.8943146 1.150519
[2,] 0.8943146 0.8334392 0.663651
[3,] 1.1505188 0.6636510 1.041901
```

eigen (S)

```
$values
[1] 2.937755824 0.274778155 0.003663353
$vectors
           [,1]       [,2]       [,3]
[1,] 0.6739616  0.1282278  0.7275530
[2,] 0.4669689 -0.8370730 -0.2850419
[3,] 0.5724647  0.5318519 -0.6240335
```

Якщо матриця більша, а потрібні тільки значення власних чисел, то можна придушити обчислення векторів за допомогою команди

```
eigen (S, only.values = TRUE) $ values
```

Списки

Список - об'єкт, що містить впорядкований набір компонентів.

```
Lst <- list (name = "John", wife = "Joan",
child. no = 3, child. ages = c (2, 5, 9))
```

Вибірка даних зі списку:

```
Lst[[1]]                    # "John"
Lst[[2]]                    # "Joan"
Lst[[3]]                    # 3
Lst[[4]]                    # 2, 5, 9
Lst[[4]][2]                 # 5
Lst[["name"]]               # "John"
x <- "wife"; Lst[[x]]       # "Joan"
```

Фрейми даних

Фрейм даних - це список класу **`data.frame`**. обмеження:

- компоненти повинні бути векторами (числовими, символьними або логічними), якісними значеннями, числовими матрицями, списками або іншими фреймами даних;
- матриці, списки і фрейми даних надають стільки змінних в новий фрейм, скільки вони мають стовпців, елементів і змінних відповідно;
- числові, якісні та логічні вектори включаються як є; символьні вектори перетворюються в якісні, причому число значень відповідає числу унікальних значень у вихідному векторі;

• вектори, що подаються змінними фрейма, повинні мати однакову довжину, матриці повинні мати однаковий розмір рядків.

Для створення фрейма даних використовується команда **data.frame**:

```
data <-data.frame (name = c ("John","Joan", "Jack","Jane"),
                   sex = c ( " m " ," f " ," m " , " f " ) ,
                   income = c (15000 ,10000 ,12000 ,17000))
data
```

```
  name sex income
1 John   m  15000
2 Joan   f  10000
3 Jack   m  12000
4 Jane   f  17000
```

```
summary ( data )
```

```
  name    sex       income
Jack:1    f:2   Min.   :10000
Jane:1    m:2   1st Qu.:11500
Joan:1          Median :13500
John:1          Mean   :13500
                3rd Qu.:15500
                Max.   :17000
```

Якщо список задовольняє обмеженням, він може бути перетворений у фрейм даних за допомогою команди **`as.data.frame.`**

Команда **`read.table`** завантажує дані з файлу і створює фрейм даних.

ВІЗУАЛІЗАЦІЯ ДАНИХ

2D

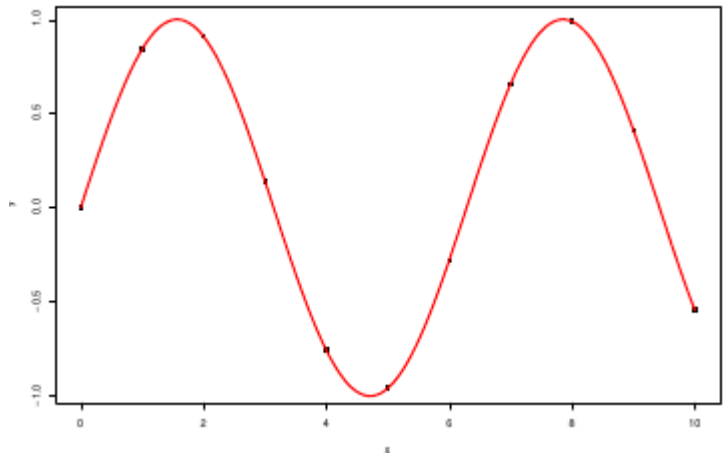
plot - узагальнена функція для малювання об'єктів **R**.

Вихідні дані і графік (за замовчуванням - точки, опція **pch** визначає вид точок):

