ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

О.В. Кудін

ЕМПІРИЧНІ МЕТОДИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Методичні рекомендації до лабораторних робіт
для студентів освітнього ступеня «бакалавр»
напряму підготовки «Програмна інженерія»

Затверджено
вченою радою ЗНУ
Протокол № ___ від _____

Запоріжжя
2015

УДК : 519.23 : 004.434 (075.8)

ББК : В172+З973.2-018.1я73

К 887

Кудін О.В. Емпіричні методи програмної інженерії: методичні рекомендації до лабораторних робіт для студентів освітнього ступеня «бакалавр» напряму підготовки «Програмна інженерія» / О.В. Кудін. – Запоріжжя: ЗНУ, 2015. – 89 с.

Методичні рекомендації до виконання лабораторних робіт з дисципліни «Емпіричні методи програмної інженерії» пропонується студентам освітнього ступеня «бакалавр» напряму підготовки «Програмна інженерія» та рекомендується для закріплення навчального матеріалу з дисципліни циклу професійної та практичної підготовки.

Методичні рекомендації містять теоретичний матеріал, лабораторні роботи, контрольні запитання з кожної теми дисципліни. Увага студентів акцентується на розширенні теоретичних знань та практичних навичок з використання методів обробки даних.

Видання призначене для студентів, викладачів, а також усіх, хто цікавиться питаннями зазначеної тематики.

Рецензент *С.М. Гребенюк*

Відповідальний за випуск *О.В. Кудін*

Зміст

Вступ

Курс «Емпіричні методи програмної інженерії» орієнтований на вивчення основних методів аналізу даних та застосування їх для розв'язання практичних задач програмної інженерії. Задачі, пов'язані з обробкою даних, виникають в різних сферах інженерної діяльності. Так, методи кластерного аналізу можуть застосовуватись в задачах розпізнавання зображень, мови тощо; методи регресійного аналізу – для наближення та прогнозування значень часових рядів; нейронні мережі використовуються у задачах машинного навчання.

Володіння сучасними інструментальними засобами дає змогу зосередитися на алгоритмі розв'язання практичної задачі та використовувати існуючі пакети алгоритмів для швидкого та ефективного розв'язання поставленої задачі.

Мета дисципліни: сформувати теоретичні знання та практичні навички в області обробки та аналізу даних у студентів освітнього ступеня «бакалавр» напряму підготовки «Програмна інженерія», які можуть бути використані при подальшому навчанні, професійній, виробничій та науковій діяльності. Курс «Емпіричні методи програмної інженерії» базується на дисциплінах «Теорія ймовірностей та математична статистика», «Математичний аналіз», «Дискретна математика», що викладаються на математичному факультеті.

Завдання навчальної дисципліни: ознайомлення студента з історичними аспектами виникнення та розвитку алгоритмів аналізу даних; засвоєння основних понять та теоретичних основ методів аналізу даних; навчання алгоритмам роботи з інструментальними засобами. Студенти повинні опанувати принципи роботи з мовою та середовищем програмування R; набути практичного досвіду розв'язання практичних задач.

Методичний посібник містить 10 лабораторних робіт, теоретичні відомості та контрольні питання.

В результаті вивчення дисципліни студент повинен:

Знати:

- методи описової статистики;
- методи кореляційного аналізу даних;
- методи регресійного аналізу даних;
- основні методи кластерного аналізу даних;
- методи проектування нейронних мереж для аналізу даних.

Уміти:

- виконувати початкову обробку емпіричних даних;
- проводити кореляційних аналіз даних;
- будувати регресійні моделі;
- виконувати кластерний аналіз;
- проектувати нейронні мережі для аналізу даних.

Загальні вказівки

1. Лабораторні роботи виконуються в години, зазначені в розкладі і для кожного студента присутність на занятті є обов'язковою. Студенту, що пропустив лабораторне заняття без поважних причин і не захистив лабораторну роботу на наступному занятті, виставляється незадовільна оцінка з відповідної лабораторної роботи.

2. У комп'ютерний клас дозволяється входити тільки після дзвінка на заняття й у присутності викладача.

3. Вхід у комп'ютерний клас дозволяється тільки за наявності документа, що засвідчує особу і завірений відповідальним органом ЗНУ.

4. Лабораторну роботу припиняють виконувати за дзвінком.

5. Забороняється виходити з аудиторії без дозволу викладача.

Вимоги до виконання лабораторних робіт

1. Ознайомитися із загальними теоретичними відомостями.

2. За номером варіанта[1] обрати завдання.

3. Уважно прочитати завдання.

4. Кожну роботу виконувати відповідно до завдання.

5. Оформити друкований звіт з виконання лабораторної роботи, який повинен містити:

– титульний аркуш;

– тему роботи;

– схеми основних моделей та алгоритмів;

– опис виконання роботи;

– висновки.

6. Захистити результати лабораторної роботи, відповісти на контрольні та додаткові питання.

[1] Номер варіанта визначає викладач.

Лабораторна робота №1. Основи роботи в системі R

Мета: засвоїти основні відомості про роботу в системі **R** та операції з базовими типами і структурами даних.

Теоретичні відомості

Основні відомості про систему R.

Система **R** – це вільно розповсюджуване програмне середовище з відкритим кодом, що розвивається в рамках проекту GNU. У системі реалізовано вбудовану мову програмування **R**.

Мова **R** виникла як вільний аналог мови **S-PLUS**, що у свою чергу є комерційною реалізацією мови розрахунків **S**.

Система **R** підтримує два режими роботи – інтерактивний режим і режим скрипта.

При запуску програми **RGui** автоматично з'являється вікно **R Console**. Це командне вікно (консоль), у якому користувач вводить команди, а програма друкує результати. У ході роботи в основному графічному вікні можуть з'явитися й інші вікна – редактор скриптів, вікна із графічним результатом виконання команд (графіки) тощо.

Команди вводяться користувачем у консолі (командному вікні) після символу запрошення, що має вигляд «>».

Після натискання кнопки Enter введена команда надходить на обробку. В одному рядку можна ввести кілька команд, розділяючи їх символом «;». Одну команду можна розташувати на двох і більше рядках. Для цього слід натиснути Enter, тоді на новому рядку замість «>» з'явиться запрошення «+».

Символ «#» означає початок коментарю. Усе, що перебуває після цього знаку в одному рядку, ігнорується програмою.

Кнопки стрілок «вгору» та «вниз» на клавіатурі дозволяють здійснювати навігацію серед раніше введених команд (можна вибрати одну з попередніх команд).

За допомогою кнопок стрілок «вліво» та «вправо» можна переміщатися у вже введеній команді, зокрема, редагуючи її.

http://www.r-project.org – сайт проекту **R**.

Робота зі скриптами.

Скрипт – це самостійно написана програма з використанням всіх можливостей **R**. Щоб створити новий скрипт, потрібно у командному меню вибирати **Файл**, а потім вибрати **Новий скрипт**. Відкриється нове вікно, що представляє із себе редактор, у якому можна писати та редагувати текст створюваної програми.

Запуск на виконання рядка або виділеного блоку – комбінація клавіш **Ctrl+R**.

Деякі основні команди R.

help(ім'я_функції) або **?ім'я_функції** – виклик контекстної допомоги.

ls() або **objects()** виводять на екран всі створені протягом сесії об'єкти (дуже корисно, якщо програма велика).

rm(ім'я_об'єкта) – видаляє об'єкт із зазначеним ім'ям.

example(ім'я_функції) – виводить приклад (якщо є) для обраної функції.

history(n) – буде виведене нове вікно, у якому перераховані **n** останніх команд.

getwd() – вивід поточної (робочої) директорії.

setwd('ім'я_нової_робочої_директорії') – зміна робочої директорії.

dir() – виводить список файлів у робочій директорії.

source('ім'я_файлу.R') – збереження **R**-коду в поточній директорії у файлі із зазначеним ім'ям і розширенням .R.

sink('ім'я_файлу.розширення') – перенаправляє потік виводу з екрану в зазначений файл; повторний виклик команди **sink()** закриває файл.

rnorm(n,mean,sd) – генерація послідовності **n** випадкових чисел, що мають нормальний розподіл з математичним очікуванням **mean** і середнім квадратичним відхиленням **sd**.

Типи даних в R.

Всі дані в **R** можна можуть мати наступні типи:

- **numeric** – об'єкти даного класу діляться на цілочислові (**integer**) і дійсні (**double**);
- **complex** – об'єкти комплексного типу;
- **logical** – логічні об'єкти, приймають тільки два значення: FALSE (F) і TRUE (T);
- **character** – символьні об'єкти (символьні змінні задаються або в подвійних лапках, або в одинарних).

Приклад 1.1. Створення нульового вектора типу **numeric** довжини 5. Для виконання команд прикладу слід створити новий скрипт та скопіювати в нього код прикладу.

```
x=numeric(5)
x
```

Приклад 1.2. Використання команд **is.integer(), is.double(), is.numeric()**.

```
x=integer(1)
x
is.integer(x)
is.double(x)
x=2
is.integer(x)
y=double(1)
y=2
y
```

```
is.double(y)
is.integer(y)
is.numeric(y)
is.numeric(x)
```

З прикладу видно, що за замовчуванням всі числа в **R** є дійсними. Щоб зробити їх цілими, треба скористатися командою **as.integer(ім'я_об'єкта).**

Приклад 1.3. Способи створення комплексних чисел.

```
z<-complex(1)
z
z=complex(1,3,4)
z
z=complex(1,modulus=2,argument=3)
z
x=3
is.double(x)
y=4
z=x+1i*y
z
is.complex(z)
```

Приклад 1.4. Перетворення з **complex** в **integer** і навпаки.

```
x
as.complex(x)
z
as.integer(z)
Warning message:
imaginary parts discarded in coercion
```

Приклад 1.5. Робота з об'єктами типу **logical**.

```
x=logical(4)
# об'єкт логічного типу. Може приймати значення T або F
x
x[1]
as.integer(x[1])
y=3
y
as.logical(y)
y=0
as.logical(y)
z=4.5
as.logical(z)
```

Приклад 1.6. Робота з об'єктами типу **character**.

```
x=character(2)
x
x[1]='1'
x[2]='2'
x
```

```
z1<-as.integer(x[1])
z1
is.integer(z1)
z2<-as.double(x[2])
z2
is.double(z2)
x[1]="a"
x
as.integer(x[1])
Warning message:
NAs introduced by coercion
```

Приклад 1.7. Функція **mode()** – перевірка й зміна типу об'єкта.

```
x=logical(1)
x
mode(x)
z1=as.integer(x)
mode(z1)
mode(z1)='logical'
z1
is.logical(z1)
```

Спеціальні змінні в R.

В **R** існує ряд особливих об'єктів:

Inf – нескінченність: додатна (Inf) і від'ємна (-Inf);

NA – відсутнє значення (Not Available);

NaN – не число (Not a Number);

NULL – ніщо.

Функції для перевірки об'єктів на приналежність до цих типів: **is.finite(ім'я_об'єкта)**, **is.nan(ім'я_об'єкта)**, **is.na(ім'я_об'єкта)**, **is.null(ім'я_об'єкта)**.

Числові послідовності в R.

Приклад 1.8. Задання числових послідовностей за допомогою команд **a:b**, **seq()**.

```
x=2:5
x
seq(1,10)
seq(1,10,0.5)
seq(from=1,to=10,by=0.5)
seq(from=1,to=10,length.out=19)
seq(5)
```

Логічні операції в R.

Базові логічні операції, які підтримує **R**, наведені у таблиці 1.1.

Таблиця 1.1 – Перелік логічних операцій **R**

Запис у R	Описання
`!x`	логічне заперечення
`x&y`	поелементна кон'юнкція
`x&&y`	кон'юнкція тільки перших елементів векторів `x`, `y`
`x|y`	поелементна диз'юнкція
`x||y`	диз'юнкція тільки перших елементів векторів `x`, `y`
`xor(x,y)`	якщо `x` та `y` різні, то отримаємо TRUE, інакше FALSE

Основні оператори **R** наведені у таблиці 1.2.

Таблиця 1.2 – Список основних операторів **R**

Оператор	Описання
>	більше
>=	більше або дорівнює
<	менше
<=	менше або дорівнює
==	тотожність
!=	не тотожність

У **R** над об'єктами можна виконувати звичайні арифметичні операції: + (додавання), − (віднімання),∗ (множення), / (ділення), ^ (піднесення до степеня), %/% (цілочислове ділення), %% (остача від ділення); %*% (множення матриць).

Структури даних в R.

Вектор (vector) – одномірний масив, що складаються з елементів одного типу даних. Можна виділити числові, логічні й символьні вектори.

Матриця (matrix) – двовимірний масив, як правило, числовий.

Багатомірний масив (array) – масив розмірності більше двох.

Фактор (factor) та **таблиця (table)** – структури, які використовуються при роботі з категоріальними даними.

Список (list) – це колекція об'єктів, доступ до яких можна здійснити по номеру або імені.

Таблиця даних (data frame) – найбільш загальна структура, яка використовується при роботі в **R**.

Розглянемо більш детально основні структури даних.

Вектор.

В **R** можна виділити кілька способів задання векторів.

1. Створюється нульовий вектор потрібного типу (логічний, числовий, символьний, комплексний) і заданого розміру, надалі елементам присвоюються необхідні значення.

2. Відразу задається вектор з потрібними елементами.

Перший спосіб реалізується за допомогою функції **vector(mode = "тип даних", довжина)**, аргументами якої є тип вектора (числовий – **numeric**, логічний – **logical**, комплексний – **complex**, символьний – **character**), зазначений у лапках, і довжина вектора – натуральне число.

Приклад 1.9. Створення вектора за допомогою функції **vector()**.

```
vector('numeric',10)
vector('complex',10)
vector('logical',10)
vector('character',10)
```

Альтернативою функції **vector()** є функції створення вектора певного типу даних: **numeric()**, **logical()**, **complex()**, **character()**.

Другий спосіб задання вектора – безпосередньо задати його елементи. Це можна здійснити за допомогою функції конкатенації **c()**. Аргументи цієї функції є елементами вектора.

Приклад 1.10. Створення векторів за допомогою функції **c()**.

```
x=c(3,2,4,10)
x
y=c(T,T,T,F,F)
y
z=c('a',"ac",'b',"bb","cc"); z
```

Якщо як аргументи функції **c()** задати дані різних типів, то вони будуть приводитися до єдиного: логічні й числові дані приводяться до числового типу даних; логічні, числові й символьні – до символьного типу; дійсні й комплексні – до комплексного типу.

Приклад 1.11. Перетворення типів даних у векторі.

```
x=c(2,3,T,F);x
y=c(2,5,6,T,2.3,3+1i*4);y
z=c(2,3,4,T,'1');z
```

Можна задати вектор, набираючи дані на клавіатурі. Це реалізується за допомогою функції **scan()**.

Приклад 1.12. Створення вектора за допомогою функції **scan()**.

```
x=scan()
#введення даних з клавіатури
Read 5 items
x
```

Розглянемо далі основні функції для роботи з векторами.

length() – визначення довжини вектора.

max() і **min()** – знаходження максимального й мінімального елементів у заданому векторі. Якщо хоча б один елемент дорівнює **NA**, то результат пошуку максимуму (мінімуму) – **NA**, якщо є **NaN** – результат **NaN**. Для вилучення **NA** (**NaN**) з розрахунків потрібно задати **max(x,na.rm=T)** і **min(x, na.rm=T)**.

mean() – середнє арифметичне вектора. Якщо є хоча б один елемент, який дорівнює **NA** (**NaN**), то сума також буде **NA** (**NaN**). Щоб уникнути цього, також потрібно задати додатковий аргумент **mean(x,na.rm = TRUE)**.

range() – вектор, що складається із двох елементів – мінімального й максимального значень свого аргументу (вектора).

sum() – сума елементів вектора. Якщо є хоча б один елемент, чиє значення **NA** (**NaN**), то результатом підсумовування також буде **NA** (**NaN**). Щоб уникнути цього, потрібно задати додатковий аргумент **sum(x,na.rm = TRUE)**.

prod() – добуток компонентів вектора.

sort() – повертає вектор тої ж довжини, що й вихідний, з елементами, відсортованими в порядку зростання (за замовчуванням), або в порядку спадання – **sort(x,decreasing=T)** (або **sort(x,dec=T)**). Значення **NA** (**NaN**) автоматично опускаються при розгляді.

which(умова) – визначення індексу елементу по деякій умові.

Матриці.

Числову матрицю можна створити із числового вектора за допомогою функції **matrix(x, nrow, ncol, byrow, dimnames)**.

Для задання матриці потрібен вектор даних **x**, треба вказати число рядків **nrow = m** та/або число стовпців **ncol = n** (за замовчуванням число рядків дорівнює числу стовпців і дорівнює 1); визначити як елементи вектора **x** заповнюють матрицю – по рядках (**byrow**) або по стовпцях (**bycol**). За замовчуванням матриця заповнюється по стовпцях. У результаті елементи з вектора будуть записані в матрицю зазначених розмірів. Аргумент **dimnames** – список із двох компонент, перша з яких задає назви рядків, а друга – назви стовпців (за замовчуванням імена рядків і стовпців не задаються).

Функції **nrow(A)**, **ncol(A)** і **dim(A)** повертають число рядків, число стовпців і розмірність матриці **A** відповідно.

Функція **cbind(A, B)** створює матрицю з матриць (векторів), приписуючи праворуч до **A** матрицю (вектор) **B** (для цього число рядків в **A** і **B** повинно збігатися).

Функція **rbind(A, B)** створює матрицю, приписуючи знизу до матриці **A** матрицю **B** (для цього число стовпців у вихідних матриць повинно збігатися).

Щоб задати діагональну матрицю слід скористатися функцією **diag(x,nrow,ncol)**.

Для редагування створеної матриці можна використати команду **fix(ім'я_об'єкта)**.

Приклад 1.13. Створення матриць.

```
m=matrix(nrow=2,ncol=3)
m
dim(m)
attributes(m)
$dim
x=1:3
```

```
y=1:10
cbind(x,y)
rbind(x,y)
```

Ряд функцій корисних при роботі з матрицями.

colSums(X, na.rm) – сума елементів по стовпцях.

rowSums(X, na.rm) – сума елементів по рядках.

colMeans(X na.rm) – середні значення по стовпцях.

rowMeans(X na.rm) – середні значення по рядках.

Операції над матрицями.

A+B, **A-B**, **A%*%B** – додавання, віднімання, множення матриць.

det(A) – визначник матриці **A**.

solve(A) – обернена матриця **A**.

Багатомірні масиви.

Робота з багатомірними масивами в **R** багато в чому аналогічна роботі з матрицями. Основний спосіб їхнього створення – функція **array(X, вектор розмірностей)**.

Списки.

Списки в **R** – це колекції об'єктів, доступ до яких можна здійснювати по номеру або імені.

