МІНІСТЕРСТВООСВІТИІ НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Математичні обчислення засобами пакету R - програмування

Навчально-методичний посібник
для студентів всіх спеціальностей

Обговорено і рекомендовано
на засіданні кафедри АТ та ГМ,
протокол № 3 від 05.01. 2017р.

Чернігів ЧНТУ 2017

Математичні обчислення засобами пакету R - програмування. Навчально-методичний посібник для студентів всіх спеціальностей./Укл.: В.В. Кальченко,В.П. Мурашковська, Ю.М. Ткач – Чернігів: ЧНТУ,2017, - 86с.

Укладачі:Кальченко Володимир Віталійович професор, доктор технічних наук кафедри автомобільного транспорту та галузевого машинобудування

Ткач Юлія Миколаївна, кандидат педагогічних наук, доцент кафедри автомобільного транспорту та галузевого машинобудування

Мурашковська Віра Петрівна, старший викладач кафедри автомобільного транспорту та галузевого машинобудування

Відповідальний за випуск:Кальченко Віталій Іванович, завідувач кафедри автомобільного транспорту та галузевого машинобудування, професор, доктор технічних наук

Рецензент:Венжега Володимир Іванович – доцент, кандидат технічних наук кафедри автомобільного транспорту та галузевого машинобудування, Чернігівського національного технологічного університету

Зміст

Вступ в R

R - це одночасно і вільно розповсюджене програмне середовище звідкритим кодом, що розвивається в рамках проекту GNU, і мова програмування для статистичної обробки даних і роботи з графікою. R можна застосовувати скрізь, де потрібна робота з даними. Це і сама математична статистика у всіх її додатках, і первинний аналіз даних, і математичне моделювання. **R** найкраще проявляєтьсясаме при статистичному аналізі даних: від обчислення середніх величин до серйозних операцій з функціональними рядами. За допомогою **R** можна підготувати дані для дослідження, яке може бути здійснено за допомогою реалізованих в різних функціях статистичних методів, а потім вивести отримані результати для подальшого аналізу. Зараз практично у всіх західноєвропейських і американських університетах вивчають і використовують **R**, щорічно видаються підручники і монографії щодо роботи як з самим пакетом **R**, так і його застосуванням при дослідженні та обробці даних в статистиці, медицині, екології, фінансовому аналізі, актуарній математиці, інженерних розрахунків та ін. **R** виник як вільний аналог середовища S-PLUS, яке в свою чергу є комерційною реалізацією мови розрахунків **S**. Перша реалізація **S** була написана на FORTRAN і працювала під керуванням операційної системи GCOS. У 1980 році реалізація була переписана під UNIX. Саме тоді в науковому середовищі і почали поширювати **S**. У 1988 році вийшла третя версія, комерційна реалізація якої стала називатися S-PLUS. Досить висока вартість запропонованого комерційного статистичного пакету і привела до виникнення **R**.

R – потужна мова і безкоштовне середовище програмування, що найчастіше застосовується для статистичних обчислень, аналізу даних та візуалізації (представлення даних у графічному вигляді). Створена Росом Іхаком (Ross Ihaka) та Робертом Джентлменом (Robert Gentleman), працівниками Оклендського Університету в Новій Зеландії. Це безкоштовний проект, що розповсюджується за а ліцензією GNU General Public License у вигляді вільнодоступого вихідного коду або відкомпільованих бінарних версій більшості операційних систем: Linux, FreeBSD, Microsoft Windows, Mac OS X, Solaris.R схожий на комерційні мову і середовище S, яка була розроблена в Bell Laboratories (раніше AT & T, зараз Lucent Technologies) Джоном Чемберсом (John Chambers) і колегами. R можна розглядати як «діалект» мови S, що широко використовується в якості навчальної мови та дослідницького інструменту. Основними

перевагамипроекту R є той факт, що R є безкоштовним і що є багато допоміжної інформації, доступної в Інтернеті. Цей проект дуже схожий на інші програмні пакети, такі як MatLab (не безкоштовний), але він більш зручний для користувача, ніж мови програмування, такі як C++ або Fortran. Ви можете використовувати R таким, якимвін є, але в освітніх цілях ми надаємо перевагу використовувати R в поєднанні з інтерфейсом RStudio (також безкоштовним), який має зручнішари і кілька додаткових опцій.

1.1. Що таке S?

✓ S це мова програмування, яка була розроблена Джоном Чемберсом та іншими в Bell Labs.

✓ S була розпочата в 1976 році в якості внутрішнього середовища для статистичного аналізу - спочатку реалізована у вигляді бібліотеки Fortran.

✓ Ранні версії мови не містили функцій для статистичного моделювання.

✓ У 1988 році система була переписана на C і стала нагадувати систему, яку ми маємо сьогодні (це була 3 версія мови). Книга «Статистичні моделі в S»Чемберса і Хесті (так звана «біла книга») документує функціональність статистичного аналізу.

✓ Версія 4 мови S була випущена в 1998 році і є версію яку ми використовуємо сьогодні. Книга «Програмування з даними» Джона Чемберса («зелена книга») документує цю версію мови.

1.2. Історичні факти

✓ У 1993 році Bell Labs дав StatSci (зараз Insightful Corp.) ексклюзивну ліцензію на розробку і продаж мови S.

✓ В 2004 Insightful придбали мову S у Lucent за $ 2 млн і є поточним власником.

✓ У 2006 році Alcatel придбала Lucent Technologies і тепер називається Alcatel-Lucent.

✓ Insightful продає свою реалізацію мови S під назвою продукту S-PLUS з великою кількістю різних можливостей, головним чином графічний користувацький інтерфейс (GUI).

✓ В 2008 Insightful купується TIBCO за $ 25 млн.

✓ Основа самої мови S суттєво не змінилася з 1998 року.

✓ У 1998 році мова S виграла премію в області систем програмного забезпечення асоціації Обчислювальної техніки.

1.3. Розвиток мови R

✓ 1991: Створена в Новій Зеландії Россом Іхаком і Робертом Джентлменом.

✓ 1993: Перший анонс R громадськості.

✓ 1995: Мартін Маклер переконує Росса і Роберта використовувати GNU General Public License, щоб зробити R безкоштовним програмним забезпеченням.

✓ 1996: Створюється список розсилки (R-help і R-Devel)

✓ 1997: Формується основна група R (що містить деяких людей, що пов'язані з S-PLUS). Основна група контролює вихідний код для R.

✓ 2000: Випущена версія 1.0.0 R.

✓ 2014: Випущена версія 3.1.1 R на липень 2014.

1.4. Особливості мови R

✓ Синтаксис дуже схожий на S, що робить його легким для переходу користувачів S-PLUS.

✓ Семантика зовні схожа на S, але насправді зовсім різна.

✓ Працює практично на будь-якій стандартній обчислювальній платформі/ОС (навіть на PlayStation 3)

✓ Часті релізи (річний + релізи з виправленням помилок); активний розвиток.

✓ Досить невелике за розмірами програмне забезпечення; Функціональність ділиться на модульні пакети

✓ Графічні можливості дуже розвинуті і краще, ніж у більшості статистичних пакетів.

✓ Зручний для інтерактивної роботи, але також містить і потужну мову програмування для розробки нових інструментів (користувач -> програміст).

✓ Дуже активне і енергійне співтовариство користувачів; R- help і R-Devel списки розсилки і Stack Overflow.

✓ Це безкоштовно!

1.5. Недоліки мови R

✓ По суті базується на основі 40-річної технології.

✓ Невелика вбудована підтримка динамічної або 3-D графіки (але все значно покращилося з так званих«старих днів»).

✓ Функціональність заснована на споживчому попиті і власних внесків користувачів. Якщо ніхто не відчуває потребив реалізації вашого улюбленого методу, то це ваша робота!

✓ Об'єкти мають бути збережені,як правило, у фізичній пам'яті.

✓ Не ідеально підходить для всіх можливих ситуацій (але це недолік всіх програмних пакетів).

2.Приступаючи до роботи

2.3. Встановлення R

Для встановлення R на ваш комп'ютер (безкоштовно), відвідайте домашню сторінку R:

http://www.r-project.org/

і виконайте наступні кроки (за умови, що ви працюєте на комп'ютері з операційною системою сімейства Windows):

- Натисніть скачати CRAN на лівій панелі (дивись рисунок 1).
- Виберіть сайт для скачування, наприклад http://cran.rstudio.com/ .
- Виберіть Windows як цільову операційну систему, для цього натисніть Download R for Windows.
- Натисніть base.
- Виберіть « Download R 3.1.1 for Windows «(або іншу новішу версію, якщо вона доступна) і натисніть зберегти.
- Запустіть файл що завантажився (R-3.1.1-win.exe)
- Виберіть мову для встановлення (дивись рисунок 2) і натисніть кнопку ОК.

Рисунок 1

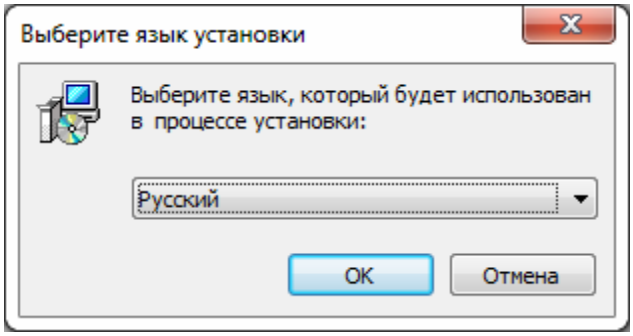


Рисунок 2

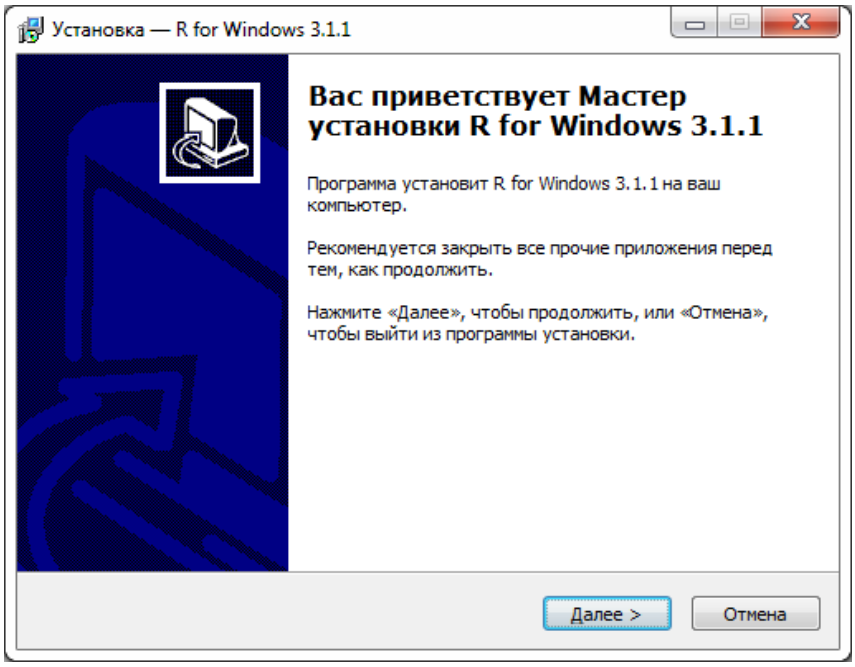


Рисунок 3

- Натисніть кнопку «Далі» (дивись рисунок 3).

- Знову натисніть кнопку «Далі» (дивись рисунок 4) .

- Виберіть місце на комп’ютері куди встановити R та натисніть «Далі» (дивись рисунок 5).

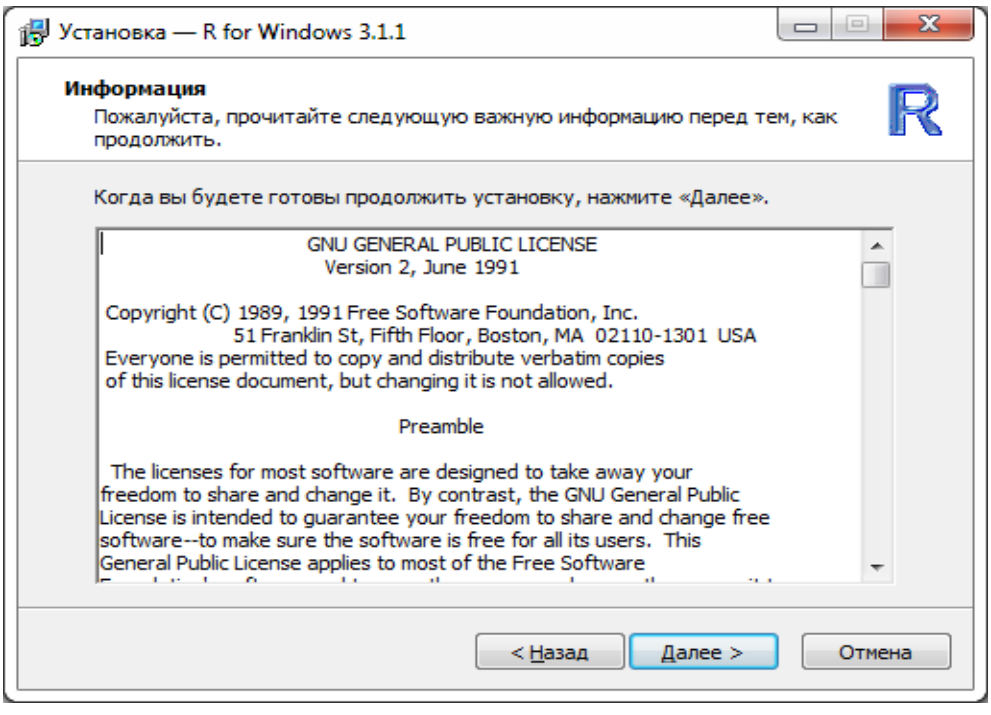


Рисунок 4

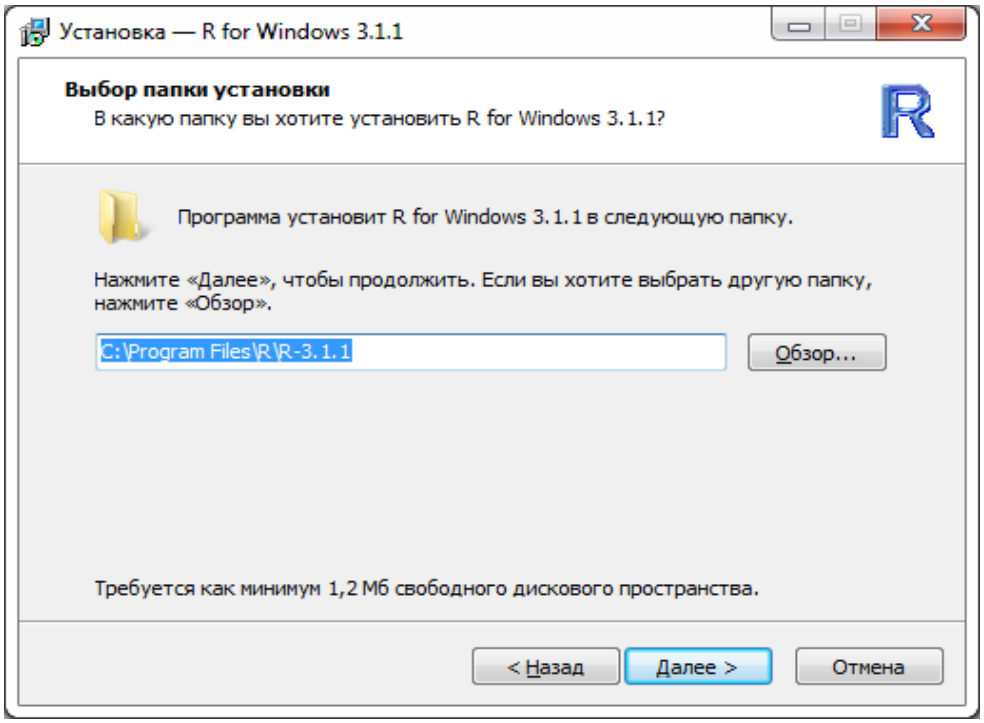


Рисунок 5

- Виберіть «Користувацьке встановлення» і натисніть «Далі» (дивись рисунок 6).

- Знову натисніть кнопку «Далі» (дивись рисунок 7) .

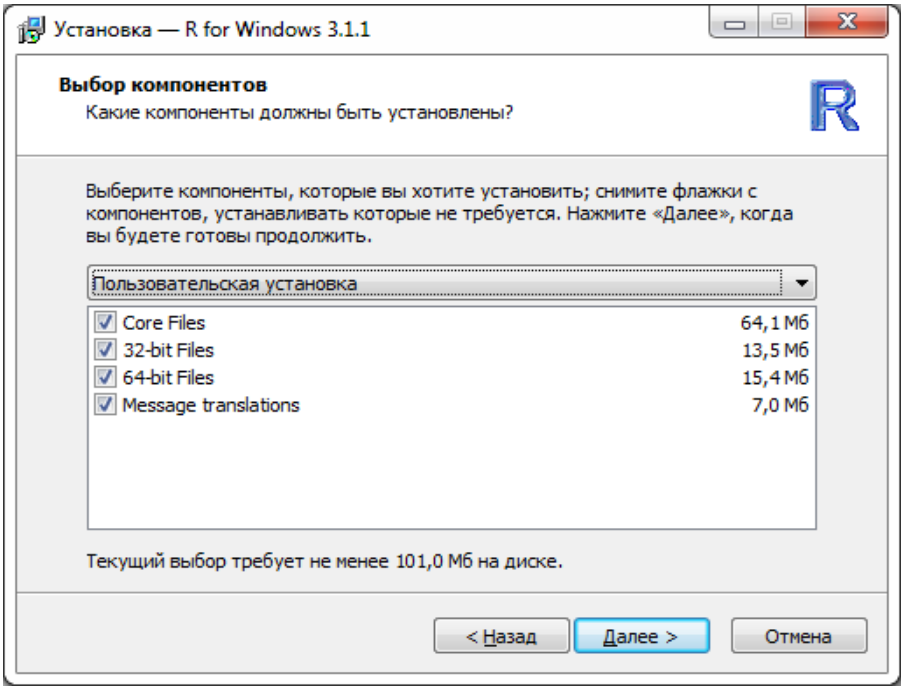


Рисунок 6

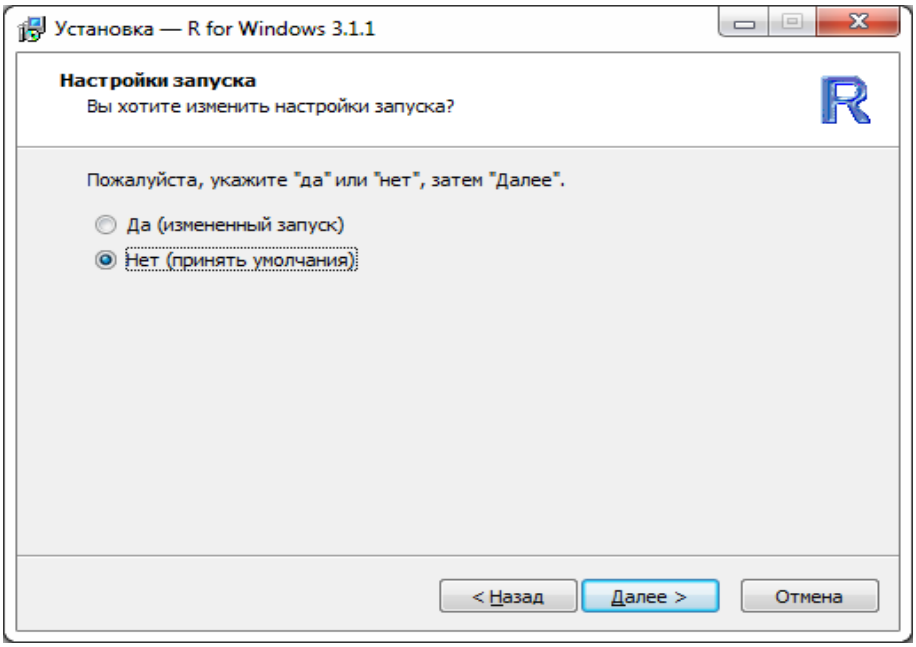


Рисунок 8

- Знову натисніть кнопку «Далі» (дивись рисунок8) .

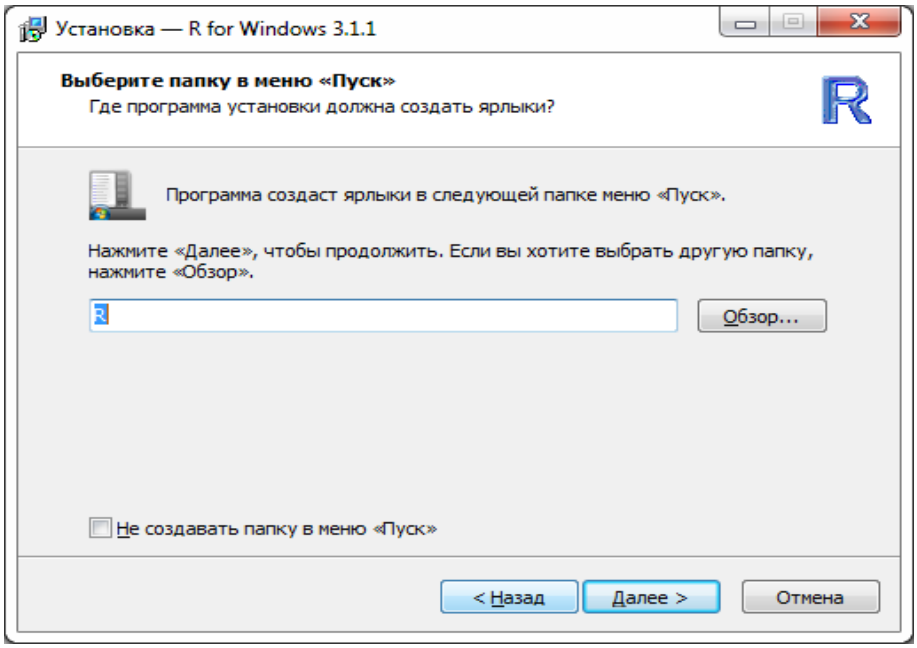


Рисунок 7

- Знову натисніть кнопку «Далі» (дивись рисунок 9) .

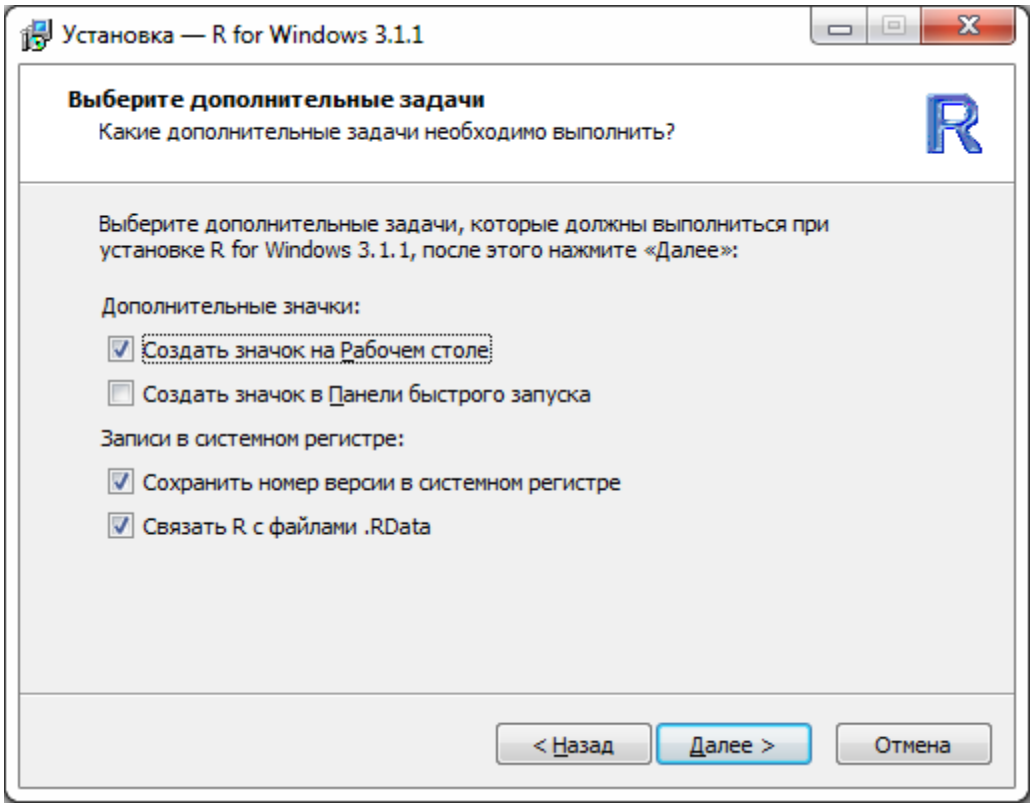


Рисунок 9

- Дочекайтеся поки R встановиться на ваш комп’ютер (дивись рисунок 10) .

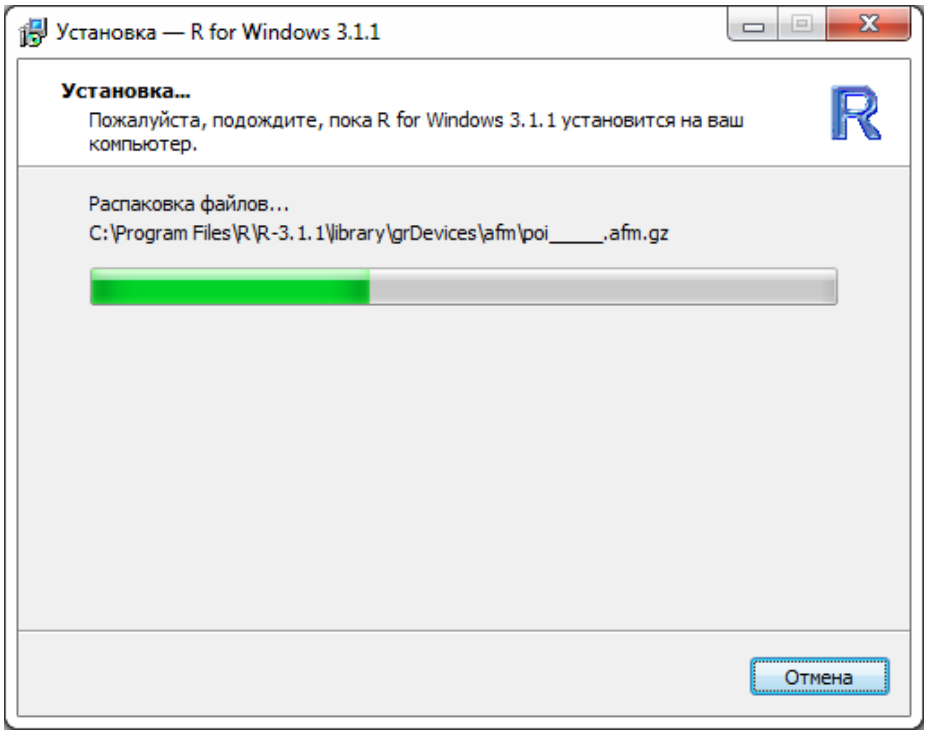


Рисунок 10

- Натисніть кнопку «Завершити». R вже встановився на ваш комп'ютер (дивись рисунок 11) .

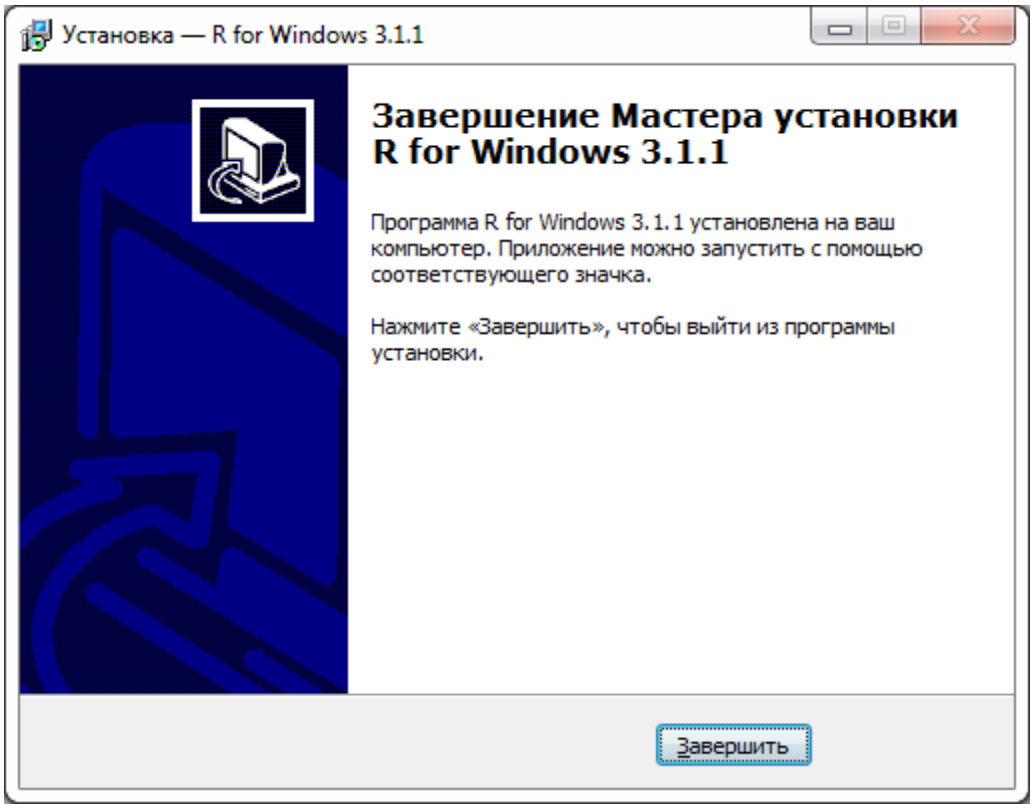


Рисунок 11

2.4. Встановлення R-Studio

Після завершення попереднього встановлення, ви повинні побачити іконку «R» на вашому робочому столі. Натиснувши на неї запуститься стандартний інтерфейс. Для більшої зручності, ми рекомендуємо використовувати інтерфейс RStudio. Крім того існує багато інших безкоштовних інтерфейсів, якими теж можна користуватися. Для встановлення RStudio на ваш комп’ютер, перейдіть по наступному посиланню:

http://www.rstudio.org/

і виконайте наступні кроки (за умови, що ви працюєте на комп’ютері з операційною системою сімейства Windows):

- Натисніть «Download RStudio» (дивись рисунок 12) .

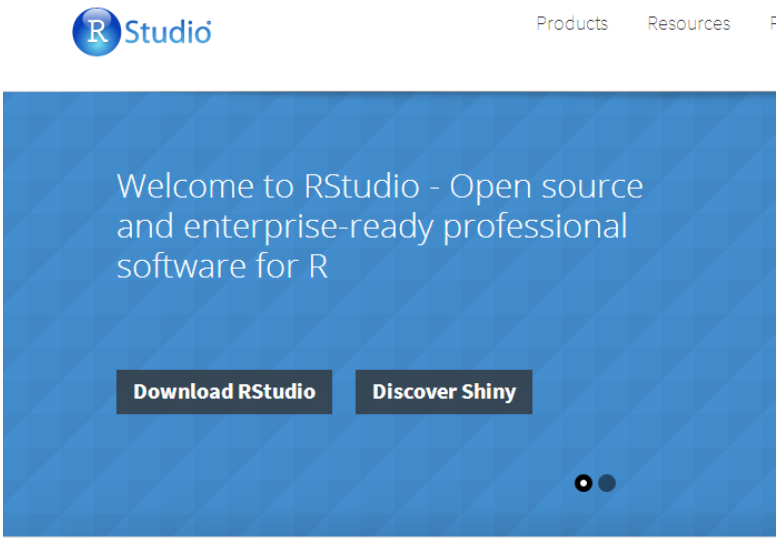


Рисунок 12

- Натисніть «Download RStudio Desktop»(дивись рисунок 13) .

Рисунок 13

- Виберіть версію RStudio для операційної системи Windows (дивись рисунок 14) .
- Завантажте .exe файл і запустіть його. Далі виберіть стандартні

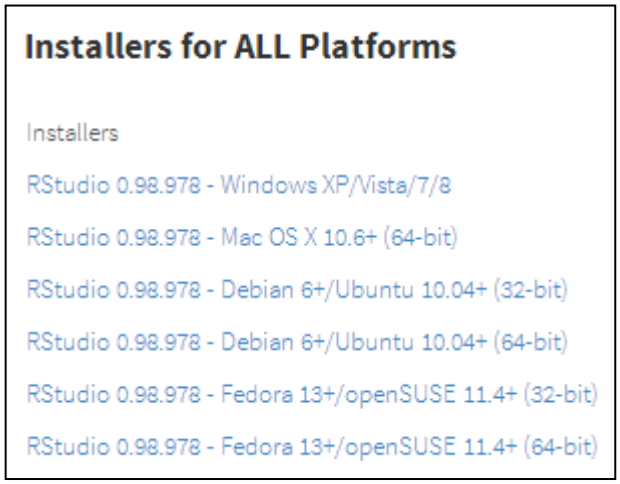


Рисунок 14

відповіді на всі запитання, аналогічно тим крокам, які виконувалися при встановленні R.

2.5. Огляд RStudio

Інтерфейс RStudio складається з декількох вікон (дивись рисунок 15).

З лівої сторони розташоване вікно консолі (яке ще можуть називати командним вікном). Тут ви можете писати різні команди після символу «>«підказки і R виконає вашу команду. Це одне з найважливіших вікон, тому що це вікно де R насправді виконує всі розрахунки.

В правому лівому куті розташоване вікно робочого простору / історії. У вікні робочого простору ви можете побачити які дані та значення R тримає в пам'яті. Ви можете продивитись та редагувати значення натиснувши на них. Вікно історії показує що було введено раніше.

В правому нижньому куті розташоване вікно файлів / графіків / пакетів /довідки. Тут ви можете відкривати файли, продивлятись графіки (також і попередні графіки), встановлювати та завантажувати пакети або використовувати функцію довідки.

Ви можете змінювати розмір вікон перетягуючи сірі смуги між вікнами.

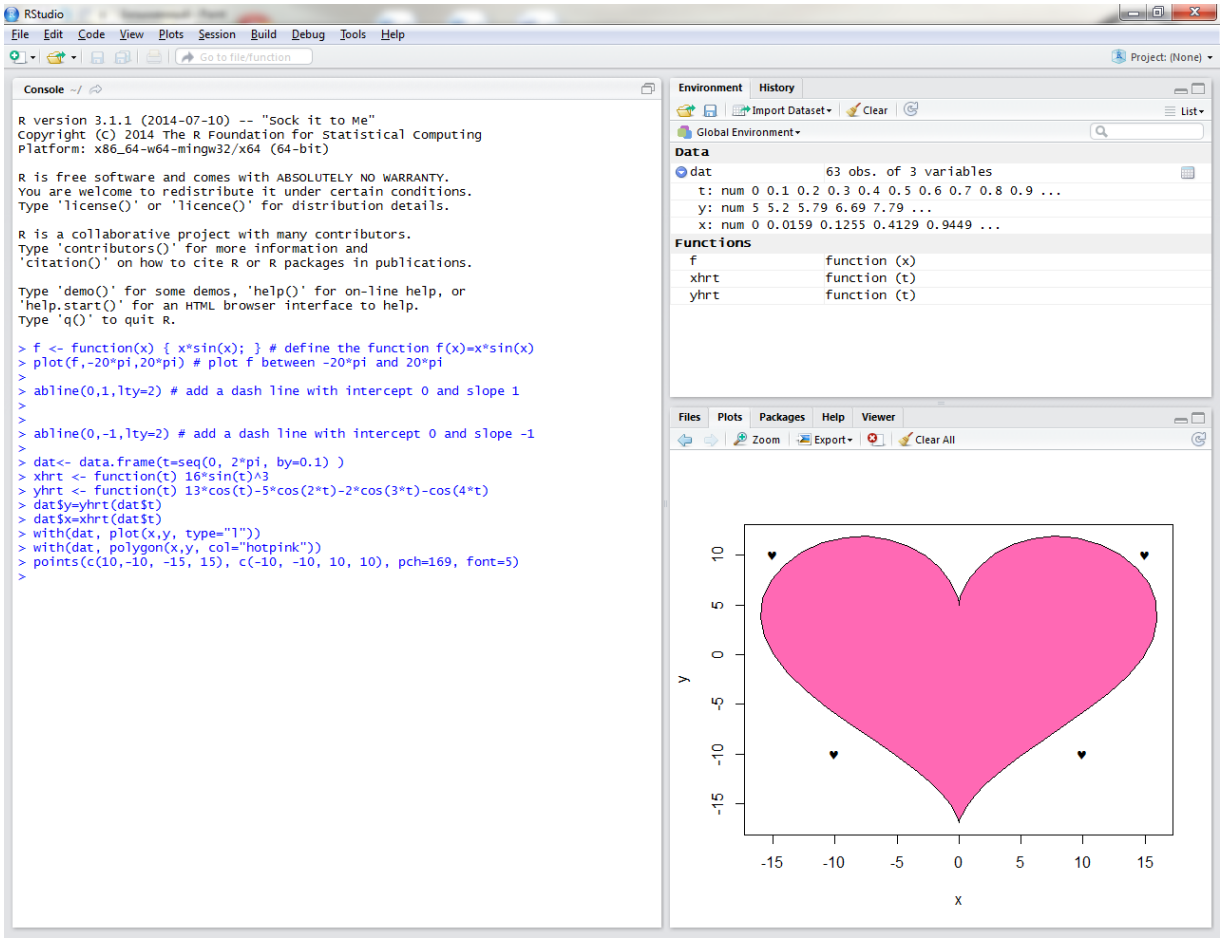


Рисунок 15

2.6. Бібліотеки

R має значні можливості для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз і багато іншого. R легко розбудовується завдяки використанню додаткових так званих пакетів або бібліотек. Разом із стандартним встановленням, більшість найбільш вживаних пакетів вже встановлено.

Щоб отримати список всіх встановлених пакетів, перейдіть у вікно пакетів, або наберіть в консолі `library()`. Якщо навпроти імені пакету стоїть прапорець, це означає що даний пакет завантажений (активований) і може використовуватися.

Є ще дуже багато пакетів доступних на сайті R. Якщо ви хочете встановити і використовувати пакет (наприклад, пакет називається «geometry») ви повинні:

- Встановити пакет: натиснути «install»у вікні пакетів і ввести geometryабо набрати install.packages(«geometry») у командному вікні.
- Завантажити пакет: поставити прапорець навпроти geometry або набрати library(«geometry») у командному вікні.

3.Застосування мови R

R-код дуже легкий для сприйняття, як тільки ви розумієте, що:

- «<-»це оператор присвоювання.
- Все, починаючи з «#», є коментарем.
- Мова R враховує регістр: «a» і «A» це два різні об'єкти.

3.1. Консоль

Команди можна вводити у вікні консолі R в командному рядку (>) і використовувати R в якості потужного наукового калькулятора.

Наприклад, ввівши у вікні консолі:

```
> pi*2.234^5; 43*5/sqrt(56); log(36)
```

Ми отримаємо відповідь

```
[1] 174.8098
[1] 28.73058
[1] 3.583519
```

Тут `sqrt`і `log`– вбудовані функції в R; `pi` – вбудована стала; крапка з комою (`;`) використовується для відокремлення R-команд.

У вікні консолі, <ВГОРУ> і <ВНИЗ> клавіші зі стрілками використовуються для переміщення між раніше введеними пропозиціями.

3.2. Отримання довідки, приклади, демонстрації

R має великий центр довідки. Крім вікна довідки, довідка також доступна з командного рядка консолі. Щоб отримати довідку про певну функцію необхідно лише ввести символ «?» перед її назвою.

Наприклад, набравши

```
> ?log
> ?sin
> ?sqrt
```

Отримаємо пояснення про функцію логарифмів і експонент, тригонометричних функцій, і інші функції.

А набравши

```
> ?Arithmetic
```

Отримаємо перераховані арифметичні операції в R.

Більшість файлів з довідкою також включають приклади. Ви можете запустити всі з них за допомогою R-команди «example».

Наприклад, набравши у вікні консолі:

```
> example(matrix)
```

виконаються всі приклади з файлу допомоги «matrix».

```
> example(pairs)
```

виконаються всі приклади з файлу допомоги «pairs». (Спробуйте це– «pairs»є дуже потужним засобом візуалізації парних відносин).

Крім того, ви можете вибрати один приклад з довідки, скопіювати його в буфер обміну (Ctrl-Cдля користувачів Windows), а потім вставити його (Ctrl-V) у вікно консолі.

Крім того, основне програмне забезпечення R та багато R-пакетів містять демонстраційний матеріал.

Введіть

```
> demo()
```

це дасть список доступних демонстрацій в головному програмному забезпеченні.

```
> demo(graphics)
```

продемонструє кілька простих графічних засобів.

3.3. Потрібно пам’ятати

1. Імена шляхів в R записуються з косою рисою «/», хоча в ОС Windows(®), використовуються зворотні косі риси, «\». Таким чином, щоб встановити робочий каталог в R:

```
>setwd(«C:/R_Code/»)
```

2. Якщо вираз на одній лінії синтаксично правильний, R буде виконувати його, навіть якщо цей вираз продовжується на наступному рядку. Наприклад, якщо ми пишемо:

```
A <- 5 + cos(pi)
     -sqrt(7)
[1] -2.645751
```

То A буде мати значення (5 + cos (pi)) і R виведе значення (-sqrt(7)).

На відміну від цього, в наступних рядках:

```
5 + cos(pi) -
     sqrt(7)
[1] 1.354249
```

R буде виводити значення 5 + cos (pi) -sqrt (7): оскільки вираз напершій лінії не єсинтаксично завершеним, R (правильно)припускає, що він буде продовжений на наступному рядку.

Будьте обережні, якщо ви хочете розділити складнийвираз на кілька рядків! Ці помилки дуже важко простежити, так що краще, уникати їх.

3.4. Використання R як калькулятора

Дуже зручно використовувати R в якості потужного калькулятора. Найкраще це можна зробити використовуючи R-консоль.

Використовуючи консоль, обчисліть значення виразів:

Приклад 1. $\left(\frac{4}{6}\cdot 8-1\right)^{\frac{2}{3}}$

```
> (4/6*8-1)^(2/3)
[1] 2.657958
```

Приклад 2. $\ln(20)$

```
> log(20)
[1] 2.995732
```

Приклад 3. $\log_2\ 4096$

```
> log2(4096)
[1] 12
```

Приклад 4. $2\cdot\pi\cdot 3$

```
> 2 * pi * 3
[1] 18.84956
```

Приклад 5. $e^{2+\cos\ 0.5\cdot\pi}$

```
> exp(2+cos(0.5*pi))
[1] 7.389056
```

Приклад 6. $\sqrt{2.3^2+5.4^2-2\cdot 2.3\cdot 5.4\cdot\cos\left(\frac{\pi}{8}\right)}$

```
> # length of 3rd side of a triangle with size 2.3 and 5.4 and angle pi/8
> sqrt(2.3^2+5.4^2-2*2.3*5.4*cos(pi/8))
[1] 3.391288
```

Порада: вам може знадобиться подивитися на файли довідки для деяких з цих функцій. Щоб це зробити, наберіть `? «+»`. Це відкриє файл довідки із загальними арифметичними операторами.

4.Обчислення з використанням змінних

4.1. Числа, вектори, матриці та масиви

4.1.1. Присвоєння

Коли використовуються змінні, вони повинні бути ініціалізовані числами.

```
> A <- 1
> B <- 2
> A + B
[1] 3
```

R може приймати в якості аргументів функцій числа, вектори, матриці або масиви.

```
> V <- factorial(10)
```

Виконується обчислення 10! (= 1 * 2 * 3 * 4 * 5 * ... * 10). Потім оператор «<- « присвоює результат цього обчислення змінній V. V може потім бути використана в подальших розрахунках:

```
> V/10
[1] 362880
```

Зверніть увагу, що присвоєння значення змінній V не відображає його у вікні. Для відображення значення V ми просто повинні написати:

```
> V
[1] 3628800
```

Крім того, ми можемо присвоїти результат розрахунків змінної і переглянути результат, взявши вираз в дужки:

```
> (X <- sin(3/2*pi))
[1] -1
```

Окрім цілих, дійсних і комплексних чисел, R також розпізнає нескінченність (Inf) і значення, які є невизначеними (NaN). Спробуйте:

```
> 1/0
[1] Inf

> 0/0
[1] NaN

> 1e-8 * 1000
[1] 1e-05
```

(Де «e-8» позначення позначає 10^{-8}).

4.1.2. Вектори

Вектори складаються з упорядкованого набору чисел і можуть бути створені різними способами:

- Використовуючи R-функцію «vector»
- Функція «c()» поєднує числа у вектор (це, напевно, найважливіша функція в R)
- Оператор «:» створює послідовність значень, кожненаступне більше (або менше), ніж попереднє
- Більш загальна послідовність може бути створена за допомогою R-функції «seq»

Наприклад:

```
> c(0, pi/2, pi, 3*pi/2, 2*pi)
[1] 0.000000 1.570796 3.141593 4.712389 6.283185

> seq(from = 0, to = 2*pi, by = pi/2 )
[1] 0.000000 1.570796 3.141593 4.712389 6.283185

> seq(0, 2*pi, pi/2)
[1] 0.000000 1.570796 3.141593 4.712389 6.283185
```

Всі ці команди створять вектор, що складається з: 0, π, ... 2 * π.

Зверніть увагу, що R-функція «seq» приймає в якості вхідних параметрів «from» (початкове значення), «to»(кінцеве значення) і «by»(крок) (другий приклад). Якщо порядок зберігається, можна не вказувати ім’я параметрів (третій приклад).

Наступна команда обчислює синус цього вектора і виводить результат:

```
> sin(seq(0, 2*pi, pi/2))
[1]  0.000000e+00  1.000000e+00  1.224606e-16 -1.000000e+00
-2.449213e-16
```

Наступний вираз:

```
> V <- 1:100
> sqrt(V)
  [1]  1.000000  1.414214  1.732051  2.000000  2.236068  2.449490  2.645751
  [8]  2.828427  3.000000  3.162278  3.316625  3.464102  3.605551  3.741657
 [15]  3.872983  4.000000  4.123106  4.242641  4.358899  4.472136  4.582576
 [22]  4.690416  4.795832  4.898979  5.000000  5.099020  5.196152  5.291503
 [29]  5.385165  5.477226  5.567764  5.656854  5.744563  5.830952  5.916080
 [36]  6.000000  6.082763  6.164414  6.244998  6.324555  6.403124  6.480741
 [43]  6.557439  6.633250  6.708204  6.782330  6.855655  6.928203  7.000000
 [50]  7.071068  7.141428  7.211103  7.280110  7.348469  7.416198  7.483315
 [57]  7.549834  7.615773  7.681146  7.745967  7.810250  7.874008  7.937254
 [64]  8.000000  8.062258  8.124038  8.185353  8.246211  8.306624  8.366600
 [71]  8.426150  8.485281  8.544004  8.602325  8.660254  8.717798  8.774964
 [78]  8.831761  8.888194  8.944272  9.000000  9.055385  9.110434  9.165151
 [85]  9.219544  9.273618  9.327379  9.380832  9.433981  9.486833  9.539392
 [92]  9.591663  9.643651  9.695360  9.746794  9.797959  9.848858  9.899495
 [99]  9.949874 10.000000
```

створює послідовність цілих чисел від 1 до 100 і обчислює квадратний корінь з усіх з них, і відображає результат на екран.

Оператор «<-»присвоює послідовність в V.

Деякі інші приклади з використанням оператора «:»:

```
> (X <- 0.5:10.5)
 [1]  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5

> 6:1
[1] 6 5 4 3 2 1
```

Примітка: Особливістю R є те, що елементам вектора можна також дати власні назви:

```
> (fruit <- c(banana = 1, apple = 2, orange = 3))
banana  apple orange
     1      2      3

> names(fruit)
[1] "banana" "apple"  "orange"
```

4.1.3. Матриці

Матриці також можуть бути створені декількома способами:

- За допомогою R-функції «matrix»
- За допомогою R-функції «diag», яка будує діагональну матрицю
- Функції «cbind» і «rbind» додають стовпці і рядки до існуючої матриці або в інший вектор

Вираз:

```
> A <- matrix(nrow = 2, data = c(1, 2, 3, 4))
```

створює матрицю A, з двома рядками, і, оскільки є чотири елементи, двома стовпчиками. Відзначимо, що дані вводяться як вектор (за допомогою функції c ()).

Наступні два вирази відображають матрицю A та матрицю квадратних коренів з її елементів:

```
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> sqrt(A)
         [,1]     [,2]
[1,] 1.000000 1.732051
[2,] 1.414214 2.000000
```

За замовчуванням, R заповнює матрицю по стовпцях (див. приклад вище). Тим не менш, це можна легко скасувати, за допомогою параметра «byrow»:

```
> (M <- matrix(nrow = 4, ncol = 3, byrow = TRUE, data = 1:12))

     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Одинична матриця (I) створюється за допомогою R-функції «diag»:

```
> diag(1, nrow = 2)
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Назви стовпців і рядків встановлюються таким чином:

```
> rownames(A) <- c("x","y")
> colnames(A) <- c("a","b")
> A

  a b
x 1 3
y 2 4
```

зверніть увагу, що ми використовуємо «c()» функцію тут! Назви рядків і стовпців насправді є векторами, що містять рядки.

Матриці також можуть бути створені шляхом об'єднання (зв'язування) векторів, наприклад по рядкам:

```
> V <- 0.5:5.5
> rbind(V, sqrt(V))

       [,1]     [,2]     [,3]     [,4]    [,5]     [,6]
V 0.5000000 1.500000 2.500000 3.500000 4.50000 5.500000
  0.7071068 1.224745 1.581139 1.870829 2.12132 2.345208
```

t(A) виконає транспонування матриці A (перестановка рядків і стовпців).

```
> t(A)

  x y
a 1 2
b 3 4
```

4.1.4. Масиви

Масиви – це багатовимірні узагальнення матриць; матриці і масиви в R насправді є векторами з атрибутом розмірності.

Багатовимірний масив створюється наступним чином:

```
> AR <- array(dim = c(2, 3, 2), data = 1)
```

В цьому випадку AR – масив 2 * 3 * 2, і всі його елементи 1.

```
> AR
, , 1

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

, , 2

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

4.2. Розмірності

Команди:

```
> length(A)
[1] 4

> dim(A)
[1] 2 2

> ncol(M)
[1] 3

> nrow(M)
[1] 4
```

Поверне довжину (загальне число елементів) V (вектора чи матриці),розмірність матриці або масиву A, число стовпців і число рядків матриці Мвідповідно.

4.3. Вибір і вилучення елементів

Для вибору підмножини векторів або матриць, ми можемо або

- вказати номери елементів, які нам потрібно (проста індексація)

- вказати вектор логічних значень (TRUE / FALSE), щоб вказати, які елементи включити (TRUE) і які не включити (FALSE). Для цього використовуються логічні вирази

4.3.1. Проста індексація

Елементи векторів, матриць і масивів індексуються використовуючи оператор «[]» :

```
> M[1, 1]
> M[1, 1:2]
> M[1:3, c(2,4)]
```

Результатом буде елемент з першого рядка і першого стовпця матриці M (1-ша лінія), потім вибірка елементів з першого рядка і перших двох стовпців (2-га лінія), а потім елементи з перших трьох рядків, 2-го і 4-го стовпця матриці M (3-а лінія).

Якщо індекс пропущений, то всі рядки (перший індекс пропущений) або стовпці (2-й індекс пропущений) вибираються. В наступному виразі:

```
> M[ , 2] <- 0
> M[1:3, ] <- M[1:3, ] * 2
```

спочатку всі елементи в другому стовпці (1-ша лінія) M обнуляються, а потім елементи на перших трьох рядках M множаться на 2 (2-га лінії).

Подібні методи вибору застосовуються до векторів:

```
> V[1:10]
[1] 0.5 1.5 2.5 3.5 4.5 5.5  NA  NA  NA  NA

> V[seq(from = 1, to = 5, by = 2)]
[1] 0.5 2.5 4.5
```

У виразі на 1-ій лінії вибираються перші 10 елементів вектора V, а у виразі на 2-ій лінії, вибираються 1-й, 3-й і 5-й елементи вектора V.

4.3.2. Логічні вирази

Логічні вирази часто використовуються, щоб вибрати елементи векторів і матриць, які підпорядковуються певним критеріям. R розрізняє логічні змінні TRUE (істина) і FALSE (хиба), які представлені числами 1 і 0.

Наступні операції

```
> ?Comparison
> ?Logic
```

перераховують реляційні і логічні оператори, доступні в R.

Наступна дія поверне TRUE для значень послідовності V, які є додатними:

```
> (V <- seq(-2, 2, 0.5))
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0

> V > 0
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

в той час як

```
> V[V > 0]
[1] 0.5 1.0 1.5 2.0
```

вибере додатні значення з V,

```
> V[V > 0] <- 0
```

обнулить всі додатні елементи в V,

```
> sum(V < 0)
[1] 4
```

поверне кількість від’ємних елементів: просумує значення TRUE (= 1), і

```
> V[V != 0]
[1] -2.0 -1.5 -1.0 -0.5
```

покаже всі ненульові елементи з V («!» це оператор логічного заперечення («не»)).

Логічні вирази також можна об'єднувати, за допомогою операторів«|» (логічний оператор «або») і «&» (логічний оператор «і»).

```
> V[V < (-1) | V > 1 ]
[1] -2.0 -1.5
```

Це виведе на екран всі значення з V, які є <-1або >1. Зверніть увагу, що ми вклали «-1» в дужки (ви можете побачити, чому це необхідно?)

І, нарешті,

```
> which(V == 0)
[1] 5 6 7 8 9

> which.min(V)
[1] 1
```

поверне індекс елемента із значенням 0 і мінімальним значенням.

4.4. Видалення елементів

Якщо перед індексом елемента поставити «-», елемент видаляється.

```
> M[,-1]
```

На екран виведеться вміст матриці M, за винятком першого стовпчика.

```
> x<-x[-1]
> M<-M[-1,]
> V<-V[-V=0]
```

Видалиться перший елемент X (перша лінія), 1-й рядок M (друга лінія), і всі додатні елементи V (третя лінія).

Для отримання додаткової інформації, наберіть

```
> ?Extract
```

5. Створення функцій

5.1. Оголошення функцій

Однією із сильних сторін R є те, що користувач може оголошувати свої власні функції, які додаються до вбудованих функцій R.

Як правило, складні функції, пишуться у **R-script** файлах, які потім підключаються до середовища R . Наприклад,

```
>Circlesurface<- function (radius) pi*radius^2
```

визначає функцію (яка має назву«Circlesurface»), яка приймає в якості вхідного аргументу змінну «radius» і яка повертає значення pi*radius^2(що є значенням площі круга з радіусом «radius»).

Після підключення цієї функції до середовища R, ми можемо використовувати її для обчисленняплощі круга із заданим радіусом:

```
> Circlesurface(10)
[1] 314.1593
```

або

```
> Circlesurface(1:20)

 [1]    3.141593   12.566371   28.274334
 [4]   50.265482   78.539816  113.097336
 [7]  153.938040  201.061930  254.469005
[10]  314.159265  380.132711  452.389342
[13]  530.929158  615.752160  706.858347
[16]  804.247719  907.920277 1017.876020
[19] 1134.114948 1256.637061
```

яка буде обчислювати площі кругів з радіусами 1, 2, ..., 20.

Більш складні функції можуть повертати більше одного елемента:

```
> Sphere <- function(radius)
+ {
+     volume <- 4/3*pi*radius^3
+     surface <- 4 *pi*radius^2
+     return(list(volume=volume,surface=surface))
+ }
```

Тут ми визначаємо:

- заголовок функції (1-я лінія), вказавши ім'я функції (Sphere)івхідний параметр (radius)
- тіло функції. Оскільки функція включає декілька операторів, то тіло функціїпишеться всередині фігурних дужок{...}.
- значення, що повертаються (остання стрічка тіла функції). Функція «Sphere»поверне об’єм і площу поверхні сфери, у вигляді списку.

Земля має приблизно радіус 6371 км, так що їїоб’єм (км^3) і площа поверхні (км^2) є:

```
> Sphere(6371)
$volume
[1] 1.083207e+12

$surface
[1] 510064472
```

В наступному виразі буде відображатися тільки об’єм сфер з радіусами 1, 2, ... 5

```
> Sphere(1:5)$volume
[1]   4.18879  33.51032 113.09734 268.08257
[5] 523.59878
```

Іноді зручно, щоб вхідні параметри мали значення за замовчуванням.

Наприклад, наступна функція оцінює щільність «стандартної океанської води» (в кг/м$^{-3}$), залежно від температури T, (і для солоності salinity = 0, тиску pressure = 1 атм) (Міллер і Пуассон, 1981); вхідний параметр T за замовчуванням, рівний 20 ° C:

```
> Rho_W <- function(T=20)
+ {
+     999.842594 + 0.06793952 * T - 0.00909529 * T^2 +
+         0.0001001685 * T^3 - 1.120083e-06 * T^4 + 6.536332e-09
* T^5
+ }
```

В кінці першого рядка стоїть «+» , цим ми вказали, що вираз не закінчився і має продовження в наступному рядку. Було б неправильно ставити «+» на початку другого рядка. (Дивись главу 3)

Виклик функції без вказування температури, використовує значення за замовчуванням:

```
> Rho_W()
[1] 998.2063
> Rho_W(20)
[1] 998.2063
> Rho_W(0)
[1] 999.8426
```

5.2. Програмування

R має всі можливості мови програмування високого рівня:

5.2.1. Конструкції if, else, ifelse

Постарайтеся зрозуміти наступне:

```
> Dummy <- function (x)
+ {
+     if ( x<0 ) string <- "x<0" else
+         if ( x<2 ) string <- "0>=x<2" else
+             string <- "x>=2"
+     print(string)
+ }
> Dummy(-1)
[1] "x<0"
> Dummy(1)
[1] "0>=x<2"
> Dummy(2)
[1] "x>=2"
```

Зверніть увагу, що ми вказали вираз «`else`» на тому ж рядку, що і частина «`if`» так, щоб R зрозумів, що вираз триває на наступному рядку!

«`if`»і «`else`»конструкції, що включають тільки один оператор можуть поєднуватися:

```
> x<-2
```

```
> ifelse (x>0, "positive", "negative,0")
[1] "positive"
```

5.2.2. Цикли

Цикли дозволяють виконувати певний набір інструкцій кілька разів.

Цикл «for»перебирає певний набір значень. У наведеному нижче прикладі змінна «i» приймає значення (1,2,3):

```
> for (i in 1:3) print(c(i,2*i,3*i))
[1] 1 2 3
[1] 2 4 6
[1] 3 6 9
```

Цикли «while» і «repeat» виконуватимуться, поки зазначена умова не буде виконана.

```
> i<-1 ; while(i<3) {print(i); i<-i+1}
[1] 1
[1] 2
```

«break»виходить із цикла.
«next» зупиняє поточну ітерацію і переходить до наступного ітерації.

```
> i<-1
> {print(i)
+ i <-i+1
+ if(i>2) break
+ }
[1] 1
```

Фігурні дужки «{...}» охоплюють кілька операторів, які виконуються в кожній ітерації.

Примітка: цикли реалізуються вкрай неефективно в R і їх слід уникати якомога частіше. На щастя, R пропонує безліч команд високого рівня, які працюють з векторами і матрицями. Їх треба використовувати якомога більше!

Для отримання додаткової інформації про, конструкцію «if» і цикли, наберіть

```
> ?Control
```

5.3. R- пакети

Пакет в R являє собою файл, що містить безліч функцій, які виконують певні завдання.

Пакети можна завантажити з офіційного веб-сайту R.

Після установки, для того щоб згенерувати список всіх доступних пакетів, завантажити пакет і отримати список з його вмістом, треба виконати наступні командами:

```
> library()
> library(deSolve)
> library(help=deSolve)
> help(package=deSolve)
```

5.4. Використання R за чотири кроки

R виник як статистичний пакет, і він як і раніше переважно використовується для цієї мети. Ви можете зробити практично будь-який статистичний аналіз в R.

Припустимо, ви хочете виконати ієрархічну кластеризацію і побудувати дендрограму багатовимірного набору даних. Під дендрограмою зазвичай розуміється дерево, тобто граф без циклів, який побудований з матриці подібності. Дендрограма дозволяє зобразити взаємні зв'язки між об'єктами з заданого переліку. Для створення дендрограми потрібно матриця подібності (відмінності), яка визначає рівень подібності між парами об'єктів. Частіше використовуються агломеративні методи. Якщо ви ніколи цього не робили в R,то ось необхідні кроки:

1. Знайти функцію, яка виконує поставлене завдання. Наприклад, використовуючи для допомоги `help.search(«cluster»)`. Залежно від кількості пакетів, які ви встановили, R перерахує ряд можливих функцій,файл довідки яких містить слово «cluster». Використовуйте функціїз пакету статистики (частина ядра R).

2. Відкрийте файл довідки (? <ім'я-функції>) і подивитися синтаксис цієї функції. Якщо у вас немає часу, щоб прочитати його повністю, принаймні, прочитайте (частину) з розділів «Опис», «Використання» і «Приклади».

3. Застосуйте приклади з файлу довідки. Ви можете:
   - Спробувати їх усі відразу (example (<ім'я-функції >)).
   - Крім того, ви можете вибрати окремі приклади, які виглядають застосовними до вашої задачі. Для цього скопіюйте і вставте їх у ваш файл сценарію (Ctrl-C / Ctrl-V) і виконайте їх.

4. Перетворіть багатообіцяючий приклад так, щоб він вирішував вашу задачу.

6. Графіка

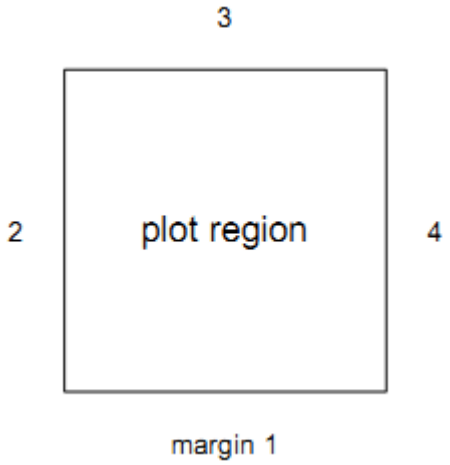
R має широкі графічні можливості. Спробуйте:

```
> demo(graphics)
> demo(image)
> demo(persp)
```

для отримання відображення можливостей R в 1-D, 2-D і 3-D зображеннях.

6.1. Основи

Графіка відображається в окремому вікні. Вся область графіка складається зобласті відображення самого графіка, та 4 граней, що його оточують. Грані нумеруються за годинниковою стрілкою, від 1 до 4, починаючи від нижньої.



R розрізняє:

1. команди високого рівня. За замовчуванням, вони створюють нову фігуру, наприклад,
   - hist, barplot, pie, boxplot, … (1-D графіки)
   - plot, curve, matplot, pairs,… (x-y графіки)
   - image, contour, filled.contour,... (2-D графіки поверхонь)
   - persp, scatterplot3d (3-D графіки)

2. команди низького рівня, які додають нові об'єкти в існуючі фігури, наприклад,

- lines, points, segments, polygon, rect, text, arrows, legend, abline, locator, rug, ...які додають об’єкти в область графіка
- box, axis, mtext (текст на гранях), title, … які додають об’єкти на грані графіка

3. графічні параметри, які контролюють зовнішній вигляд.
   - Графічні об’єкти:cex (розмір тексту і символів), col (колір), font, las (орієнтація надпису осей), lty (тип ліній), lwd (ширина ліній), pch (тип точок),…
   - Графічне вікно:mar (розмір граней), mfrow (кількість фігур на рядку), mfcol(кількість фігур у стовпці),…

```
> ?plot.default
> ?plot.window
> ?par
> ?points
```

відкриє файли довідки, в той час як

```
> example(plot.default)
> example(points)
```

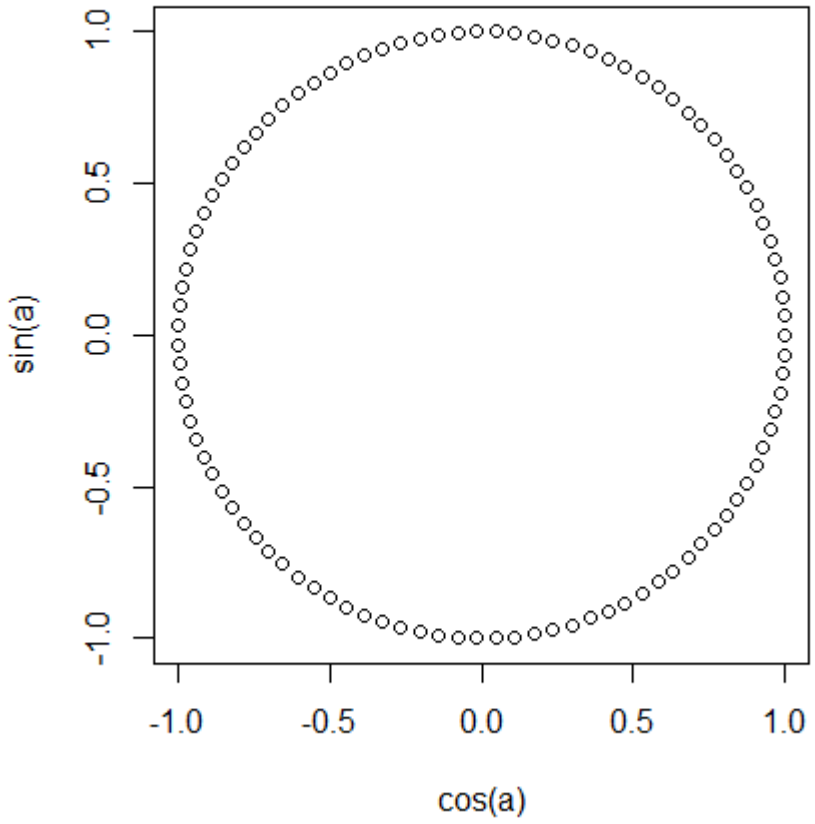
буде запускати приклади, показуючи кожен новий графік, після натискання клавіші <ENTER> (спробуйте!)

6.2. X-Y графіки

Круг можна побути по набору (x, y) точок, де x= r*cos(α) і y= r*sin(α), з кутом α, від 0 до 2π, та радіусом r.

У наступному прикладі ми спочатку створили послідовність значень кута α, від 0 до 2π, що включає 100 значень (length.out), а потім побудували коло з одиничним радіусом:

```
> a <- seq(0,2*pi, length.out=100)
> plot(x=cos(a),y=sin(a))
```



«plot» – це команда високого рівня, вона створює нову фігуру.

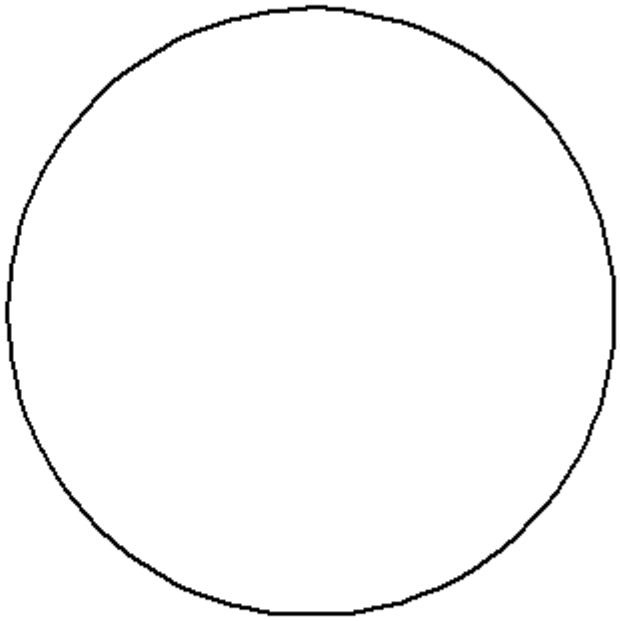
За замовчуванням, R додає осі і надписи осей, і представляє (x, y) дані у вигляді маленьких крапок. Зауважимо, що графік не є симетричним.

Тепер ми зробимо більш складну фігуру, яка нагадує мішень, наприклад, для практикуючих стрілянину з лука або метання дротиками.

Спочатку використаємо ту ж команду («plot»), що описано вище, але ми додамо ряд графічних параметрів, які визначають, що:

- замість крапок, точки повинні бути з'єднані лініями (type);
- лінія повинна бути в два рази ширше ніж за замовчуванням (lwd) ;
- надписи X і У-осей (xlab, ylab) повинні бути порожніми ;
- осі і анотації осей (axes) повинні бути видалені ;
- графік має бути симетричним, тобто x / y співвідношення сторін = 1(asp).

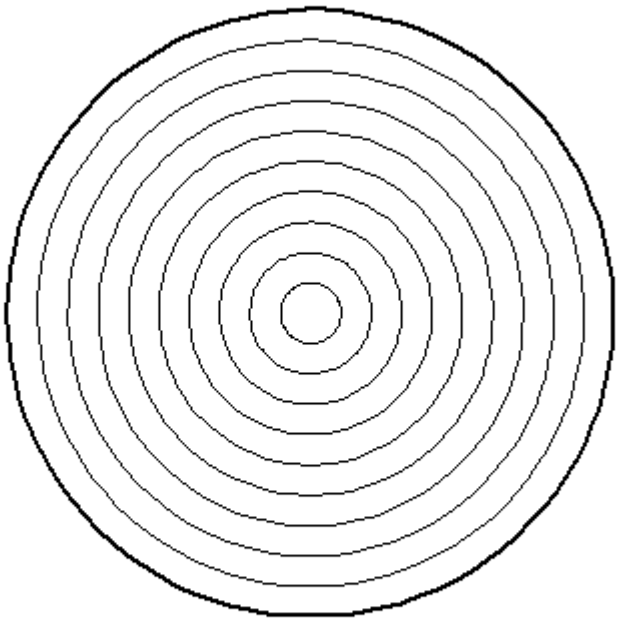
```
     >plot(cos(a),sin(a),type="l",lwd=2,xlab="",ylab="",axes=FAL
SE, asp=1)
```



До цієї фігури, ми можемо тепер додати кілька об'єктів низького рівня:

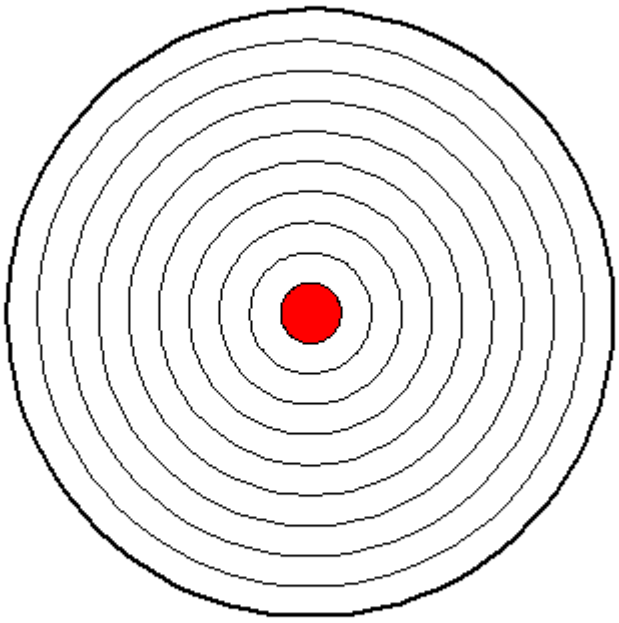
- серія ліній, що представляють все менші й менші кола (lines).

```
> for (i in seq( 0.1,0.9,by=0.1)) lines(i*sin(a), i*cos(a))
```



- внутрішній червоний багатокутник (polygon).

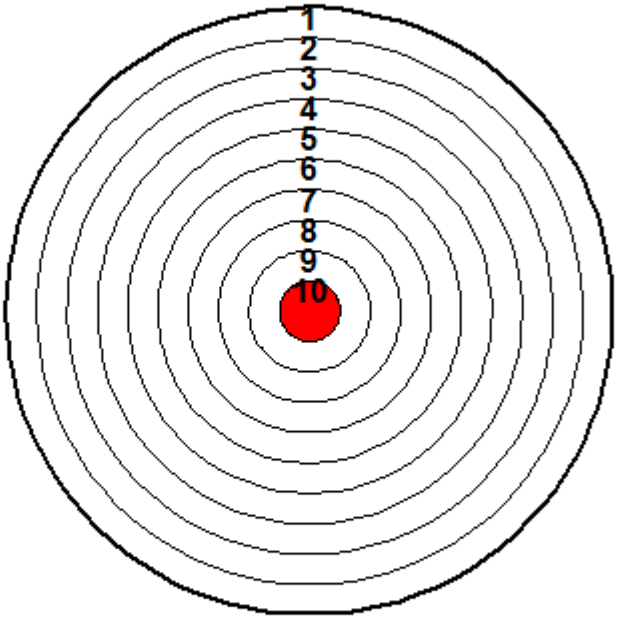
```
> polygon(sin(a)*0.1,cos(a)*0.1,col="red")
```



- точки позначити текстовими мітками, починаючи від 10 до 1 (text).

Чим ближче до центру, тим більше значення

> for (i in 1:10) text(x=0,y=i/10-0.025,labels=11-i,font=2)



Тепер два лучника зробили по 10 пострілів у мішень. Ми імітуємо їхні постріли, генеруючи нормальний розподіл (x, y) чисел, з математичним очікуванням = 0 (центр!) і де досвід лучника імітується за допомогою стандартного відхилення. Лучник більш досвідчений, чим ближче стріли будуть до центру, тобто чим нижче стандартне відхилення.

- R-вираз `rnorm` генерує нормально розподілені числа; нам потрібні 20 з них, які розташовані у вигляді матриці з двомастовпчиками.

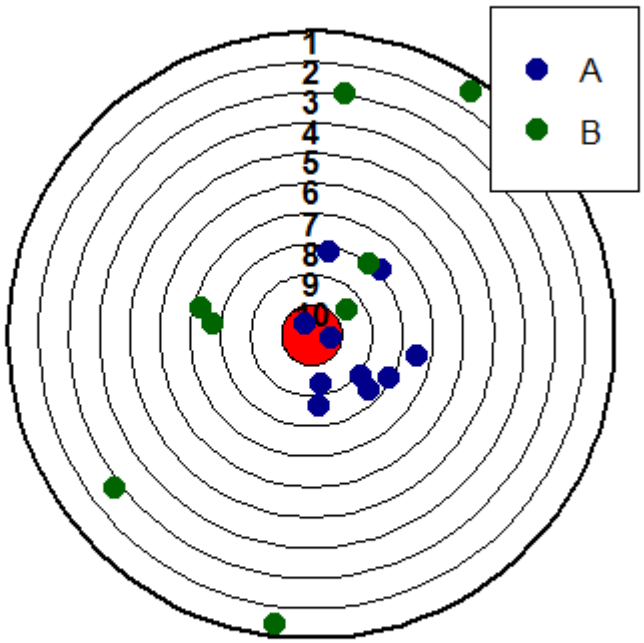
```
> shots1 <- matrix(ncol=2, data=rnorm(n=20,sd=0.2))
> shots2 <- matrix(ncol=2, data=rnorm(n=20,sd=0.5))
```

- Постріли додаються до фігури як точки (`points`), розфарбовуються синім `darkblue` (досвідчений стрілець) і темно-зеленим `darkgreen` (початкового рівня стрілець). Зверніть увагу, що ми вибираємо на 50% збільшений розмір точок (`cex`), і ми вибираємо точки круглої форми (`pch = 16`)

```
> points(shots1,col="darkblue",pch=16,cex=1.5)
> points(shots2,col="darkgreen",pch=16,cex=1.5)
```

- Нарешті, ми додаємо легенду, пояснюючи, хто зробив постріли:

```
> legend("topright",legend=c("A","B"),pch=16,
+        col=c("darkblue","darkgreen"),pt.cex=1.5)
```



Зверніть увагу, що текст і колір легенди вводяться у вигляді вектора рядків, використовуючи функцію «`c ()`» (наприклад, `c («A», «B»)`).

6.3. X-Y графіки – умовна побудова

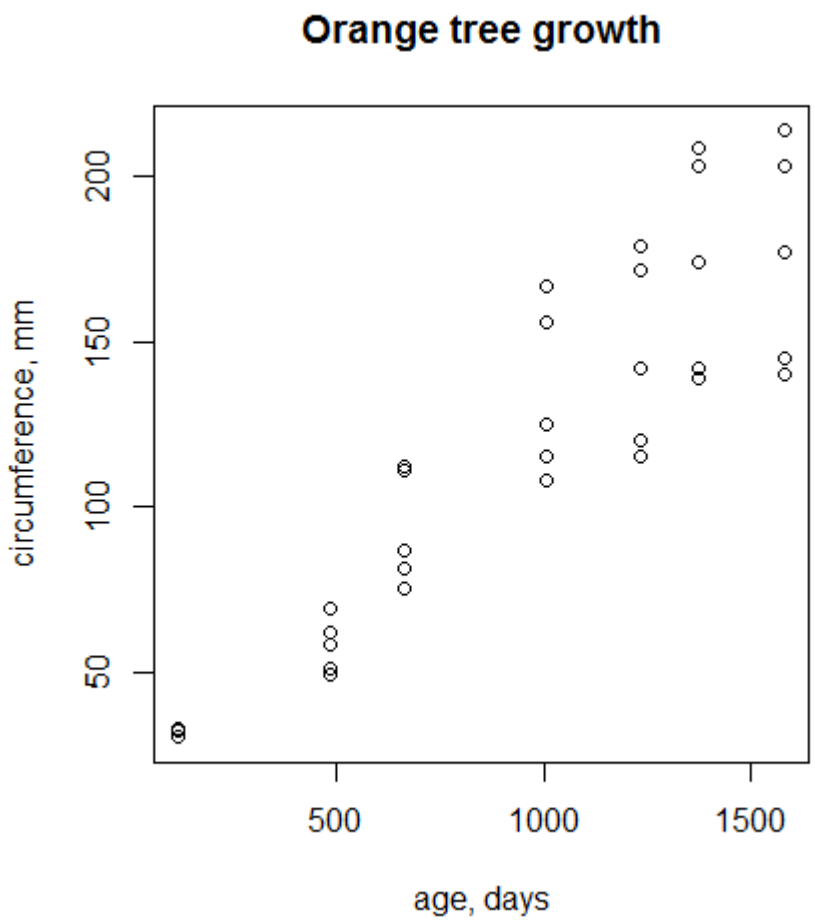
У якості більш витонченої демонстрації використання символів в R-графіках, ми будемо працювати на біологічному прикладі, із набору даних R під назвою «`Orange`».

Цей набір даних містить окружності (у мм, на висоті грудей), виміряних в різному віці на п'яти апельсинових деревах. Ми почнемо з розгляду даних (відображається тільки частина).

```
>head(Orange)
Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142

> tail(Orange)
   Tree  age circumference
30    5  484            49
31    5  664            81
32    5 1004           125
33    5 1231           142
34    5 1372           174
35    5 1582           177
```

і зробимо грубий графік окружностей в залежності від віку:

```
>   plot(Orange$age,   Orange$circumference,xlab="age,   days",
ylab="circumference, mm", main= "Orange tree growth")
```



(Оскільки Orange є data.frame, колонки можуть бути розглянуті за їх іменами, `Orange$age` і `Orange$circumference`).

Результат (рисунок) показує, що існує великий розкид данних, що пов'язано з тим, що п'ять дерев не ростуть з однією швидкістю.

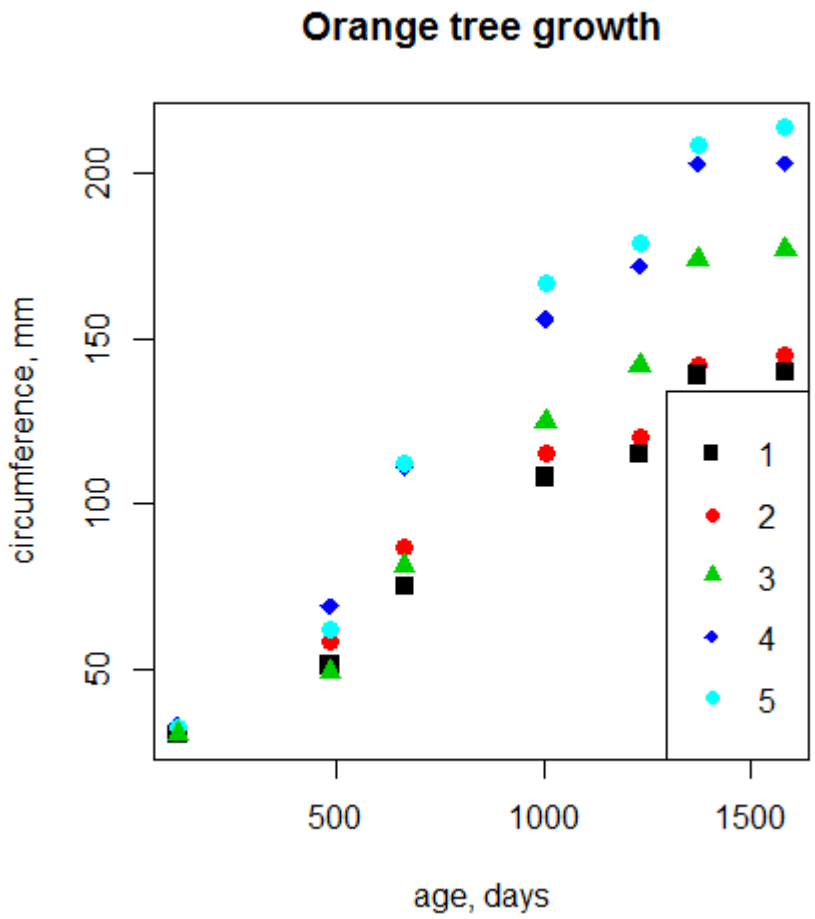
Корисно побудувати залежність між окружністю і віком окремо для кожного дерева. В R, це зробити просто: ми можемо використати деякі графічні параметри (типи символів, кольору, розміру, ...) в залежності від певних «факторів». Фактори відіграють дуже важливу роль в статистичних використаннях R. Для нашого застосування, досить знати, що фактори є цілими числами, починаючи з 1.

В R-виразі нижче, ми просто використовуємо різні символи (`pch`) і коліру (`col`) для кожного дерева: `pch=(15:20)[Orange$Tree]` означає, що, в залежності від значення `Orange$Tree` (тобто номера дерева), символ (`pch`) прийме значення 15 (дерево = 1), 16 (дерево = 2), ... 20 (дерево = 5). `col=(1:5) [Orange$Tree]` робить те ж саме для кольору точок. В останньому виразі додається легенда, розташована у правому нижньому куті.

```
> plot(Orange$age,
Orange$circumference,xlab="age,days",ylab="circumference,mm"
,main= "Orange tree
growth",pch=(15:20)[Orange$Tree],col=(1:5)
[Orange$Tree],cex=1.3)

> legend("bottomright",pch=15:20,col=1:5,legend=1:5)
```

Результат (рисунок) показує, що дерево число номер 5 росте швидко, а дерево номер 1 повільно зростає.

(Зверніть увагу: корисно виконати приклади з файлу довідки Orange.)



6.3.1. Темпи росту зоопланктону

«Zoogrowth» з пакету «marelac» представляє собою набір літературних даних, складений Хансена та ін. (1997) з результатами вимірювань максимальних швидкостей росту організмів зоопланктону в залежності від об'єму тіла. Запустіть приклад для цього набору даних (вам

спочатку потрібно буде завантажити пакет «marelac»виконавши install.packages(«marelac»)):

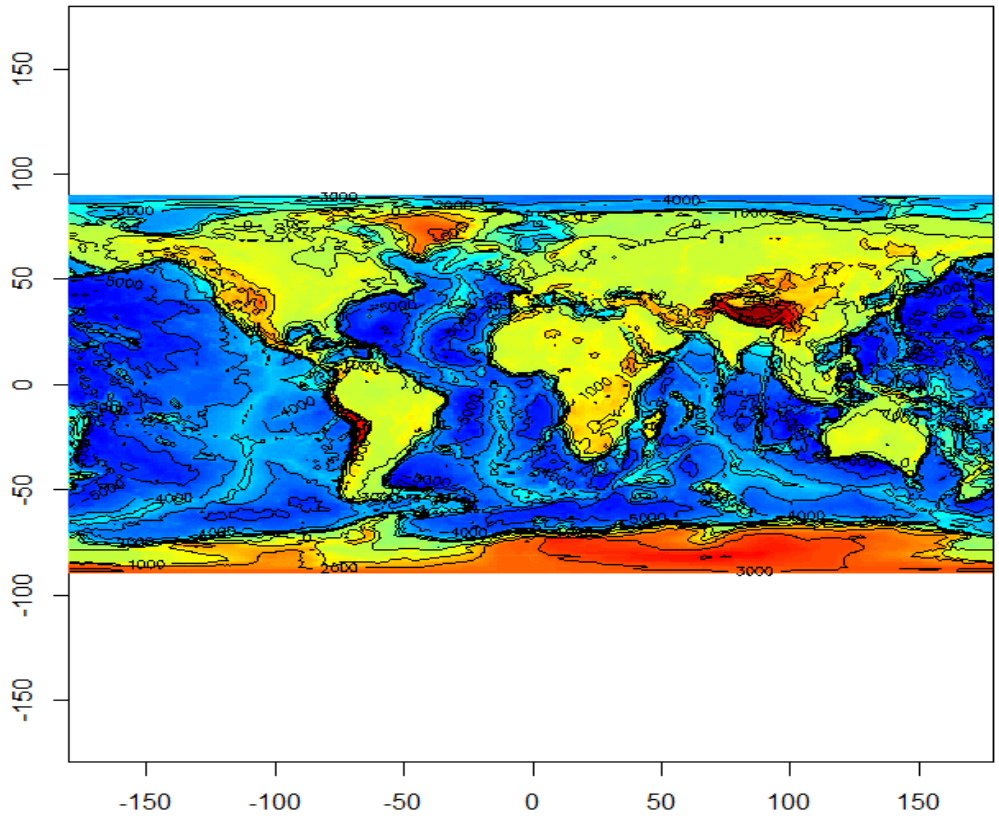
```
> require(marelacTeaching)
> example(Zoogrowth)
```

6.4. Зображення та контурні графіки

R має деякі дуже потужні функції для створення зображень і додавання контурів. Наприклад, набір даних «Bathymetry» (сукупність даних про глибини водного об'єкта) з пакету marelac може використовуватися для генерації батіметрії (і гіпсометрії) Світового океану (і землі):

```
> require(marelac)
>image(Bathymetry$x,Bathymetry$y,Bathymetry$z,col=femmecol(
100,asp=TRUE,xlab="",ylab="")

> contour(Bathymetry$x,Bathymetry$y,Bathymetry$z,add=TRUE)
```

Зверніть увагу на використання «asp=TRUE», яка підтримує співвідношення сторін.



6.5. Побудова математичних функцій

Зображенняграфіків математичних функцій будуються швидко за допомогою R-команди«curve»:

```
>curve(sin(3*pi*x))
```

Намалює графік для y=sin(3*π*x), використовуючи налаштування за замовчуванням (рис. ліворуч), в той час як:

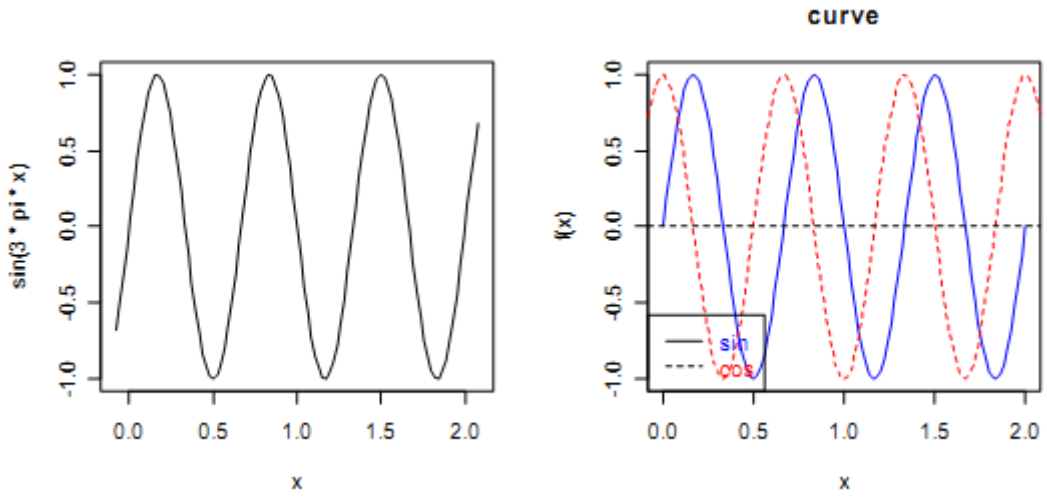
```
> curve(sin(3*pi*x),from=0,to=2,col="blue",
+ xlab="x",ylab="f(x)",main="curve")

> curve(cos(3*pi*x),add=TRUE,col="red",lty=2)
```

спочатку малює графік y=sin(3*π*x), синього кольору (col), і для значень x в діапазоні від 0 до 2 (from, to), додаючи назву графіка (main) та назви для X і Y осей (xlab, ylab) (перша команда).

Друга R-команда додає графік функції y=cos(3*π*x), червоного кольору (col) пунктирною лінією (lty). Зверніть увагу на використання параметра «add=TRUE», оскільки команда «curve» за замовчуванням створює новий графік.

Остання команда додає вісь X, пунктирною (lty=2) лінією (abline) при y = 0 і легенду.

```
> abline(h=0,lty=2)
> legend("bottomleft",c("sin","cos"),
+text.col=c("blue","red"),lty=1:2)
```



6.6. Декілька фігур в одному вікні

Є декілька способів, як можна відображати декілька фігур на одному графіку.

1. Найпростіший, вказати кількість фігур у рядку (`mfrow`) і в стовпчику (`mfcol`):

```
> par(mfrow=c(3,2))
```

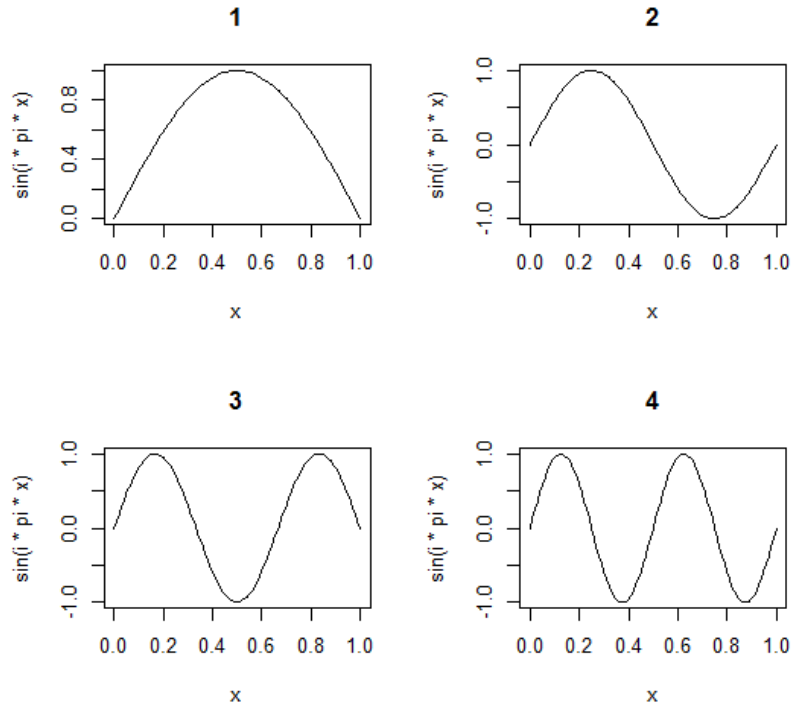
організовує наступні графіки в 3 рядки і 2 колонки. Графіки будуть побудовані по-рядково.

```
>par(mfcol=c(3,2))
```

організує наступні графіки в 3 колонки і 2 рядки по стовпцях.

Слід зазначити, що як `mfrow`так і`mfcol` повинні бути введені в якості вектора. Спробуйте:

```
> par(mfrow=c(2,2))
> for ( i in 1:4) curve(sin(i*pi*x),0,1,main=i)
```



2. R-функція «layout» дозволяє набагато більш складні налаштування відображення дкількох графіків.

7. Робота з матрицями

Працювати з матрицями в R – дуже просто. Практично все можливо!
Ось найбільш важливі R-функцій, які працюють на матрицях:

- `%*%` множення матриць
- `t(A)` транспонування матриці A
- `diag(A)` діагональна матриця до A
- `solve(A)` обернена матриця до A
- `solve(A,B)` розв'язок матричного рівняння$Ax=B$ відносно x
- `eigen(A)` власні значення і власні вектори матриці A
- `det(A)` визначник матриці A

Наприклад:

```
> (A <-matrix(nrow=2,data=c(1,2,3,4)))

     [,1] [,2]
[1,]    1    3
[2,]    2    4

> solve(A) %*% A
```

спочатку обчислюється матриця обернена до A (`solve(A)`), і множиться на A (%*%), результатом є одинична матриця:

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

в той час як `t(A)` виконає транспонування матриці A (перестановка рядків і стовпців).

```
> t(A)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Наступний набір операторів розв'яже лінійну систему $Ax = B$ для невідомого вектора x:

```
> B <- c(5,6)
> solve(A,B)
[1] -1  2
```

Нарешті, власні значення і власні вектори матриці A оцінюються з використанням R-функції «eigen». Ця функція повертає список, що містить як власні значення($values) так і власні вектори ($vectors), (стовбчики).

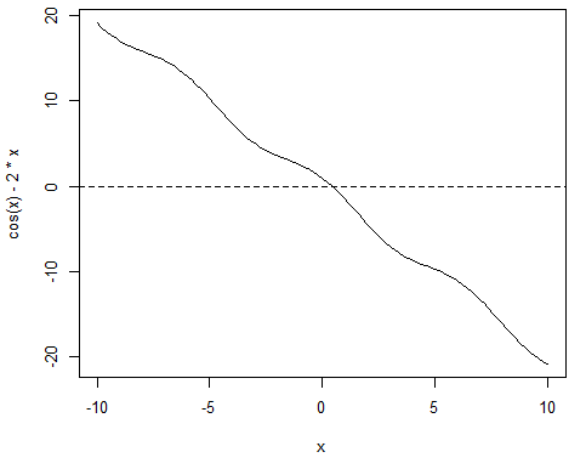
```
> eigen(A)
$values
[1]  5.3722813 -0.3722813

$vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

8. Корені функцій

8.1. Корінь функції однієї змінної

Припустимо, ми хочемо розв'язати наступну задачу: `cos(x) = 2*x`відноснох.Математично, ми шукаємо корінь функції `y = cos(x) - 2*x`, це значення x, при яких y = 0. Оскільки, функція є досить складною, то не можливо знайти точнийрозв'язок (явний вираз) для цього кореня. Це завжди гарна ідея, розв'язати дане рівняння графічним методом.

```
>curve(cos(x)-2*x,-10,10)
> abline(h=0,lty=2)
```



З графіка видно, що насправді існує таке значення x, для якого Y = 0.

Тепер можна використати R-функцію«`uniroot»,` щоб знайти цезначення.

Функції, які шукають корінь нелінійного рівняння як правило, працюють «ітераційно», тобто вони наближаються все ближче і ближче до кореня поетапно (ітераційно).

Ці методи, як правило, не мають можливості, щоб знайти цей корінь точно, а знаходять корінь з певною точністю (`tol`, дуже маленькезначення). Більш конкретно: коренямиу `= cosx-2 * x`є значення x, для яких| `cosx-2 * x` | <`tol ,` або для яких послідовні зміни x є < `tol`

Для того, щоб метод працював, повинен бути принаймні один корінь на інтервалі. Вираз нижче знаходить корені рівняння; він повертає кілька значень, у вигляді списку.

```
> uniroot(f = function(x) cos(x)-2*x, interval=c(-10,10))
$root
[1] 0.4501686

$f.root
[1] 3.655945e-05

$iter
[1] 5

$init.it
[1] NA

$estim.prec
[1] 6.103516e-05
```

Найважливіше значення сам корінь ($root), який є 0.4501686; значення функції в корені було 3.66e-5, метод виконав 5 ітерацій.

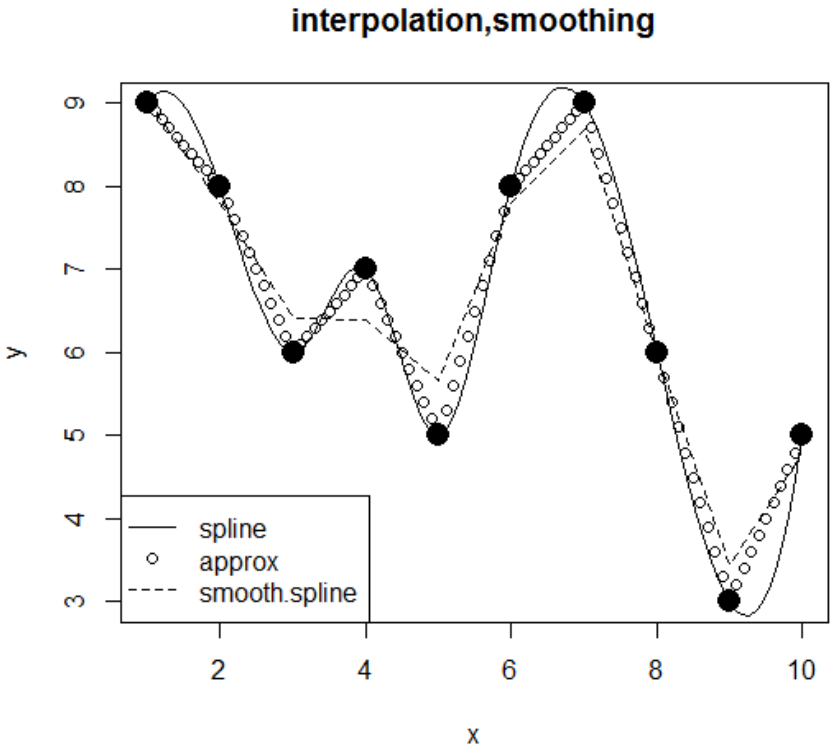
9. Інтерполяція, згладжування

Інтерполяції і згладжування в R можна зробити декількома способами:

- `approx` лінійно інтерполює через точки
- `spline` використовує інтерполяцію сплайн, яка є більш гладкою
- `smooth.spline` розгладжує набори даних; це означає, що цей метод не пов'язує оригінальні точки.

Використання цих функцій ілюструється наступним сценарієм і відповідним результатом:

```
> x <- 1:10
> y <- c(9,8,6,7,5,8,9,6,3,5)
> plot(x,y,pch=16,cex=2,main="interpolation,smoothing")
> lines (spline(x,y, n=100),lty=1)
> points(approx(x,y, xout=seq(1,10,0.1)),pch=1)
> lines (smooth.spline(x,y),lty=2)
> legend("bottomleft",lty=c(1,NA,2),pch=c(NA,1,NA),
+ legend=c("spline","approx","smooth.spline"))
```



10. Диференціальні рівняння

Диференціальні рівняння виражають швидкість зміни функції уздовж однієї або декількох розмірностей, зазвичай часу та/або простору.

Розглянемо наступнусистему з двох диференціальних рівнянь:

$$\frac{dA}{dt} = r \cdot (x - A) - k \cdot A \cdot B$$

$$\frac{dB}{dt} = r \cdot (y - B) + k \cdot A \cdot B$$

- `A` і `B` називаються диференціальними змінними (або змінними стану),
- $\frac{dA}{dt}$ є похідною (або швидкістю зміни),
- r,x,y і k параметри (константи).

Для розв'язання систем диференціальних рівнянь в R ми визначимо функцію (тут називається `model`), яка має в якості вхідних параметрів час (`t`), значення змінних стану (`state`) і значення параметрів (`pars`). Ця функція просто обчислює швидкість зміни змінних стану (`dA` і `dB`) і повертає їх у вигляді списку. R-вираз«`with (as.list (c(state,pars)),{ }`»гарантує, що змінні стану і параметри можуть бути розглянуті за їх іменами.

```
> model <- function(t,state,pars)
+ {
+     with (as.list(c(state,pars)),
+ {
+     dA <- r*(x-A)-k*A*B
+     dB <- r*(y-B)+k*A*B
+     return (list(c(dA,dB)))
+ }
+     )
+ }
```

Перш ніж ми зможемо розв'язати цю модель, потрібно

- згенерувати послідовність значень часу, в якому ми хочемо отримати результат (`times`),

- задати початкові умови для змінних стану (state) та
- задати значення для параметрів (parms):

```
> times <- seq(0,300,1)
> state <- c(A=1,B=1)
> parms <- c(x =1, y = 0.1, k = 0.05, r = 0.05)
```

Модель тепер може бути розв'язана. Щоб зробити це, ми використовуємо функцію інтегрування ode з R, яку можна знайти в R-пакеті deSolve. Спочатку потрібно завантажити цей пакет.

```
> require(deSolve)
```

У кожен момент часу t, «ode» викликає функцію «model», з поточними значеннями змінних стану і значень параметрів. Результат зберігається в data.frame, що називається «out».

```
> out <- as.data.frame(ode(state,times,model,parms))
```

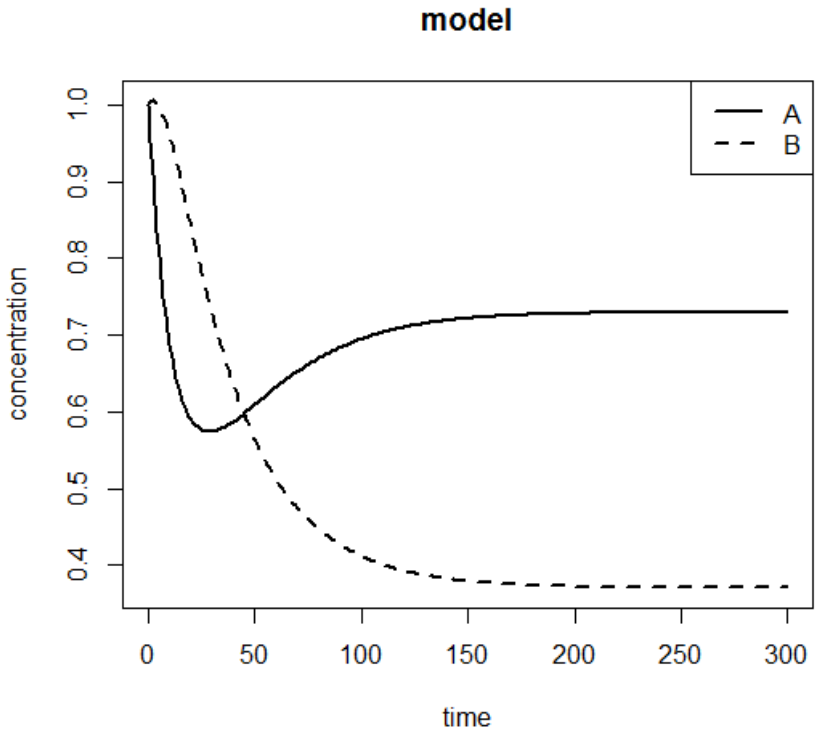
Все, що ми повинні зробити зараз, це побудувати графік моделі. Перш, ніж ми це зробимо, ми повинні переглянути отримані результати в data.frame«out»:

```
> head(out)

time          A         B
1         0    1.0000000      1.0000000
2         1    0.9523189      1.0037869
3         2    0.9090687      1.0052854
4         3    0.8699226      1.0047151
5         4    0.8345728      1.0022854
6      5  0.8027203      0.9982009
```

Дані розташовані в три стовпчики: перший – час, наступні – значення A і B. Оскільки результат «out»це data.frame, то ми можемо мати доступ до даних, використовуючи їх імена (out$time, out$A, out$B).

Перед побудовою графіка моделі, діапазон значень A і B оцінюється; це виконується для того, щоб встановити межі осі y (ylim). R-функція plot створює новий графік; lines додає лінію на цей графік; ltyвибирає тип лінії; lwd=2робить лінії в два рази товще ніж за замовчуванням;legend додає легенду.

```
> ylim <- range(c(out$A,out$B))
> plot(out$time,out$A,xlab="time",ylab="concentration",
+      lwd=2,type="l",ylim=ylim,main="model")
> lines(out$time,out$B,lwd=2,lty=2)
> legend("topright",legend=c("A","B"),lwd=2,lty=c(1,2))
```



11. Статистика

Ось деякі пакети, які часто використовуються:

• `fastR` (пакет з деякими хорошими утилітами; є на CRAN)
• `lattice` (пакет для графіки)
• `mosaic` (для вивчення статистики; є на CRAN)
• `Hmisc` (пакет з деякими хорошими утилітами; є на CRAN)

11.1. Обробка числових даних

R включає в себе функції, які обчислюють широкий спектр чисельних і графічних даних для статистичного аналізу.

Для прикладу будемо використовувати набір даних `mtcars`. Дані витягуються з журналу Motor Trend в США1974 року, і включають в себе споживання палива і 10 аспектів автомобільного дизайну і продуктивності для 32 автомобілів (моделей 1973-1974 років).

Цей набір даних містить наступні дані:

Індекс	Назва змінної	Опис
[,1]	mpg	Міль/галон(США)
[,2]	cyl	Кількість циліндрів
[,3]	disp	Об’єм (в кубічних дюймах)
[,4]	hp	Механічна кінська сила
[,5]	drat	Передатне відношенняголовної передачі
[,6]	wt	Вага (фунт/1000)
[,7]	qsec	Час проходження ¼ милі
[,8]	vs	V/S
[,9]	am	Передача (0 = автоматична, 1 = механічна)
[,10]	gear	Кількість передач
[,11]	carb	Кількість карбюраторів

Декілька прикладів обчислення різних статистичних значень:

Середнє значення набору даних `mtcars`по показнику mpg:

```
> mean(mtcars$mpg)
[1] 20.09062
```

Медіана набору даних mtcarsпо показнику mpg:

```
> median(mtcars$mpg)
[1] 19.2
```

Квантиль набору даних mtcarsпо показнику mpg:

```
> quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900
```

Інтерквантиль набору даних mtcarsпо показнику mpg:

```
> IQR(mtcars$mpg)
[1] 7.375
```

Функція favstats()з пакетуmosaicобчислює декілька статистичних значеньодразу.

```
> favstats(mtcars$mpg)
  min     Q1 median   Q3  max     mean       sd  n missing
 10.4 15.425   19.2 22.8 33.9 20.09062 6.026948 32       0
```

В пакеті Hmisc є функція summary() яка віводить данні в зручному вигляді.

```
> summary(mpg ~ am, data = mtcars, fun=favstats)
mpg    N=32

+-------+---+--+----+------+------+----+----+--------+--------+--+--------+
|       |   |N |min |Q1    |median |Q3  |max |mean    |sd      |n |missing |
+-------+---+--+----+------+------+----+----+--------+--------+--+--------+
|am     |No |19|10.4|14.950|17.3   |19.2|24.4|17.14737|3.833966|19|0       |
|       |Yes|13|15.0|21.000|22.8   |30.4|33.9|24.39231|6.166504|13|0       |
+-------+---+--+----+------+------+----+----+--------+--------+--+--------+
|Overall|   |32|10.4|15.425|19.2   |22.8|33.9|20.09062|6.026948|32|0       |
+-------+---+--+----+------+------+----+----+--------+--------+--+--------+
```

11.2. Граткова графіка

Є кілька способів, щоб побудувати графіки в R. Один з підходів є використовування системи граткових графіків. Першим кроком для використання граток («lattice»), потрібно завантажити пакет «lattice»,

використовуючи прапорець на вкладці Packages або за допомогою наступної команди:

```
> require(lattice)
```

граткові графіки використовують формульний інтерфейс:

```
> plotname( y ~ x | z, data=dataname, groups=grouping_variable, ...)
```

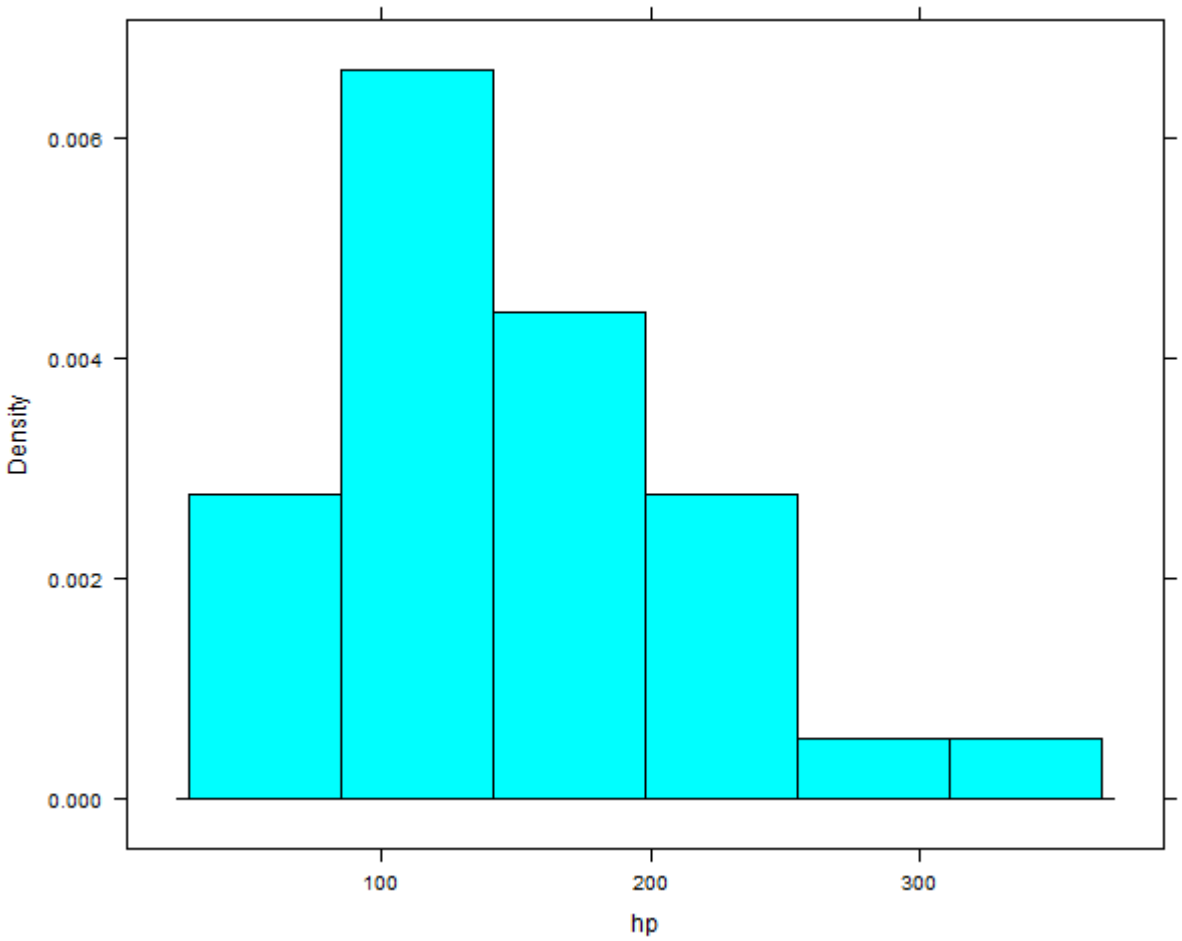
- Ось імена кількох граткових графіків:
  - histogram(для гістограм)
  - bwplot (для коробкових графіків або графіків«ящик з вусами»)
  - xyplot (для діаграм розсіювання)
  - qqmath(для графіків квантиль-квантиль (Q-Q))
- x це ім'я змінної, яка відкладається по горизонтальній (*x*) осі.
- y це ім'я змінної, яка відкладається по вертикальній (*y*) осі. (Для деяких графіків, цеполе порожнє, бо R обчислює ці значення із значень x.)
- zзміннастану, використовується для поділу графіка на декілька областей, так званих панелей.
- grouping_variableвикористовується для відображення різних груп по-різному (різні кольори або символи, наприклад) в рамках тієї ж панелі.
- ... Є багато додаткових аргументів цих функцій, які дозволяють контролювати, як виглядає графік.

11.3. Гістограма

Гістограма — спосіб графічного представлення табличних даних. Являє собою діаграму, що складається з прямокутників без розривів між ними.

Наприклад, щоб побудувати гістограму набору даних mtcarsпо показнику hp, треба:

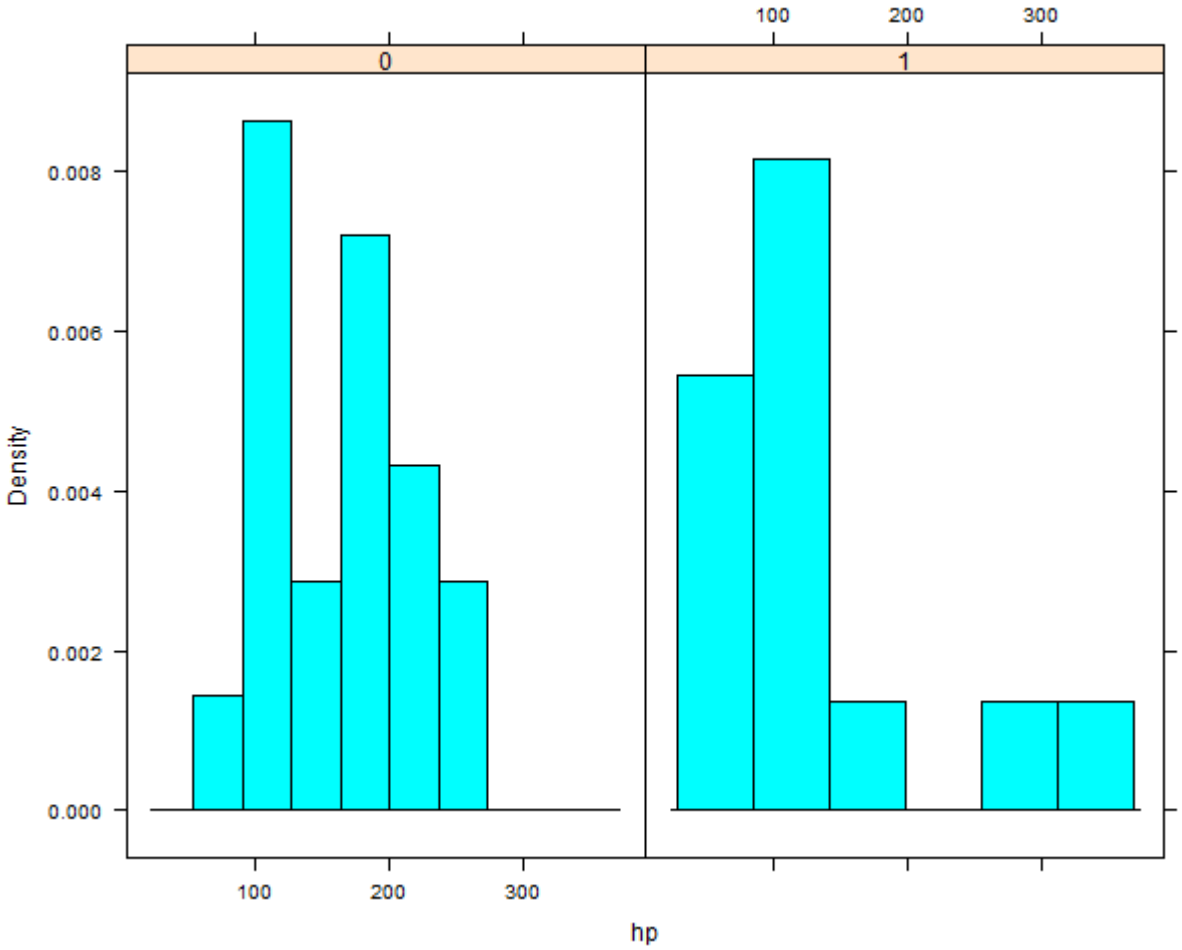
```
> histogram(~ hp, data=mtcars)
```



У компонент формули порожній, так як ми дозволяємо R обчислити висоту стовпців.

Ми можемо використовувати змінністану, щоб побудувати окремі гістограми для кожного значенняцієї змінної. Наприклад,щоб побудувати гістограму набору даних `mtcars`по показнику `hp`в залежності від значення параметра`am`, треба:

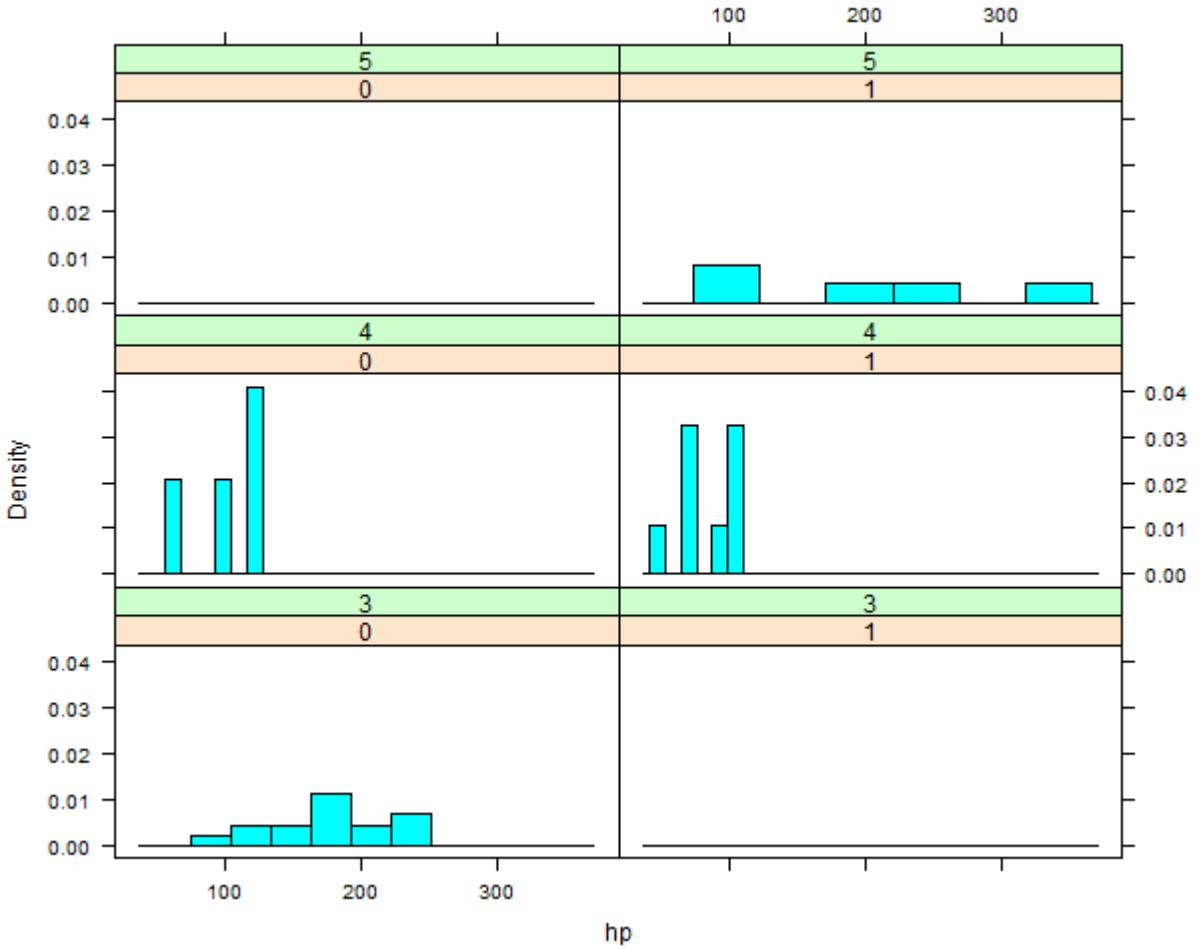
```
> histogram(~ hp | factor(am), data=mtcars)
```



На графіку зображено дві підгрупи, які називаютьсяпанелями, та надписизверху, які називаються смужками. (Смужки можуть бути розміщені на лівій стороні, якщо ви віддаєте перевагу.)

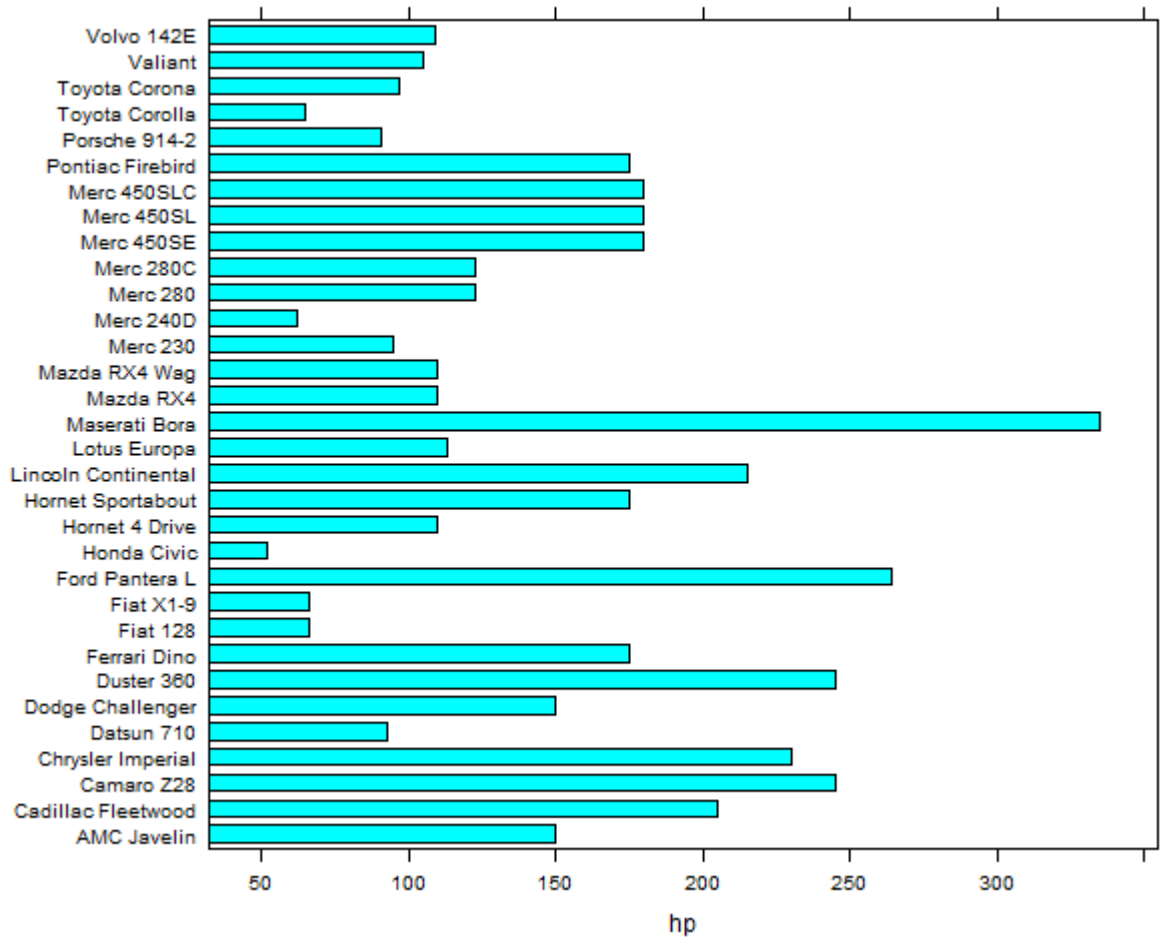
Ми можемо використовувати одразу декілька змінних:

```
> histogram(~ hp | factor(am) + factor(gear), data=mtcars)
```

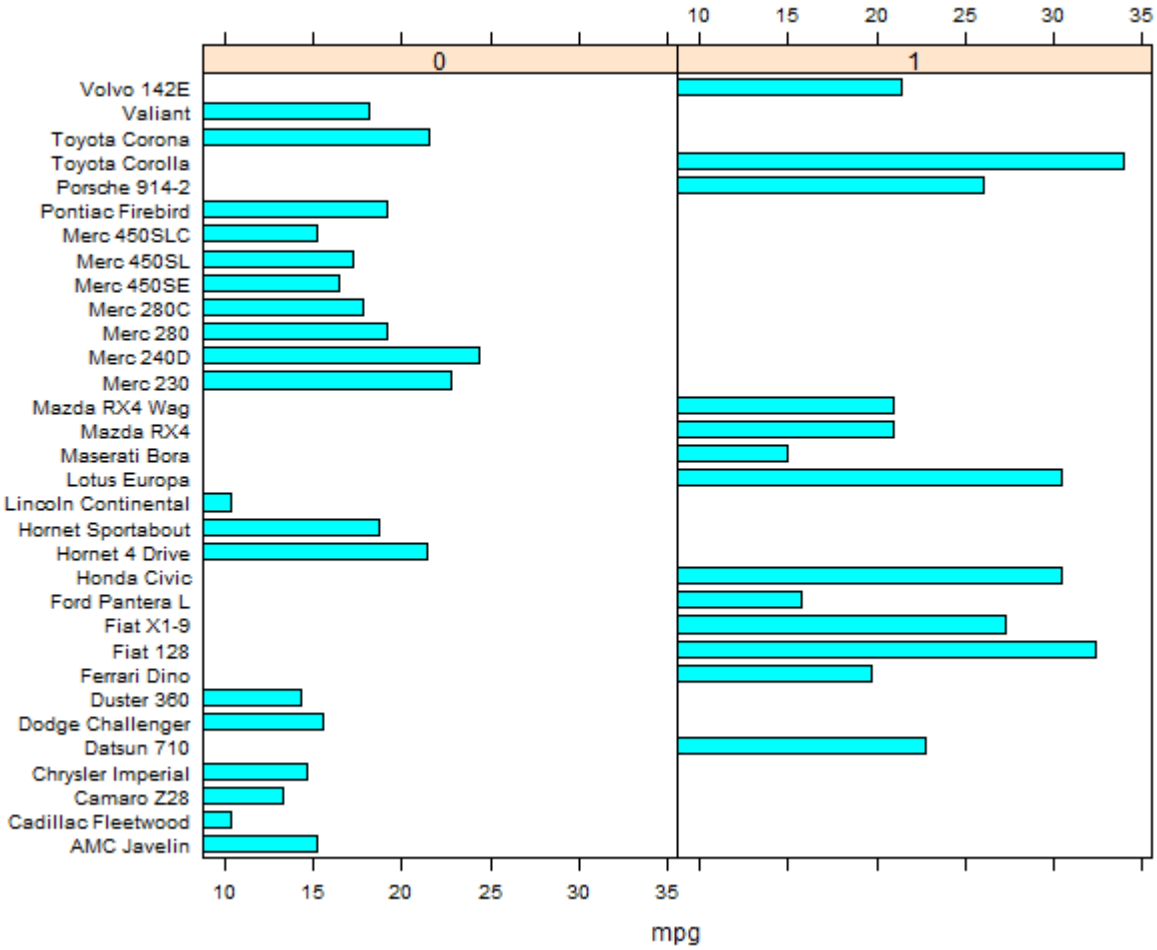


Ще одним із видів гістограм є стовпчикова діаграма. Щоб її побудувати, треба:

```
>barchart(row.names(mtcars) ~ hp, data = mtcars)
```



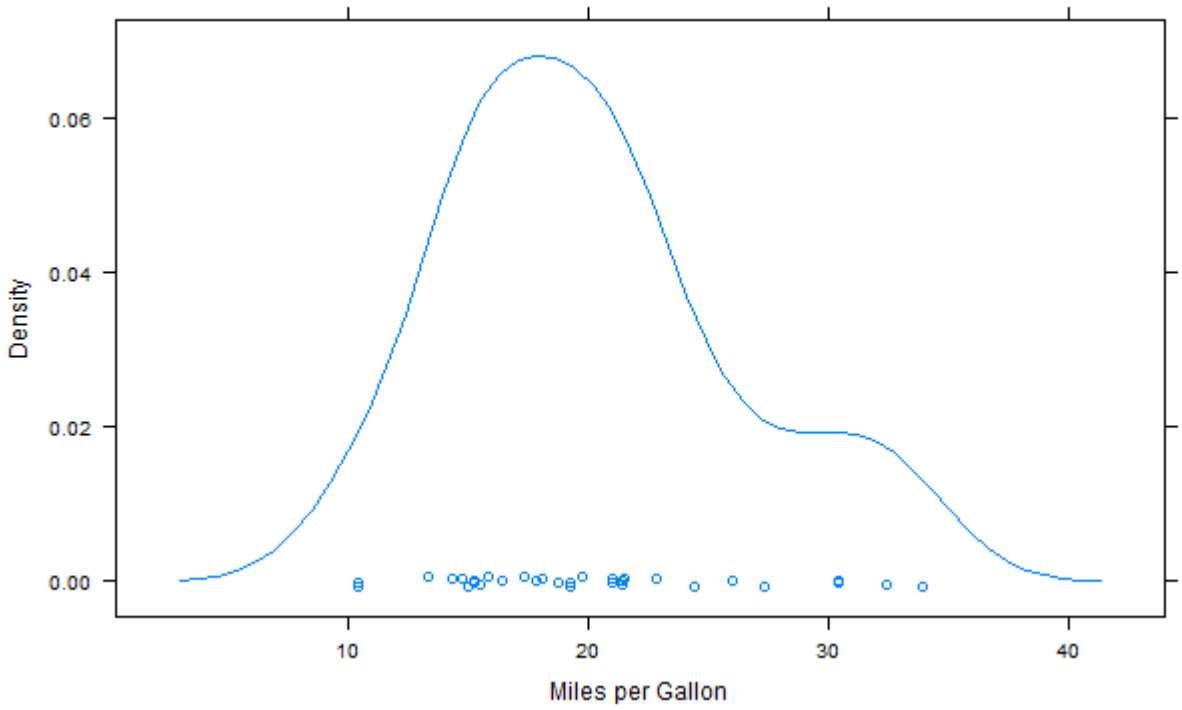

```
> barchart(row.names(mtcars) ~ mpg | factor(am), data = mtcars)
```



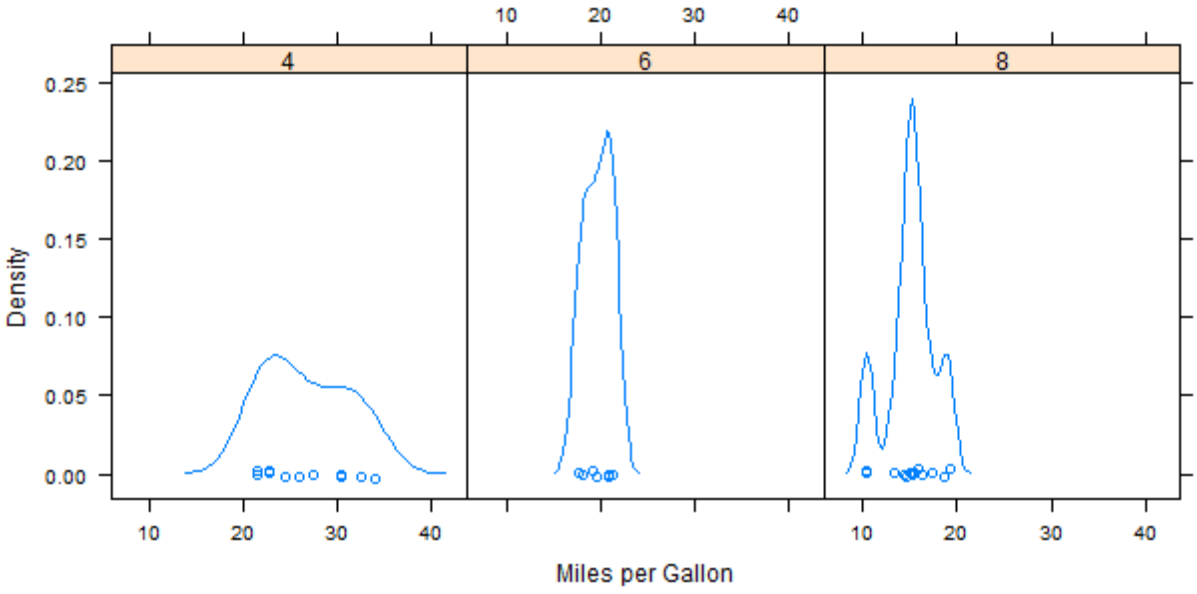
11.4. Графік щільності (густини) ймовірності

Функція `densityplot()`будує оцінку щільності ядра в одновимірному наборі даних; тобто оцінку функції густини ймовірностей.

```
> densityplot(~mpg, data=mtcars, xlab=«Miles per Gallon»)
```

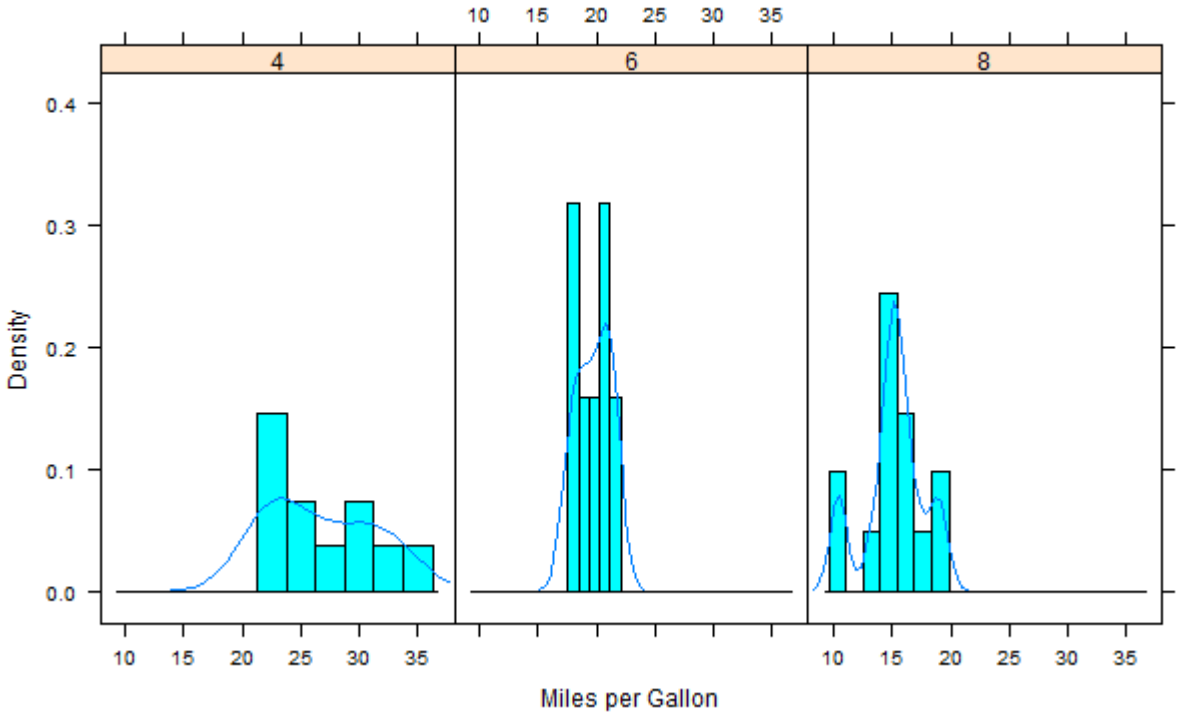



```
> densityplot(~mpg|factor(cyl), data=mtcars, xlab=«Miles per Gallon»)
```



Для більшої зручності, можна побудувати гістограму та графік густини ймовірностей разом:

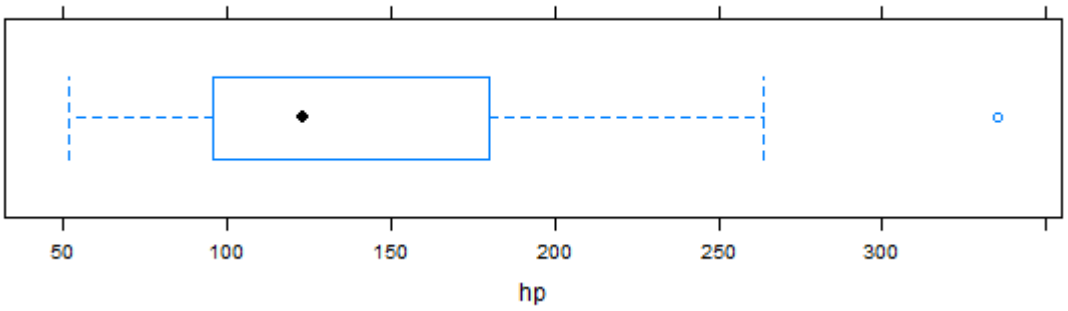
```
>histogram(~mpg|factor(cyl), data=mtcars, xlab=«Miles perGallon»,type='density',panel=function(...)
{panel.histogram(...); panel.densityplot(...)})
```



11.5. Графік «ящик з вусами»

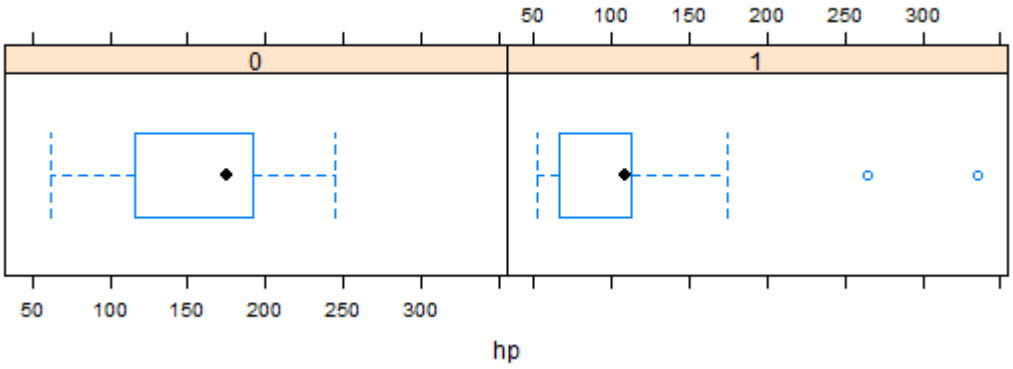
Функція bwplotгенерує графік типу «ящик з вусами». Ця функція використовується таким самим чином, як і функція що будує гістограми.

```
> bwplot(~ hp, data=mtcars)
```



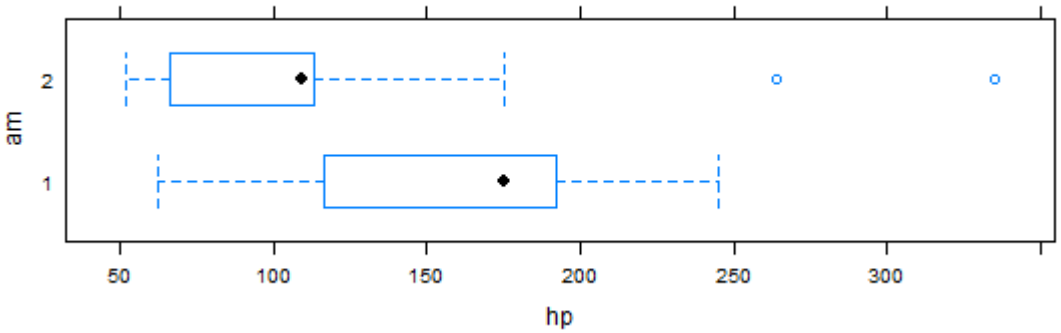
Ми можемо використовувати умовні змінні, так як ми робили і для гістограм:

```
> bwplot(~ hp | format(am), data=mtcars)
```

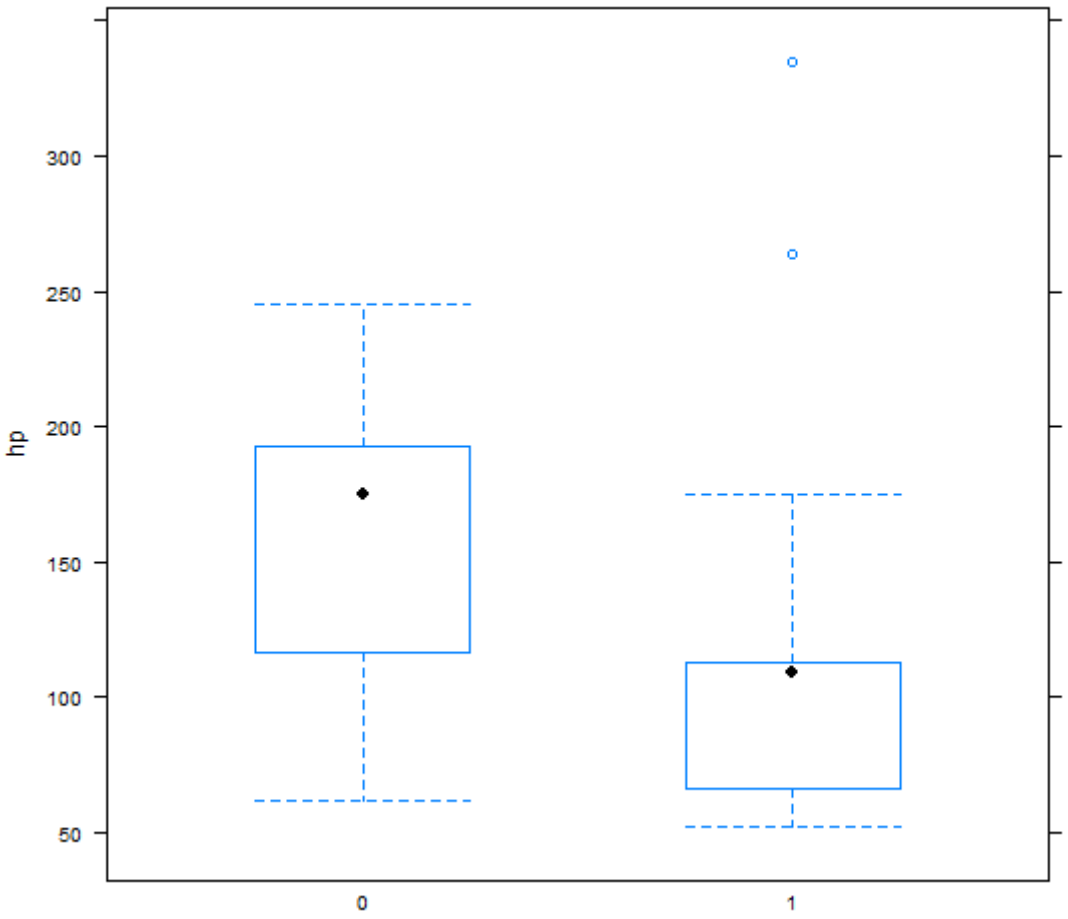


Але є кращі способи зробити це:

```
> bwplot(am ~ hp, data=mtcars)
```

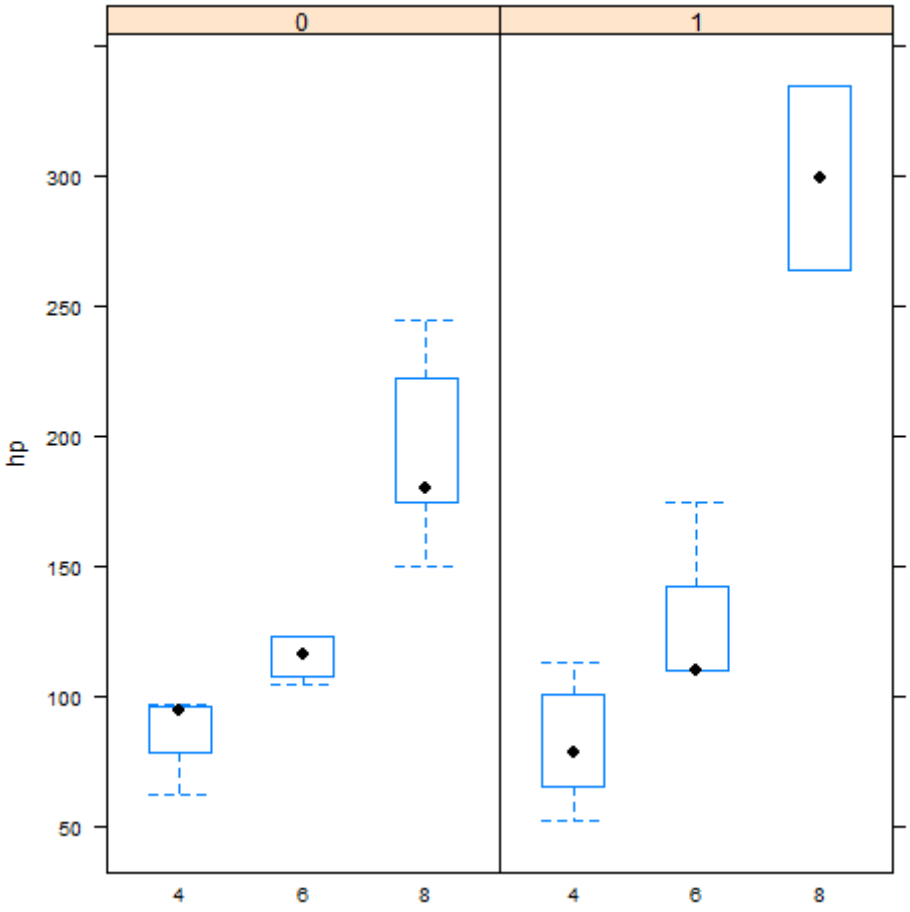



```
> bwplot(hp ~ format(am), data=mtcars)
```



Ми можемо поєднати цей спосіб з умовними змінними, якщо потрібно:

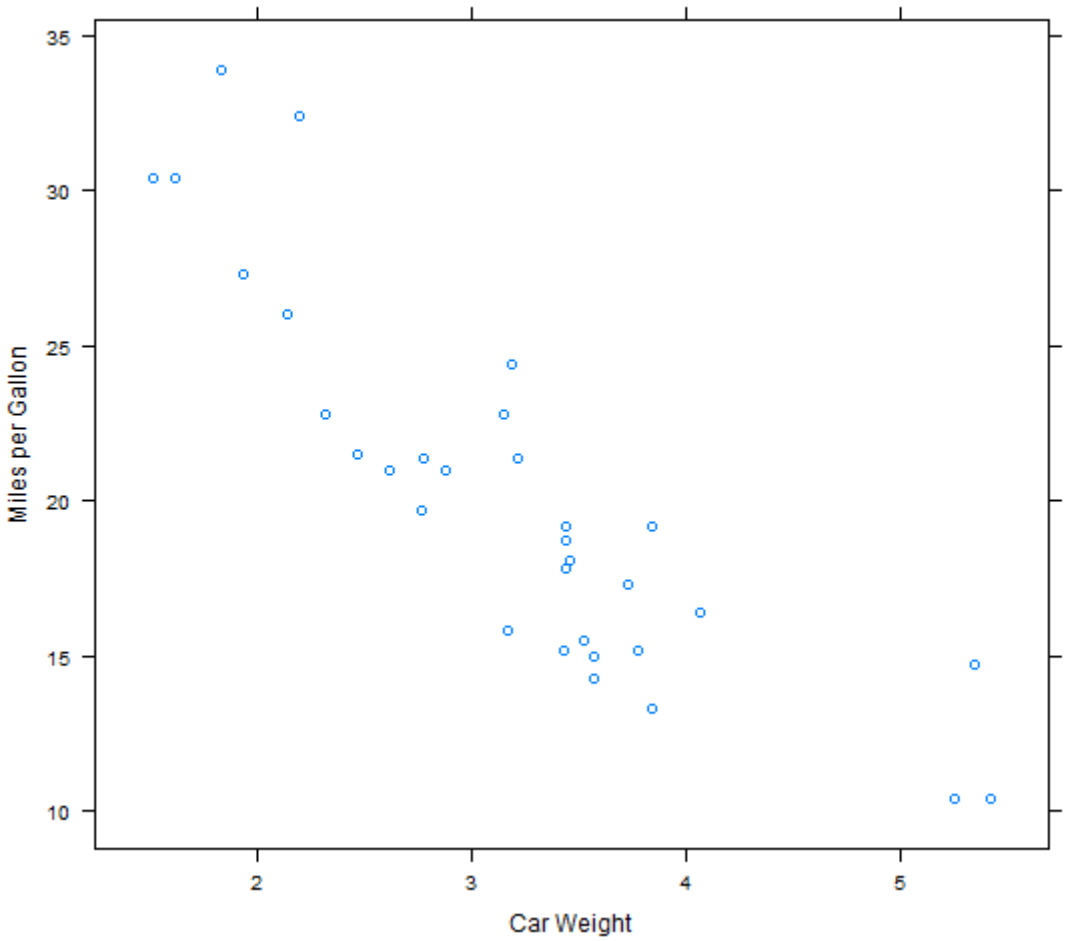
```
> bwplot(hp ~ format(cyl) | format(am), data=mtcars)
```



11.6. Діаграма розсіювання

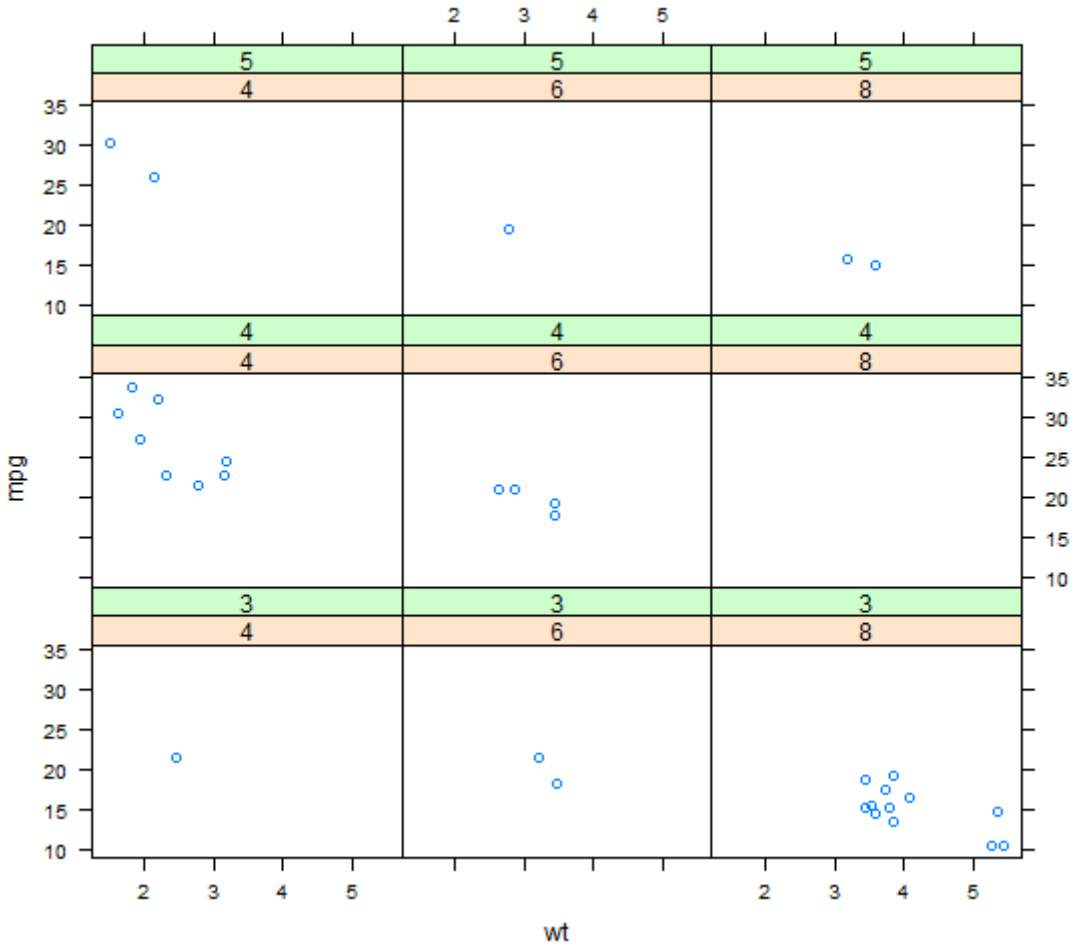
Діаграму розсіювання можна побудувати за допомогою функції `xyplot()`.

```
> xyplot(mpg ~ wt,data=mtcars, ylab=«Miles per Gallon», xlab=«Car Weight»)
```



Знову ж таки, ми можемо використовувати умовні змінні, щоб зробити панель для кожного виду.

```
> xyplot(mpg ~ wt | format(cyl) * format(gear),data=mtcars)
```



11.7. Часовий ряд

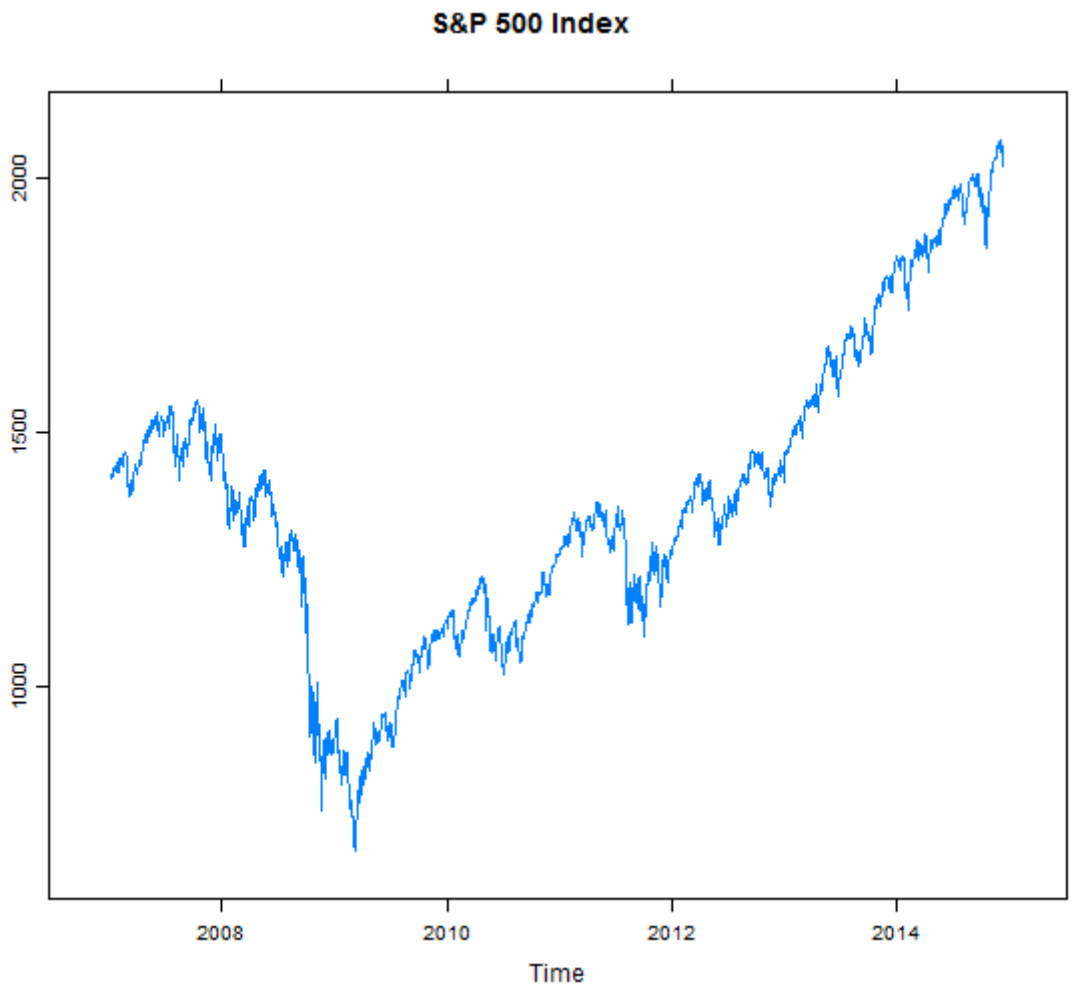
Часовий ряд – це послідовність значень досліджуваної ознаки (статистичного показника), впорядкована у хронологічному порядку.

Функція `xyplot()` має методи для побудови графіків часовихрядів (`ts, zoo, xts`).

Для прикладу використаємо часовий ряд індексуS&P 500.

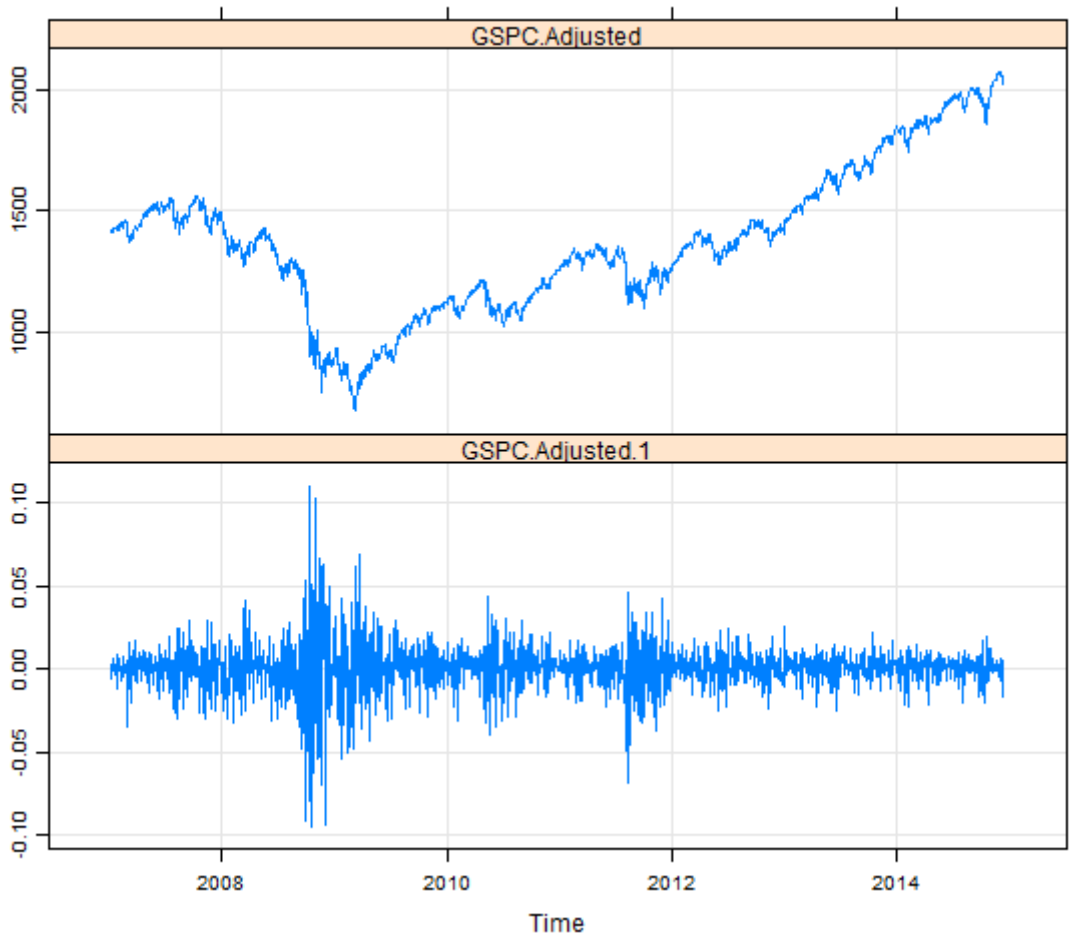
S&P 500 — фондовий індекс, у кошик якого включено 500 акціонерних компаній США, що мають найбільшу капіталізацію.

```
> install.packages(«quantmod»)
> library(quantmod)
> getSymbols(«^GSPC»)

> xyplot(Ad(GSPC),main=«S&P 500 Index»)
```



Графік рівня S&P500 і графік надходжень:

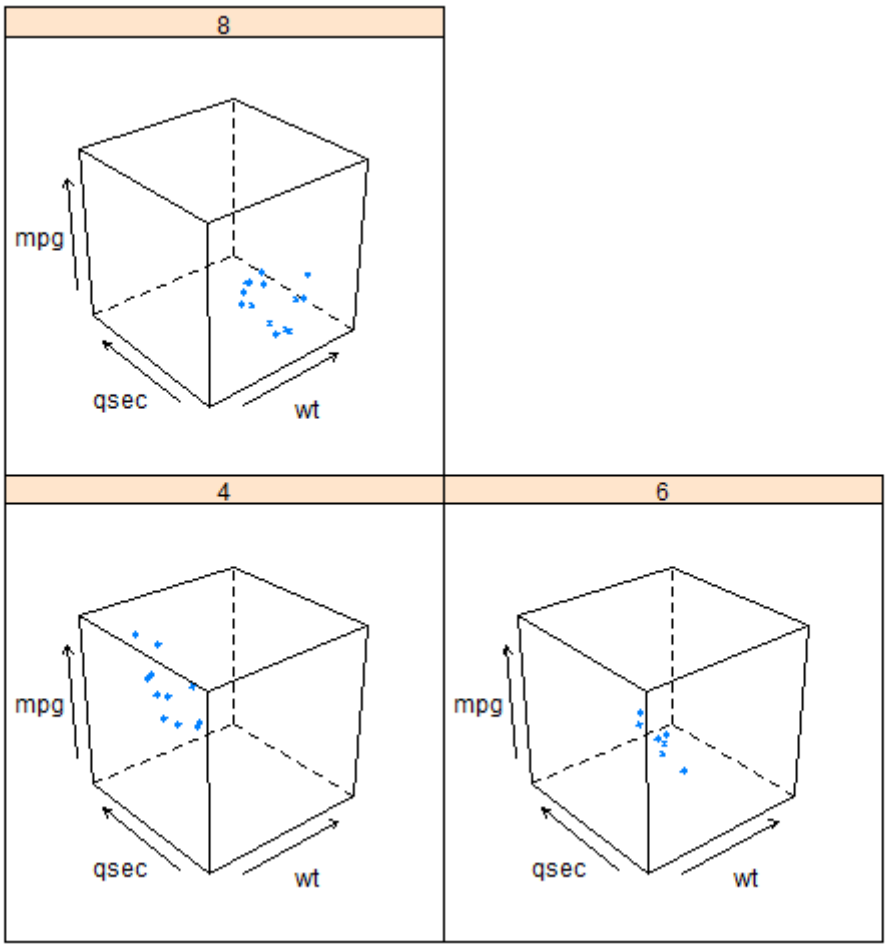
```
>xyplot(merge(Ad(GSPC),diff(log(Ad(GSPC)))),type=c(«l»,»g»)
)
```



11.8. 3D графіки

Функція `cloud()` використовується для побудови 3-D хмари точок.

```
> cloud(mpg ~ wt * qsec | format(cyl),data=mtcars)
```



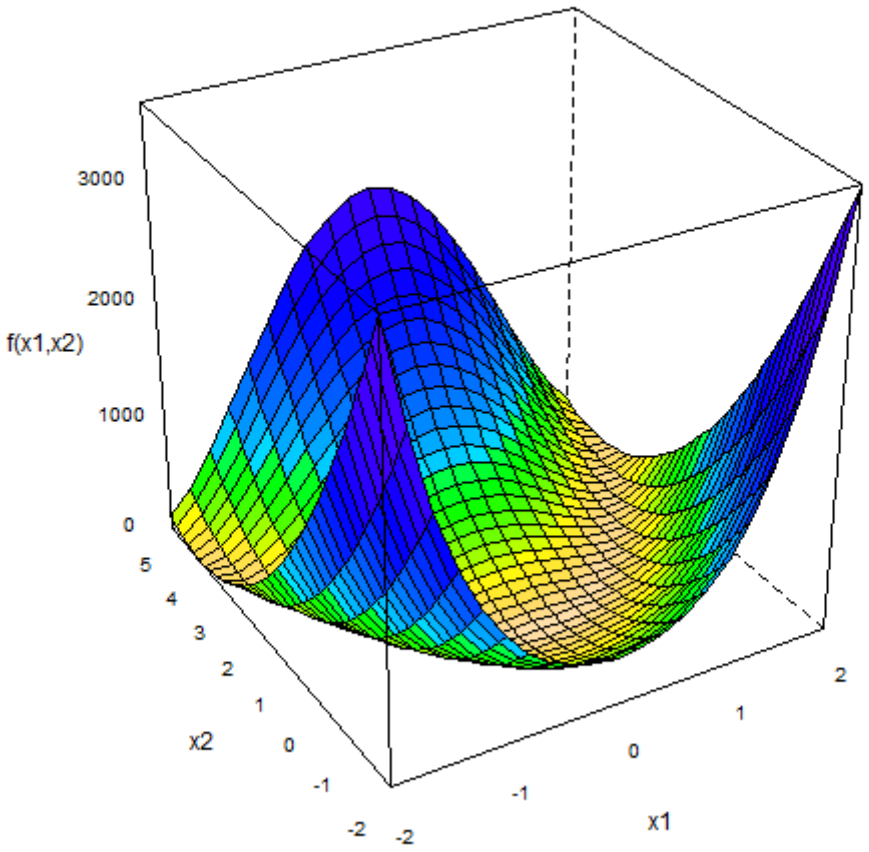
Приклад. Побудуємо графік функції Розенброка (яку ще називають банан)

$$f(x1, x2) = (1 - x_1)^2 + 100(x_2 - x^2{}_1)^2$$

```
> banana = function(x) {val = 100*(x[2]-x[1]^2)^2+(1-x[1])^2}
> x <- seq(-2, 2, .2)
> y <- seq(-2, 5, .2)
> YX <- expand.grid(x,y)
> head(YX,3)
  Var1 Var2
1 -2.0   -2
2 -1.8   -2
3 -1.6   -2
> z <- apply(YX,1,banana)
> XYZ <- data.frame(y=YX[,2],x=YX[,1],z=z)
> breaks = as.numeric(quantile(z,(0:1000)/1000))
```

Для побудови поверхні використаємо функцію `wireframe()`.

```
     >    par.set    <-    list(axis.line=list(col=«transparent»),
clip=list(panel=«off»))

     > print(wireframe(z ~ x*y, data = XYZ,
     xlab=«x1»,ylab=«x2»,       zlab       =«\n\nf(x1,x2)       «,
scales = list(arrows = FALSE),
     drape = F,
     shade=F,
     at=breaks,
     col.regions = rev(topo.colors(length(breaks)-1)),
     screen = list(z = 30, x = -60),
     colorkey=F,
     par.settings=par.set,
     zoom=0.95))
```



Додаток А. Довідкова карта по R

Отримання довідки

Більшість функцій R мають потужну онлайн документацію.

help(topic)	документація по темі topic
?topic	документація по темі topic
help.search("topic")	пошук довідкової системи
apropos("topic")	імена всіх об'єктів у списку пошуку, що відповідають регулярному виразу "topic"
help.start()	відкриття HTML версії допомоги
str(a)	відображає внутрішню структуру об'єкта R
summary(a)	дає "резюме" a, зазвичай як статистичне резюме, але і як загальне значення, має різні операції для різних класів a
ls()	показує об'єкти в шляху пошуку; вкажіть pat="pat" для пошуку за шаблоном
ls.str()	str() для кожної змінної в шляху пошуку
dir()	показує файли в поточній директорії
methods(a)	S3 методи a
methods(class=class(a))	перераховані всі методи для обробки об'єктів класу a

Ввід і вивід даних

load()	завантажити набори даних, записані за допомогою save
data(x)	завантажує вказаний набір даних
library(x)	завантаження додаткових пакетів
read.table(file)	читає файл у форматі таблиці і створює data.frame з нього; роздільник за замовчуванням sep=""; використовуйте header=TRUE, щоб зчитувати перший рядок як заголовок імен стовпців; використовуйте as.is=TRUE, щоб запобігти перетворення векторів символів у фактори; використовуйте comment.char="", щоб запобігти інтерпретації "#" як коментаря; використовуйте skip=n, щоб пропустити n рядків перед читанням даних;

`read.csv("filename",header=TRUE)`	аналогічно, але з встановленим за замовчуванням зчитуванням розділених комами файлів
`read.delim("filename", header=TRUE)`	аналогічно, але з встановленим за замовчуванням зчитуванням розділених табуляцією файлів
`read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE)`	зчитування таблиці фіксованої ширини відформатованих даних в «data.frame»; `widths` – цілий вектор, що задає ширини полів фіксованої ширини
`save(file,...)`	зберігаються задані об'єкти (...) в платформо незалежномубінарному форматі XDR
`save.image(file)`	зберігає всі об'єкти
`cat(..., file="", sep=" ")`	друкує аргументи після приєднання до символу; `sep`– символ роздільник між аргументами
`print(a, ...)`	друкує свої аргументи; тобто в загальному, може мати різні методи для різних об'єктів
`format(x,...)`	форматування об'єкта R для красивого друку
`write.table(x,file="", row.names=TRUE,col.names=TRUE, sep=" ")`	друк `x` після перетворення в `data.frame`; якщо `quote=TRUE`, символ або фактор стовпців береться в лапки ("); `sep` – роздільник полів; `eol`– роздільник кінця рядка; `na`– це стрічка для відсутніх значень, використовуйте `col.names=NA`, щоб додати порожню назву стовпця, для того, щоб отримати заголовки стовпців правильно вирівняні для введення таблиць

Більшість функцій введення / виводу мають аргумент файлу. Це часто може бути символьний рядок назвифайлу або з'єднання. `file=""`означає стандартний ввід або вивід. З'єднанняможуть включати в себе файли, канали, стислі файли, і R змінні.

Для взаємодії з базою даних, дивіться пакети`RODBC`, `DBI`, `RMySQL`, `RPgSQL`, `iROracle`. Подивіться всі пропозиції `XML`, `hdf5`, `NetCDF` для читання інших форматів файлів.

Створення даних

`c(...)`	узагальнена функція поєднання аргументіву вектор; якщо`recursive=TRUE`,спускається по списку, об'єднуючи всіелементи в один вектор
`from:to`	генерує послідовність; ":" має пріоритет оператора; 1: 4 + 1 буде вектор "2,3,4,5"
`seq(from,to)`	генерує послідовність `by`=вказуєприріст; `length`=визначає потрібну довжину
`seq(along=x)`	формує 1, 2, ..., `length(along);` корисно для`for`циклів
`rep(x,times)`	повторити `x` разів; Використовуйте `each`=щоб повторити "кожен" елемент з`xeach` раз; Rep (C (1,2,3), 2) буде 1 2 3 1 2 3; Rep (C (1,2,3), `each` = 2) буде 1 1 2 2 3 3
`data.frame(...)`	створенняблоку даних з поіменованими або безіменними аргументами; `data.frame(v=1:4,ch=c("a","B","c","d"),n=10);` короткі вектори використовуються повторно, до довжини найдовшого вектора
`list(...)`	створення спискуз поіменованими або безіменними аргументами; `list(a=c(1,2),b="hi",c=3i);`
`array(x,dim=)`	створення масиву з даними `x`; розмірність вказується, наприклад як `dim=c(3,4,2)`; елементи x використовуються повторно, якщо `x` не достатньо довгий
`matrix(x,nrow=,ncol=)`	матриця; елементи x використовуються повторно
`factor(x,levels=)`	кодує вектор `x` в якості фактора
`gl(n,k,length=n*k, labels=1:n)`	створення рівнів (факторів), вказавши шаблон їх рівнів; k – число рівнів; n – число повторень;
`expand.grid()`	`data.frame` від усіх даних векторів або факторів
`rbind(...)`	об'єднання аргументів по рядках для матриць, масивів даних та ін.
`cbind(...)`	аналогічно, по стовпчикам

Доступ до даних

Індексація векторів

x[n]	n-й елемент
x[-n]	всі крім n-ого елементу
x[1:n]	перші n елементів
x[-(1:n)]	елементи від n+1 до кінця
x[c(1,4,2)]	вказані елементи
x["name"]	елемент з ім'ям "name"
x[x > 3]	всі елементи, значення яких більші 3
x[x > 3 & x < 5]	всі елементи, значення яких від 3 до 5
x[x%in%("a","and","the")]	елементи у вибраній множині

Індексація списків

x[n]	список з елементами n
x[[n]]	n-й елемент списку
x[["name"]]	елемент списку з ім'ям "name"
x$name	елемент з ім'ям "name" у списку

Індексація матриць

x[i,j]	елемент матриці i-го рядка, j-го стовпчика
x[i,]	i-й рядок
x[,j]	j-й стовпчик
x[,c(1,3)]	стовпчики 1 та 3
x["name",]	рядок з ім'ям "name"

Математичні функції

sin,cos,tan,asin,acos,atan,atan2,log,log10,exp

max(x)	максимум елементів x
min(x)	мінімум елементів x
sum(x)	сума елементів x
mean(x)	середнє значення елементів x
median(x)	медіана елементів x
quantile(x,probs=)	вибірковий квантиль, відповідний заданому значенню ймовірності probs
rank(x)	ранг елементів x
sd(x)	стандартне відхилення x
round(x, n)	округлення елементів xдоnдесяткових знаків
log(x, base)	обчислення логарифму x за основою base
Re(x)	дійсна частина комплексного числа
Im(x)	уявна частина комплексного числа

Mod(x)	модуль x
abs(x)	модуль x
Arg(x)	аргумент комплексного числа (в радіанах)

Матриці

t(x)	транспонування
diag(x)	діагональ матриці x
%*%	множення матриць
solve(a,b)	розв'язання a %*% x = b для x
solve(a)	обернена матриця

Графіка

plot(x)	побудова значень x(на осіy),впорядкованих на осі x
plot(x, y)	двовимірний графік побудови значень x (на осіx) і значень y (на осіy)
hist(x)	побудова гістограми послідовності x
barplot(x)	побудова гістограми значень x; використовуйте horiz=FALSEдля горизонтальних смуг
pie(x)	кругова діаграма
boxplot(x)	«коробочка з вусами»
symbols(x, y, ...)	Побудова символу (коло, квадрат, прямокутник, зірка …) в заданих координатах (x, y)

Рекомендована література

1. R Development Core Team (2009). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

2. Brian S. Everitt and Torsten Hothorn (2005). A Handbook of Statistical Analyses Using R, London and Erlangen, http://www.R-project.org.

3. Michael J. Crawley (2007). The R Book, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester. ISBN-13: 978-0-470-51024-7.

4. John M. Chambers (2008). Software for Data Analysis. Programming with R, Springer Science+Business Media, LLC, USA, ISBN: 978-0-387-75935-7

5. Phil Spector (2008). Data Manipulation with R, Springer Science+Business Media, LLC, USA, ISBN 978-0-387-74730-9

6. Yosef Cohen and Jeremiah Y. Cohen (2008). Statistics and Data with R, John Wiley & Sons Ltd,The Atrium, Southern Gate, Chichester, ISBN 978-0-470-75805-2

7. J. H. Maindonald (2001). Using R for Data Analysis and Graphics An Introduction. Statistical Consulting Unit of the Graduate School, Australian National University.

8. John Verzani. simpleR — Using R for Introductory Statistics, http://cran.R-project.org/other-docs.html.

9. Jim Lemon. Kickstarting R, http://cran.R-project.org/other-docs.html.

10. Зарядов И.С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика. М.: Издательство Российского университета дружбы народов, 2010. 207 с.

11. Зарядов И.С. Статистический пакет R: теория вероятностей и математическая статистика. М.: Издательство Российского университета дружбы народов, 2010. 141 с.

12. Vikram Dayal. *An Introduction to R for Quantitative Economics: Graphing, Simulating and Computing*. Springer, 2015. ISBN 978-81-322-2340-5. http://www.springer.com/978-81-322-2340-5

13. K. Soetaert, J. Cash, and F. Mazzia. *Solving Differential Equations in R*. Use R! Springer, 2012. ISBN 978-3-642-28070-2.

14. Michael Lawrence. *Programming Graphical User Interfaces in R*. Chapman & Hall/CRC the R series. Chapman & Hall/CRC Press, Boca Raton, FL, 2012. ISBN 978-1-4398-5682-6. http://www.crcpress.com/product/isbn/9781439856826

15. Paul Teetor. *R Cookbook*. O'Reilly, first edition, 2011. ISBN 978-0-596-80915-7. http://oreilly.com/catalog/9780596809157