```
x <- 0:10; y <- sin (x)
plot (x, y, pch = 15)
```

Щоб додати новий графік до вже існуючого малюнку використовуються функції **lines** (опція **col** задає колір, опція **lwd** - товщину лінії) і points:

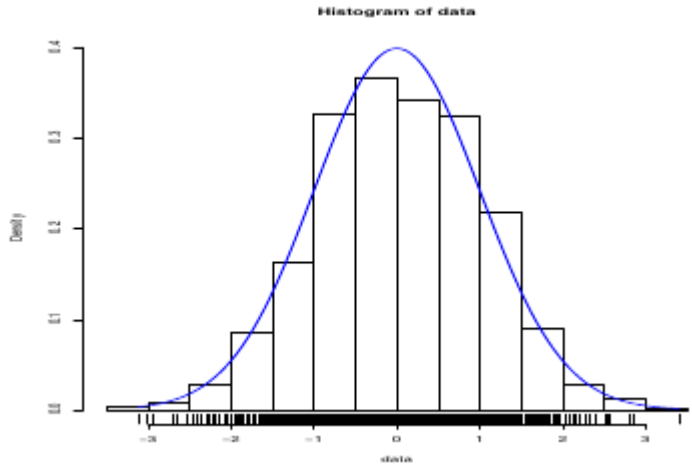
```
t <- seq(from=min(x), to=max(x), by=0.01)
lines (t, sin (t), col = "red", lwd = 3)
```


1.0
0.5
0.0
−0.5
−1.0
y
0
2
4
6
8
10
x

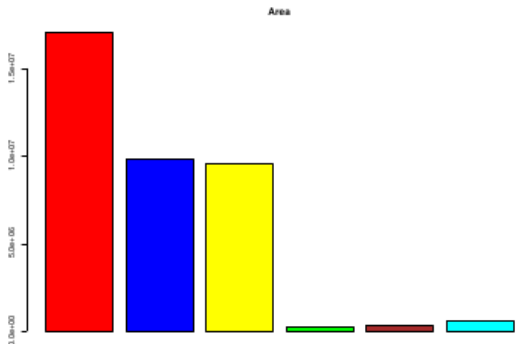
Гістограми малюються за допомогою функції **hist**. Функція **rug** дозволяє відзначити на осі абсцис розташування вихідних даних.

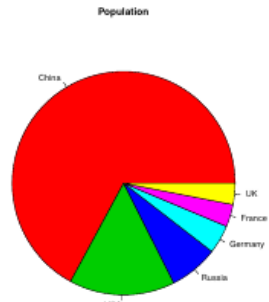
Приклад. Гістограма вибірки нормально розподілених випадкових чисел.

```
data <- rnorm (1000)  # N (0, 1)
x <- seq ( from = min ( data ) , to = max
( data ) , length =100)
y <- dnorm ( x )  # щільність ймовірності
hist ( data, prob =T, ylim = c (0, max( y )))
# гістограма (T - викор-ся відносні частоти)
rug ( data ) # Вихідні дані на осі абсцис
lines (x ,y , col = " blue " , lwd =2)
```

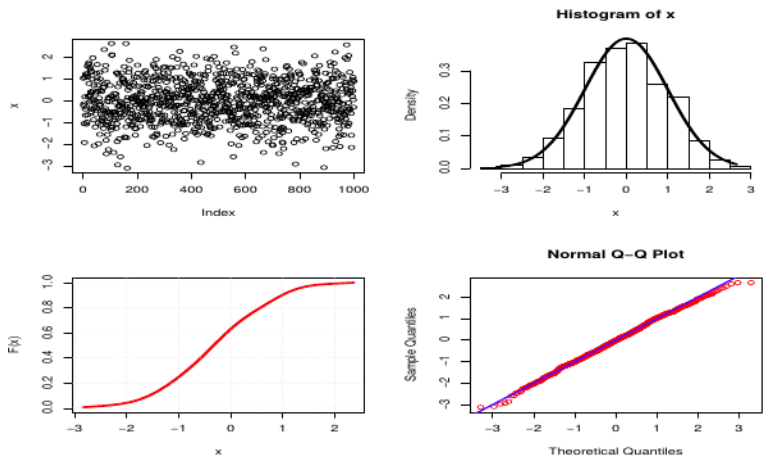

Histogram of data
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
data

Діаграми. Для малювання стовпчастих і кругових діаграм використовуються функції **`barplot`** і **`pie`** відповідно.





Декілька графиків на одному малюнку



Завантаження даних

Нехай потрібно завантажити файл **data.txt**, що знаходиться в кореневому каталозі на диску **E:**.

Якщо дані в файлі розташовані в стовпцях:

	DATE	PI	DPI	TPE	FOOD	...
1	1959	544.90	479.70	440.40	99.70	...
2	1960	559.70	489.70	452.00	100.90	...
3	1961	575.40	503.80	461.40	102.50	...
.....						

то для завантаження даних використовується команда **read.table**:

```
d <- read.table ( " e:/data.txt " )
```

Якщо всі стовпці (включаючи перший) мають заголовки-назви, то вказується параметр **header**:

```
d <- read.table ( " e:/data.txt " , header=T)
```

ЕЛЕМЕНТИ ПРОГРАМУВАННЯ

Загальний вигляд функції в **R**:

```
ім'я <- function (параметри)
{
тіло функції
}
```

Результат - значення останнього обчисленого виразу.

Умовні оператори:

```
if (умова) вираз
if (умова) вираз else альтернатива
```

цикли:

```
for (змінна in послідовність) дія
while (умова) дія
repeat дія
```

Команда **break** дозволяє перервати виконання циклу, а **next** - перейти до наступної ітерації.

ПРИКЛАД ПРОГРАМИ

(Числа Фибоначчи).

$$f_0 = f_1 = 1, \qquad f_i = f_{i-1} + f_{i-2}, \quad i = 2, 3, \ldots$$

```r
fib1 <- function (n)          # использование рекурсии
{
  if (n==0 || n==1) 1
  else fib1(n-1)+fib1(n-2)
}

fib2 <- function (n)          # использование итерации
{
  if (n==0 || n==1) 1
  else
  {
    f1 <- 1
    f2 <- 1
    for (i in 2:n)
    {
      f <- f1+f2
      f1 <- f2
      f2 <- f
    }
    f
  }
}
```

Після визначення, функції користувача використовуються як звичайні функції **R**:

```
fib1(20); fib2(20)              # вычисление 20-го числа Фибоначчи
```

```
[1] 10946
[1] 10946
```

Визначення часу, витраченого на обчислення (в секундах):

```
start <- proc.time(); fib1(20); proc.time()-start
start <- proc.time(); fib2(20); proc.time()-start
```

```
[1] 10946
пользователь      система       прошло
       0.067        0.015        0.081
[1] 10946
пользователь      система       прошло
           0            0            0
```

РЕКОМЕНДОВАНА ЛІТЕРАТУРА ПО МОВІ R

Corawley, M. J. Statistics: an introduction using R / M. J. Corawley. — John Wiley& Sons, Ltd., 2005. — 327 p.

Cowpertwait, P. S. P. Introductory time series with R / P. S. P. Cowpertwait, A. V. Metcalse. — Springer, 2009. — 254 p.

Crawer, J. D. Statistics: an introduction using R / J. D. Crawer, K.-S. Chan. — 2nd edition. — Springer, 2008. — 491 p.

Everett, B. S. A handbook of statistical analyses using R / B. S. Everett, T. Hothorn. — 2nd edition. — CRC Press, 2010. — 348 p.

Horton, N. Using R for data management, statistical analysis, and graphics / N. Horton, K. Kleinman. — CRC Press, 2011. — 238 p.

Lumley, T. Complex surveys: a guide to analysis using R / T. Lumley. — John Wiley & Sons, Ltd., 2010. — 276 p.

Maindonald, J. Data analysis and graphics using R — an example–based approach / J. Maindonald, W. J. Brawn. — 3rd edition. — Cambridge University Press, 2003. — 525 p.

Pfaff, B. Analysis of integrated and cointegrated time series with R / B. Pfaff. — Springer, 2008. — 188 p.

Віктор Гнатюк. Вступ до R на прикладах. - Харків: 2010.
https://cran.r-project.org/doc/contrib/Hnatyuk-R-book-ua.pdf

Роберт Кабаков. R в действии = R in Action. — ДМК-Пресс, 2014. — 588 с.

Хэдли Уикем, Гарретт Гроулмунд. Язык R в задачах науки о данных: импорт, подготовка, обработка, визуализация и моделирование данных = R for Data Science: Visualize, Model, Transform, Tidy, and Import Data. — Вильямс, 2017. — 592 с.

Гришин В.А., Тихов М.С. Методы обработки данных и моделирование на языке R. - Нижний Новгород: Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского (ННГУ), 2019. — 54 с.

На наступній лекції ми розглянемо основи програмування в мультипарадигменній мові OCaml