Лабораторна робота

Основи роботи в системі R

ЗАГАЛЬНІ ВІДОМОСТІ

Мова програмування R є діалектом (реалізацією) мови S, яка була розроблена як ефективний і зручний засіб для роботи з даними, моделювання та візуалізації. R розвивається в рамках opensource-проекту і є доступною для різних платформ: Windows, MacOS і UNIX / UNIX-подібних систем (включаючи FreeBSD і Linux).

Корисні посилання:

- Офіційний сайт проекту: http://www.r-project.org
- Короткий довідник часто використовуваних функцій: http://cran.r-project.org/doc/contrib/Short-refcard.pdf (англ.)
- Популярне вільне середовище розробки: http://www.rstudio.com

ОСНОВИ РОБОТИ В СИСТЕМІ R

Основним способом роботи користувача в системі R є використання інтерактивного режиму роботи в консолі інтерпретатора, або написання програм-скриптів для їх подальшого виконання в системі. Як правило, скрипти зберігаються в файлах, що мають розширення .R. Виклик скрипта на виконання з консолі інтерпретатора мови здійснюється функцією **source("path/to/file.R")**. При цьому можна вказувати відносні шляхи, відправною точкою яких є робоча директорія, дізнатися яку можна за допомогою функції **getwd()**, а змінити за допомогою **setwd("new/working/directory")**.

Всі обчислення проводяться в деякому оточенні (environment), яке визначає зв'язок між іменами і значеннями: між ім'ям змінної і її значенням, між ім'ям функції і її реалізацією. Даний зв'язок здійснюється шляхом оголошення і ініціалізації змінних за допомогою оператора присвоювання (= або <-). Додатково вказувати тип змінної при її створенні не потрібно.

```
> x = 7
> x
[1] 7
> y <- 1.5
> y
[1] 1.5
```

В даному прикладі наведено результат інтерактивного виконання коду в консолі інтерпретатора, де кожна нова команда пишеться після запрошення (за замовчуванням символу «>»), при цьому в тексті скриптів дані символи повинні бути опущені. Результат роботи коду однаковий в цих двох режимах за винятком виведення результату на екран, для чого в скриптах необхідно використовувати функції **print(x)** або **sprintf (fmt,...)**. Функція sprintf служить для форматованого виведення і має такий самий формат, як і аналогічна функція в мові С. Для перенаправлення виведення в файл використовується функція **sink(file)**. Відновлення виведення в консоль здійснюється викликом даної функції без аргументів.

Дізнатися всі доступні в поточному оточенні імена можна за допомогою функції **ls()**. Видалити деяке ім'я x можна за допомогою функції **rm(x)**, повністю очистити поточне оточення можна наступним чином **rm(list=ls())**.

Кожна команда або вираз в мові R повертає деякий результат. Вираз може бути частиною іншого виразу. Вирази можна об'єднувати в блоки. Блок – це кілька виразів, вкладених у фігурні дужки. Самі вирази розділяються крапкою з комою або символом переходу на новий рядок. Результатом виконання всього блоку буде результат останнього з виразів, що складають блок.

Для отримання довідки по тій чи іншій функції можна виконати команду **help("function_name")**.

БАЗОВІ ТИПИ

Вектор

Вектори є основною структурою даних мови R, що представляє собою деяку кількість однотипних елементів неперервно розташованих в пам'яті. В R є 6 базових типів векторів (або, що те ж саме, їх елементів): логічний (**logical**), цілочисельний (**integer**), дійсний (**double**), комплексний (**complex**), символьний (**character**) і двійкові дані (**raw**). Будь-яка константа і змінна одного з перерахованих вище типів також є вектором довжини 1. Для визначення довжини вектора використовується функція **length(x)**.

```
> length(7.8)
[1] 1
> length("text")
[1] 1
```

До числових векторів можна застосовувати арифметичні операції (**+, -, *, /, ^**), які будуть виконані поелементно. Якщо довжини векторів різні, то вектор