Список може містити об'єкти (компоненти) різних типів, що відрізняє списки від векторів. Компонентами списку можуть бути в тому числі вектори й інші списки.

Функція **list(об'єкт1, об'єкт2, ...)**.
створює список, що містить зазначені об'єкти.

Приклад 1.14. Створення списку.

```
x<-list(1,"a",T,1+4i)
x
```

Приклад 1.15. До елементів списку можна звертатися по іменам.

```
x=list(a=1,b=2,c=3)
x
$a
$b
$c
x[1]
$a
x$a
x$b
```

Фактори й таблиці.

Фактор – це векторний об'єкт, що кодує категоріальні дані (класи). Фактори створюються за допомогою функції **factor()**.

Інша функція, що дозволяє працювати з категоріальними даними – **table(ім'я_об'єкта)**. Особливість цієї функції полягає в тому, що вона не тільки виділяє різні категорії даних, але й систематизує їх, указуючи скільки елементів перебуває в кожній категорії. У результаті роботи **table()** одержимо таблицю,

перший рядок якої – це різні категорії, а друга – кількість елементів в кожній категорії.

Приклад 1.16. Створення фактору.

```
x=factor(c("yes","no","yes","no","no","no"))
x
table(x)
x
```

Таблиці даних.

Таблиці (фрейми даних, **data frames**) – один з найважливіших типів даних в **R**, що дозволяє поєднувати дані різних типів разом.

Таблиця даних – це двовимірна таблиця, у якій, на відміну від числових матриць, різні стовпці можуть містити дані різних типів (але всі дані в одному стовпці мають один тип). Наприклад, така таблиця може містити результати експерименту. Створити фрейм даних можна за допомогою функції **data.frame()**:

```
frm <- data.frame(data1, data2, ...)
```

Тут три крапки позначають, що список даних може містити довільне число елементів. У якості даних (data1, data2, . . .) можуть виступати вектори (числові, символьні або логічні), фактори, матриці (числові, символьні або логічні), списки або інші структури даних.

При цьому всі вектори повинні мати однакову довжину, а матриці й таблиці – однакове (таке ж) число рядків. Можуть також зустрічатися вектори, довжина яких менше, але в цьому випадку ця довжина повинна бути дільником максимальної довжини, що зустрічається. Та ж вимога висувається до компонентів списків. Функція **data.frame** просто збирає всі дані разом. Символьні вектори конвертуються у фактори. Інші дані збираються у фрейм такими, які вони є.

Приклад 1.17. Створити таблицю даних по чотирьом студентам – рік народження й рік вступу до ВНЗ.

```
y=matrix(c(1988,1987,1989,1989,2005,2006,2005,2005),nrow=
4)
y
rownames=c("Іванов","Петров","Сидоров","Воєводін")
colnames=c("рік народження","рік вступу")
y
colnames(y)=c("рік народження","рік вступу")
rownames(y)=c("Іванов","Петров","Сидоров","Воєводін")
y
# додамо нові стовпці
n=c(F,T,F,F)
y1=c(2,3,1,2)
y2=data.frame(y,n,y1)
y2
> colnames(y2)[3]="заборгованість"
```

```
> colnames(y2)[4]="курс"
> y2
```

Завдання до лабораторної роботи

Напишіть скрипт, що виконує перераховані нижче завдання.

1. Згенерувати цілочисловий вектор *X*, що складається з 100 випадкових чисел, що підкоряються нормальному розподілу з математичним сподіванням 50 і середнім квадратичним відхиленням 10.
2. Визначити максимальний, мінімальний елементи і їхні індекси у векторі *X*. Визначити середнє геометричне й медіану значень вектора *X*.
3. Згенерувати вектор *X1*, що складається з елементів вектора *X*, менших 40 або більших 60.
4. Визначити моду значень вектора *X*.
5. З вектора *X* створити матрицю *M1* із заповненням по рядках і матрицю *M2* із заповненням по стовпцях.
6. Створити таблицю даних (data frame) Res з результатами сесії студентів своєї підгрупи.
7. Розв'язати систему лінійних алгебраїчних рівнянь

$$\sum_{j=1}^{6} a_{ij} x_j = b_i, \quad i = 1,2,..,6$$

матричним методом згідно з номером варіанта.

Таблиця 1.3 – Варіанти завдань

№ варіанта	Матриця коефіцієнтів системи A						Стовпець вільних членів $\bar{b}$
1	2,1	1,3	-5,2	1,4	3,3	4,6	14,35
	6,3	-0,4	-14,4	7,8	14,0	21,0	90,95
	2,1	-3,0	-1,7	9,2	7,8	24,6	129,28
	4,2	-1,7	-6,9	10,6	11,1	29,2	143,63
	-2,1	3,0	-0,6	-13,4	-0,5	0,0	-65,30
	4,2	2,6	-15,0	-5,6	13,5	0,6	-43,71
2	1,7	-3,1	2,2	0,6	1,8	-0,1	-7,31
	3,3	4,2	0,7	5,4	2,1	1,6	40,53
	-0,4	5,7	4,1	3,6	-0,5	2,7	26,42
	4,7	-0,6	3,3	7,1	1,9	-4,6	23,24
	-0,1	7,2	4,8	2,3	5,1	3,3	52,40
	2,6	3,1	7,4	-4,1	8,2	-0,1	31,99
3	-3,7	0,2	5,3	-1,4	2,2	3,4	-3,25

	0,6	2,4	-1,7	3,0	-7,1	0,7	-30,32
	4,8	-5,2	-2,3	1,6	0,8	3,5	19,01
	1,1	4,4	0,8	3,1	-2,8	-5,1	6,19
	-0,3	1,5	-1,4	6,2	1,7	0,4	7,04
	2,2	-0,1	4,5	-3,3	2,6	6,6	20,35
4	1,7	-0,1	8,2	-1,4	2,3	0,0	41,48
	-1,3	-0,9	2,1	6,7	0,0	1,1	6,27
	-0,1	-1,3	0,5	2,4	5,6	0,6	16,44
	6,1	-0,1	3,2	1,2	0,7	-0,4	40,62
	2,2	0,1	-0,2	0,4	-1,8	5,1	5,08
	0,4	7,6	2,0	-0,3	1,8	2,1	3,65
5	0,9	-1,7	0,3	6,9	1,2	-0,1	0,78
	-0,3	1,9	7,4	-1,1	0,9	0,2	-13,80
	7,2	0,2	-0,9	1,3	-2,0	1,9	12,35
	0,4	0,6	2,2	-1,2	-0,1	5,2	12,69
	1,6	8,8	-0,4	0,7	2,3	-1,1	10,57
	-1,3	-0,8	0,1	0,6	5,7	2,0	1,91
6	2,4	-1,0	0,2	0,0	1,4	6,8	13,12
	0,0	0,4	5,4	-1,0	-1,9	0,1	11,74
	-1,9	1,5	-0,7	0,3	7,4	0,0	18,81
	8,3	-1,4	0,0	1,0	2,1	-1,7	-6,40
	2,2	0,6	0,0	6,6	-0,1	0,4	4,64
	-0,5	7,8	1,1	0,1	0,0	2,4	13,78
7	-2,5	0,0	-0,4	0,0	6,7	1,2	4,13
	1,6	0,3	0,0	8,0	-1,1	-0,7	24,59
	1,2	-1,0	0,7	-0,9	0,0	5,9	-5,39
	-1,0	8,4	0,2	-1,3	0,0	2,1	15,64
	9,0	0,7	-1,4	2,1	-0,2	-0,8	12,98
	0,2	-1,4	7,7	0,1	1,9	0,0	-10,12
8	-0,9	0,3	-0,5	0,0	5,4	-1,2	-3,19
	0,7	-0,4	6,4	-2,0	1,4	0,0	-9,30
	5,9	0,0	-0,8	1,1	0,3	0,7	3,83
	0,0	1,6	1,7	-2,1	0,0	6,6	19,43
	1,2	-1,0	0,0	7,9	-0,2	0,4	8,16
	-0,1	6,5	1,2	0,0	-0,9	-2,3	19,45
9	-1,4	1,7	-0,1	8,2	2,3	0,0	-22,23
	0,4	-1,8	5,1	2,2	0,1	-0,2	1,65
	0,7	-0,4	-2,0	0,0	1,4	6,4	19,00
	0,0	-0,8	1,1	0,3	5,9	0,7	3,17
	6,7	0,0	1,1	-1,3	2,1	-0,9	12,13
	1,2	7,9	0,0	-1,0	-0,2	0,4	0,72
10	-0,4	0,7	1,2	3,2	6,1	-0,1	-22,40
	0,3	0,0	8,0	1,6	-1,1	-0,7	-9,39
	-0,8	-0,2	2,1	-1,4	0,7	9,0	-3,09

	0,0	6,7	1,1	2,1	-0,9	-1,3	3,94
	5,6	2,4	0,5	-1,3	-0,1	0,6	-1,69
	1,8	-0,3	2,0	7,6	0,4	2,1	21,40
11	2,1	1,8	-0,3	2,0	0,4	7,6	-1,52
	0,0	1,6	6,8	0,2	-1,0	2,4	10,34
	1,9	2,0	1,3	-0,9	7,2	0,2	10,61
	0,6	5,6	2,4	-0,1	-1,3	0,5	-1,98
	6,9	1,2	-1,0	0,3	-1,7	0,7	22,74
	2,0	0,6	0,1	5,7	-0,8	-1,3	28,30
12	-1,0	1,2	0,3	-1,7	6,9	0,9	-0,65
	-0,1	-1,2	5,2	2,2	0,6	0,4	-4,52
	5,9	0,7	0,3	1,1	-0,8	0,0	6,23
	0,2	0,9	-1,1	1,9	-0,3	7,4	25,28
	1,9	-2,0	1,3	7,2	0,2	-0,9	1,74
	1,4	6,8	0,0	0,2	-1,0	2,4	-4,26
13	0,2	0,9	-1,1	1,9	7,4	-0,3	-5,94
	-0,9	0,2	7,2	1,9	-2,0	1,3	7,40
	-1,1	2,3	0,7	-0,4	1,6	8,8	-6,04
	2,0	5,7	0,6	0,1	-0,8	-1,3	-7,38
	6,6	0,0	-2,1	1,7	1,6	0,0	15,74
	-1,0	1,2	0,3	6,9	0,9	-1,7	15,55
14	2,4	0,0	0,1	1,1	7,8	-0,5	-11,76
	0,1	-1,9	5,4	-1,0	0,4	0,0	4,66
	0,0	0,3	-0,7	1,5	-1,9	7,4	28,60
	5,7	2,0	0,6	0,1	-0,8	-1,3	-16,18
	0,4	-0,1	0,0	6,6	0,6	2,2	20,94
	1,4	6,8	0,0	0,2	-1,0	2,4	5,92
15	1,2	0,0	-0,4	6,6	0,1	-2,5	-8,86
	0,0	-0,9	6,1	0,7	-1,0	1,2	7,98
	0,0	1,9	0,1	-1,3	0,2	7,8	4,70
	-0,7	-1,1	0,0	0,3	8,1	1,3	29,47
	-0,1	5,2	-1,2	2,2	0,6	0,4	6,09
	7,9	0,4	-0,2	0,0	-1,0	1,2	-27,93

Контрольні запитання

1. Поясніть призначення мови **R**.
2. Охарактеризуйте типи даних в **R**.
3. Назвіть основні команди для роботи з типами даних.
4. Охарактеризуйте структури даних в **R**.
5. Назвіть основні команди для роботи зі структурами даних.

Лабораторна робота №2. Основи програмування в R

Мета: отримати практичні навички роботи з операторами циклу, умови, створення функцій; робота з функціями введення та виведення даних.

Теоретичні відомості

Оператор **if**.

Оператор умови має наступну форму:

if(умова) вираз

або

if(умова) вираз1 else вираз2.

Умова – будь-який оператор (<, >, >=, <=, ==, !=), результатом виконання якого є логічний вектор одиничний довжини. Якщо значення вектора TRUE, то виконується **вираз**.

Вираз – один або кілька операторів, які виконуються у випадку вірності умови. Якщо задано кілька виразів, то вони повинні бути укладені у фігурні дужки {} і розділятися крапкою з комою (якщо на одному рядку). Оператор **if** повертає значення виразу у випадку вірності умови або нічого не повертає (NULL).

Приклад 2.1. Коротка форма оператора **if**.

```
x=5;y=4
if (x>y) {z1=x+y; z2=x*y}
z1;
z2;
```

Приклад 2.2. Повна форма оператора **if**.

```
x=5;y=4
if (x<y) z=x+y else z=x-y
z;
```

Оператор **ifelse**

ifelse(умова, yes, no)

дозволяє змінній залежно від виконання (невиконання) деякої умови приймати різні значення. Відмінність від оператора **if** полягає в тому, що тут умова є логічним вектором будь-якої заданої розмірності (залежить від розмірності об’єктів, що порівнюються). Порядковий номер елемента логічного вектора і його значення визначає, який елемент вектора (**yes** або **no)** привласнюється новій змінній.

Приклад 2.3. Створити два вектори **x** і **y** однакової довжини й привласнити змінній **z** або зі знаком «+» номери елементів, що співпадають, або зі знаком «-» номери елементів, які не співпадають.

```
x=c(1,3,1,5,1,7,1,9)
```

```
y=c(2,3,4,5,2,7,1,8)
z=ifelse(x==y,1:10,(-1):(-10))
z
```

Оператор циклу **for**

for(змінна in послідовність) вираз.

Поки змінна перебуває в рамках заданої числової послідовності, виконується **вираз** (або блок виразів).

Приклад 2.4. Створення вектора.

```
x=1:10
y=10:1
x
y
w=vector('numeric',10)
w
for (i in 1:10)
{
if (x[i]<y[i]) w[i]=x[i]/y[i]
else w[i]=x[i]*y[i]
}
w
```

Оператор **while**

while(умова) вираз.

Поки виконується умова, обчислюється вираз; як тільки результатом умови стає FALSE, виконується вихід із циклу.

Приклад 2.5. Оператор **while**.

```
> x=-10
> while(x<0) {z=x;x=x+1}
> z
[1] -1
```

Оператор **repeat**

repeat вираз

створює нескінченний цикл, у якому обчислюється вираз (або блок виразів). Для виходу із цього циклу потрібно використати **break**, а також умовний оператор у якості одного з виразів.

Приклад 2.6. Застосування оператора **repeat**.

```
t=-10
t
repeat{
if (t>0) break
f=t^2
t=t+1}
f
```

Оператор **next,** як і **break,** може застосовуватися тільки всередині циклів. Відмінність від **break** полягає в тому, що відбувається не переривання циклу, а перехід на наступну ітерацію.

Оператор **switch**

switch(керуючий вираз, альтернативні дії)

дозволяє виконувати одну з декількох операцій залежно від результатів керуючого виразу. Керуючий вираз повертає:

- або ціле число (від 1 до числа альтернатив), що є номером дії, яка виконується;
- або символьну змінну (рядок), що відповідає імені операції, що використовується.

Якщо значення, яке повертається керуючим виразом, не відповідає ні номеру можливої операції, ні її імені, то результатом оператора **switch** буде **NULL**.

Оператор **switch** найчастіше використовується всередині інших керуючих конструкцій.

Приклад 2.7. Створити числовий вектор **x** довжини 5, елементи якого приймають значення залежно від значення **i** – або cos(π), або e, або $\log_2(4)$, або $\lg_{10}(0.01)$, або число, що відповідає логічному значенню TRUE.

```
x=numeric(5)
for (i in 1:5)
x[i]=switch(i,cos(pi),exp(1),log2(4),log10(0.001),T)
x
```

Стандартна форма визначення функції в R.

Загальний вид оператора визначення функції:

function(аргументи) {вираз}.

Тут:

- **function** – ключове слово, яке повідомляє **R** про те, що буде створена функція;
- **аргументи функції** – список формальних аргументів (може мати довільну довжину), від яких залежить вираз; аргументи розділяються комами. Формальним аргументом може бути: символ (тобто ім'я змінної, наприклад, **x** або **y**), вираз виду **символ=вираз** (наприклад, **x=TRUE**), спеціальний формальний аргумент – три крапки «**…**»;
- **вираз** (тіло функції) – команда або блок команд (укладених у фігурні дужки {}), які залежать від визначених раніше аргументів функції. Окремі команди в блоці пишуться з нового рядка (але можна й на одному рядку через символ «;»).

Функція як своє значення повертає результат останнього у фігурних дужках виразу.

Звертання до функції має вигляд **ім'я(арг1, арг2, ...)**. Тут значення виразів **арг1, арг2, ...** є фактичними аргументами, які підставляються замість відповідних формальних аргументів, визначених при заданні функції.

Приклад 2.8. Створимо функцію, що обчислює норму – квадратний корінь скалярного добутку векторів *x* і *y*.

```
x=1:10
y=10:20
norm = function(x,y) sqrt(x%*%y)
norm(1:4,2:5)
```

Якщо фактичні аргументи задані в іменованому виді **ім'я = вираз**, то вони можуть бути перераховані в довільному порядку. Більше того, список фактичних аргументів може починатися з аргументів, представлених у звичайній позиційній формі, після чого можуть йти аргументи в іменованій формі.

При виклику функції фактичні аргументи заміняють формальні, прописані при створенні функції. Це може відбуватися різними способами:

- **точна відповідність** – якщо фактичні аргументи задані в іменованому виді, то при виклику функції відбувається пошук по повному імені відповідних формальних аргументів (тобто відбувається пошук тих формальних аргументів, чиє ім'я повністю відповідає імені фактичного аргументу);
- **часткова відповідність** – при виклику функції іменовані фактичні аргументи можуть замінити формальні й у тому випадку, якщо ім'я фактичного аргументу збігається із частиною імені формального;
- **позиційна відповідність** – неіменовані формальні аргументи заміняються неіменованими фактичними відповідно до порядку їхнього розташування (перший фактичний аргумент заміняє перший формальний і т.д.).

При розробці скриптів іноді буває необхідно створити функції для довільного числа формальних аргументів. Це можна зробити за допомогою формального аргументу три крапки «**...**», це означає, що до функції може бути передано довільне число фактичних аргументів.

Приклад 2.9. Створити функцію, що знаходить для довільного числа векторів їх мінімальні, максимальні й середні значення.

```
x=rnorm(100)
y=rnorm(200)
z=rnorm(300)
fun= function (...) {
data = list(...)
n =length(data)
maxs = numeric(n)
mins = numeric(n)
means<- numeric(n)
for (i in 1:n) {
maxs[i] = max(data[[i]])
mins[i]<-min(data[[i]])
means[i]<-mean(data[[i]])
```

```
}
print(maxs)
print(mins)
print(means)
invisible(NULL)
}
fun(x,y,z)
```

Формальні аргументи, локальні змінні й вільні змінні.

Всі ідентифікатори, що зустрічаються в тілі функції, діляться на три групи: це формальні аргументи, локальні змінні й вільні змінні.

Формальні аргументи (формальні параметри) – це аргументи, перераховані в заголовку функції (у круглих дужках після ключового слова **function**).

Локальні змінні – це змінні, що не є формальними аргументами, значення яких визначаються під час виконання функції.

Змінні, що не є формальними аргументами й локальними змінними, є **вільними змінними.**

Наприклад, у функції

```
f = function(x)
{
y = 2*x
print(x)
print(y)
print(z)
}
```

x – формальний аргумент, **y** – локальна змінна, **z** – вільна змінна. Область видимості формальних аргументів і локальних змінних деякої функції – тільки сама ця функція. Це означає, що зміни цих змінних усередині функції ніяк не відбиваються на змінних з такими ж іменами в зовнішній функції. Область видимості вільних змінних поширюється до зовнішньої функції, у якій вони були визначені. Зміна таких змінних у тілі функції впливає також на відповідні змінні в цій зовнішній функції.

Повна форма запису функції від короткої відрізняється тільки додатковим записом у тілі функції:

> ім'я = function(аргументи) {вираз
return(значення)
}

return(значення) – значення, що повертається функцією.

Сильне присвоєння в R.

Діючи звичайним чином, тобто змінюючи значення формальних аргументів усередині функції, не можна змінити відповідних фактичних аргументів.

Єдиний спосіб зробити це – використання в тілі функції сильного присвоювання, яке позначається символом «<<-»:

arg1 <<- вираз.

Приклад 2.10. Використання сильного присвоєння.

```
y=5;z=6
ff=function(x){
y=sum(x); z<<- sqrt(y)
return(y)}
ff(1:10)
y
z
```

Команда apply().

Команда **apply()** використовується тоді, коли потрібно застосувати яку-небудь функцію до рядка або стовпця матриці (таблиці даних). Повна форма запису:

apply(X, вказівник, функція, ...).

Тут:

- **X** – ім'я матриці або таблиці даних;
- **вказівник** – вказує, до чого застосовується функція: 1 – до рядків, 2 – до стовпців, **c(1,2)** – до рядків і стовпців одночасно (якщо функція застосовується до всіх елементів матриці й результат – матриця, то визначається порядок виводу елементів);
- **функція** – ім'я функції, що застосовується. Якщо потрібно застосувати прості операції виду «+», «-» і т.д., то їх необхідно задати в лапках.

Приклад 2.11. Застосування функції **apply()**.

```
#Спочатку створимо матрицю X.
(X=matrix(1:25,nrow=5))
#Знайдемо суму елементів по рядках
apply(X,1,sum)
#і по стовпцях
apply(X,2,sum)
#В обох випадках одержали вектори, чиї довжини
#відповідають числу стовпців
# і рядків відповідно.
#Знайдемо корінь квадратний.
apply(X,c(1,2),sqrt)
apply(X,1,sqrt)
apply(X,2,sqrt)
```

Команда sapply().

Функція **sapply()** застосовується аналогічно **apply()**, але тільки до векторів і списків. Вона корисна при складних ітераційних обчисленнях, тому що дозволяє уникнути створення циклів:

sapply(X, функція, ..., simplify = TRUE, USE.NAMES = TRUE).

Параметри функції наступні:

- **X** – об'єкт, до якого застосовується команда;

- **функція** – ім’я функції, яка застосовується до об’єкта **X**;
- **simplify** – логічний аргумент: чи потрібно представляти виведений результат у вигляді вектора або матриці (значення TRUE – за замовчуванням);
- **USE.NAMES** – логічний аргумент: якщо вказаний аргумент приймає значення TRUE (за замовчуванням) і **X** символьного типу, то як назва для виведених результатів використовується **X**.

У результаті застосування **sapply()** створюється список тієї ж розмірності, що й об’єкт **X**.

Команда lapply().

Функція

lapply(X, функція, ...)

є версією функції **sapply()**, а саме:

sapply(X, функція, ..., simplify = FALSE, USE.NAMES = FALSE).

lapply() є корисною при роботі зі списками, дозволяє застосовувати різні функції до елементів списку (числові функції можна застосовувати тільки в тому випадку, якщо всі елементи списку **X** належать класу **numeric**).

Приклад 2.12. Застосування функції **lapply()**.

```
#Створимо список із трьох компонентів – символьного,
#числового і логічного векторів.
a=c("a","b","c","d")
b=c(1,2,3,4,4,3,2,1)
c=c(T,T,F)
X=list(a,b,c)
#Перевіримо, до якого класу належить створений об'єкт і
#виведемо його елементи на екран.
class(X)
X
#Знайдемо за допомогою lapply() довжини елементів списку
lapply(X,length)
# і середні значення
lapply(X,mean)
```

Введення і виведення даних в R.

Функція **scan()**.

Дані зчитуються у вектор або в список з консолі або з файлу. Повний вигляд (форма запису):

scan(file = "", what = double(0), nmax = -1, n = -1, sep = "",
quote = if(identical(sep, "\n")) "" else "'\"", dec = ".",
skip = 0, nlines = 0, na.strings = "NA",
flush = FALSE, fill = FALSE, strip.white = FALSE,
quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,
comment.char = "", allowEscapes = FALSE,
encoding = "unknown").

Аргументи:

- **file** – ім’я файлу, звідки зчитуються дані; якщо file=, то дані вводяться з клавіатури; якщо задано тільки ім’я файлу, то пошук файлу ведеться в поточній директорії; інакше задається повний шлях до файлу; іменем файлу може бути URL;
- **what** – задається тип зчитуваних даних: **logica**l, **integer**, **numeric**, **complex**, **character**, **list**; якщо зчитується список, то рядки у файлі сприймаються як поля списку зазначених вище типів;
- **nmax** – ціле додатне число – максимальне число даних для читання або максимальне число записів у списку; при пропуску цього аргументу або при неправильному його заданні файл зчитується до кінця;
- **n** – ціле додатне число – максимальне число даних для читання; неправильні значення або не тип **integer** ігноруються;
- **sep** – роздільник полів; за замовчуванням – пробіл;
- **quote** – вид лапок (подвійні або одинарні);
- **dec** – десятковий роздільник (крапка або кома);
- **skip** – ціле позитивне число – число рядків файлу, які варто пропустити перед читанням;
- **nlines** – ціле додатне число – максимальне число рядків для зчитування;
- **na.strings** – символьний вектор – його елементи інтерпретуються як пропущені значення **NA**; порожні поля за замовчуванням зчитуються як **NA**;
- **flush** – логічний аргумент – значення TRUE дозволяє додавати коментарі до даних, що були зчитані;
- **fill** – логічний аргумент – при рівності значенню TRUE додаються порожні поля до рядків, у яких кількість полів даних менше визначеного параметром **what**;
- **strip.white** – логічний вектор; використовується тільки якщо задано параметр **sep**, видаляє порожній простір (пробіл) перед символьними змінними й після них;
- **quiet** – логічний аргумент; при значенні FALSE функція виведе повідомлення про те, скільки елементів було прочитано;
- **blank.lines.skip** – логічний аргумент; при значенні TRUE порожні рядки ігноруються (не зчитуються) (параметри skip і nlines однаково будуть ураховувати всі порожні рядки);
- **multi.line** – логічний аргумент; використовується якщо аргумент **what** приймає значення list; при значенні FALSE всі записи будуть зчитані в один рядок; якщо ж і fill=T, то зчитування при досягненні кінця рядка буде припинено;
- **comment.char** – символьний аргумент, визначає знак коментарю;
- **allowEscapes** – логічний аргумент – чи потрібно наступні послідовності символів \n, \a, \b, \f, \r, \t, \v при читанні розглядати як команди (TRUE) або просто як символи (FALSE);

– **encoding** – символьний аргумент, задає кодування файлу, що зчитується.

Функції read.table() і read.csv().

Якщо вихідні дані представлено у вигляді таблиці, й підсумковий результат повинен бути таблицею (фреймом даних), то зручніше скористатися функцією **read.table( )** або **read.csv()**.

Повний запис функції:

read.table(file, header = FALSE, sep = "", quote = "\"'",
dec = ".", row.names, col.names,
as.is = !stringsAsFactors,
na.strings = "NA", colClasses = NA, nrows = -1,
skip = 0, check.names = TRUE, fill = !blank.lines.skip,
strip.white = FALSE, blank.lines.skip = TRUE,
comment.char = "#",
allowEscapes = FALSE, flush = FALSE,
stringsAsFactors = default.stringsAsFactors(),
fileEncoding = "", encoding = "unknown").

Аргументи:

– **row.names** – вектор імен рядків; представляє собою або вектор з іменами рядків підсумкової таблиці; або число – номер стовпця вихідної таблиці з назвами рядків; або ім’я стовпця таблиці, де наведені назви рядків; якщо цей параметр не заданий, то рядка в підсумковій таблиці будуть пронумеровані;

– **col.names** – вектор імен стовпців у підсумковій таблиці;

– **as.is** – чи потрібно символьні змінні, не перетворені в числові або логічні, переводити у фактори. **as.is** – або логічний, або числовий вектор, що визначає стовпці, які не конвертуються у фактори;

– **colClasses** – символьний вектор; визначає класи даних у стовпцях (символьні, логічні, числові). Можливі значення: **NA** – автоматична конвертація типів даних, NULL – стовпець пропускається (дані не перетворюються), тип даних у який будуть переведені елементи стовпця, factor;

– **nrows** – цілочисловий аргумент; визначає максимальне число рядків, що зчитуються;

– **check.names** – логічний аргумент; при значенні TRUE імена змінних будуть перевірені на синтаксичну правильність і відсутність дублювання.

Функція write().

Функція **write()** призначена для запису даних (в основному матриць) в файл. Її вигляд:

write(x, file = "data",
ncolumns = if(is.character(x)) 1 else 5,
append = FALSE, sep = " ").

Аргументи:

- **x** – дані, які потрібно записати;
- **file** – ім’я файлу, куди буде записано інформацію;
- **ncolumns** – число стовпців, у яких буде записано інформацію;
- **append** – логічний аргумент – якщо значення TRUE, то дані допишуться у вихідний файл, якщо ж FALSE – файл буде переписано;
- **sep** – роздільник стовпців (\t – табуляція)

Якщо як аргумент функції задати тільки дані – write(x), то їх буде виведено на екран.

Функція cat().

Функція **cat()** перетворює вихідні дані в символьні, поєднує їх у єдиний символьний вектор і записує в заданий файл. Функція має вигляд:

cat(... , file = "", sep = " ", fill = FALSE, labels = NULL,
append = FALSE).

Її аргументи:

- **…** – об’єкти **R**, які будуть записані у файл;
- **file** – ім’я файлу, куди буде записано інформацію; якщо цей аргумент відсутній, то дані буде виведено на екран;
- **sep** – роздільник елементів об’єкта, що записується;
- **fill** – логічний або додатний цілочисловий аргумент – контролює створення нових рядків у файлі. Якщо **fill** – числовий, то задається довжина рядка (кількість символів у рядку). Якщо **fill** – логічний і його значення FALSE, то нові рядки створюються тільки при наявності в записуваних даних символу «\n»; якщо значення TRUE, то задається додатковий аргумент **width**, що визначає довжину створюваного у файлі рядка;
- **labels** – символьний вектор, що задає назви рядків. Ігнорується, якщо якщо аргумент **fill**=FALSE;
- **append** – логічний аргумент – використовується, якщо задано ім’я файлу. Якщо значення аргументу TRUE, то нові дані додаються до вихідного файлу, якщо FALSE, то записуються замість старих.

Функції write.table(), write.csv().

Функції **write.table()** записує таблицю даних (або матрицю) у заданий файл. Якщо записуваний об’єкт не є таблицею (фреймом даних), то він автоматично буде конвертований.

Вид функції:

write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
eol = "\n", na = "NA", dec = ".", row.names = TRUE,
col.names = TRUE, qmethod = c("escape", "double")).

Аргументи:

- **x** – об’єкт, що записується. Краще, щоб це була матриця або таблиця (фрейм) даних;
- **file** – ім’я файлу, у який буде записано дані;

- **append** – логічний аргумент – використовується, якщо тільки задане ім’я файлу. Якщо значення аргументу TRUE, то нові дані додаються до вихідного файлу, якщо FALSE – записуються замість старих;
- **quote** – логічний або числовий аргумент. Якщо **quote** є логічним аргументом і його значення TRUE, то всі символьні змінні й фактори буде записано у файл у подвійних лапках. Якщо значення аргументу FALSE – символьні змінні й фактори записуються без лапок. Якщо **quote** – числовий вектор, то його елементи задають номера стовпців, у яких дані повинні бути записані в лапках. Назви стовпців і рядків за замовчуванням буде записано в лапках;
- **sep** – аргумент, що визначає роздільник полів; значення в кожному рядку вихідних даних розділяються цим символом;
- **eol** – символ закінчення рядка; для Windows і Unix – "\n", "\r"; для MacOS – "\r";
- **na** – символьний аргумент для позначення відсутніх елементів у вихідних даних;
- **dec** – десятковий роздільник у числах (крапка або кома);
- **row.names** – аргумент, що задає назви рядків у файлі. Аргумент або логічний (якщо значення TRUE, то використаються імена рядків, зазначені у вихідних даних), або представляє собою символьний вектор, що безпосередньо задає імена рядків;
- **col.names** – аналогічний аргумент, що задає імена стовпців;
- **qmethod** – символьний вектор, що визначає як обробляти вкладені лапки – ""a"". Можливі два значення – **escape** (зовнішні лапки вбираються – за замовчуванням) і **double** (всі лапки залишаються).

Для функції **write.csv()** аргументи **col.names**, **sep**, **dec** і **qmethod** не можуть бути зміненими.

Завдання до лабораторної роботи

1. Написати власну функцію для сортування елементів вектора. Для перевірки роботи функції виконати сортування вектора випадкових чисел. Вхід функції – вектор, який треба відсортувати. Вихід функції – відсортований вектор.
2. Написати власну функцію пошуку заданого елемента у векторі. Вхід функції – вектор, елемент. Вихід функції – індекс елемента у векторі.

Для наступних завдань використайте таблицю даних **airquality** – вимірів якості повітря, яку можна завантажити, виконавши наступні команди:

```
library(datasets)
data(airquality)
```

Довідкову інформацію можна одержати виконавши

```
?airquality
```

3. Написати скрипт для визначення середньої температури повітря за кожний місяць. Записати результати розрахунків у файл.
4. Написати скрипт для визначення середньої швидкості вітру при показнику сонячної радіації більше 100.
5. Написати скрипт для визначення: у якому місяці середній зміст озону максимальний.

Контрольні запитання

1. Назвіть основні керуючі конструкції **R**.
2. Охарактеризуйте команди **apply()**, **sapply()** і **lapply()**.
3. Дайте визначення сильного присвоювання в **R**.
4. Опишіть способи організації функцій в **R**.
5. Охарактеризуйте команди введення\виведеня в **R**.

Лабораторна робота №3. Графічні можливості R

Мета: ознайомитися з функціями роботи з графікою в **R**.

Теоретичні відомості

Графічні засоби – важливий і надзвичайно універсальний компонент середовища **R**. Системою надається можливість використання стандартних засобів **R** для виводу на екран широкого спектра статистичних графіків, а також створення повністю нових типів графіків. У базовій конфігурації **R** один з пакетів відповідає за графіку – пакет **graphics**. Виділимо три групи функцій цього пакету:

- функції високого рівня (високорівневі);
- функції низького рівня (низькорівневі);
- інтерактивні функції.

Функції першого типу (високорівневі) приводять до створення графічного вікна, у якому будується задане зображення.

Низькорівневі функції дозволяють змінити (доповнити) вже побудоване раніше зображення. Самі вони не можуть створити графічне вікно. Як правило, більшість низькорівневих функцій дублюють аргументи високорівневих функцій.

Інтерактивні функції дозволяють за побудованим зображенням (за графіком, діаграмою) одержати деяку інформацію

Для того, щоб ознайомитися з можливостями пакета graphics, слід виконати команду

```
demo(graphics)
```

Функції високого рівня.

Виділимо основні функції високого рівня:

- **par()** – створення графічного вікна із заданими параметрами;
- **barplot()** – побудова мозаїчної діаграми (статистика);
- **boxplot()** – побудова boxplot – особої діаграми для статистичного аналізу даних щодо передбачуваного розподілу (статистика);
- **contour()** – побудова графіка з контурними лініями (лініями рівня);
- **curve()** – побудова лінії (кривих);
- **dotchart()** – точкова діаграма (статистика);
- **frame()** – задання нового графічного вікна;
- **plot.new()** – задання нового графічного вікна;
- **hist()** – побудова гістограми (статистика);
- **image()** – побудова кольорової прямокутної сітки згідно із заданими кольорами;

- **matplot()** – зображення елементів стовпців однієї матриці щодо елементів відповідних стовпців іншої матриці;
- **mosaicplot()** – побудова мозаїчної діаграми (статистика);
- **persp()** – побудова тривимірних графіків;
- **pie()** – побудова кругової діаграми;
- **plot()** – основна функція побудови двовимірних графіків;
- **plot.data.frame()** – графічний аналіз таблиць даних;
- **plot.default()** – побудова діаграми розсіювання;
- **plot.factor()** – побудова діаграми розсіювання для факторів;
- **plot.formula()** – побудова діаграми розсіювання за допомогою структури formula;
- **plot.histogram()** – побудова гістограми (статистика);
- **plot.table()** – графічний аналіз таблиці даних (побудова мозаїчних діаграм);
- **screen()** – керування графічним вікном;
- **stripchart()** – побудова діаграми розсіювання (аналог boxplot()).

Функція par().

Функція **par()**, визначає всі параметри графічного вікна

par(..., no.readonly = FALSE).

Аргументи:

- **...** – аргументи виду **ім'я=значення** або список іменованих аргументів, що визначають вид графічного вікна. Для того, щоб довідатися всі аргументи функції **par()**, необхідно звернутися до неї в **R**;
- **no.readonly** – логічний аргумент, якщо його значення TRUE і не задані інші аргументи, то будуть виведені тільки ті параметри функції **par()**, які встановлені для поточного графічного вікна й можуть бути встановлені (змінені) надалі.

Функція plot().

Функція

plot(x, y, ...),

дозволяє графічно відображати залежність `y` від `x`.

Аргументи функції:

x – координати точки на графіку.

y – координати по осі `y`, якщо задано відповідні значення по осі `x`.

... – додаткові аргументи. До них відносяться:

- **type** – символьний аргумент, що визначає тип побудови графіка. Можливі значення: "**p**" – точки (кола), "**l**" – лінії, "**b**" – лінії й точки, "**c**" – будуються тільки лінії з "**b**", "**n**" – нічого не будується (але визначаються області по осях);
- **log** – символьний аргумент – задається назва осі ("x", "y" або "xy"), координати по якій переводяться в логарифмічну шкалу.
- **lty** – стиль лінії. Можливі значення: "solid" (1) – суцільна лінія; "blank" (0) – відсутня лінія; "dashed" (2) – штрихова лінія; "dotdash"

(4) – штрих-пунктир; "dotted" (3) – пунктирна лінія; "longdash" (5) – довгий штрих; "twodash" (6) – подвійний штрих (слід зазначити, що повинен бути заданий параметр type="l").

- **xlab**, **ylab**, **main** і **sub** – символьні змінні – назви осей, основного й додаткового заголовків графіка.
- **col** – символьний або числовий аргумент – колір графіка.
- **bg** – символьна змінна, відповідальна за колір фону графічного вікна. За замовчуванням білий – bg="white".
- **col.axis** – символьна змінна – колір осей. За замовчуванням чорний – "black".
- **col.main** – символьна змінна – колір основного заголовка. За замовчуванням чорний – "black".
- **col.sub** – символьна змінна – колір додаткового заголовка. За замовчуванням чорний – "black".
- **fg** – символьна змінна – колір рамки навколо графіка.
- **font.axis**, **font.lab**, **font.main** і **font.sub** – числові аргументи – тип шрифту для назви осей, основного й додаткового заголовків. Можливі значення: 1 – простий (за замовчуванням), 2 – жирний; 3 – курсив і 4 – жирний курсив.
- **lwd** – додатне ціле число – товщина лінії; за замовчуванням lwd=1.
- **xlim** – числовий вектор **c(x1, x2)** – межі по осі x.
- **ylim** – числовий вектор **c(y1, y2)** – межі по осі y.
- **frame** – логічний аргумент – визначає, чи потрібно будувати рамку навколо графіка.

Приклад 3.1. Побудувати графік функції y = cos(x) на відрізку $[-2\pi;2\pi]$ двома способами й вивести в єдиному графічному вікні.

```
#Спочатку створимо графічне вікно й розділимо його на два
#вікна.
par(mfrow=c(1,2))
#Перший графік побудуємо, задаючи масиви точок по осі x і
осі y.
x=seq(-2*pi,2*pi,by=0.01)
y=cos(x)
#Будуємо графік:
plot(x,y,type='l',lty=1,col='black',ylab='cos(x)')
#Тепер візьмемо як перший аргумент plot() ім'я функції -
#cos, а також задамо початкове й кінцеве значення по осі
#x -2π;2π, відповідно.
plot(cos,-2*pi,2*pi)
#Результат:
```

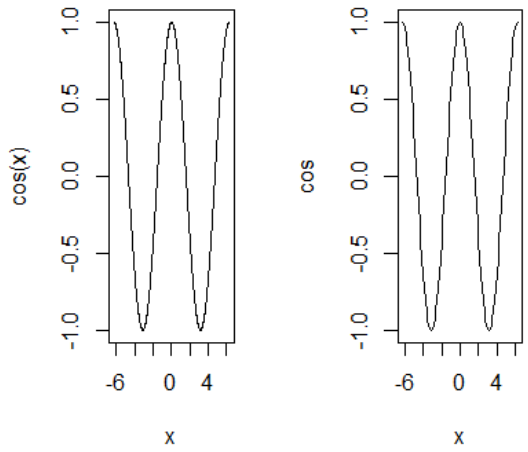

Рисунок 3.1 – Використання графічних функцій **R**

Функції низького рівня.

До функцій низького рівня можна віднести:

- **abline()** – побудова прямих ліній у вже існуючому графічному вікні;
- **arrows()** – зображення стрілок;
- **axis()** – побудова осі графіка;
- **box()** – побудова рамки навколо графіка;
- **grid()** – задання прямокутної сітки на графіку;
- **legend()** – задання різних легенд до графіка;
- **lines()** – побудова ліній, що з'єднують задані точки;
- **mtext()** – вивід тексту у відповідній області;
- **points()** – додавання точок на графік;
- **polygon()** – побудова багатокутників;
- **rect()** – побудова прямокутників;
- **segments()** – з'єднання точок прямими відрізками;
- **symbols()** – побудова одного з шести видів фігур (коло, квадрат, прямокутник, зірка, термометр, boxplot) на графіку;
- **text()** – додавання тексту до графіка;
- **title()** – додавання заголовків;
- **xspline()** – побудова сплайну відносно заданих контрольних точок.

Завдання до лабораторної роботи

1. Побудувати графіки тригонометричних функцій. Для кожної функції побудувати 5 графіків з різними типами\стилями ліній функції **plot()**.
2. Побудувати гістограму за даними температури з таблиці даних airquality. За допомогою команди низького рівня додати легенду на графік.

3. Побудувати точкову діаграму за даними температури з таблиці даних airquality.
4. Створити нове графічне вікно. Побудувати в цьому вікні кілька точок. З’єднати ці точки відрізками. Використати команди низького рівня.

Контрольні запитання

1. Назвіть основні команди роботи із графікою в R.
2. Поясніть відмінності графічних команд низького та високого рівня.

Лабораторна робота №4. Основи теорії ймовірностей в R

Мета: ознайомитися з функціями **R** для генерування та аналізу основних розподілів ймовірностей.

Теоретичні відомості

Базові ймовірнісні розподіли.

Функцією розподілу називають функцію $F(x)$, яка визначає ймовірність того, що випадкова величина X прийме значення менше за x (див. додаток А).

В базовій конфігурації системи **R** (пакет **stats**) реалізовано велику кількість стандартних ймовірнісних розподілів.

Розглянемо більш детально наступні базові розподіли.

1. Дискретні:
   - біноміальний;
   - пуассоновський;
   - геометричний;
   - гіпергеометричний.
2. Неперервні:
   - нормальний розподіл;
   - χ^2-розподіл;
   - розподіл Фішера;
   - розподіл Стьюдента.

Для кожного з розподілів в **R** вводиться чотири типи функцій.

1. Щільність розподілу (для неперервних випадкових величин) і ймовірність прийняття величиною конкретного значення (для дискретних випадкових величин) – префікс **d** перед назвою розподілу.
2. Функція розподілу випадкової величини – префікс **p** перед назвою розподілу.
3. Квантилі розподілу – префікс **q** перед назвою розподілу.
4. Випадкова вибірка по даному розподілу – префікс **r** перед назвою розподілу.

Далі розглянемо більш детально згадані розподіли.

Біноміальний розподіл.

Нехай проводиться n незалежних випробовувань, в кожному з яких подія A може наступити або не наступити. Ймовірність того, що подія наступить є постійною для всіх випробовувань та дорівнює p. Така схема проведення випробовувань називається схемою Бернуллі.

Випадкова величина ξ, яка описує число «успіхів» в ряді випробувань Бернуллі, належить до біноміального розподілу з параметрами p – ймовірність «успіху» в випробувані і n – загальне число випробувань Бернуллі.

Ймовірність $P\{\xi = k\}$ має вигляд:

$$P\{\xi = \mathrm{k}\} = C_n^k p^k (1-p)^{n-k}.$$

В **R** для біноміального розподілу реалізовано наступні функції:
dbinom(x, size, prob, log = FALSE),
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE),
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE),
rbinom(n, size, prob).

Аргументи функцій:

- **x** – цілочисловий невід'ємний вектор – вектор значень випадкової величини;
- **q** – невід'ємний вектор – вектор квантилей;
- **p** – вектор ймовірностей;
- **n** – довжина вектора, що створюється;
- **size** – один з параметрів біноміального розподілу – число випробовувань;
- **prob** – ймовірність успіху в одному випробовуванні;
- **log** – логічний аргумент (за замовченням – FALSE). Показує, чи потрібно обчислювати логарифм ймовірності;
- **log.p** – аналогічно **log**;
- **lower.tail** – логічний аргумент. Якщо його значення TRUE, то використовується $P\{\xi \le \mathrm{x}\}$, в іншому випадку використовується $F(\mathrm{x}) = P\{\xi > \mathrm{x}\}$.

В результаті роботи вказаних функцій отримаємо наступне.

dbinom(x, size, prob) – ймовірність того, що випадкова величина прийме задане значення. Якщо **x** – нецілe або від'ємне число, то результатом буде нуль.

pbinom(q, size, prob) – значення функції розподілу в точці **q**.

qbinom(p, size, prob) – значення квантиля Q_p для заданої ймовірності.

rbinom(n, size, prob) – створюється вектор випадкових чисел, які мають біноміальний розподіл.

Приклад 4.1. Нехай проводиться 10 випробовувань. Ймовірність «успіху» в кожному з них – 0,5. Знайдемо ймовірності $P\{\xi = k\}$ отримати k $(k = 1..10)$ успіхів в 10 випробовуваннях.

```
size=10; prob=0.5
dbinom(0:10,size,prob)
#Значення функції розподілу
pbinom(-1:10,size,prob)
#Квантилі
qbinom(seq(0,1,by=0.05),size,prob)
```

Пуассонівський розподіл.

Дискретна випадкова величина ξ має розподіл Пуассона з параметром λ, якщо:

$$P\{\xi = i\} = \frac{\lambda^i}{i!} e^{-\lambda}.$$

Функцій **R**, які відповідають за розподіл:

dpois(x, lambda, log = FALSE),
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE),
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE),
rpois(n, lambda).

Параметри:

- **x** – невід'ємний цілочисловий вектор;
- **q**, **lambda** – невід'ємні аргументи.

Геометричний розподіл.

Випадкова величина ξ, яка описує число «невдач» до появлення першого «успіху» в послідовності випробовувань Бернуллі, має геометричний закон розподілу з параметром p – ймовірністю «успіху» в одному випробовувані Бернуллі.

$$P\{\xi = i\} = (1-p)^i p,\ i \geq 0.$$

Функції в **R**:

dgeom(x, prob, log = FALSE),
pgeom(q, prob, lower.tail = TRUE, log.p = FALSE),
qgeom(p, prob, lower.tail = TRUE, log.p = FALSE),
rgeom(n, prob).

Параметри:

- **prob** – ймовірність «успіху» p.

Гіпергеометричний розподіл.

Розглянемо наступну задачу. У кошику знаходяться m білих та n чорних кульок. З кошику без повернення виймають k кульок ($0 < k < n + m$). Випадкова величина ξ, яка описує число i ($0 \leq i \leq \min(k,m)$) вийнятих білих кульок, має гіпергеометричний розподіл:

$$P\{\xi = i\} = \frac{C_m^i C_n^{k-i}}{C_{n+m}^k},\ 0 \leq i \leq \min(k,m).$$

Функції в **R**:

dhyper(x, m, n, k, log = FALSE),
phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE),
qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE),
rhyper(nn, m, n, k).

Параметри:

- **x** – невід'ємне число (вектор) – кількість вийнятих (без повернення) білих кульок;

– **m, n** – початкова кількість білих та чорних кульок (невід'ємні аргументи);
– **k** – загальна кількість вийнятих кульок.

Приклад 4.2. У кошику знаходиться 20 кульок: 6 білих, 14 червоних. З кошику без повернення виймають 6 кульок. Побудуйте розподіл випадкової величини ξ – кількості вийнятих білих кульок.

```
ph=numeric(7)
for(i in 0:6) ph[i+1]<-dhyper(i,6,14,6)
ph
sum(ph)
#Побудуємо мозаїчну діаграму (рис. 4.1)
barplot(ph,names=as.character(0:6),ylim=c(0,0.4),density=
16)
```

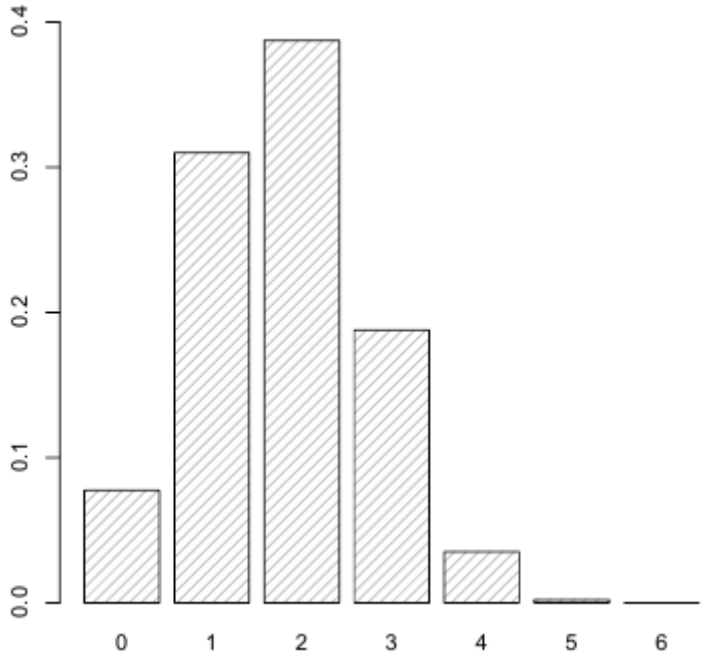

Рисунок 4.1 – Мозаїчна діаграма

Як можна побачити, найбільш ймовірно вийняти 2 білих та 4 червоних кульки.

Нормальний розподіл.

Нормальний розподіл – це найбільш значущий розподіл в теорії ймовірностей та математичній статистиці.

Щільність нормального розподілу

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}},$$

де m – це математичне сподівання, σ – середнє квадратичне відхилення.

Функції **R:**

dnorm(x, mean = 0, sd = 1, log = FALSE),

pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE),
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE),
rnorm(n, mean = 0, sd = 1).

Аргументи функцій:

- **x, q** – значення випадкової величини;
- **mean, sd** – параметри нормального закону розподілу – математичне сподівання (m) та середнє квадратичне відхилення (σ) відповідно.

χ^2-розподіл.

Розподіл, який має таку щільність:

$$p(x)=\begin{cases}0, & x<0,\\ \dfrac{1}{2^{\nu/2}\Gamma(\dfrac{\nu}{2})}x^{\frac{\nu}{2}-1}e^{-\frac{x}{2}}, & x\geq 0.\end{cases}$$

χ^2-розподіл є важливим тому, що велика кількість квадратичних форм (сум квадратів) випадкових величин мають даний закон розподілу, якщо вихідні випадкові величини мають нормальний розподіл.

Функції **R**:

dchisq(x, df, ncp=0, log = FALSE),
pchisq(q, df, ncp=0, lower.tail = TRUE, log.p = FALSE),
qchisq(p, df, ncp=0, lower.tail = TRUE, log.p = FALSE),
rchisq(n, df, ncp=0).

Аргументи функцій:

- **x, q** – невід'ємні числові вектори.
- **df** – параметр χ^2-розподіл – число степенів свободи.
- **ncp** – параметр зміщення (невід'ємне число).

Зміщений χ^2-розподіл з **df=**ν степенями свободи та параметром зміщення **ncp=**λ має щільність вигляду

$$p(\mathrm{x})=\mathrm{e}^{-\frac{\lambda}{2}}\sum_{r=0}^{\infty}\frac{\left(\dfrac{\lambda}{2}\right)^r}{r!}p_{\nu+2r}(x),$$

де $p_{\nu+2r}(x)$ – незміщений χ^2-розподіл з $\nu+2r$ степенями свободи.

Розподіл Фішера.

Розподіл названо в честь Фішера (R.A. Fisher). Щільність розподілу:

$$p(x)=\begin{cases}0, & x<0,\\ \dfrac{r\Gamma\left(\dfrac{1}{2(r+s)}\right)}{s\Gamma\left(\dfrac{1}{2r}\right)\Gamma\left(\dfrac{1}{2s}\right)}\cdot\dfrac{\left(\dfrac{rx}{s}\right)^{(r-1)/2}}{\left(1+\dfrac{rx}{s}\right)^{(\mathrm{r}+\mathrm{s})/2}} & x>0,\end{cases}$$

де r та s – число степенів свободи.

Функції **R**:
df(x, df1, df2, ncp, log = FALSE),
pf(q, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE),
qf(p, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE),
rf(n, df1, df2, ncp).
Аргументи:

- **df1, df2** – числа степенів свободи (r та s відповідно);
- **ncp** – параметр зміщення.

Розподіл Стьюдента.

Цей розподіл опублікував W.S. Gosset (під псевдонімом Student) у 1908 році. Щільність розподілу:

$$p(x) = \frac{\Gamma\left(\frac{1}{2(r+1)}\right)}{\sqrt{\pi r}\,\Gamma\left(\frac{1}{2r}\right)}\left(1+\frac{x^2}{r}\right)^{-(r+1)/2}.$$

Функції в **R**:
dt(x, df, ncp, log = FALSE),
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE),
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE),
rt(n, df, ncp).

Завдання до лабораторної роботи

1. Згенерувати випадкові вибірки випадкових величин, які мають розподіли ймовірностей, що розглянуті в теоретичній частині, об’ємом 50 елементів. Побудувати гістограми варіаційних рядів для отриманих наборів даних.
2. В кошику знаходиться 20 кульок: 6 білих та 14 червоних. З кошику без повернення виймається 6 кульок. Побудувати розподілення ймовірностей випадкової величини ξ – числа білих кульок, які були вийняті. Побудувати гістограму отриманого розподілення ймовірностей.
3. Після відповіді на питання екзаменаційного білету викладач задає студенту додаткові питання. Викладач перестає задавати питання як тільки студент не може відповісти на задане питання. Ймовірність того, що студент відповість на задане питання – 0,9. Завдання:
   - побудувати закон розподілу ймовірностей випадкової дискретної величини X – числа додаткових питань;
   - знайти найймовірніше число додаткових питань.

Контрольні запитання

1. Дайте визначення закону розподілу випадкової величини.
2. Дайте визначення функції розподілу.
3. Дайте визначення щільності розподілу.
4. Охарактеризуйте основні дискретні та неперервні розподіли.

Лабораторна робота №5. Основи математичної статистики в R

Мета: засвоїти функції первинної статистичної обробки даних **R**.

Теоретичні відомості

Аналіз категоріальних даних.

Деякі функції, які дозволяють працювати з категоріальними даними, було розглянуто в **лабораторній роботі № 1**. Це функції **factor()** та **table()**. Ці функції дозволяють визначати різні категорії в масиві даних, а також кількість елементів в кожній категорії.

Категоріальні дані можливо також аналізувати за допомогою графічних функцій **barplot()** (побудова мозаїчних діаграм) та **pie()** (побудова кругових діаграм).

Для переводу числових даних у категоріальні (побудови інтервального варіаційного ряду) можливо скористатись функцією **cut()**

cut(x, breaks, labels = NULL,
include.lowest = FALSE, right = TRUE, dig.lab = 3,
ordered_result = FALSE, ...),

де **x –** числовий вектор, який потрібно перетворити в категоріальні дані; **breaks** – будь-яке число (≥ 2), яке задає число інтервалів розбиття, або числовий вектор, який задає границі розбиття; **labels** – символьний вектор, який містить назви створюваних категорій; **include.lowest** – логічний аргумент – чи потрібно елемент x_i вектора **x**, який співпадає з нижньою чи верхньою границею одного з інтервалів розбиття, включити в цей інтервал; **right** – логічний аргумент – задає вид інтервалу розбиття: (a,b] (за замовчуванням) або [a,b); **dig.lab** – числовий аргумент – число знаків після коми в назві категорій (якщо назва – інтервали розбиття); **ordered_result** – логічний аргумент – чи потрібно впорядковувати створені категорії.

Приклад 5.1. Дослідити динаміку зміни оцінок з курсу «Теорія ймовірностей та математична статистика» за останні два роки: оцінки за стобальною шкалою, п’ятибальною шкалою та по системі ECTS. Початковими даними є вектори **year1**, **year2** – бали студентів за стобальною шкалою. Побудуйте інтервальний ряд та варіаційний ряд.

Виділимо наступні категорії:

- за п’ятибальною шкалою: 2 – бали від 0 до 59, 3 – від 60 до 74, 4 – від 75 до 89, 5 – від 90 до 100;
- за системою ECTS: FX – від 0 до 34, F – від 35 до 59, E – від 60 до 69, D – від 70 до 74, C – від 75 до 84, B – від 85 до 89, A – від 90 до 100.

```
year1=c(83, 70, 86, 51, 61, 67, 80, 84, 70, 64, 83, 55,
88, 75, 61, 70, 95)
year2=c(70, 31, 70, 46, 78, 69, 36, 33, 65, 70, 74, 51,
90, 26, 62, 70, 86, 60, 83)
# для розбиття на категорії скористаємось функцією cut()
cut1_1=cut(year1,breaks=c(-
1,59,74,89,100),labels=c("2","3","4","5"))
cut1_1
cut1_2=cut(year1,breaks=c(-
1,34,59,69,74,84,89,100),labels=c("F","FX","E","D","C","B
","A"))
cut1_2
cut2_1=cut(year2,breaks=c(-
1,59,74,89,100),labels=c("2","3","4","5"))
cut2_1
cut2_2=cut(year2,breaks=c(-
1,34,59,69,74,84,89,100),labels=c("F","FX","E","D","C","B
","A"))
cut2_2
#Таким чином, числові вектори перетворені у фактори.
#Визначимо число елементів у тій чи іншій категорії.
table1_1=table(cut1_1);table1_1
cut1_1
table2_1=table(cut2_1);table2_1
cut2_1
table1_2=table(cut1_2);table1_2
cut1_2
table2_2=table(cut2_2);table2_2
cut2_2
#Побудуємо таблиці відносних частот
table(cut1_1)/length(cut1_1)
table(cut2_1)/length(cut2_1)
table(cut1_2)/length(cut1_2)
table(cut2_2)/length(cut2_2)
```

Графічне дослідження вибірки.

Для графічного дослідження вибірки частіше використовують такі графічні функції: **barplot(), pie(), hist()**.

Приклад 5.2. Побудувати мозаїчні діаграми на основі таблиць, отриманих у прикладі 5.1.

```
matrix1=cbind(table1_1,table2_1);matrix1
matrix2=cbind(table1_2,table2_2);matrix2
par(mfrow=c(3,2))
barplot(height=table1_1)
```

```
barplot(height=table2_1)
barplot(height=table1_2)
barplot(height=table2_2)
barplot(height=matrix1,beside = TRUE)
barplot(height=matrix2,beside = TRUE)
```

Для створення кругових діаграм використовується функція **pie()**.

Приклад 5.3. Для таблиць, отриманих у прикладі 5.1 побудувати кругові діаграми.

```
par(mfrow=c(2,2))
pie(table1_1,labels=c('2','3','4','5'),edges=40,radius=1,
clockwise=T,
init.angle=60,density=c(2,4,6,8),angle=c(15,45,60,75),col
=1:4,
border=NULL,main='table1_1')
pie(table2_1,labels=c('2','3','4','5'),edges=60,radius=0.
8,clockwise=F,
init.angle=45,col=3:6,border='purple',main='table2_1')
pie(table1_2,labels=c('F','FX','E','D','C','B','A'),edges
=80,radius=1,
clockwise=T,init.angle=30,density=c(2,4,6,8,10,12,14),
angle=c(15,45,60,75,90,105,120),col=1:7,border=NULL,main=
'table1_2')
pie(table2_2,labels=c('F','FX','E','D','C','B','A'),edges
=100,radius=1,
clockwise=F,init.angle=30,col=1:7,border=NULL,main='table
2_2')
```

Статистичні характеристики.

До базових числових характеристик вибірки відносяться:

- вибіркове середнє та дисперсія;
- вибіркова коваріація;
- вибіркові квантилі;
- медіана.

Вибіркове середнє значення заданої вибірки **x**

$$\bar{m} = \frac{1}{n}\sum_{i=1}^{n} x_i ,$$

обчислюється функцією **mean()**:

mean(x, trim = 0, na.rm = FALSE, ...).

Функція **weighted.mean** обчислює за вибіркою **x** зважене середнє значення

$$\tilde{m} = \sum_{i=1}^{n} x_i w_i .$$

Середнє квадратичне відхилення вибірки **x** обчислюється функцією **sd()**.

Приклад 5.4. Для заданої вибірки розрахувати вибіркове середнє та середнє квадратичне відхилення.

```
#вибіркове середнє та
#середнє квадратичне відхилення
x=rnorm(100,3,2)
mean(x)
sd(x)
#обчислимо зважене середнє
#за статистичним рядом
y=rbinom(200,10,0.5)
table(y)
y1=1:9
w=as.vector(table(y))
weighted.mean(y1,w)
mean(y)
```

Вибіркова дисперсія вектора чи матриці **x** обчислюється функцією **var()**.

Функція **cov()** дозволяє будувати коваріаційну матрицю для заданих вибірок.

Функція **quantile()** дозволяє знаходити за вибіркою квантилі Q_α заданого розміру α $(0 \le \alpha \le 1)$.

Функції **summary()** та **fivenum()** виводять вектори, які містять деякі характеристики вибірки. **summary()** – мінімальне та максимальне значення вибірки, медіана, середнє, перший та третій квартилі. **fivenum()** – максимальне та мінімальне значення, медіана.

Функція **ecdf()** будує емпіричну функцію розподілу

$$F_n(x) = \sum_{x_i < x} \nu_i ,$$

де x_i – елементи статистичного ряду деякої вибірки, ν_i – частоти цих елементів.

Приклад 5.5. По виборці, елементи якої підкоряються біноміальному закону розподілу, побудуємо емпіричну функцію розподілу.

```
set.seed(0)
x=rbinom(100,100,0.5);x
Fn=ecdf(x);Fn
summary(Fn)
#вивід значень
#емпіричного розподілу
#у вигляді графіку
par(mfrow=c(2,2))
plot(Fn)
plot.ecdf(Fn)
plot.ecdf(x)
plot(Fn, verticals=TRUE, col.points="blue",
col.hor="red", col.vert="bisque")
#у вигляді переліку значень
Fn(x)
```

Функція **ecdf()** тільки обчислює значення емпіричної функції розподілу, але не виводить їх. Для отримання значень емпіричної функції розподілу потрібно скористатись функцією **plot()**, або присвоїти результат якійсь змінній та вивести її.

Завдання для лабораторної роботи

1. Згенеруйте послідовність випадкових 100 чисел, які мають біноміальний закон розподілення ймовірностей.
2. Для отриманої послідовності виконайте наступне:
   - побудуйте інтервальний варіаційний ряд;
   - обчислить вибіркове середнє, дисперсію, середнє квадратичне відхилення;
   - побудуйте гістограму розподілу ймовірностей та кругову діаграму;
   - обчисліть та побудуйте графічно емпіричну функцію розподілу.

Контрольні запитання

1. Назвіть основні числові характеристики вибірки.
2. Дайте визначення варіаційного ряду.
3. Дайте визначення теоретичної та емпіричної функції розподілу.

Лабораторна робота №6. Перевірка статистичних гіпотез

Мета: засвоїти функції **R** для перевірки статистичних гіпотез.

Теоретичні відомості

Випадкову величину можна розглядати як сукупність деяких значень, для яких є відомими ймовірності появи кожного значення. Ймовірності утворюють розподіл. Однак, кожна вибірка є лише окремими значеннями, зафіксованими в результаті спостережень над цією випадковою величиною. Крім цього, кожна вибірка має лише скінченну кількість елементів, а статистичний розподіл вибірки збігається по ймовірності до розподілу ймовірностей генеральної сукупності при нескінченному зростанні об'єму вибірки.

Кожне судження про генеральну сукупність на основі даної вибірки будемо називати **статистичною гіпотезою**. Обґрунтування гіпотези про генеральну сукупність називається **перевіркою статистичних гіпотез** або **статистичним доведенням.**

При статистичному доведенні можливі похибки двох типів:

- відкинути істинну гіпотезу (похибка першого типу);
- прийняти хибну гіпотезу (похибка другого типу).

Статистична перевірка гіпотез здійснюється по єдиній схемі. Частіше за все маємо одну або декілька вибірок.

1. На основі вибірок формулюється нульова гіпотеза, тобто початкове твердження про генеральну сукупність, яке хочемо підтвердити або спростувати.
2. Вибирається **рівень значущості**. Під рівнем значущості розуміється ймовірність відкинути істинну гіпотезу (допустити похибку першого рівня).
3. Вибирається відповідний до нульової гіпотези критерій її перевірки.
4. Обчислюються критичні значення обраного критерію перевірки гіпотези. Сукупність критичних значень утворює критичну область.
5. На основі отриманих даних робиться висновок про правомірність висунутої нульової гіпотези.

Більш докладно процес перевірки статистичних гіпотез розглянемо на прикладі задачі визначення чи підкорюється вибірка нормальному закону розподілу.

Перевірка статистичних гіпотез для однієї вибірки в R.

Тест Шапіро-Уілка.

Функція **shapiro.test(x)** виконує тест Шапіро-Уілка. Нульова гіпотеза полягає в тому, що випадкова величина, вибірка якої **x** відома, розподілена за

нормальним законом розподілу. Об'єм вибірки повинен бути не менше 3 і не більше 5000 елементів.

Об'єкт, що повертається функцією **shapiro.test(x)** – це список з наступними полями:

- **statistics** – значення статистики Шапіро-Уілка;
- **p.value** – величина, що використовується при тестуванні статистичних гіпотез – ймовірність допустити похибку першого роду.

Приклад 6.1. Протестувати стандартний датчик випадкових чисел, який має нормальний розподіл.

```
set.seed(0)
shapiro.test(rnorm(100, mean = 2, sd = 5))
```

Результат роботи:

```
Shapiro-Wilk normality test

data:  rnorm(100, mean = 2, sd = 5)
W = 0.9896, p-value = 0.6303
```

При рівні значущості, наприклад, $\alpha = 0{,}05$ гіпотезу слід прийняти, оскільки $p - value > \alpha$.

Приклад 6.2. Протестувати стандартний датчик випадкових чисел, який має розподіл Пуссона.

```
set.seed(0)
shapiro.test(rpois(100,0.5))
```

Результат роботи:

```
Shapiro-Wilk normality test

data:  rpois(100, 0.5)
W = 0.7026, p-value = 6.195e-13
```

На рівні значущості $\alpha = 0{,}05$ нульову гіпотезу слід відхилити, оскільки $p - value < \alpha$.

Приклад 6.3. Таблиця даних **trees** з бібліотеки **datasets** містить виміри діаметру, висоти та об'єму дерев. Перевірити гіпотезу про те, що висота дерев розподілена за нормальним законом розподілу.

```
colnames(trees)
x <- trees[, "Height"]
hist(x, col = "green", xlab = "Tree heights",
main = "Tree height frequencies")
shapiro.test(x)
```

Результати роботи функції **shapiro.test(x)**:

```
Shapiro-Wilk normality test

data:  x
W = 0.9655, p-value = 0.4034
```

Оскільки $p-value > \alpha$, гіпотезу про нормальність розподілу висот дерев при рівні значущості $\alpha = 0,05$ слід прийняти.

Завдання до лабораторної роботи

1. Проаналізувати дані з фрейму даних **mtcars** та **iris**, щодо відповідності їх нормальному розподілу. Для цього потрібно проаналізувати признаки з фрейму за допомогою тесту Шапіро-Уілка та виділити ті признаки, які розподілені за нормальним законом.

Контрольні запитання

1. Дайте визначення статистичної гіпотези.
2. Охарактеризуйте типи помилок, які можливі при статистичному доведенні.
3. Поясніть, яким чином можна перевірити вибірку на відповідність нормальному закону розподілу ймовірностей.

Лабораторна робота №7. Основи кореляційного аналізу в R

Мета: засвоїти основні засоби проведення кореляційного аналізу в **R**

Теоретичні відомості

Величини можуть бути або незалежними, або зв'язаними **функціональною** або **стохастичною** (ймовірнісною) залежністю.

Функціональна залежність величин реалізується тоді, коли кожному значенню однієї величини (аргументу) відповідає певне значення іншої величини. Прикладом функціональної залежності є довжина кола $C = 2\pi R$ залежно від її радіуса R.

В більшості випадків між змінними існують залежності, при яких кожному значенню однієї величини (аргументу) відповідає не якесь певне значення іншої величини, а множина її можливих значень – певний розподіл. Така залежність називається **стохастичною**, або ймовірнісною.

Окремим випадком ймовірнісної залежності є **кореляційна залежність** – стохастична залежність між випадковими величинами, при якій спостерігається функціональна залежність між значеннями однієї величини й середніми значеннями іншої величини.

Кореляційний зв'язок, на відміну від функціонального, показує лише тенденцію зміни однієї величини під дією іншої, отже, на підставі кореляції можна стверджувати лише про ступінь зв'язку між змінними, але не про існування причинно-наслідкової залежності між ними.

Кореляційні зв'язки розрізняються по тісноті (силі) зв'язку й кількості ознак. Прийнято виділяти такі види сили зв'язку: тісна (сильна), середня (помірна), слабка й нульова (відсутня) сили зв'язку.

По кількості ознак кореляція може бути парною (між двома ознаками) і множинною (між декількома ознаками).

Форма парної кореляції може бути лінійною, яка описується лінійною функцією регресії, і нелінійною (криволінійною), що описується нелінійними функціями.

Парна лінійна кореляція, у свою чергу, може бути додатною («прямою») і від'ємною («зворотною»). При прямій кореляції при зростанні однієї ознаки в середньому збільшується інша, у випадку ж зворотної кореляції при зростанні однієї ознаки інша в середньому зменшується.

Зображені на координатній площині точки (x_i, y_i), де x_i і y_i – значення першого і другого параметру, називаються кореляційним полем (рис. 7.1).

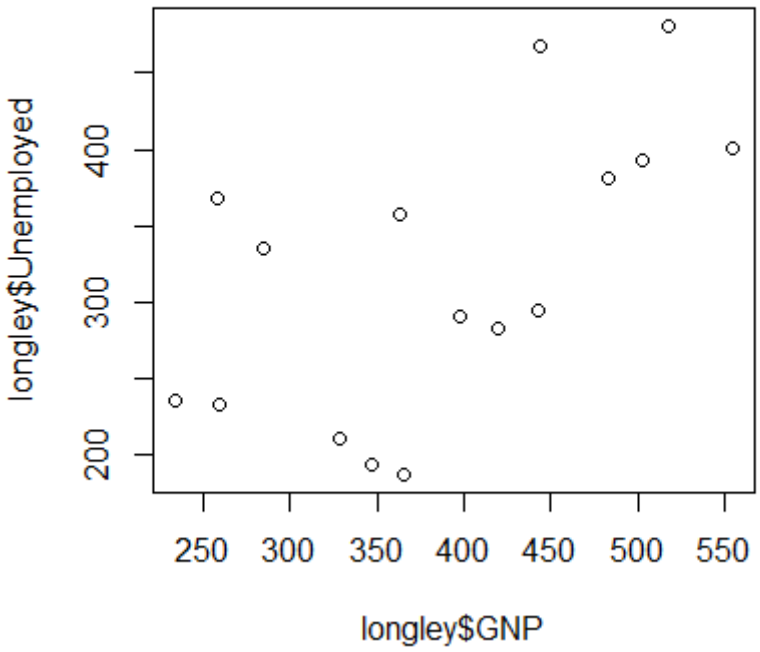

Рисунок 7.1 – Кореляційне поле

Висновок про форму парного кореляційного зв'язку можна зробити, зобразивши кореляційне поле на координатній площині.

Коефіцієнт кореляції.

Коефіцієнт кореляції генеральної сукупності ρ – це кореляція, обчислена з використанням усіх пар значень ознак (x, y) генеральної сукупності.

Коефіцієнт кореляції є безрозмірною величиною, що змінюється в межах від -1 до 1.

При $\rho = \pm 1$ кореляційний зв'язок представляє собою **лінійну функціональну залежність**, при цьому всі спостережувані значення розташовуються на прямій.

Для **незалежних випадкових величин коефіцієнт кореляції дорівнює нулю**. Однак **рівність нулю коефіцієнта кореляції** не завжди означає незалежність випадкових величин: це **свідчить лише про відсутність лінійної кореляційної залежності між досліджуваними величинами**, але не кореляційної залежності взагалі. Коефіцієнт кореляції може дорівнювати нулю у випадку нелінійного зв'язку.

Напрямок лінійного кореляційного зв'язку визначається знаком коефіцієнта кореляції: для додатного зв'язку $\rho > 0$, для від'ємного зв'язку $\rho < 0$.

Тісnота (сила) лінійного зв'язку між випадковими величинами визначається абсолютною величиною коефіцієнта кореляції, можливі діапазони значень наведено у таблиці 7.1.

Таблиця 7.1 – Можливі значення коефіцієнту кореляції

Значення коефіцієнта кореляції	Сила зв'язку
$\|\rho\|=1$	лінійна функціональна залежність
$0,95 \le \|\rho\| < 1$	дуже сильних зв'язок, майже функціональний
$0,75 \le \|\rho\| < 0,95$	зв'язок сильний
$0,5 \le \|\rho\| < 0,75$	зв'язок середній
$0,2 \le \|\rho\| < 0,5$	зв'язок слабкий
$0 \le \|\rho\| < 0,2$	практично немає зв'язку

Коефіцієнт лінійної кореляції Пірсона.

Коефіцієнт лінійної кореляції Пірсона r використовується для оцінки ρ, якщо виконано наступні припущення:

- змінні x та y лінійно залежні;
- змінні є випадковими;
- обидві змінні мають нормальний розподіл.

Коефіцієнт кореляції Пірсона розраховується за формулою:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\left(\sum_{i=1}^{n}(x_i - \overline{x})^2\right)\left(\sum_{i=1}^{n}(y_i - \overline{y})^2\right)}},$$

де $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ – середні арифметичні значення кожного з параметрів.

Величина вибіркового коефіцієнта лінійної кореляції Пірсона, як і генерального, змінюється в межах від –1 до +1. При малому об'ємі вибірки (n < 100) значення коефіцієнта лінійної кореляції Пірсона необхідно корегувати за формулою:

$$r' = r\left(1 + \frac{1 - r^2}{2(n-3)}\right).$$

Вибірковий коефіцієнт лінійної кореляції Пірсона, як і всі вибіркові характеристики, є випадковою величиною й при повторенні вимірів може приймати інші значення. Тому для незалежних випадкових величин, для яких генеральний коефіцієнт кореляції ρ дорівнює нулю, вибірковий коефіцієнт r може помітно відрізнятися від нуля, і навпаки. У зв'язку із цим завжди виникає

важлива практична задача, яка полягає в перевірці значущості вибіркового коефіцієнта кореляції.

Нульова гіпотеза h_0 полягає у відсутності лінійного кореляційного зв'язку між досліджуваними змінними в генеральній сукупності: $\rho = 0$. Альтернативною гіпотезою h_1 є твердження про те, що генеральний коефіцієнт кореляції ρ відмінний від нуля: $|\rho| \neq 0$.

Рангова кореляція.

На практиці часто виникає потреба аналізу зв'язку між змінними, які не можуть бути виміряні в інтервальній шкалі, але проте піддаються впорядкуванню й можуть бути проранжовані за степенем убування або зростання ознаки. Для визначення тісноти зв'язку між ознаками, вимірюваних в порядкових шкалах, застосовуються методи рангової кореляції. До них відносяться: коефіцієнти рангової кореляції Спірмена й Кендалла (використовуються для визначення тісноти зв'язку між двома величинами).

Використання коефіцієнта лінійної кореляції Пірсона у випадку, коли про закон розподілу й про тип вимірювальної шкали відсутня будь-яка надійна інформація, може привести до істотних помилок.

Методи рангової кореляції можуть бути використані для визначення тісноти зв'язку не тільки між кількісними змінними, але й між якісними ознаками за умови, що їхні значення можна впорядкувати й проранжувати. Ці методи також можуть бути застосовані до ознак, які вимірюються в інтервальних шкалах, однак їхня ефективність у цьому випадку завжди буде нижче.

Коефіцієнт рангової кореляції Спірмена.

Кожна із двох сукупностей розташовується у вигляді варіаційного ряду із присвоєнням кожному члену ряду відповідного порядкового номера (рангу), вираженого натуральним числом. Однаковим значенням ряду присвоюється середнє рангове число.

Порівнювані ознаки можна ранжувати у будь-якому напрямку: як у бік погіршення якості (ранг 1 одержує найбільший, швидкий, розумний і т.д.), так і навпаки. Головне, щоб обидві змінні були проранжовані однаковим способом.

Коефіцієнт рангової кореляції Спірмена підраховується за формулою

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n^3 - n}$$

де d_i – різниця рангів для кожної *i*-пари з *n* спостережень.

Ранговий коефіцієнт кореляції Спірмена, як і Пірсона, змінюється від –1 до +1, однак значення рангового коефіцієнта кореляції Спірмена завжди менше значення коефіцієнта лінійної кореляції Пірсона: $r_s < r$.

Коефіцієнт рангової кореляції Кендалла.

Застосовується для виявлення взаємозв'язку між кількісними або якісними показниками, якщо їх можна ранжувати. Коефіцієнт кореляції Кендалла має ті ж властивості, що й коефіцієнт Спірмена (змінюється від -1 до

+1, для незалежних випадкових величин дорівнює нулю), однак він вважається більше інформативним.

Обчислення коефіцієнтів кореляції в R.

В **R** коефіцієнт кореляції Пірсона, так само як і інші коефіцієнти, можна розрахувати за допомогою функцій **cor()** і **cor.test()**. Різниця між цими двома функціями полягає в тому, що **cor()** дозволяє обчислити тільки сам коефіцієнт кореляції, тоді як **cor.test()** виконує ще й оцінку статистичної значущості коефіцієнта, перевіряючи нульову гіпотезу про рівність його нулю.

Завдання до лабораторної роботи

1. Для заданої таблиці даних обчислити коефіцієнти кореляції та зробити висновок про тісноту зв'язку між параметрами x та y згідно варіанта.

№ варіанта	Дані
1	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(32,36,42,46,48,50,52,54,58,62,64,66,70,74,76,78,80,84,88,90)
2	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(5,9,15,19,21,23,25,27,31,35,37,39,43,47,49,51,53,57,61,63)
3	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(31,39,66,94,111,130,151,174,226,286,319,354,430,514,559, 606,655,759,871,930)
4	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(30.5,34.5,48,62,70.5,80,90.5,102,128,158,174.5,192,230,272, 294.5,318,342.5,394.5,450.5,480)
5	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(-79,-53,136,432,649,920,1251,1648,2664,4016,4833,5752, 7920,10568,12087,13744,15545,19603,24309,26920)
6	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(30,0,-210,-530,-762,-1050,-1400,-1818,-2882,-4290,-5138, -6090,-8330,-11058,-12620,-14322,-16170,-20328,-25142,-27810)
7	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(17,29,47,59,65,71,77,83,95,107,113,119,131,143,149, 155,161,173,185,191)
8	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(6,8,11,13,14,15,16,17,19,21,22,23,25,27,28,29,30,32,34,35)
9	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(15,21,30,36,39,42,45,48,54,60,63,66,72,78,81,84,87,93,99,102)
10	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(-6,2,14,22,26,30,34,38,46,54,58,62,70,78,82,86,90,98,106,110)

11	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(-26,-18,-6,2,6,10,14,18,26,34,38,42,50,58,62,66,70,78,86,90)
12	x=c(1,3,6,8,9,10,11,12,14,16,17,18,20,22,23,24,25,27,29,30) y=c(38,34,28,24,22,20,18,16,12,8,6,4,0,-4,-6,-8,-10,-14,-18,-20)

Контрольні запитання

1. Визначте мету кореляційного аналізу.
2. Дайте визначення коефіцієнта кореляції.
3. Охарактеризуйте засоби **R** для проведення кореляційного аналізу.

Лабораторна робота №8. Основи регресійного аналізу в R

Мета: засвоїти функції регресійного аналізу в **R**.

Теоретичні відомості

Регресія або **регресійний аналіз** – це сукупність статистичних методів, які застосовуються для знаходження, аналізу, моделювання й прогнозування залежностей між змінними. В **R** існує широкий набір функцій для проведення різних видів регресійного аналізу. Кожна із цих функцій, як правило, містить ключовий параметр **formula** – модель, відповідно до якої аналізуються дані.

Лінійна регресія.

Найпростішим видом регресії є лінійна регресія, що моделює лінійну залежність між змінною y і незалежними змінними $x_1, x_2, ..., x_n$. У загальному випадку множинної лінійної регресії залежність між змінними має вигляд:

$$y_i = b_0 + b_1 x_1 + b_2 x_2 + ... + b_i x_i + b_n x_n + e_i,$$

де y_i – значення змінної *y* при *i*-му спостереженні, b_i – коефіцієнти (параметри) регресії, $x_{i,j}$ – незалежні змінні, $e_1, e_2, ..., e_n$ – залишки (нев'язки) – незалежні між собою випадкові величини.

Лінійний регресійний аналіз в **R** реалізується функцією

lm(formula, data, …),

де **formula** – символьний опис моделі регресії, **data** – об'єкт даних (фрейм даних).

Параметр **formula** відіграє важливу роль у регресійному аналізі за допомогою **R** тому, що описує зв'язок між змінними, відповідно до якого аналізуються дані з **data**.

У найпростішому випадку лінійної залежності

$$y_i = b_0 + b_1 x_i + e_i$$

формула моделі виглядає так: $y \sim x$

Можливі й інші види залежностей. На рисунку 8.1 наведено основні моделі та відповідні їм формули **R**.

Формула	*Модель*
`Y ~ A`	$Y = \beta_0 + \beta_1 A$
`Y ~ -1 + A`	$Y = \beta_1 A$
`Y ~ A + I(A^2)`	$Y = \beta_0 + \beta_1 A + \beta_2 A^2$
`Y ~ poly(A, 2)`	$Y = \beta_0 + \beta_1 A + \beta_2 A^2$
`Y ~ A + B`	$Y = \beta_0 + \beta_1 A + \beta_2 B$
`Y ~ A:B`	$Y = \beta_0 + \beta_1 AB$
`Y ~ A*B`	$Y = \beta_0 + \beta_1 A + \beta_2 B + \beta_3 AB$
`Y ~ (A + B)^2`	$Y = \beta_0 + \beta_1 A + \beta_2 B + \beta_4 AB$

Рисунок 8.1 – Види формул регресійних моделей

Проілюструємо побудову лінійної регресії на прикладі аналізу даних економічних показників – **longley** з бібліотеки **datasets**.

Приклад 8.1. Побудова моделі лінійної регресії для даних **longley**.

```
library(datasets)
data(longley)
?longley
plot(longley)
x=lm(GNP~Population,data=longley)
summary(x)
```

Результат використання функції **lm** зберігається в цьому випадку в об’єкті **x**, що є об’єктом класу **lm**. Функція **summary()** у випадку застосування її до об’єктів класу **lm** повертає формулу моделі, залишки (residuals), коефіцієнти регресії, середнє квадратичне відхилення оцінки регресії, коефіцієнт детермінації R^2, статистику дисперсійного аналізу. Результат виконання наведеного коду:

```
Call:
lm(formula = GNP ~ Population, data = longley)

Coefficients:
(Intercept)   Population
   -1275.21        14.16

Call:
lm(formula = GNP ~ Population, data = longley)

Residuals:
     Min       1Q   Median       3Q      Max
```

```
-21.2942 -11.3519  -0.6804  11.2152  18.1277

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1275.2100    59.8256  -21.32 4.53e-12 ***
Population     14.1616     0.5086   27.84 1.17e-13 ***
-i-
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1

Residual standard error: 13.7 on 14 degrees of freedom
Multiple  R-squared:   0.9823,       Adjusted  R-squared:
0.981
F-statistic: 775.2 on 1 and 14 DF,  p-value: 1.168e-13
```

Достовірність отриманих результатів можна перевірити представивши результати аналізу в графічному вигляді

plot(x).

Функція **update()** дозволяє змінити параметри аналізу. Наприклад, команда

x2=update(x,~ +Year)

дозволяє обновити модель лінійного аналізу з урахуванням додаткової незалежної змінної **Year**. Результат нового аналізу збережеться в об'єкт **x2**:

```
Call:
lm(formula = GNP ~ Population + Year, data = longley)

Coefficients:
(Intercept)   Population         Year
 -34306.363        2.175       17.620
```

Нелінійна регресія.

Функція **nls()** дозволяє проводити нелінійний регресійний аналіз і знаходити параметри моделі нелінійної регресії вигляду

$$y_i = f(x_i, \beta) + e_i,$$

де y_i – значення змінної y в *i*-му спостереженні, f – певна нелінійна функція, x_i – незалежні змінні, $\beta = (\beta_1, ..., \beta_p)$ – коефіцієнти (параметри) регресії, які розраховуються на основі даних (x_i, y_i), $e_1, ..., e_n$ – залишкові компоненти (нев'язки) – незалежні величини.

Синтаксис функції **nls()** має вигляд:

nls(formula,data,start,...),

де **formula** – формула нелінійної моделі, **data** – дані у вигляді фрейму даних, **start** – вектор початкових (наближених) значень параметрів регресії. На відміну від функції **lm()** у функції **nls()** формула має звичайну математичну нотацію.

Приклад 8.2. Побудова моделі нелінійної регресії.

```
# початкові дані
xdata = c(-2,-1.64,-1.33,-0.7,0,0.45,1.2,1.64,2.32,2.9)
ydata =
c(0.699369,0.700462,0.695354,1.03905,1.97389,2.41143,1.91
091,0.919576,-0.730975,-1.42001)
# побудова графіку
plot(xdata,ydata)
# задаємо початкові значення
p1 = 1
p2 = 0.2
# будуємо модель
fit = nls(ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata),
start=list(p1=p1,p2=p2))
# загальні дані моделі
summary(fit)
# інші початкові значення
p1 = 2
p2 = 0.5
# побудова моделі
fit1 = nls(ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata),
start=list(p1=p1,p2=p2))
# загальні дані моделі
summary(fit1)
```

Завдання до лабораторної роботи

1. За допомогою вбудованої довідки вивчити параметри функцій **lm()** та **nls()**.
2. Для фреймів даних **airmiles**, **freeny**, **pressure**, **uspop** дослідити залежності між даними. Побудувати рівняння регресії.
3. Для набору даних **x** та **y** з лабораторної роботи №7 дослідити залежність між даними та побудувати рівняння регресії. Для цього необхідно:
   - побудувати кореляційне поле вибраних даних;
   - по виду розташування точок на кореляційному полі оцінити вид залежності;
   - обчислити вибірковий коефіцієнт кореляції для вибраних даних;
   - на основі виду кореляційного поля та значень коефіцієнту кореляції зробити висновок про тип рівняння регресії (лінійна або нелінійна регресія);
   - побудувати регресійну модель за допомогою функцій **lm()** або **nls()**.

– виписати рівняння регресії з числовими значеннями коефіцієнтів регресії;
– оцінити якість побудованої моделі регресії.

Контрольні запитання

1. Назвіть мету регресійного аналізу.
2. Охарактеризуйте види регресійного аналізу.
3. Поясніть яким чином визначається вид функціональної залежності в лінійній і нелінійній моделі регресії в **R**?

Лабораторна робота №9. Основи кластерного аналізу в R

Мета: засвоїти основні засоби кореляційного аналізу в **R**.

Теоретичні відомості

Один з фундаментальних процесів у науці й практиці – класифікація. Факти і явища повинні бути впорядковані, класифіковані, перш ніж дослідник може зрозуміти їх. Аналіз даних має у своєму розпорядженні великий арсенал методів класифікації.

Залежно від наявності попередньої вибіркової інформації говорять про класифікацію без навчання (інформація про самі класи відсутня) і про класифікацію з навчанням (є навчальні вибірки). Розв'язання задач класифікації без навчання проводиться методами кластерного аналізу.

Постановка задачі кластерного аналізу.

Є множина $O=\{O_1,O_2,...,O_N\}$, що складається з N об'єктів. Кожний об'єкт описується за допомогою n ознак $x_1,x_2,...,x_n$. Сукупність значень ознак зведена в матрицю X:

$$X=\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ & \dots & & \\ x_{N1} & x_{N2} & \dots & x_{Nn} \end{bmatrix}.$$

Матрицю X можна інтерпретувати як множину точок $X_1=(x_{11},x_{12},...,x_{1n})$, $X_2=(x_{21},x_{22},...,x_{2n}),\dots,$ $X_N=(x_{N1},x_{N2},...,x_{Nn})$ в n-вимірному евклідовому просторі E_n.

Нехай m – ціле число, $m<N$. Задачу кластерного аналізу можна сформулювати таким чином: на підставі даних, що містяться в X, розбити множину об'єктів O на m кластерів (підмножин) $K_1, K_2,..., K_m$ так, щоб кожний об'єкт належав одній й тільки одній підмножині, і щоб об'єкти, що належать тому самому кластеру, були подібними (близькими), тоді як об'єкти, що належать різним кластерам, були несхожими (відмінними).

Розв'язання задачі кластерного аналізу вимагає формалізації понять близькості й відмінності. Розглянемо поняття відстані між точками X_i й X_j з E_n.

Функції відстані й подібності.

Невід'ємна дійсна функція $d(X_i,X_j)$ називається функцією відстані (метрикою), якщо виконуються такі властивості:

1. $d(X_i, X_j) \geq 0 \ \forall \ X_i, X_j \in E_n$.
2. $d(X_i, X_j) = 0$ лише для $X_i = X_j$.
3. $d(X_i, X_j) = d(X_j, X_i)$.
4. $d(X_i, X_j) \leq d(X_i, X_k) + d(X_k, X_j)$, де X_i, X_j, X_k – будь-які три точки з E_n (так зване «правило трикутника»).

Значення функції d для двох заданих точок X_i, X_j еквівалентно відстані між об'єктами O_i і O_j.

Як приклад функцій відстаней приведемо найбільш уживані.

Евклідова відстань:

$$d_2(X_i, X_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} .$$

Сума абсолютних відхилень:

$$d_1(X_i, X_j) = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right| .$$

Відстані між N об'єктами можуть бути зведені у квадратну симетричну матрицю відстаней

$$D = \begin{bmatrix} 0 & d_{12} & \ldots & d_{1N} \\ d_{21} & 0 & \ldots & d_{2N} \\ & & \ldots & \\ d_{N1} & d_{N2} & \ldots & 0 \end{bmatrix}.$$

Поняттям, протилежним відстані, є поняття подібності. Мірою подібності називають невід'ємну дійсну функцію s, що задовольняє наступним аксіомам:

1. $0 \leq s(X_i, X_j) \leq 1, \ X_i \neq X_j$.
2. $s(X_i, X_i) = 1$.
3. $s(X_i, X_j) = s(X_j, X_i)$.

Значення функції подібності елементів множини O можна об'єднати в матрицю подібності:

$$S = \begin{bmatrix} 1 & s_{12} & \ldots & s_{1N} \\ s_{21} & 1 & \ldots & s_{2N} \\ & & \ldots & \\ s_{N1} & s_{N2} & \ldots & 1 \end{bmatrix}.$$

Міра близькості між кластерами.

Нехай K_i – i-й кластер, що містить N_i об'єктів; $\bar{X}_i$ – середнє арифметичне спостережень, що входять в K_i, тобто

$$\bar{X}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} X_j, \ X_j \in K_i .$$

Найбільш уживані відстані між кластерами.

Відстань, вимірювана за принципом найближчого сусіда:

$$D_{\min}(K_l, K_m) = \min_{X_i \in K_i, X_j \in K_m} d(X_l, X_m).$$

Відстань, вимірювана за принципом далекого сусіда:

$$D_{\max}(K_l, K_m) = \max_{X_i \in K_i, X_j \in K_m} d(X_l, X_m).$$

Відстань, вимірювана по центрах ваги кластерів:

$$D_{cp}(K_l, K_m) = d(\bar{X}_l, \bar{X}_m).$$

Ілюстрація трьох наведених мір:

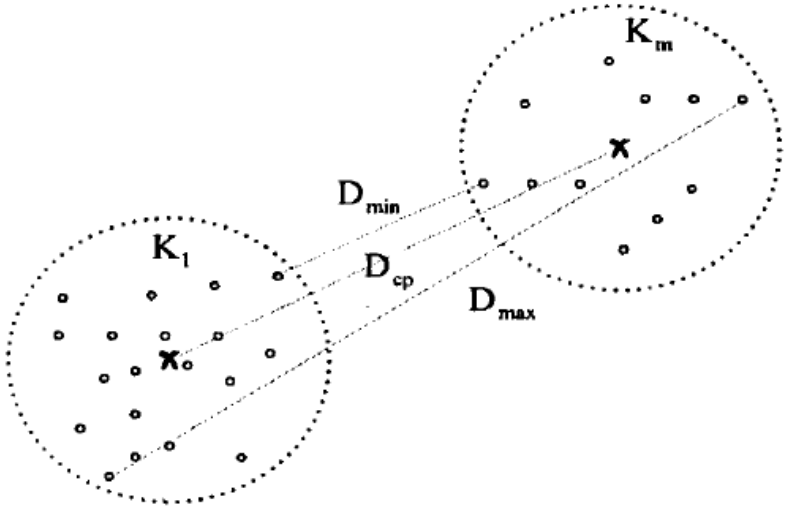

Рисунок 9.1 – Різні міри близькості між кластерами

Функціонали якості розбиття на кластери.

Розбиття вихідної сукупності об’єктів на кластери може здійснюватися різними способами. Природно визначити той кількісний критерій, виходячи з якого можна було б робити висновок про якість розбиття. З цією метою в задачах кластерного аналізу вводять поняття функціонала якості розбиття $G(K)$. Цей функціонал визначається на множині всіх можливих варіантах розбиття. Розбиття K^*, що приводить до екстремуму обраного функціоналу, вважається найкращим.

Слід зазначити, що вибір функціоналу, так само як і вибір метрики, здійснюється довільно.

Функціонал якості розбиття повинен відбивати різнорідність множини об’єктів. Користуючись поняттям відстані, введемо спочатку міри розсіювання множини об’єктів, які будуються на базі матриці відстаней D. Величину $S_d(O)$, рівну напівсумі всіх елементів матриці D, називають загальним розсіюванням, а величину

$$\bar{S}_d(O) = \frac{S_d(O)}{N_d},$$

де

$$N_d = \frac{N(n-1)}{2},$$

називають середнім розсіюванням множини O.

Визначимо функціонал якості розбиття. Нехай $K = \{K_1, K_2, ..., K_m\}$ – деяке фіксоване розбиття спостережень $X_1, X_2, ..., X_N$ на задане число m кластерів $K_1, K_2, ..., K_m$. У якості функціоналів часто обираються наступні характеристики.

Сума загальних розсіювань:

$$G_1(K) = \sum_{i=1}^{m} S_d(K_i).$$

Сума середніх розсіювань:

$$G_2(K) = \sum_{i=1}^{m} \bar{S}_d(K_i).$$

Алгоритми роздільної кластеризації.

Задача кластерного аналізу носить комбінаторний характер. Прямий спосіб розв'язання такої задачі полягає в повному переборові всіх можливих варіантів розбиття на кластери й виборі розбивки, що забезпечує екстремальне значення функціоналу. Такий спосіб розв'язання називають кластеризацією повним перебором. Аналогом кластерної проблеми комбінаторної математики є задача розбиття множини з n об'єктів на m підмножин. Число таких варіантів розбиття позначається через $S(n,m)$ і називається числом Стірлінга другого роду. Кластеризація повним перебором можлива в тих випадках, коли число об'єктів і кластерів не дуже велике.

Найбільш широке застосування одержали алгоритми послідовної кластеризації. У цих алгоритмах проводиться послідовний вибір чергових спостережень і для кожного з них вирішується питання, до якого з m кластерів його віднести. Розглянемо одну з послідовних кластер-процедур – метод k-середніх.

Метод k-середніх.

Особливість цього алгоритму в тому, що він виконується в два етапи: на першому етапі в просторі E_n виконується пошук точок – центрів кластерів, а потім спостереження розподіляються по тих кластерах, до центрів яких вони тяжіють. Алгоритм працює в припущенні, що число m кластерів відомо. Перший етап починається з відбору m об'єктів, які приймаються як нульове наближення центрів кластеризації. Це можуть бути перші m зі списку об'єктів або випадково відібрані.

Кожному центру приписується одинична вага. На першому кроці алгоритму отримується перша з точок, що залишилися (позначимо її X_{m+1}) і з'ясовується, до якого з центрів вона виявилася ближче всього в сенсі обраної d метрики. Цей центр заміняється новим, обумовленим як зважена комбінація

старого центру й нової точки. Вага центру збільшується на одиницю і т.д., доки не будуть пройдені всі точки.

Ієрархічний кластерний аналіз.

Поряд з роздільним кластерним аналізом широко застосовується ієрархічний кластерний аналіз, ціль якого полягає в одержанні всієї ієрархії об'єктів, а не окремого розбиття. Якщо використати неорієнтований граф, то структура такої розбивки являє собою дерево. Сам процес класифікації представляє собою побудову ієрархічного дерева досліджуваної сукупності об'єктів. Графічне зображення неорієнтованого графа ієрархії на площині називають дендрограмою.

Кластерний аналіз в R.

Основні процедури кластерного аналізу реалізовані в **R** у пакеті **stats**, **cluster**. Є також і інші пакети, які необхідно завантажити і встановити окремо. Наприклад, **pvclust**.

Розглянемо як приклад множину O, що складається з восьми об'єктів ($N=8$), що мають єдину ознаку ($n=1$); результати спостережень зведено в матрицю (для даного прикладу – вектор) $X=\{3,4,7,4,3,3,4,4\}^T$.

Приклад 9.1. Кластеризація в **R**.

```
data=data.frame(c(3,4,7,4,3,3,4,4))
#визначення матриці відстаней
matr_D=dist(data, method = "euclidean")
#побудова дендрограми за ієрархічним алгоритмом
tree=hclust(matr_D)
plot(tree)
#метод k-середніх
cl=kmeans(matr_D,3,10)
#значення до якого кластеру відноситься
cl$cluster
```

Завдання до лабораторної роботи

1. За допомогою вбудованої довідки вивчити параметри функцій **hclust()** і **kmeans()**.
2. Виконати кластеризацію країн за рівнем ВВП використовуючи дані з файлу ny.gdp.mktp.kd.zg_Indicator_en_csv_v2.csv.
3. Виконати кластеризацію країн за рівнем безробіття використовуючи дані з файлу sl.uem.totl.zs_Indicator_en_csv_v2.csv.

Контрольні запитання

1. Визначте мету кластерного аналізу.
2. Охарактеризуйте постановку задачі кластерного аналізу.
3. Розкрийте алгоритми кластерного аналізу.
4. Охарактеризуйте засоби **R** для проведення кластерного аналізу.

Лабораторна робота №10. Нейронні мережі в R

Мета: ознайомитися з основними засобами нейронних мереж в **R**.

Теоретичні відомості

Нейронні мережі – це моделі біологічних нейронних мереж мозку, в яких нейрони моделюються відносно простими елементами (штучними нейронами).

Нейронна мережа може бути представлена як направлений граф зі зваженими дугами, в якому штучні нейрони є вершинами, а синаптичні зв'язки – дугами.

Серед задач Data Mining, що розв'язуються за допомогою нейронних мереж, можна виділити такі:

- класифікація (навчання з вчителем). Приклади задач класифікації: розпізнавання тексту, мови. На відміну від задач кластеризації, вважається, що класи, за якими потрібно розподілити досліджувані об'єкти, є відомими;
- прогнозування. Наприклад: знайти найкраще наближення функції, заданої скінченним набором початкових значень (навчальні приклади);
- кластеризація (навчання без вчителя). Прикладом може бути задача сегментації зображень з метою визначення його границь.

Елементи та типи нейронних мереж.

Штучний нейрон – елемент нейронних мереж, який моделює деякі функції біологічного нейрону. Головна функція штучного нейрону – формувати вихідний сигнал в залежності від вхідного сигналу, який поступає на вхід нейрону.

Найпоширеніша схема нейрону показана на рисунку 10.1. Нейрон має n входів $x_1, x_2, x_3 ..., x_n$, на які можуть поступати сигнали або з зовнішньої середи, або з інших нейронів. Нейрон характеризується поточним станом та має групу **синапсів** – односпрямованих вхідних зв'язків, поєднаних з виходами інших нейронів. **Аксон** – вихідний зв'язок даного нейрону. Через аксон даний нейрон пов'язаний з іншими нейронами.

Кожний синапс характеризується величиною синаптичного зв'язку – вага w_i. Вага визначає як відповідний вхід впливає на стан нейрону.

На виході з ядра нейрону виникає сигнал S:

$$S = \sum_{i=1}^{n} x_i w_i \text{ або } S = \sum_{i=1}^{n} x_i w_i + b,$$

де b – деяка константа – зміщення.

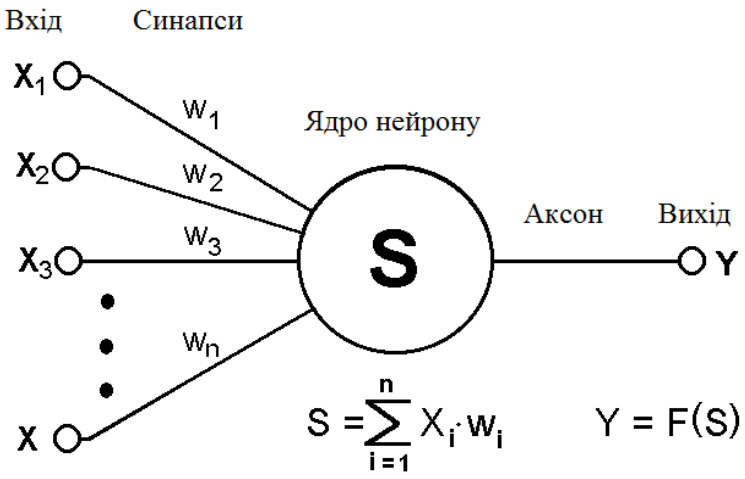

Рисунок 10.1 – Схема нейрону

Аксон пов’язує ядро з виходом та виконує додаткове перетворення, як правило, нелінійне:

$$y = F(S).$$

Функція $F(S)$ називається функцією активації нейрону.

Нейронні мережі можуть бути синхронними та асинхронними. В **синхронних** мережах в кожний момент часу свій стан може змінювати лише один нейрон. В **асинхронних** – стан змінюється у цілої групи нейронів, як провило, у цілого шару.

Можна виділити дві основні архітектури: шаруваті та повнозв’язні мережі.

Шарувата нейронна мережа – це сукупність нейронів, що складають шари. В кожному шарі нейрони між собою ніяк не пов’язані, але нейрони з попереднього та наступного шарів пов’язані між собою. Ключовим в шаруватих нейронних мережах є поняття шару.

Шар – один чи декілька нейронів, на входи яких подається один загальний сигнал.

В шаруватих мережах нейрони *i*-го шару отримують вхідні сигнали, перетворюють їх та передають через свої виходи нейронам (*i*+1)-го шару. Число нейронів в кожному шарі не є пов’язаним з кількістю нейронів в інших шарах.

В рамках одного шару дані обробляються паралельно, а в рамках всієї мережі послідовно – від шару до шару. При цьому, перший шар називається вхідним, наступні шари – внутрішніми або прихованими, останній шар – вихідним.

Основні моделі нейронних мереж.

Розглянемо далі найбільш прості моделі нейронних мереж: одношаровий та багатошаровий **персептрон.**

Одношаровий персептрон (персептрон Розенблатта) – одношарова нейронна мережа без зворотних зв’язків .

На рисунку 10.2 представлена нейронна мережа – одношаровий тринейронний персептрон.

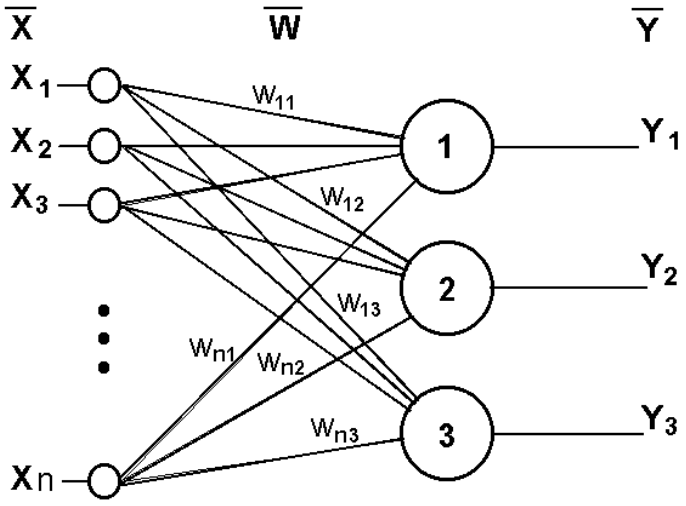

Рисунок 10.2 – Одношаровий тринейронний персептрон.

Багатошаровий персептрон – нейронна мережа прямого поширення сигналу (рис. 10.3)

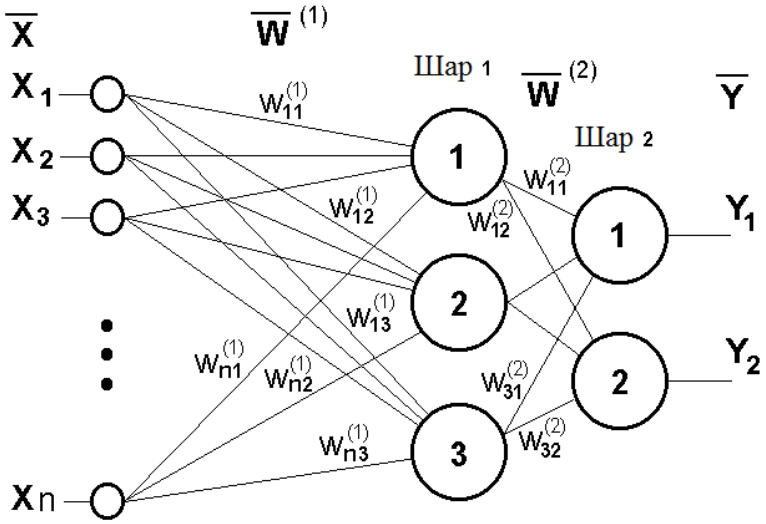

Рисунок 10.3 – Багатошаровий персептрон.

Навчання нейронних мереж.

Процес навчання нейронної мережі полягає в налаштуванні її внутрішніх параметрів під конкретну задачу.

Алгоритм навчання нейронної мережі є ітеративним, його шаги називаються **епохами**, або циклами.

Епоха – одна ітерація в процесі навчання, яка включає задання всіх прикладів з навчальної множини та перевірку якості навчання на контрольній множині.

Процес навчання відбувається на навчальній вибірці.

Навчальна вибірка включає вхідні значення та відповідні їм вихідні значення. Під час навчання нейронна мережа обирає внутрішні параметри для того, щоб вхідні параметри були пов'язані з вихідними. Еталонний набір вихідних параметрів часто називають «вчителем».

Різниця між отриманим в мережі виходом та виходом з навчальної вибірки називається **помилкою навчання**. З отриманих різниць можна сформувати **функцію помилок.**

Функція помилок – це цільова функція, яка мінімізується під час керованого навчання.

Одним з базових алгоритмів навчання нейронних мереж з архітектурою багатошарових персептронів є **алгоритм зворотного розповсюдження похибок.** Цей алгоритм заснований на обчисленні градієнта функції помилок. В процесі навчання ваги нейронів кожного шару корегуються з урахуванням сигналів, які поступають з попереднього шару, та нев'язки кожного шару, яка обчислюється у зворотному напряму від останнього шару до першого.

Нейронні мережі в R.

Існує досить велика кількість пакетів – реалізацій нейронних мереж у **R.** Серед них можна виділити такі: **neuralnet**, **nnet**, **AMORE**, **kohonen** і інші.

Пакет **neuralnet** розроблено для навчання багатошарових персептронів з метою проведення регресійного аналізу. Основні функції пакету:

- **neuralnet()** – навчання нейронної мережі;
- **plot()** – графічне зображення нейронної мережі;
- **compute()** – обчислення у мережі значень вихідного параметру.

Приклад 10.1. Побудуємо нейронну мережу, яка моделює дані з датафрейму **infert**.

```
library(neuralnet)
library(datasets)
head(infert)
#навчання нейронної мережі
nn=neuralnet(case~age+parity+induced+spontaneous,data=inf
ert,hidden=2,
err.fct="ce",linear.output=FALSE)
nn
#результуючі параметри нейромережі
nn$result.matrix
#зображення нейронної мережі
plot(nn)
#обчислення нових значень, використовуючи побудовану
нейромережу
new.output <- compute(nn,
covariate=matrix(c(22,1,0,0,22,1,1,0,22,1,0,1,22,1,1,1),
byrow=TRUE, ncol=4))
new.output$net.result
```

Завдання до лабораторної роботи

1. За допомогою команди **install.packages('neuralnet')** встановити пакет **neuralnet**
2. За допомогою вбудованої довідки вивчити параметри функцій **neuralnet()**.
3. Використати дані свого варіанта з лабораторної роботи №7 для побудови регресійної моделі за допомогою нейронної мережі. Для цього, перші 15 значень використати для навчання нейронної мережі. Останні 5 значень спрогнозувати за допомогою нейронної мережі та порівняти з табличними значеннями.
4. Виконати завдання пункту 3 для даних файлу forex.dat.

Контрольні запитання

1. Дайте визначення нейронної мережі.
2. Поясніть призначення функції активізації.
3. Охарактеризуйте основні підходи до навчання нейронної мережі.

Зміст індивідуального завдання

Теоретичні відомості

Банк даних «Показники світового розвитку (ПСР) Всесвітнього банку» охоплює великий набір економічних, соціальних і екологічних показників, заснованих на даних «Всесвітнього банку». Банк даних містить у собі більше 900 показників по 210 країнам за період з 1960 року.

Доступ до даних можна отримати за адресою http://data.worldbank.org/

Для подальшого аналізу виберемо наступні показники. Дані у форматі .xls і .csv перебувають у папці Data_Idz.

Agriculture, value added (% of GDP)
http://data.worldbank.org/indicator/NV.AGR.TOTL.ZS

Central government debt, total (% of GDP)
http://data.worldbank.org/indicator/GC.DOD.TOTL.GD.ZS

External debt stocks, total (DOD, current US$)
http://data.worldbank.org/indicator/DT.DOD.DECT.CD

Exports of goods and services (% of GDP)
http://data.worldbank.org/indicator/NE.EXP.GNFS.ZS

Imports of goods and services (% of GDP)
http://data.worldbank.org/indicator/NE.IMP.GNFS.ZS

Inflation, consumer prices (annual %)
http://data.worldbank.org/indicator/FP.CPI.TOTL.ZG

Population ages 0-14 (% of total)
http://data.worldbank.org/indicator/SP.POP.0014.TO.ZS

Unemployment, total (% of total labor force)
http://data.worldbank.org/indicator/SL.UEM.TOTL.ZS

Alternative and nuclear energy (% of total energy use)
http://data.worldbank.org/indicator/EG.USE.COMM.CL.ZS

Energy production (kt of oil equivalent)
http://data.worldbank.org/indicator/EG.EGY.PROD.KT.OE

Energy use (kt of oil equivalent)
http://data.worldbank.org/indicator/EG.USE.COMM.KT.OE

Access to electricity (% of population)
http://data.worldbank.org/indicator/EG.ELC.ACCS.ZS

CO2 emissions (kt)
http://data.worldbank.org/indicator/EN.ATM.CO2E.KT

Bank capital to assets ratio (%)
http://data.worldbank.org/indicator/FB.BNK.CAPA.ZS

Incidence of tuberculosis (per 100,000 people)
http://data.worldbank.org/indicator/SH.TBS.INCD

Teenage mothers (% of womenages15-19 who have had children or are currently pregnant)
http://data.worldbank.org/indicator/SP.MTR.1519.ZS

High-technologyexports (currentUS$)
http://data.worldbank.org/indicator/TX.VAL.TECH.CD

Urban population (% of total)
http://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS

Childreninemployment, total (%ofchildrenages7-14)
http://data.worldbank.org/indicator/SL.TLF.0714.ZS

Приклад можливого аналізу даних.

Розглянемо два параметри:

- виробництво енергії

Energy production (kt of oil equivalent)
http://data.worldbank.org/indicator/EG.EGY.PROD.KT.OE

- викиди вуглекислого газу

CO_2 emissions (kt)
http://data.worldbank.org/indicator/EN.ATM.CO2E.KT

Таблиці даних перебувають у файлах
eg.egy.prod.kt.oe_Indicator_en_csv_v2.csv
та
en.atm.co2e.kt_Indicator_en_csv_v2.csv відповідно.

План кореляційно-регресійного аналізу двох параметрів X і Y.

1. Визначити, чи існує істотний лінійний зв’язок між параметрами X і Y?
2. Якщо лінійний зв’язок існує, то, швидше за все, можна побудувати лінійну регресію на цих параметрах.
3. Якщо сильний лінійний зв’язок не спостерігається, то можна спробувати визначити вид нелінійного зв’язку й побудувати нелінійну регресію.

Приклад 1. Скрипт для визначення середнього виробництва енергії по всіх країнах за обраний період і середній рівень викидів CO_2 записано нижче.

```
factor1=read.csv("D:/Data_Idz2/eg.egy.prod.kt.oe_Indicato
r_en_csv_v2.csv",header = TRUE, sep = ",", quote = "\"",
dec = ".", fill = TRUE, comment.char = "")
x=factor1$"X2012"
world_factor1_0=mean(x, na.rm = TRUE)
x=factor1$"X2011"
world_factor1_1=mean(x, na.rm = TRUE)
x=factor1$"X2010"
world_factor1_2=mean(x, na.rm = TRUE)
x=factor1$"X2009"
world_factor1_3=mean(x, na.rm = TRUE)
x=factor1$"X2008"
world_factor1_4=mean(x, na.rm = TRUE)
x=factor1$"X2007"
```

```
world_factor1_5=mean(x, na.rm = TRUE)
x=factor1$"X2006"
world_factor1_6=mean(x, na.rm = TRUE)
x=factor1$"X2005"
world_factor1_7=mean(x, na.rm = TRUE)
x=factor1$"X2004"
world_factor1_8=mean(x, na.rm = TRUE)
x=factor1$"X2003"
world_factor1_9=mean(x, na.rm = TRUE)
x=factor1$"X2002"
world_factor1_10=mean(x, na.rm = TRUE)
x=factor1$"X2001"
world_factor1_11=mean(x, na.rm = TRUE)
x=factor1$"X2000"
world_factor1_12=mean(x, na.rm = TRUE)
x=factor1$"X1999"
world_factor1_13=mean(x, na.rm = TRUE)
x=factor1$"X1998"
world_factor1_14=mean(x, na.rm = TRUE)
x=factor1$"X1997"
world_factor1_15=mean(x, na.rm = TRUE)
x=factor1$"X1996"
world_factor1_16=mean(x, na.rm = TRUE)
x=factor1$"X1995"
world_factor1_17=mean(x, na.rm = TRUE)
x=factor1$"X1994"
world_factor1_18=mean(x, na.rm = TRUE)
x=factor1$"X1993"
world_factor1_19=mean(x, na.rm = TRUE)
x=factor1$"X1992"
world_factor1_20=mean(x, na.rm = TRUE)
x=factor1$"X1991"
world_factor1_21=mean(x, na.rm = TRUE)
x=factor1$"X1990"
world_factor1_22=mean(x, na.rm = TRUE)

factor1_x=c(world_factor1_22,world_factor1_21,world_facto
r1_20,world_factor1_19,world_factor1_18,world_factor1_17,
world_factor1_16,world_factor1_15,world_factor1_14,world_
factor1_13,world_factor1_12,world_factor1_11,world_factor
1_10,world_factor1_9,world_factor1_8,world_factor1_7,worl
d_factor1_6,world_factor1_5,world_factor1_4,world_factor1
_3,world_factor1_2)
```

```
factor2=read.csv("D:/Data_Idz2/en.atm.co2e.kt_Indicator_e
n_csv_v2.csv",header = TRUE, sep = ",", quote = "\"", dec
= ".", fill = TRUE, comment.char = "")
x=factor2$"X2012"
world_factor2_0=mean(x, na.rm = TRUE)
x=factor2$"X2011"
world_factor2_1=mean(x, na.rm = TRUE)
x=factor2$"X2010"
world_factor2_2=mean(x, na.rm = TRUE)
x=factor2$"X2009"
world_factor2_3=mean(x, na.rm = TRUE)
x=factor2$"X2008"
world_factor2_4=mean(x, na.rm = TRUE)
x=factor2$"X2007"
world_factor2_5=mean(x, na.rm = TRUE)
x=factor2$"X2006"
world_factor2_6=mean(x, na.rm = TRUE)
x=factor2$"X2005"
world_factor2_7=mean(x, na.rm = TRUE)
x=factor2$"X2004"
world_factor2_8=mean(x, na.rm = TRUE)
x=factor2$"X2003"
world_factor2_9=mean(x, na.rm = TRUE)
x=factor2$"X2002"
world_factor2_10=mean(x, na.rm = TRUE)
x=factor2$"X2001"
world_factor2_11=mean(x, na.rm = TRUE)
x=factor2$"X2000"
world_factor2_12=mean(x, na.rm = TRUE)
x=factor2$"X1999"
world_factor2_13=mean(x, na.rm = TRUE)
x=factor2$"X1998"
world_factor2_14=mean(x, na.rm = TRUE)
x=factor2$"X1997"
world_factor2_15=mean(x, na.rm = TRUE)
x=factor2$"X1996"
world_factor2_16=mean(x, na.rm = TRUE)
x=factor2$"X1995"
world_factor2_17=mean(x, na.rm = TRUE)
x=factor2$"X1994"
world_factor2_18=mean(x, na.rm = TRUE)
x=factor2$"X1993"
world_factor2_19=mean(x, na.rm = TRUE)
x=factor2$"X1992"
world_factor2_20=mean(x, na.rm = TRUE)
```

```
x=factor2$"X1991"
world_factor2_21=mean(x, na.rm = TRUE)
x=factor2$"X1990"
world_factor2_22=mean(x, na.rm = TRUE)

factor2_y=c(world_factor2_22,world_factor2_21,world_facto
r2_20,world_factor2_19,world_factor2_18,world_factor2_17,
world_factor2_16,world_factor2_15,world_factor2_14,world_
factor2_13,world_factor2_12,world_factor2_11,world_factor
2_10,world_factor2_9,world_factor2_8,world_factor2_7,worl
d_factor2_6,world_factor2_5,world_factor2_4,world_factor2
_3,world_factor2_2)

plot(factor1_x)
plot(factor2_y)

plot(factor1_x,factor2_y)
factor1=as.data.frame(factor1_x)
factor2=as.data.frame(factor2_y)
cor(factor1,factor2)
factors=data.frame(factor1,factor2)
lregr=lm(factor2_y~factor1_x,factors)
summary(lregr)
write.table(factors, "D:/factors.txt", sep="\t", dec =
",")
```

Таблицю з файлу factors.txt можна використати для перевірки в MSExcel (файл idz.xls).

Висновок: між параметрами середній рівень виробництва енергії й середній рівень викидів вуглекислого газу за період з 1990 по 2010 р. можна спостерігати сильний лінійний зв'язок (коефіцієнт кореляції Пірсона 0,99).

Побудовано лінійну регресійну модель, що зв'язує розглянуті параметри формулою:

$$CO2 = 1.792 \cdot eg\_prod - 9253,$$

де eg_prod – середній рівень виробництва енергії, $CO2$ – середній рівень викидів вуглекислого газу.

Коефіцієнт детермінації регресійної моделі $R^2 = 0{,}98$, що свідчить про тісний лінійний зв'язок середнього рівня викидів вуглекислоти, розрахованого за наведеною формулою, і табличних значень. Отже, якість регресійної моделі задовільна. Однак, ґрунтуючись на результатах проведеного аналізу, не можна стверджувати остаточно, що рівень виробництва енергії безпосередньо впливає на рівень викидів вуглекислого газу. Швидше за все, існують інші фактори, які побічно впливають на обидва параметри.

Аналогічним чином проводиться нелінійний регресійний аналіз при відсутності тісної лінійної кореляції.

Змістовна частина завдання

Провести кореляційний, регресійний та кластерний аналіз мінімуму 10 із запропонованих 19-ти показників світового розвитку. (Можна самостійно вибрати параметри для аналізу на сайті http://data.worldbank.org/)

Завдання для самостійної роботи

Аналіз даних за допомогою карт Кохонена

Навчання без «вчителя» – це тип навчання нейронної мережі, при якому навчання відбувається без еталонного вихідного набору даних. Для навчання використовується тільки деякий вхідний набір даних. Нейронна мережа виявляє закономірності у вхідних даних та навчається самостійно.

Один з алгоритмів навчання без «вчителя» був розроблений фінським вченим Тойво Кохоненом у 1982 р. Нейронні мережі, що використовують цей алгоритм навчання, називаються картами Кохонена або самоорганізаційними картами (Self Organizing Map, MOP).

Карта Кохонена, на відміну від багатошарової нейронної мережі, має тільки два шари: вхідний та вихідний (рис. 1).

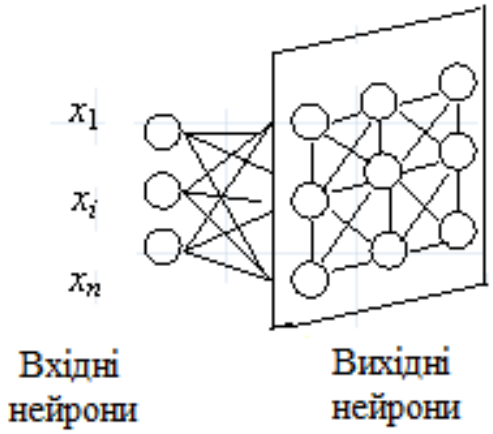

Рисунок 1 – Карта Кохонена.

Одним з найпоширеніших застосувань карт Кохонена є розв’язання задачі класифікації без вчителя, тобто кластеризації.

Карти Кохонена реалізовано в **R** у пакеті **kohonen**.

Під час самостійної роботи рекомендується побудувати карту Кохонена для розв’язання задачі кластеризації. Дані для кластеризації беруться з лабораторної роботи №9. Результати аналізу слід порівняти з результатами, отриманими в лабораторній роботі №9.

Глосарій

Варіанти – це різні значення ознаки, що спостерігаються у елементів сукупності.

Варіаційний ряд – ранжований у порядку зростання або спадання ряд варіант з частотами, що їм відповідають.

Величина випадкова – це величина, яка при кожному здійсненні відповідного ймовірнісного експерименту набуває того чи іншого числового значення, наперед невідомого та залежного від випадкових причин, які наперед не можуть бути враховані.

Величина випадкова дискретна – випадкова величина, множина значень якої скінченна або зчисленна.

Величина випадкова неперервна – випадкова величина, функція розподілу якої скрізь неперервна, а похідна функції неперервна в усіх точках, за винятком зліченного числа точок на будь-якому скінченному інтервалі.

Гіпотези – події, які утворюють повну групу подій, причому разом з однією з цих подій може відбутися деяка інша подія.

Гістограма розподілу частот – графік інтервального варіаційного ряду, коли на осі абсцис відкладають межі класових інтервалів, а осі ординат – частоти інтервалів.

Дисперсія – це математичне сподівання квадратів відхилень випадкової величини від її математичного сподівання.

Добування даних (англ. Data Mining) – збиральна назва, що використовується для позначення сукупності методів виявлення в даних раніше невідомих, нетривіальних та практично корисних знань, необхідних для прийняття рішень в різних сферах людської діяльності.

Добуток подій – це така подія, яка полягає в тому, що всі події відбудуться.

Закон розподілу дискретної випадкової величини – перелік її можливих значень і відповідних їм ймовірностей.

Залежність кореляційна між двома змінними величинами – функціональна залежність, яка існує між значеннями однієї з них та груповими середніми іншої величини.

Ймовірність – число, яке є мірою об’єктивної можливості здійснення подій.

Ймовірність довірча – ймовірність того, що абсолютна величина відхилення вибіркової середньої від генеральної середньої не перевищить заданого числа.

Кластерний аналіз – статистична процедура, яка виконує збір даних про вибірку та упорядкування об’єктів у відносно однорідні групи.

Коваріація – міра лінійної залежності двох випадкових величин.

Критерій узгодженості – оцінка ймовірності того, що одержана вибірка не суперечить зробленому припущенню про вид закону розподілу розглянутої

випадкової величини.

Медіана – число, яке ділить варіаційний ряд на дві частини, що мають однакову кількість елементів.

Мода – значення варіанти варіаційного ряду, в якої найбільша відносна частота.

Момент випадкової величини – числова характеристика розподілу ймовірностей випадкової величини.

Момент початковий порядку k випадкової величини – математичне сподівання k-го степеня випадкової величини.

Момент центральний порядку k випадкової величини – математичне сподівання k-го степеня відхилення величини від її математичного сподівання.

Нейронна мережа – це модель біологічних нейронних мереж мозку, в яких нейрони моделюються відносно простими елементами (штучними нейронами).

Об'єм сукупності – загальна сума всіх частот варіант сукупності.

Означення ймовірності класичне – відношення числа сприятливих випадків настання події до загально числа можливих випадків.

Події залежні – такі події, коли настання однієї залежить від настання іншої.

Події незалежні – це такі події, коли ймовірність однієї з них виключає появу іншої.

Події несумісні – це такі події, коли поява однієї з них виключає появу іншої.

Події сумісні – події, які можуть відбутися одночасно.

Подія випадкова – подія, яка може відбутися або на відбутися в результаті досліду при здійсненні певного комплексу умов.

Подія достовірна – це подія, що обов'язково відбудеться.

Подія неможлива – це подія, протилежна достовірній.

Полігон розподілу частот – геометрична фігура у вигляді багатокутника, що одержується з'єднанням точок, які побудовано у прямокутній системі координат відкладанням по осі абсцис – значень варіант, по осі ординат – частот.

Регресійний аналіз – метод моделювання вимірюваних даних та дослідження їх властивостей. Регресійна модель є функцією незалежної змінної та параметрів з додаванням випадкової змінної.

Розмах варіації – різниця між найбільшою та найменшою варіантами.

Середнє квадратичне відхилення – арифметичне значення квадратного кореня з дисперсії.

Статистики – числові показники, що характеризують вибірку.

Сукупність вибіркова (вибірка) – частина об'єктів, яка потрапила на дослідження.

Сукупність генеральна – загальна група об'єктів, з якими пов'язане статистичне дослідження.

Сума випадкових подій – подія, яка полягає у здійсненні хоча б однієї з

даних подій.

Функція розподілу випадкової величини – функція, яка виражає для кожного значення випадкової величини ймовірність того, що вона набуде яке-небудь значення менше заданого.

Функція розподілу емпірична – функція, яка виражає для кожного значення змінної x частку тих елементів сукупності даного об'єму, в яких ознака приймає значення менше за x.

Штучний нейрон – елемент нейронних мереж, який моделює деякі функції біологічного нейрону. Головна функція штучного нейрону – формувати вихідний сигнал в залежності від вхідного сигналу, який поступає на вхід нейрону.

Щільність розподілу – похідна від функції розподілу випадкової величини.

Рекомендована література

Основна:

1. Афанасьева Т.В. Моделирование нечетких тенденций временных рядов / Т.В. Афанасьева. – Ульяновск : УлГТУ, 2013. – 215 с.
2. Барсегян А.А. Анализ данных и процессов: учебное пособие / А.А. Барсегян, М.С. Куприянов, И.И. Холод, М.Д. Тесс, С.И. Елизаров. – СПб : БХВ-Петербург, 2009. – 512 с.
3. Вьюгин В.В. Математические основы теории машинного обучения и прогнозирования / В.В. Вьюгин. – М., 2013. – 387 с.
4. Гнатюк В. Вступ до R на прикладах / Віктор Гнатюк. – Харків : ХНЕУ, 2010. – 101 с.
5. Дебок Г. Анализ финансовых данных с помощью самоорганизующихся карт / Г. Дебок, Т. Кохонен. – М. : «Альпина», 2001. – 317 с.
6. Дюк В. Data mining: учебный курс / В. Дюк, А. Самойленко. – СПБ : Питер, 2001. – 368 с.
7. Зарядов И.С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика / И.С. Зарядов. – М. : Изд-во РУДН, 2010. – 207 с.
8. Зарядов И.С. Введение в статистический пакет R: теория вероятностей и математическая статистика / И.С. Зарядов. – М. : Изд-во РУДН, 2010. – 207 с.
9. Осипов О.Ю. Теорія ймовірностей та математична статистика : навчальний посібник для студентів фізичного факультету / О.Ю. Осипов. – Запоріжжя : ЗНУ, 2011. – 190 с.
10. Снитнюк В.Е. Прогнозирование. Модели, методы, алгоритмы : учебное пособие / В.Е. Снитнюк. – К : «Маклаут», 2008. – 364 с.
11. Ситник В.Ф. Інтелектуальний аналіз даних (дейтамайнінг) : навчальний посібник / В. Ф. Ситник, М.Т. Краснюк. – К. : КНЕУ, 2007. – 376 с.
12. Ярушкина Н.Г. Интеллектуальный анализ данных временных рядов : учебное пособие / Н.Г. Ярушкина, Т.В. Афанасьева, И.Г. