

# Аналіз даних за допомогою пакету R

Нікіта Скибицький

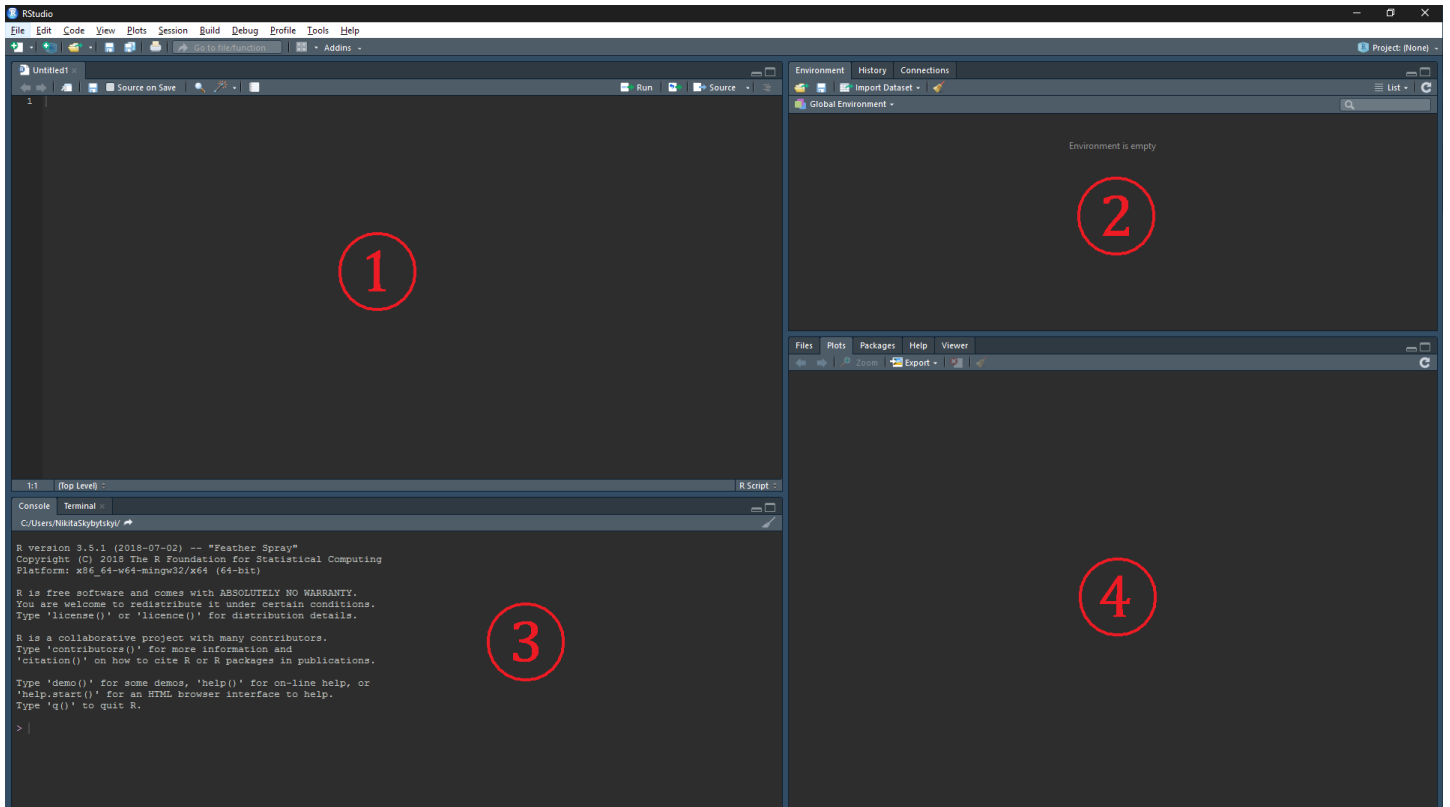
30 листопада 2018 р.

# 1 Початок роботи з системою R

Завантажте R (для Windows) за посиланням (станом на 30 листопада 2018 р.остання версія 3.5.1) та встановіть його.

Завантажте RStudio (для Windows) за посиланням (станом на 30 листопада 2018 р.остання версія 1.1.463) та встановіть її.

Запустіть RStudio (наприклад виконавши `start rstudio.exe` у командному рядку). Ви побачите приблизно наступний екран (швидше за все із світлою темою):



Як бачимо, у нього є 4 основні частини:

1. Текстовий редактор, у ньому ви набиратимете код.
2. Інспектор середовища, через нього можна подивитися які імена визначені у середовищі виконання програми на даний момент і які значення їм відповідають.
3. R-консоль і термінал, у них відображатимуться результати виконання R- та bash-команд відповідно.
4. Інспектор графіків, у ньому відображатимуться усі графіки які ви малюватимете.

Для початку можете погратися із R як із калькулятором, вводячи у R-консоль (третя область на малюнках вище) арифметичні вирази і спостерігаючи результат їх обчислення:

```
> 1 + 2
[1] 3
> 3 - 4
[1] -1
```

```
> 4 * 5
[1] 20
> 6 / 7
[1] 0.8571429
> 8~9
[1] 134217728
```

Виконати поточну команду у консолі можна клавішею **Return** (також відома як **Enter**), а у текстовому редакторі – комбінацією клавіш **Ctrl+Return**. (Командою вважається рядок на якому знаходиться курсор, або виділена частина скрипта якщо така є).

Як і у мові програмування **python**, більшість функціональності не вбудована в мову програмування, а реалізована у R-пакетах (англ. R-packages). Підключити встановлений пакет можна за допомогою функції **library()**, а встановити новий пакет – функцією **install.packages()**.

```
> library(splines) # підключаємо встановлений за замовчуванням пакет
```

```
> library(abctools) # намагаємося підключити не встановлений пакет і ловимо помилку:
Error in library(abctools) : there is no package called 'abctools'
```

```
> install.packages("abctools") # встановлюємо пакет, разом зі всіма його залежностями:
Installing package into 'C:/Users/NikitaSkybytskyi/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
also installing the dependencies 'SparseM', ..., 'Hmisc'
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/SparseM_1.77.zip'
Content type 'application/zip' length 1374637 bytes (1.3 MB)
downloaded 1.3 MB
```

```
...
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/Hmisc_4.1-1.zip'
Content type 'application/zip' length 3013429 bytes (2.9 MB)
downloaded 2.9 MB
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/abctools_1.1.3.zip'
Content type 'application/zip' length 2803903 bytes (2.7 MB)
downloaded 2.7 MB
```

```
package 'SparseM' successfully unpacked and MD5 sums checked
```

```
...
```

```
package 'Hmisc' successfully unpacked and MD5 sums checked
```

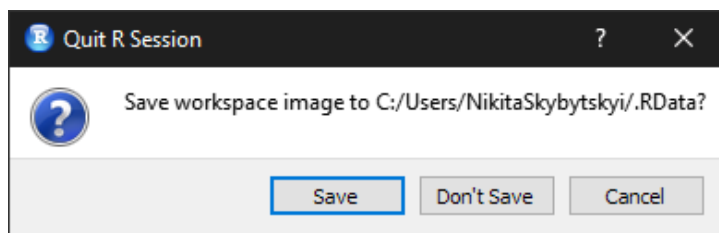
```
package 'abctools' successfully unpacked and MD5 sums checked
```

```
The downloaded binary packages are in
```

```
C:\Users\NikitaSkybytskyi\AppData\Local\Temp\RtmpUNppEz\downloaded_packages
```

```
> library(abctools) # успішно підключаємо щойно встановлений пакет:
Loading required package: abc
...
abctools 1.1.3 loaded
```

З цікавих особливостей зауважимо ще, що R може зберігати образ середовища, що дозволяє продовжувати роботу з того ж місця на якому ви зупинилися минулого разу без необхідності заново визначати значення всіх визначених імен. Якщо це саме те що вам потрібно, то у діалозі при закритті RStudio виберіть “зберегти образ робочого середовища”:



У мові програмування S (так, R не є мовою програмування, а радше просто інтерпретатором), як і будь-якій іншій потужній мові легко загубитися початківцю. Як завжди, рекомендуємо звертатися до офіційної документації окремих пакетів, і знаходити відповіді на свої запитання (котрих майже напевно буде не мало) на теренах [stackoverflow](https://stackoverflow.com).

## 2 Робота з даними (найпростіші операції і типи даних)

### 2.1 Арифметичні операції в R

Позначаються, як ми вже бачили вище, звичайним чином: `+`, `-`, `*`, `/` і `^` для піднесення до степеню. Порядок виконання операцій звичайний і може бути змінений за допомогою дужок:

```
> 7 * (3 + 12) / 2 - 7^2
[1] 3.5
```

Також є в наявності елементарні функції: `log()`, `log10()`, `exp()`, `sin()`, `cos()`, `tan()`, `sqrt()`, а також `abs()`. Функція `round(x, n)` округлює число `x` до `n` десяткових знаків після коми.

### 2.2 Логічні операції

Це операції: `<`, `>`, `<=` (менше або дорівнює); `>=` (більше або дорівнює); `==` (дорівнює); `!=` (не дорівнює); `&` (логічне “і”); `|` (логічне “або”).

### 2.3 Присвоєння значення імені

Традиційно здійснюється за допомогою символу `<-`. Також використовується знак `=`:

```
> x <- 7
> x + 3
[1] 10
```

### 2.4 Статистичні операції

- `mean(x)` обчислює вибіркове середнє масиву `x`;
- `sd(x)` обчислює вибіркове середньоквадратичне відхилення `x`;
- `var(x)` обчислює вибіркиму дисперсію масиву `x`;
- `summary(x)` виводить елементи описативної статистики масиву `x`: мінімальне значення, максимальне, обидва кватилі, медіану і середнє.

- `range(x)` повертає найбільше і найменше значення в `x`. Якщо нас цікавить різниця між найбільшим і найменшим значеннями, можна скористатись функцією `diff(range(x))`.

```
> range(1:10) # 1:10 - послідовність цілих чисел від 1 до 10
[1] 1 10
> diff(range(1:10))
[1] 9
```

- `cor(x, y)` обчислює коефіцієнт кореляції Пірсона між масивами `x` і `y` однакової довжини. Якщо `x` – матриця, то команда `cor(x, x)` видає матрицю кореляцій даних.
- Для обчислення коефіцієнта кореляції Спірмена необхідно в тій самій команді задати додатковий параметр `method`:

```
> x <- c(-100, 0, 11) # c() створює вектор із своїх аргументів
> y <- c(134, 2, -6)
> cor(x, y)
[1] -0.9992336
> cor(x, y, method = "spearman")
[1] -1
```

## 2.5 Вектори

Створюємо вектор за допомогою функції `c()`:

```
> x <- c(1.5, 2, 3)
> x
[1] 1.5 2.0 3.0
```

У вектор можна помістити дані інших типів. Наприклад, текстові:

```
> y <- c("A", "Hello", "world!")
> y
[1] "A" "Hello" "world!"
```

Можна з'єднати 2 вектори в одну структуру з двох стовпчиків. Для цього використовують функцію `cbind()`:

```
> x <- c(2, 3, 4, 1)
> y <- c(1, 1, 1, 10)
> cbind(x, y)
      x y
[1,] 2  1
[2,] 3  1
[3,] 4  1
[4,] 1 10
```

Об'єднання двох рядків в одну структуру здійснюється за допомогою `rbind()`:

```
> rbind(x, y)
      [,1] [,2] [,3] [,4]
x         2     3     4     1
y         1     1     1    10
```

Операції додавання, різниці, множення векторів відбуваються поелементно. Якщо ж вектори, що, наприклад, додаються, мають різні довжини, то коротший вектор “циклічно” продовжується до розміру довгого, і після цього проводиться додавання поелементно, причому генерується застереження:

```
> x <- c(-100, 0, 1, 5, -9, 8, 4)
> y <- c(1, 2)
> y + x
[1] -99 2 2 7 -8 10 5
Warning message:
In y + x : longer object length is not a multiple of shorter object length
```

## 2.6 Послідовності

Команда `seq()` створює послідовність чисел. Її часто використовують при графічному аналізі. Три аргументи, які зазвичай використовують в команді: початкове значення, кінцеве і крок (приріст). Якщо ж `приріст = 1`, то достатньо двох аргументів:

```
> seq(0, 1, 0.2)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
> seq(0, 8)
[1] 0 1 2 3 4 5 6 7 8
```

Для стислого запису послідовності цілих від 0 до 8 можна скористатись діапазоном:

```
> 0:8
[1] 0 1 2 3 4 5 6 7 8
```

Для повторення в послідовності `n` разів однакових чисел або символів використовують команду `rep(a, n)`:

```
> rep(c(0, "x"), 3)
[1] "0" "x" "0" "x" "0" "x"
```

Також можна регулювати довжину послідовності.

```
> rep(c(1, 2, 3), length = 10)
[1] 1 2 3 1 2 3 1 2 3 1
```

Відмітимо, що пакет R (як і будь-яка мова програмування яка себе поважає) використовує круглі дужки `()` для аргументів функцій і квадратні `[]` для того, щоб звернутись до конкретного елемента в послідовності, векторі, масиві, списку. Також R підтримує концепції зрізу (англ. *slice*), індексації по бітовій, індексній та логічній масках, а також від’ємні індекси для індексації (у якомусь сенсі) з кінця:

```
> t <- c(2, 3, -1, 77, 128, 3)
> t[2]   # другий елемент
[1] 3
> t[2:4] # елементи з другого по четвертий, включно
[1] 3 -1 77
> t[c(1, 4, 6)] # елементи з індексами 1, 4 і 6
[1] 2 77 3
> t[t < 0] # всі від’ємні елементи
```

```
[1] -1
> t[abs(t) > 70] # всі елементи, за модулем більші ніж 70
[1] 77 128
> t[-1] # всі окрім першого
[1] 3 -1 77 128 3
```

Зауважимо, що пакет R (на відміну від будь-якої мови програмування яка себе поважає) починає індексацію з 1 а не з 0, тому будьте обачні.

Абсолютно аналогічно індексація працює над іншими типами даних, наприклад над матрицями.

## 2.7 Списки

Список – це структура, тобто вектор, елементи якого можуть мати різні типи: числові, текстові і т.д. Елементом списку може бути інший список. Списки створюються за допомогою функції `list()`:

```
> student <- list(name = "Alex", surname = "Koval", major = "Math", entry.year = 2015)
> student[[4]] # індексація списків трохи відрізняється: [[]] замість []
[1] 2015
> student$entry.year # також можливе поймає звертання до полів списку
[1] 2015
```

## 2.8 Матриці

$$b = \begin{pmatrix} 190 & 8 & 22.0 \\ 191 & 4 & 1.7 \\ 223 & 80 & 2.0 \end{pmatrix}.$$

Створити таку матрицю в R можна за допомогою команди `matrix()`. Присвоїмо їй ім'я `b.data`. Якщо ми хочемо, щоб дані записувались в матрицю по рядках, використовуємо опцію `byrow = TRUE`. В даному прикладі опція `nrow = 3` показує кількість рядків матриці:

```
> Data <- c(190, 8, 22, 191, 4, 1.7, 223, 80, 2)
> b.data <- matrix(Data, nrow = 3, byrow = TRUE)
> b.data
      [,1] [,2] [,3]
[1,]  190    8 22.0
[2,]  191    4  1.7
[3,]  223   80  2.0
> dim(b.data) # розмірність матриці
[1] 3 3
> region <- c("East", "Middle", "West") # імена розмірностей
> dimnames(b.data) <- list(region, NULL)
> b.data
      [,1] [,2] [,3]
East   190    8 22.0
Middle 191    4  1.7
West   223   80  2.0
> type <- c("type A", "type B", "type C") # імена розмірностей
> dimnames(b.data) <- list(NULL, type)
> b.data
      type A type B type C
```

```
[1,] 190 8 22.0
[2,] 191 4 1.7
[3,] 223 80 2.0
> dimnames(b.data) <- list(region, type) # імена розмірностей
> b.data
      type A type B type C
East    190    8  22.0
Middle  191    4   1.7
West    223   80   2.0
```

Додавання матриць  $A + B$  відбувається поелементно. Множення  $A * B$  також – поелементно. Якщо матриці мають різну розмірність, то операція не виконується.

Операція множення матриць в математичному сенсі виглядає як  $A \%*\% B$ :

```
> Data <- c(1, 0, 0, 1)
> A <- matrix(Data, nrow = 2, byrow = TRUE)
> A
      [,1] [,2]
[1,]    1    0
[2,]    0    1
> Data <- c(1, -9, 1, 1)
> B <- matrix(Data, nrow = 2, byrow = TRUE)
> C <- A \%*\% B
> C
      [,1] [,2]
[1,]    1  -9
[2,]    1    1
```

Операція `diag()` перетворює діагональні елементи матриці в елементи вектора:

```
> z <- diag(C)
> z
[1] 1 1
```

## 2.9 Фрейми даних

Фрейм – найбільш широкоживаний тип змінних в R, який використовується для зберігання даних. Фрейми складаються з різноманітних типів даних (числових, текстових, логічних і т.д.). Традиційно, стовпчики розглядаються як змінні, рядки містять характеристики об'єктів. Приклад:

1. Створюємо файл `*.txt` з даними. Зчитуємо ці дані у фрейм за допомогою команди

```
> da <- read.table(file.choose(), header = TRUE)
> da
      Value_A Value_B Value_C
First     190      8   22.0
Second    191      4    1.7
Third     223     80    2.0
```

2. Тепер ми хочемо залучити дві додаткові змінні: логічну змінну `Opinion` і текстову `Danger`. Створюємо відповідні стовпчики-масиви і заносимо їх у фрейм:



```

> Opinion <- c(FALSE, TRUE, FALSE)
> Danger <- c("no", "no", "yes")
> y <- data.frame(da, Opinion, Danger)
> y
      Value_A Value_B Value_C Opinion Danger
First      190      8    22.0   FALSE    no
Second     191      4     1.7    TRUE    no
Third      223     80     2.0   FALSE    yes
> rm("Danger")
> rm("Opinion")

```

3. Тепер ми хочемо надрукувати дані, що відповідають лише змінній `Value_A`:

```

> y$Value_A
[1] 190 191 223

```

4. Для того, щоб звертатись до стовпчиків фрейму спрощено, лише за ім'ям, треба скористатись операцією `attach()`. А щоб зняти це маскування, треба застосувати `detach()`:

```

> attach(y)
> Value_A
[1] 190 191 223

```

5. Відсортуємо дані відповідно змінній `Value_C`:

```

> y[sort.list(y[,3]),]
      Value_A Value_B Value_C Opinion Danger
Second     191      4     1.7    TRUE    no
Third      223     80     2.0   FALSE    yes
First      190      8    22.0   FALSE    no

```

## 2.10 Читання і запис даних. Редагування

Як вже зазначалося, спершу треба створити файл `*.txt` з даними (це можна зробити в текстовому редакторі RStudio). В пакеті можна зчитати ці дані у фрейм `da` за допомогою команди

```

> da <- read.table(file.choose(), header = TRUE)

```

Перша з опцій вказує на те, що файл для зчитування треба задати вручну. При виконанні операції треба відкрити необхідний файл, вказавши шлях до нього. Друга опція вказує на те, що перший рядок таблиці – імена змінних. В іншому випадку, якщо файл одразу починається з даних, замість `TRUE` друкуємо `FALSE`. Тоді знов-таки дані зчитуються в фрейм, але змінні будуть називатись `V1`, `V2` тощо.

Нехай ми зчитали дані у фрейм, і вирішили їх трохи змінити. Наприклад, дати ім'я змінним. Для цього є зручна команда `edit()`

```

> da <- edit(da)

```

Відкриється таблиця, що містить фрейм, і можна вручну виправити всі дані чи константи.

### 3 Робота з розподілами

Кожному розподілу в R відповідають чотири функції. Корінь назви функції вказує на вид розподілу. Перша літера – префікс – розшифровується наступним чином:

- **p** позначає функцію розподілу
- **q** позначає квантильну функцію, тобто, обернену до функції розподілу.
- **d** позначає щільність розподілу.
- **r** – функція з таким префіксом генерує випадкову величину, яка має вказаний розподіл.

Для дискретної випадкової величини під поняттям “щільність” розуміємо  $f(x) = P\{X = x\}$ .

Розподіл	F	Q	f	random
Бета	pbeta	qbeta	dbeta	rbeta
Біноміальний	pbinom	qbinom	dbinom	rbinom
Коші	pcauchy	qcauchy	dcauchy	rcauchy
Хі-квадрат	pchisq	qchisq	dchisq	rchisq
Експоненціальний	pexp	qexp	dexp	rexp
Гама	pgamma	qgamma	dgamma	rgamma
Геометричний	pgeom	qgeom	dgeom	rgeom
Гіпергеометричний	phyper	qhyper	dhyper	rhyper
Логістичний	plogis	qlogis	dlogis	rlogis
Логнормальний	plnorm	qlnorm	dlnorm	rlnorm
Від’ємний біноміальний	pnbinom	qnbinom	dnbinom	rnbinom
Нормальний	pnorm	qnorm	dnorm	rnorm
Пуассона	ppois	qpois	dpois	rpois
Стюдента	pt	qt	dt	rt
Рівномірний	punif	qunif	dunif	runif
Вейбулла	pweibull	qweibull	dweibull	rweibull

Трохи прикладів:

1. Біноміальний розподіл. Генерування вибірки обсягу 100 з біноміальним розподілом з параметрами  $n = 5$ ,  $p = 0.5$ :

```
> x <- rbinom(100, 5, 0.5)
> x
[1] 1 2 5 3 1 2 3 3 2 3 2 3 1 2 3 2 1 3 2 2 4 2 1 3 2 3 4 2 2 3 2 3 3 2 1 0 4
[38] 3 2 2 1 4 2 2 4 1 0 3 2 2 3 3 1 3 2 3 3 3 2 2 2 2 4 0 1 5 1 2 3 2 2 5 3 1
[75] 4 4 3 3 4 2 2 2 3 3 3 5 2 3 1 0 1 2 1 1 1 2 2 3 3 2
```

2. Розподіл Пуассона. Знаходження  $P\{X \leq 3\}$  для розподілу Пуассона з параметром  $\lambda = 8$ .

```
> ppois(3, 8)
[1] 0.04238011
```

3. Геометричний розподіл. Знаходження  $P\{X = 3\}$  для геометричного розподілу з параметром  $p = 0.2$ . Геометричний розподіл визначений як  $P\{X = k\} = p(1 - p)^k$ ,  $k = 0, 1, \dots$

```
> dgeom(3, 0.2)
[1] 0.1024
```

4. Гіпергеометричний розподіл. Знаходження  $P\{X = 25\}$  для гіпергеометричного розподілу з параметрами  $m = 147$ ,  $n = 3$ ,  $k = 25$ . Гіпергеометричний розподіл визначений як

$$P\{X = i\} = \frac{C_m^i C_n^{k-i}}{C_{n+m}^k}.$$

```
> dhyper(25, 147, 3, 25)
[1] 0.576365
```

5. Рівномірний розподіл. Генерування вибірки обсягу 100 з рівномірного розподілу на відрізку  $[3, 5]$ .

```
> runif(100, 3, 5)
[1] 4.708462 4.148540 3.911683 4.404110 3.504621 4.503454 4.671398 3.472399
[9] 4.536314 4.403720 3.065475 3.309524 4.716290 3.498786 4.120027 3.759941
[17] 4.654183 4.446405 3.544601 4.837599 3.681989 4.703181 4.457970 4.703332
[25] 3.778597 3.880072 3.051437 4.037928 4.233277 3.552749 4.269502 4.793736
[33] 3.931118 3.256315 3.589360 3.592465 3.595315 3.102945 4.393103 3.221857
[41] 4.856228 3.928156 3.364604 3.871798 3.445257 4.852213 4.631684 3.255897
[49] 4.443173 4.317664 3.889451 4.081390 3.132968 3.895170 3.065156 3.401125
[57] 3.061146 3.909691 3.145242 3.426145 3.629901 3.327554 3.687854 4.343435
[65] 4.960866 3.510414 4.442051 3.701867 4.907187 3.132439 4.000265 4.573773
[73] 4.877307 3.222933 3.040170 4.706944 4.330527 4.063808 3.446007 3.762330
[81] 3.886572 3.157416 3.938254 4.661562 3.827205 4.479402 3.511800 3.615316
[89] 4.569078 4.572645 3.304311 4.447321 4.354902 3.251734 4.115363 3.007065
[97] 3.309011 4.714925 4.810585 3.534666
```

6. Експоненціальний розподіл. Знаходження квантиля рівня 0.95 експоненціального розподілу з параметром  $\lambda = 1/8$ .

```
> qexp(0.95, 1/8)
[1] 23.96586
```

7. Нормальний розподіл. Знайдемо  $P\{X < 27.4\}$ , де  $X$  – нормальна випадкова величина з параметрами  $m = 50$ ,  $\sigma = 20$ .

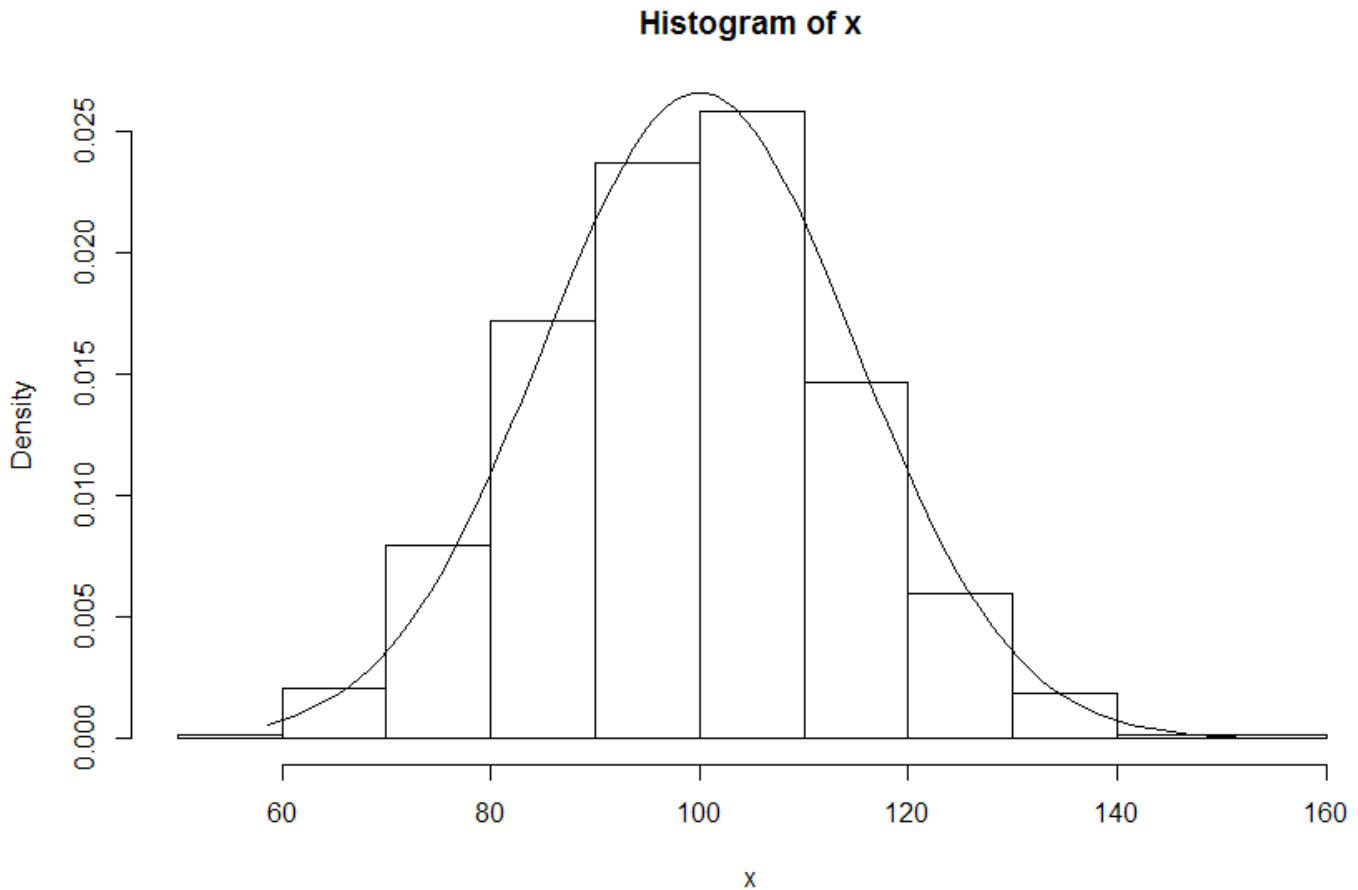
```
> pnorm(27.4, mean = 50, sd = 20) # або просто pnorm(27.4, 50, 20)
[1] 0.1292381
```

Знайдемо квантиль рівня 0.95 з нормального розподілу з іншими параметрами.

```
> qnorm(0.95, mean = 100, sd = 15)
[1] 124.6728
```

Тут ми моделюємо 1000 значень нормальної в.в. з параметрами 100 і 15, будемо для них гістограму і графік оцінки щільності за даними.

```
> x <- rnorm(1000, mean = 100, sd = 15)
> hist(x, probability = TRUE)
> xx <- seq(min(x), max(x), length = 100)
> lines(xx, dnorm(xx, mean = 100, sd = 15))
```



8. Гама-розподіл. Знайдемо  $P\{X > 1\}$ , де  $X$  має гама-розподіл з параметрами  $a = 2$ ,  $\lambda = 3$ .

```
> 1 - pgamma(1, 2, 3)
[1] 0.1991483
```

9. Розподіл  $\chi^2$ . Знайдемо  $P\{40 \leq \chi^2 \leq 50\}$ , де  $\chi^2$  розподілена з 65 степенями свободи.

```
> pchisq(50, 65) - pchisq(40, 65)
[1] 0.07861696
```

10. Розподіл Стюдента. Знайдемо значення щільності розподілу Стюдента з 2 степенями свободи в точці 0.8.

```
> pt(0.8, 1)
[1] 0.7147767
```

Тут другий параметр обчислений за формулою = кількість степенів свободи  $-1 = 2 - 1 = 1$ .

11. Розподіл Фішера. Знайдемо  $c$ ,  $d$  зі співвідношень  $P\{X < c\} = 0.95$  та  $P\{X < d\} = 0.05$ , де  $X$  має розподіл Фішера зі степенями свободи 5, 10.

```
> c <- qf(0.95, 5, 10)
> c
[1] 3.325835
> d <- qf(0.05, 5, 10)
> d
[1] 0.2111904
```

## 4 Графічний аналіз

В файлі `Cancer.txt` наводяться дані по кількості викурених цигарок на душу населення в 43 регіонах Америки та округу Колумбія за 1960 рік, а також дані по кількості смертей від різних форм раку на 100 000 душ населення. Обсяг вибірки  $n = 44$ . Назви змінних:

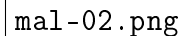
1. `CIG` = Кількість викурених цигарок (умовних упаковок на душу населення).
2. `BLAD` = Кількість смертей на 100 тис. населення від раку сечового міхура
3. `LUNG` = Кількість смертей на 100 тис. населення від раку легенів.
4. `KID` = Кількість смертей на 100 тис. населення від раку нирки.
5. `LEUK` = Кількість смертей на 100 тис. населення від лейкемії.

1. Читаємо дані з файлу у фрейм `da`.

```
> da <- read.table(file.choose(), header = TRUE)
```

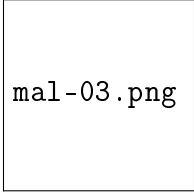
2. Побудуємо гістограму для змінної `CIG`. Розбиття на часткові інтервали обирається за замовчанням.

```
> attach(da)
> hist(CIG)
```



3. Якщо ми хочемо самотужки розбити на часткові інтервали наші дані, слід скористатись опцією `breaks`. Часткові інтервали довжини 3, від 0 до 50. `Xlab` використовується для надпису на осі.

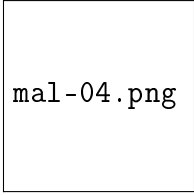
```
> bin <- seq(0, 50, 3)
> hist(CIG, breaks = bin, xlab = "Cigarettes", right = FALSE)
```



mal-03.png

4. Для побудови вусатої коробочки (англ. `boxplot`) можна скористатись оператором

```
> boxplot(CIG)
```

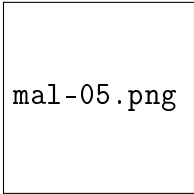


mal-04.png

В пакеті R вусата коробочка має традиційну структуру. Середня риска коробочки відмічає вибірккову медіану. Прямокутник відмічає нижній ( $H_l$ ) і верхній ( $H_u$ ) кuartилі. Тобто,  $H_l$  – вибіркковий квантиль рівня  $1/4$ ;  $H_u$  – вибіркковий квантиль рівня  $3/4$ . Позначимо ширину прямокутника  $H = H_u - H_l$ . Тоді відстань від прямокутника до верхнього вуса коробочки становить 1.5; в нижній частині коробочки так само 1.5.

5. Побудуємо 4 коробочки на одному графіку: BLAD, LUNG, KID, LEUK.

```
> boxplot(BLAD, LUNG, KID, LEUK, horizontal = TRUE,  
col = c(13, 4, 1, 2), names  
= c("BLAD", "LUNG", "KID", "LEUK"))
```

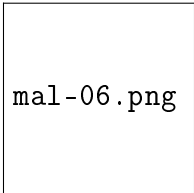


mal-05.png

Опція `horizontal = TRUE` задає горизонтальне положення вусатих коробочок, опція `col` описує кольори, в які вони пофарбовані, `names` друкує надписи до коробочок.

6. Побудувати Q-Q (quantile-quantile) діаграму на порівняння з нормальним розподілом можна наступним чином

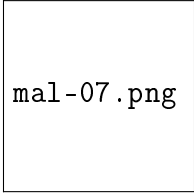
```
> qqnorm(CIG)  
> qqline(CIG)
```



mal-06.png

Тепер побудуємо Q-Q-діаграму на порівняння змінної **CIG** з розподілом  $\chi^2$  з п'ятьма степенями свободи. Як ми знаємо, обсяг вибірки = 44. Опція **ppoints** задає послідовність чисел  $\frac{i-1/2}{n}$ , в даному випадку  $n = 44$ .

```
> qqplot(CIG, qchisq(ppoints(44), df = 5))
```

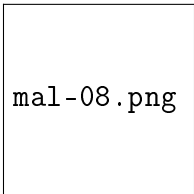


mal-07.png

7. Побудувати Р-Р (probability-probability plot) діаграму на порівняння з нормальним розподілом можна наступним чином. Як відомо, абсциса точки на р-р-діаграмі є емпірична функція розподілу, куди замість аргументу підставляємо дані з вибірки, ордината – теоретична функція розподілу, куди так само замість аргументу підставляємо дані з вибірки. Емпіричну функцію розподілу можна побудувати за допомогою функції **ecdf()**. В наступному прикладі ми генеруємо вибірку зі стандартним нормальним розподілом і будуємо для неї р-р-діаграму на порівняння з тим самим нормальним розподілом.

```
> x <- rnorm(100, 0, 1)
> plot(ecdf(x), pnorm(x, mean(x), sd(x)))
```

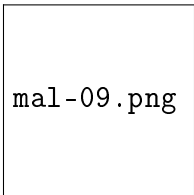
Друга з команд проводить пряму.



mal-08.png

8. Побудуємо матричну діаграму всіх змінних, розміщених в **Cancer.txt**, тобто, тепер в фреймі **da**.

```
> pairs(da)
```



mal-09.png

Схоже, що змінні **BLAD** та **LUNG** залежні лінійно від **CIG**, а от **KID** та **LEUK** – не дуже.