меншої довжини буде повторений потрібну кількість разів і обрізаний до потрібного розміру. До цілочисельних векторів аналогічно використовуються операції отримання цілої частини **%/%** і залишку **%%** від ділення. Порівняння векторів здійснюється за допомогою операцій **==, !=, <, <=, >, >=**, результатом застосування яких є логічні вектори. До логічних векторів, в свою чергу, можна застосовувати логічні операції **&** (кон'юнкція), **|** (диз'юнкція), **!** (заперечення). Елементи векторів крім звичайних значень, обумовлених їх типом, можуть приймати спеціальні значення: **NA** (недоступне, пропущене значення), **NaN** (не число), **Inf** (нескінченність), **-Inf** (мінус нескінченність).

Функції створення векторів

Наведемо короткий опис деяких базових функцій для створення векторів.

- **integer(n), double(n), numeric(n), complex(n), logical(n), character(n), raw(n)**. Створюють вектори довжини n відповідного типу, ініціалізовані значеннями за замовчуванням. numeric(n) створює вектор чисел з плаваючою комою, тобто типу double.

```
> integer(10)
[1] 0 0 0 0 0 0 0 0 0 0
> logical(5)
[1] FALSE FALSE FALSE FALSE FALSE
```

- **c(...)**. Об'єднує (конкатенує) кілька векторів в один.

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

- **Оператор :** . Створює вектор, що складається з послідовних цілих чисел, межі інтервалу при цьому зазначаються наступний чином: **<початкове_значення>:<кінцеве_значення>**.

```
> 1:5
[1] 1 2 3 4 5
```

- **seq(from, to, by=d)**, **seq(from, to, length=n)** – розбиває відрізок [from, to] на частини з кроком d, або на (n-1) рівних частин.

```
> seq(0, 1, by = 0.1)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(0, 1, len = 5)
[1] 0.00 0.25 0.50 0.75 1.00
```

- **rep(x, times)**. Об'єднуються times копій вектора x.

```
> rep(c(TRUE, FALSE), 3)
[1]  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

- **Оператор [**. Створює вектор, що складається з заданих елементів вихідного вектора. Таким чином, a[idx] представляє собою новий вектор, що містить елементи вектора a, зазначені в idx. При цьому вектор idx може складатися з додатних цілих чисел і містити в собі індекси елементів в a (індексація починається з 1), які потрібно включити в новий вектор, або складатися з від'ємних цілих чисел і містити в собі індекси елементів в a, які потрібно виключити з вихідного вектора, або бути логічною маскою.

```
> (10:20)[c(1, 3, 5)]
[1] 10 12 14
> (10:20)[-c(1, 3, 5)]
[1] 11 13 15 16 17 18 19 20
```

- **sample(x, n, replace=FALSE, prob=NULL)**. Створює вектор, який є випадковою вибіркою n значень з вектора x з поверненням або без в залежності від значення параметра replace. Розподіл ймовірностей можна задати за допомогою параметра prob, за замовчуванням всі значення рівноймовірні.

