Лабораторна 1: Вступ до R.
*Базові команди.*

R використовує функції для виконання операцій. Щоб запустити функцію з назвою **funcname**, ми вводимо funcname(input1, input2). Функція може мати будь-яке число входів. Наприклад, для створення вектора чисел ми використовуємо функцію c () (для конкатенації). Будь-які числа в дужках об'єднуються.

Наступна команда вказує R об'єднати числа 1, 3, 2 і 5, і зберегти їх у вигляді вектора з іменем x.

```
> x <- c (1,3,2,5)
> x
[1] 1 3 2 5
```

Ми також можемо використовувати =, а не <-:

```
> x = c (1,6,2)
> x
[1] 1 6 2
> y = c (1,4,3)
```

Натискання стрілки вгору кілька разів відображатиме попередні команди, які потім можна редагувати. Це корисно, оскільки часто хочеться повторити подібну команду. Крім того, введення **?funcname** призводить до відкриття нового вікна файлу довідки з додатковою інформацією про функцію funcname.

Ми можемо додавати дві множини чисел. До першого числа з **x** додасться перше число з **y** і т. д. Однак **x** і **y** мають бути однакової довжини. Ми можемо перевірити їх довжину, використовуючи функцію **length()**.

```
> length(x)
[1] 3
> length(y)
[1] 3
> x+y
[1] 2 10 5
```

Функція ls () дозволяє нам переглянути список усіх таких об'єктів, які ми зберегли до цього часу. Функція rm () може бути використана для видалення тих об'єктів, які більше не потрібні.

```
> ls()
[1] "x" "y"
> rm(x, y)
> ls()
```

character(0)

Також можна видалити всі об’єкти одночасно:

> rm(list=ls())

Функцію **matrix**() можна використати для створення матриці чисел. Перед її використанням можемо дізнатись більше про неї:

> ?matrix

Файл довідки показує, що функція **matrix**() приймає декілька аргументів, але зараз ми зосередимося на перших трьох: даних (записах у матриці), кількість рядків і кількість стовпців. По-перше, ми створюємо просту матрицю.

```
> x= matrix(data=c(1,2,3,4),nrow=2,ncol=2)
> x
     [, 1] [, 2]
[1,]    1    3
[2,]    2    4
```

Ми могли б пропустити введення data=, nrow= та ncol= у команді matrix() вище: тобто ми могли б просто ввести

> x= matrix(c (1,2,3,4), 2,2)

і це мало б такий самий ефект. Однак іноді це може бути корисно вказувати імена аргументів, оскільки інакше R буде вважати що аргументи функції передаються у функцію в тому ж порядку що вказано у файлі довідки функції. Бачимо, що за замовчуванням R створює матриці шляхом послідовного заповнення стовпців. Можна задати параметр byrow = TRUE для заповнення матриці в по рядках.

> matrix(c(1,2,3,4),2,2, byrow=TRUE)

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

Тут ми не призначили матрицю змінній, у цьому випадку матриця друкується на екрані, але не зберігається для майбутніх розрахунків. Функція **sqrt**() повертає квадратний корінь кожного елемента вектора або матриці. Команда x^2 підносить кожен елемент x до степеня 2 (можливі будь-які степені, включаючи дробові або негативні).

```
> sqrt(x)
     [, 1] [, 2]
```

```
[1,]    1,00  1,73
[2,]    1,41  2,00
> x^2
         [, 1] [, 2]
[1,]      1      9
[2,]      4      16
```

Функція **rnorm**() генерує вектор випадкових нормальних величин, з першим аргументом *n* – розмір вибірки. Кожного разу, коли ми викликаємо цю функцію, ми отримає інший результат. Тут ми створюємо два корельовані набори чисел, **x** та **y** та використовуємо функцію **cor**() для обчислення кореляції між ними.

```
> x=rnorm(50)
> y=x+rnorm(50, mean=50, sd=.1)
>cor (x, y)
[1] 0,995
```

За замовчуванням **rnorm**() генерує стандартні нормальні випадкові величини із середнім значенням 0 і стандартним відхиленням 1. Однак середнє і стандартне відхилення можуть бути змінені за допомогою аргументів mean та sd, як проілюстровано вище. Іноді ми хочемо, щоб наш код відтворював точно такий же набір випадкових числа; для цього ми можемо використовувати функцію **set.seed**(). Ця функція приймає (довільний) цілочисельний аргумент.

```
> set.seed(1303)
> rnorm(50)
[1] -1,1440 1,3421 2,1854 0,5364 0,0632 0,5022 -0,0004 . . .
```

Функції **mean**() та **var**() можуть бути використані для обчислення середнього значення та дисперсії вектора чисел. Застосування **sqrt**() до результату **var**() дасть стандартне відхилення. Або ми можемо просто використовувати функцію **sd**().

```
> set.seed(3)
> y=rnorm (100)
> mean(y)
[1] 0,0110
>var (y)
[1] 0,7329
> sqrt(var (y))
[1] 0,8561
>sd (y)
[1] 0.8561
```

*Графіки*

Функція **plot**() - це основний спосіб побудови графічних даних у R. Наприклад, **plot**(x, y) створює діаграму розсіювання чисел у **x** проти чисел у **y**. Є багато

додаткових опцій, які можна передати **plot**() функції. Наприклад, передача аргументу xlab призведе до мітки на осі x. Щоб дізнатись більше про функцію plot (), введіть ?**plot**.

```
> x=rnorm(100)
> y=rnorm(100)
> plot(x, y)
> plot(x, y, xlab = "це вісь x", ylab = "це вісь y", main = "Графік X проти Y")
```

Ми часто хочемо зберегти вихідні дані графіку R. Команда, яку ми використання для цього залежатиме від типу файлу, який ми хотіли б створити. Наприклад, для створення pdf ми використовуємо функцію **pdf**(), а для створення jpeg, ми використовуємо функцію **jpeg**().

```
> pdf("Figure.pdf")
> plot(x, y, col="green")
> dev.off()
null device
          1
```

Функція **dev.off**() вказує R, що ми закінчили створення графіку. Як варіант, ми можемо просто скопіювати вікно графіку та вставити його у файл відповідного типу, наприклад документа Word.

Функцію **seq**() можна використовувати для створення послідовності чисел. Наприклад, **seq**(a, b) робить вектор цілих чисел між **a** і **b**. Існує багато інших варіантів: наприклад, seq(0,1, length=10) робить послідовність з 10 чисел, розташованих на однаковій відстані від 0 до 1. Ввід 3:11 – це скорочення для **seq**(3,11) для цілочисельних аргументів.

```
> x=seq(1, 10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x=1: 10
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x=seq (-pi, pi, length=50)
```

Зараз ми створимо кілька складніших графіків. **contour**() функція створює контурний графік для представлення тривимірних даних; це як топографічна карта. Для цього потрібні три аргументи:

1. Вектор значень **x** (перший вимір),
2. Вектор значень **y** (другий вимір),
3. Матриця, елементи якої відповідають значенню **z** (третій вимір) для кожної пари координат (**x**, **y**).

Як і у випадку з функцією **plot**(), існує безліч інших аргументів, які можна використовувати для точної настройки виводу функції **contour**(). Щоб дізнатись більше про них, ?contour.

```
> y=x
> f=outer(x, y, function(x, y) cos(y)/(1+x^2))
> contour(x, y, f)
> contour(x, y, f, nlevels = 45, add=T)
> fa=(f-t(f))/2
> contour(x, y, fa, nlevels=15)
```

Функція **image**() працює так само, як і **contour**(), за винятком того, що вона створює кольорову схему, кольори якої залежать від значення **z**. Як варіант, **persp**() можна використовувати для створення тривимірного графіку. Аргументи **theta** та **phi** контролюють кути, під якими знаходиться графік.

```
> image(x, y, fa)
> persp(x, y, fa)
> persp(x, y, fa, theta=30)
> persp(x, y, fa, theta=30, phi=20)
> persp(x, y, fa, theta=30, phi=70)
> persp (x, y, fa, theta=30, phi=40)
```

*Індексація даних*

Ми часто хочемо вивчити частину набору даних. Припустимо, що наші дані зберігаються у вигляді матриці **A**.

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

Потім, набравши, наприклад,

```
> A[2,3]
[1] 10
```

виберемо елемент, що відповідає другому рядку та третьому стовпцю. Перше число після символу відкритої дужки [ завжди посилається на рядок, а друге число завжди відноситься до стовпця. Ми також можемо вибирати кілька рядків і стовпців одночасно, надаючи вектори як індекси.

```
> A[c(1,3) ,c(2,4)]
     [,1] [,2]
[1,]    5   13
[2,]    7   15
> A[1:3 ,2:4]
     [,1] [,2] [,3]
[1,]    5    9   13
[2,]    6   10   14
[3,]    7   11   15
> A[1:2 ,]
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
> A[ ,1:2]
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Останні два приклади ілюструють або відсутність індексу для стовпців, або відсутність індексу для рядків. Це вказує на те, що R має включати всі стовпці або всі рядки, відповідно. R розглядає один рядок або стовпець матриці як вектор.

```
> A[1,]
[1] 1 5 9 13
```

Використання знака - в індексі зберігає всі рядки або стовпці крім зазначених в покажчику.

```
> A[-c(1,3),]
     [, 1] [, 2] [, 3] [, 4]
[1,]    2    6   10   14
[2,]    4    8   12   16
> A[-c(1,3), -c(1,3,4)]
[1] 6 8
```

Функція **dim**() виводить кількість рядків, за якою слідує кількість стовпців заданої матриці.

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

*Завантаження даних*

У більшості випадків перший крок передбачає імпорт набору даних у R. Функція **read.table**() є одним з основних способів зробити це. Файл довідки містить

детальну інформацію про те, як користуватися цією функцією. Ми можемо використовувати функцію **write.table**() для експорту даних. Перш ніж намагатися завантажити набір даних, ми повинні переконатися, що R знає каталог для пошуку даних.

Ми починаємо з завантаження набору даних Auto. Для ілюстрації функції **read.table**() завантажимо дані із текстового файлу. Наступна команда завантажить файл Auto.data в R і збереже як об'єкт, що називається Auto, у форматі, який називається фреймом даних. Після завантаження даних функцію **fix**() можна використовувати для перегляду у вікні як в електронній таблиці. Однак вікно повинно бути зачинене перед тим, як будуть можливі подальші команди R.

```
> Auto=read.table("Auto.txt")
> fix(Auto)
```

Цей конкретний набір даних завантажено неправильно, оскільки R включила імена змінних як частину даних у першому рядку. Набір даних також включає ряд пропущених спостережень, які позначені знаком питання ? Використання опції **header**=T (або **header**=TRUE) у функція **read.table**() повідомляє R, що перший рядок файлу містить імена змінних, а використання опції **na.strings** повідомляє R, що певний символ або набір символів (наприклад, знак запитання), слід розглядати як відсутній елемент таблиці даних.

```
> Auto=read.table("Auto.txt", header=T, na.strings="?")
> fix(Auto)
```

Простий спосіб завантажити дані з Excel в R - це зберегти їх як файл CSV, а потім використовувати функція **read.csv**() для завантаження.

```
> Auto=read.csv("Auto.csv", header=T, na.strings="?")
> fix(Auto)
> dim(Auto)
[1] 397 9
> Auto[1:4,]
```

Функція **dim**() повідомляє нам, що дані мають 397 спостережень, або рядків, і дев'ять змінних або стовпців. Існують різні способи роботи зі пропущеними даними. У цьому випадку лише п'ять рядків містять відсутні спостереження, і тому ми вирішили використати функцію **na.omit**(), щоб просто видалити ці рядки.

```
> Auto=na.omit(Auto)
> dim(Auto)
[1] 392 9
```

Як тільки дані завантажуються правильно, ми можемо використовувати names() для перевірки імен змінних.

```
> names(Auto)
[1] "mpg " "cylinders " " displacement" "horsepower "
[5] "weight " " acceleration" "year" "origin "
[9] "name"
```

*Додаткові графічні та числові відомості*

Ми можемо використовувати функцію **plot**() для створення графіків кількісних змінних. Однак просто введення імен змінних призведе до помилки, оскільки R не знає, що потрібно шукати відповідні дані у наборі Auto

```
> plot(cylinders , mpg)
Error in plot(cylinders , mpg) : object ’cylinders ’ not found
```

Щоб звернутися до змінної, ми повинні ввести набір даних та ім'я змінної об'єднані символом $. Крім того, ми можемо використовувати функцію **attach**(),щоб сказати R зробити змінні в цьому фреймі даних доступними за іменами.

```
> plot(Auto$cylinders , Auto$mpg )
> attach (Auto)
> plot(cylinders , mpg)
```

Змінна cylinders зберігається як числовий вектор, тому R обробив її як кількісну. Однак, оскільки є лише невелика кількість можливих значень для cylinders, можна розглядати її як якісну змінну. Функція **as.factor**() перетворює кількісні змінні в якісні.

```
> cylinders =as.factor (cylinders )
```

Якщо змінна, нанесена на вісь x, є категоріальною, тоді це буде прямокутна діаграма (boxplot) автоматично створена функцією **plot**(). Можна використати додаткові параметри цієї функції

```
> plot(cylinders, mpg)
> plot(cylinders, mpg, col="red ")
> plot(cylinders, mpg, col="red", varwidth=T)
> plot(cylinders, mpg, col="red", varwidth=T,horizontal=T)
> plot(cylinders, mpg, col="red", varwidth=T, xlab="cylinders", ylab ="MPG")
```

Функцію **hist**() можна використовувати для побудови гістограми. Зверніть увагу, що col = 2 має той самий ефект, що і col="red".

```
> hist(mpg)
```

```
> hist(mpg ,col =2)
> hist(mpg ,col =2, breaks =15)
```

Функція **pairs**() створює матрицю графіків, тобто набір графік для кожної пари змінних для будь-якого заданого набору даних. Ми також можемо створити матрицю графіків для підмножини змінних.

```
> pairs(Auto)
> pairs(∼ mpg + displacement + horsepower + weight +acceleration , Auto)
```

У поєднанні з функцією **plot**() **identify**() надає корисний інтерактивний метод ідентифікації значення певної змінної для точки на графіку. Ми передаємо три аргументи для **identify**():змінна осі **x**, змінна осі **y** та змінна, значення якої ми хотіли б побачити надруковані для кожної точки. Потім клацніть на задану точку на графіку. Це призведе до того, що R надрукує значення змінної, що цікавить. Клацніть правою кнопкою для виходу з функції **identify**().

```
> plot(horsepower ,mpg)
> identify (horsepower ,mpg ,name)
```

Функція **summary**() створює підсумок для кожної змінної в певному наборі даних.

```
> summary(Auto)
mpg cylinders displacement
Min. : 9.00 Min . :3.000 Min. : 68.0
1st Qu .:17.00 1st Qu .:4.000 1st Qu .:105.0
Median :22.75 Median :4.000 Median :151.0
Mean :23.45 Mean :5.472 Mean :194.4
3rd Qu .:29.00 3rd Qu .:8.000 3rd Qu .:275.8
Max. :46.60 Max . :8.000 Max. :455.0
horsepower weight acceleration
Min. : 46.0 Min . :1613 Min . : 8.00
1st Qu.: 75.0 1st Qu .:2225 1st Qu .:13.78
Median : 93.5 Median :2804 Median :15.50
Mean :104.5 Mean :2978 Mean :15.54
3rd Qu .:126.0 3rd Qu .:3615 3rd Qu .:17.02
Max. :230.0 Max . :5140 Max . :24.80
year origin name
Min. :70.00 Min . :1.000 amc matador : 5
1st Qu .:73.00 1st Qu .:1.000 ford pinto : 5
Median :76.00 Median :1.000 toyota corolla : 5
Mean :75.98 Mean :1.577 amc gremlin : 4
3rd Qu .:79.00 3rd Qu .:2.000 amc hornet : 4
Max. :82.00 Max . :3.000 chevrolet chevette : 4
(Other) :365
```

Для якісних змінних, таких як **name**, буде вказано кількість спостережень що потрапляють у кожну категорію. Ми також можемо зробити короткий підсумок лише для однієї змінної.

```
> summary (mpg)
Min. 1st Qu. Median Mean 3rd Qu. Max .
9.00 17.00 22.75 23.45 29.00 46.60
```

Після того, як ми закінчили використовувати R, ми вводимо **q**(). При виході з R ми маємо можливість зберегти поточну робочу область так, що всі об’єкти (наприклад, набори даних), які ми створили в цій сесії R будуть доступні наступного разу. Перш ніж вийти з R, ми можемо зберегти запис всіх команд, які ми ввели в останній сесії за допомогою функції **savehistory**(). Наступного разу, ми можемо завантажити цю історію за допомогою функції **loadhistory**().