```
> sample(1:10, 5, replace = FALSE)
[1] 5 8 2 1 9
> sample(1:10, 5, replace = TRUE)
[1] 10  2  6  3 10
```

Математичні функції

Наведемо опис деяких математичних функцій, що застосовуються до векторів.

- **sin(x), cos(x), tan(x), asin(x), acos(x), atan(x)**. Тригонометричні функції.
- **exp(x), log(x), log10(x), log(x, base)**. Експонента і логарифм.
- **max(x), min(x)**. Пошук максимального / мінімального значення.
- **range (x)**. Пошук мінімуму і максимуму, тобто c(min(x), max(x))**.**
- **sum(x)**. Сума елементів вектора.
- **prod(x)**. Добуток елементів вектора.
- **mean(x)**. Середнє арифметичне елеметів вектора.

Деякі інші функції

- **sort(x, decreasing=FALSE)**. Створює вектор, що складається з елементів вектора x, відсортованих в порядку спадання (decreasing=TRUE) або зростання (decreasing=FALSE).
- **order(x, decreasing=FALSE)**. Створює вектор-перестановку елементів вектора x для його впорядкування за зростанням або спаданням. Сортування значень вектора x може бути здійснена наступним чином: **x[order(x)]**.
- **round(x), trunc(x), floor(x), ceiling(x)**. Функції округлення векторів з числами з плаваючою комою.
- **is.integer(x), is.double(x), is.numeric(x), is.complex(x), is.logical(x), is.character(x), is.raw(x)**. Функції для перевірки типу вектора, повертають логічне значення.
- **as.integer(x), as.double(x), as.numeric(x), as.complex(x), as.logical(x), as.character(x), as.raw(x)**. Функції конвертування типів векторів.
- **is.na(x), is.nan(x), is.infinite(x)**. Функції для перевірки чи є елементи вектора рівними NA, NaN, Inf / -Inf.
- **paste(..., sep="", collapse=NULL)**. Функція для об'єднання векторів, зконвертованих в символьні. Ця функція конвертує всі, передані в неї вектори в символьні, після чого поелементно об'єднує їх, використовуючи роздільник sep. Якщо вказано параметр collapse, то всі елементи отриманого вектора також об'єднуються в один рядок, використовуючи роздільник collapse.

```
> paste(c("a", "b", "c"), 1:3, sep = "-")
[1] "a-1" "b-2" "c-3"
> paste(c("a", "b", "c"), 1:3, sep = "-", collapse = "/")
[1] "a-1/b-2/c-3"
> paste(c("a", "b", "c"), collapse = "")
[1] "abc"
```

Матриця

В той час як вектори представляють одновимірну послідовність даних базових типів, найчастіше зручніше оперувати багатовимірними даними. Матриці в мові R є двовимірними векторами.

Для перетворення вектора в матрицю можна присвоїти вектору атрибут розмірності за допомогою функції **dim(x)**. Ця функція приймає в якості аргумента вектор, матрицю або масив, повертаючи вектор розмірності. Вектор може мати

або порожній вектор розмірності, або вектор розмірності довжини один, що містить в собі довжину вектора. Матриця повинна мати вектор розмірності довжини 2, перша компонента якого позначає кількість рядків, друга – стовпців. Дані в матриці зберігаються по стовпчиках.

```
> x = 1:16
> dim(x)
NULL
> dim(x) = c(16)
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
> dim(x)
[1] 16
> dim(x) = c(2, 8)
> x
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    3    5    7    9   11   13   15
[2,]    2    4    6    8   10   12   14   16
```

Іншим способом створення матриць є використання функції **matrix(data=NA, nrow=1, ncol=1, byrow=FALSE)**. Параметр data приймає вектор, елементами якого буде проініціалізована створена матриця. Параметри nrow і ncol визначають кількість рядків і стовпців відповідно, byrow визначає як буде заповнена матриця елементами з data (по рядках чи по стовпцях).

```
> x = 1:16
> y = matrix(data = x, nrow = 2, ncol = 8)
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
> y
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    3    5    7    9   11   13   15
[2,]    2    4    6    8   10   12   14   16
> dim(y)
[1] 2 8
> matrix(1:3, nrow = 2, ncol = 3, byrow = FALSE)
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    1    3
> matrix(1:3, nrow = 2, ncol = 3, byrow = TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    2    3
```

Дізнатися кількість рядків і стовпців матриці можна, отримавши весь вектор розмірності за допомогою функції **dim**, або використовуючи функції **nrow(x)** і **ncol(x)**. Доступ до елементів матриці здійснюється за індексом. A[i, j] посилається на елемент i-го рядка і j-го стовпця матриці A. На місці індексів і та j

можуть стояти вектори. Інтерпретація така ж, як і для векторів. Якщо, наприклад, і та j це вектори з цілими додатними елементами, то A[i, j] є підматрицею матриці A, утвореною елементами, які стоять на перетині рядків з номерами з і та стовпців з номерами з j. Якщо, і та j це вектори з цілими відєємними елементами, то A[i, j] є підматрицею, отриманою з A викреслюванням відповідних рядків та стовпців. Вектори і та j можуть бути логічними або символьними. В останньому випадку рядкам і стовпцям матриці повинні бути присвоєні імена функцій **rownames(x)** і **colnames(x)**. A[i,] еквівалетно A[i, 1: ncol(A)], а A[,j] еквівалетно A [1: nrow(A), j]. Можливий доступ до елементів матриці за допомогою одного індексу. Наприклад, A[5] означає 5-й елемент матриці A, якщо вважати, що елементи перенумеровані по стовпчиках. Якщо і вектор, то A[i] означає вибірку відповідних елементів і т. д.

```
> m = matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> m[1, 3]
[1] 5
> m[, 3]
[1] 5 6
> m[, c(1, 3)]
     [,1] [,2]
[1,]    1    5
[2,]    2    6
> m[-1,]
[1] 2 4 6
> A = matrix(c(23, 31, 58, 16), nrow = 2)
> rownames(A) <- c("petal", "sepal")
> colnames(A) <- c("length", "width")
> A
      length width
petal     23    58
sepal     31    16
> A["petal", "length"]
[1] 23
```

Для того, щоб об’єднати кілька матриць в одну, використовуються функції **rbind(...)** (приписує матриці одна до одної знизу) і **cbind (...)** (приписує матриці одна до одної праворуч).

```
> A = matrix(1:4, nrow = 2, ncol = 2)
> B = matrix(5:8, nrow = 2, ncol = 2)
> cbind(A, B)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> rbind(A, B)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    5    7
[4,]    6    8
```

До матриць можна застосовувати арифметичні операції, проте, вони здійснюватимуться поелементно. Для виконання матричного множення застосовується оператор **%*%**.

- **t(A)**. Транспонує матрицю.
- **diag(A)**. Повертає вектор діагональних елементів матриці, або створює діагональну матрицю з вказаними значеннями діагоналі.

```
> m = matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> diag(m)
[1] 1 4
> diag(m) = c(11, 14)
> m
     [,1] [,2] [,3]
[1,]   11    3    5
[2,]    2   14    6
> a = diag(1:3, nrow = 3, ncol = 4)
> a
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
```

Масив

Подальшим розвитком ідеї багатовимірності даних в мові R є масиви, які можуть мати будь-яку кількість розмірностей. Робота з масивами майже ідентична роботі з матрицями. Створити масив можна або змінивши вектор розмірності вектора або матриці за допомогою функції **dim(x)**, або використовуючи функцію **array(data=NA, dim=length(data))**.

```
> A  = array(1:24, c(2, 3, 4))
> A
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18

, , 4

     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24

> x = 1:24
> dim(x) = c(2, 3, 4)
> x
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18

```

```
, , 4

     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
```

Для перевірки чи є деяка змінна вектором, матрицею або масивом використовуються фунции **is.vector(x), is.matrix(x), is.array(x)**.

Список

Список є узагальним (generic) вектором, тобто його елементи можуть мати різний тип. Елементами списку можуть бути в тому числі вектори і інші списки. Для створення списку використовується функція **list(...)**, яка приймає на вхід будь-яку кількість об'єктів, які потрібно об'єднати в список.

```
> L1 = list(1:5, c("a", "b"))
> L1
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "a" "b"

> L2 = list(rep(c(TRUE, FALSE), 3), L1)
> L2
[[1]]
[1]  TRUE FALSE  TRUE FALSE  TRUE FALSE

[[2]]
[[2]][[1]]
[1] 1 2 3 4 5

[[2]][[2]]
[1] "a" "b"
```

Для додавання нових елементів у список може використовуватися функція конкатенації списків **c(...)**. Для доступу до елементів списку здійснюється індексування за допомогою подвійних квадратних дужок.

```
> x = list(1:5, c("a", "b", "c"), rep(c(FALSE, TRUE), 3),
    seq(0, 1, length = 11))
> x
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "a" "b" "c"

[[3]]
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE

[[4]]
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> x[c(1, 3)]
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE

> x[2]
[[1]]
[1] "a" "b" "c"

> x[[2]]
[1] "a" "b" "c"
```

Також, за допомогою функції **names(x)** елементам списку можна присвоювати імена, а потім використовувати їх в якості індексів, вказуючи в квадратних дужках (одинарних або подвійних) після імені списку, або використовуючи оператор **$**.

```
> x = list(1:5, c("a", "b", "c"), rep(c(FALSE, TRUE), 3),
    seq(0, 1, length = 11))
> names(x) = c("integers", "letters", "booleans", "doubles")
> x
$integers
[1] 1 2 3 4 5

$letters
[1] "a" "b" "c"

$booleans
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE

$doubles
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
> x[c("integers", "doubles")]
$integers
[1] 1 2 3 4 5

$doubles
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> x["letters"]
$letters
[1] "a" "b" "c"

> x[["letters"]]
[1] "a" "b" "c"
> x$letters
[1] "a" "b" "c"
```

Фактор

Фактори представляють собою структуру даних для зберігання векторів категоріальних даних (класів), тобто величин, які можуть приймати значення зі скінченної, в загальному випадку, невпорядкованої множини. Фактори створюються за допомогою функції **factor(x=character(), levels, labels=Levels)**. Припустимо, що у нас є 3 класи Yes, No, Perhaps. І деяка вибірка з 6 об'єктів, кожен з яких належить одному з цих класів. Тоді

```
> v = c("Yes", "No", "Yes", "Perhaps", "No", "Perhaps")
> f = factor(v)
> f
[1] Yes     No      Yes     Perhaps No      Perhaps
Levels: No Perhaps Yes
```

Для отримання і зміни текстового вектора, що містить імена рівнів фактора служить функція **levels(x)**.

Фрейм

Фрейми даних (**data frames**) один з найважливіших типів даних в R, дозволяє об'єднувати дані різних типів разом. Фрейм є спеціальною версією списку, де всі елементи мають однакову довжину. Можна вважати, що фрейм даних це двовимірна таблиця, в якій (на відміну від числових матриць), різні стовпці можуть містити дані різних типів (але всі дані в одному стовпці мають один тип). Наприклад, така таблиця може містити результати експерименту.

Створити фрейм даних можна за допомогою функції **data.frame(...)**, аргументами якої є будь-яка кількість елементів (стовпців) фрейму. Як елементи

фрейму даних можуть виступати вектори, фактори, матриці, списки або інші фрейми. При цьому всі вектори повинні мати однакову довжину, а матриці і фрейми однакове (таке ж) число рядків. Функція **data.frame(...)** просто збирає всі дані разом. Символьні вектори конвертуються у фактори. Інші дані збираються у фрейм так як є.

```
> a = matrix(1:8, nrow = 4, ncol = 2)
> b = c("a", "b", "c", "a")
> d = (1:4 %% 2 == 0)
> e = factor(c("soft", "hard", "soft", "medium"))
> f = data.frame(a, b, d, e)
> f
  X1 X2 b     d      e
1  1  5 a FALSE   soft
2  2  6 b  TRUE   hard
3  3  7 c FALSE   soft
4  4  8 a  TRUE medium
> f[[1]]
[1] 1 2 3 4
> f[["b"]]
[1] a b c a
Levels: a b c
> f$d
[1] FALSE  TRUE FALSE  TRUE
```

За допомогою функції **colnames(x)** можна змінити імена стовпців фрейму, за допомогою **rownames(x)** – імена рядків.

Для завантаження наборів даних з файлу в фрейм може бути використана функція **read.table(file, header=FALSE, sep="", ...)**, якій на вхід подається шлях до текстового файлу з даними, в якому значення в кожному рядку розділені символом sep. Параметр header дозволяє вказати, чи треба інтерпретувати перший рядок файлу, як імена стовпців таблиці.

ПАКЕТИ РОЗШИРЕНЬ

Однією з причин популярності мови R є велика кількість безкоштовних пакетів (packages), що розширюють базові можливості мови. Основним репозиторієм пакетів розширень є репозиторій CRAN (http://cran.rproject.org), на даний момент нараховує більше 4000 пакетів.

Для використання певного пакета, потрібно встановити його з усіма залежностями і підключити. Для того, щоб встановити пакет, можна завантажити його зі сховищ в бінарному вигляді або у вигляді вихідного коду (в цьому випадку доведеться виконувати збірку пакета вручну), а потім встановити за допомогою

функції **install.packages(pkgs, lib)**, першим аргументом якої є шлях до zip-файлу з пакетом в бінарному вигляді, а другим – шлях до директорії (бібліотеки), в яку буде встановлений даний пакет. Список відомих бібліотек можна подивитися, викликавши функцію **.libPaths()**. Якщо при виклику **install.packages** параметр lib не було зазначено, то використовується перша компонента вектора **.libPaths()**. Іншим способом встановлення пакета є його автоматичне закачування зі сховищ, для цього в якості параметра pkgs функції **install.packages** потрібно вказати тільки ім'я пакета, який у разі вдалого пошуку в репозиторії буде викачаний і встановлений разом з усіма залежностями.

Подивитися список встановлених пакетів можна за допомогою функції **library()**. Щоб завантажити пакет для використання, необхідно викликати функцію **library(package)** або **require(package)**, де package – ім'я пакета. Для того, щоб отримати коротку довідку про встановлений пакет необхідно викликати функцію **library(help=package)**.

Оновити встановлені пакети можна функцією **update.packages()**.

ГРАФІКА

Для побудови графіків в R використовується функція **plot(x, y, ...)**. Тут x – вектор значень абсцис і y – вектор значень ординат.

Функція **plot** має ряд додаткових параметрів. Наведемо опис деяких з них:

- **type** дозволяє вказати тип графіка. Деякі значення параметра: **"p"** (точки, за замовчуванням), **"l"** (лінії), **"b"** (лінії і точки), **"o"** (лінії і точки перекриваються);
- **xlab** і **ylab** дозволяють встановити підписи до осей абсцис і ординат відповідно;
- **main** і **sub** створюють написи зверху і знизу графіка;
- **col** задає колір графіка. Деякі можливі значення: **"blue"**, **"red"**, **"green"**, **"cyan"**, **"magenta"**, **"yellow"**, **"black"**. Можна задати колір rgb-вектором функцією **rgb(r, g, b)**, де **r**, **g** і **b** приймають значення від 0 до 1.
- **lty** визначає стиль лінії. Деякі можливі значення **"blank"**, (немає лінії), **"solid"** (суцільна лінія), **"dashed"** (пунктирна лінія), **"dotted"** (точкова лінія), **"dotdash"** (точка-тире).

Наступні функції ніколи не затирають наявний рисунок. Їм також можуть бути передані додаткові параметри, аналогічні відповідним параметрам функції **plot**.

- **points(x, y, ...)** виводить додаткові точки, **lines(x, y, ...)** – ломану лінію, що з'єднує задані точки.
- **text(x, y=NULL, labels=seq_along(x), ...)** додає текст. За замовчуванням він центрується навколо точки (*x; y*). Вказані параметри можуть бути векторами. В цьому випадку буде розміщено декілька написів.
- **abline(k, b, ...)** використовується для відображення прямої *y=kx+b*.
- **abline(h=y, ...)** і **abline(v=x, ...)** виводять горизонтальні і вертикальні прямі відповідно.
- **legend(x, y=NULL, legend, ...)** додає легенду у вказану точку.

Розглянемо приклад.

```
> x = seq(0, 2 * pi, length = 250)
> plot(x, sin(x), col = "red", type = "l", main = "sine and
    cosine", lty = "dashed")
> lines(x, cos(x), col = "blue")
> legend(0, -0.75, legend = c("sine", "cosine"), lty =
    c("dashed", "solid"), col = c("red", "blue"))
```

Результат виконання даного набору команд наведено на рис. 1.

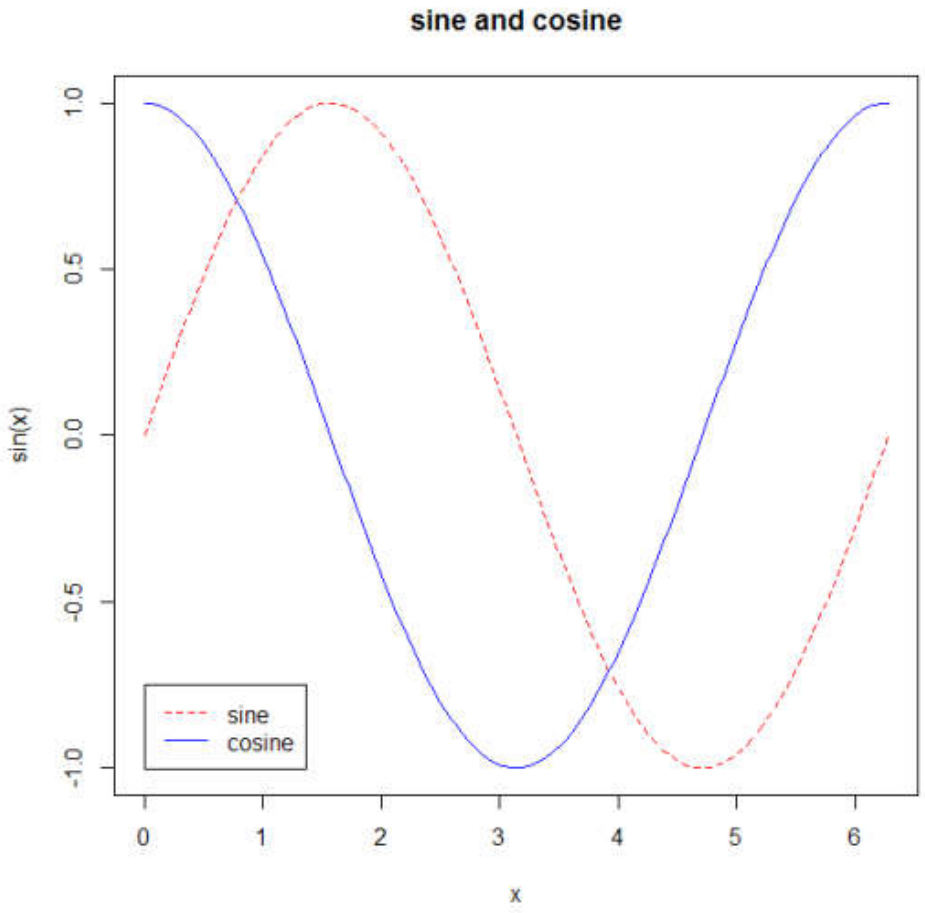

Рис. 1. Побудова графіків в R

Завдання

1. Завантажити дані з файла **hospital-data.csv** в таблицю даних **data**.
2. Вивести декілька перших та останніх записів на екран (функції **head(…)** та **tail(…)** відповідно).
3. Переглянути загальну інформацію про об’єкт (**summary(data)**). Яка різниця у відображенні загальної інформації для стовпців типу фактор і числових (наприклад, стовпці **State** і **ZIP.Code**)?
4. Побудувати графік виду **plot(стовпець типу фактор)**.
5. Відібрати записи з таблиці даних за певним критерієм (придумати три критерії). Наприклад, в результаті виконання команди

   data[data$State == “MA”,]

   будуть відібрані тільки ті записи, для який значення у стовпці **State** дорівнює **“MA”**.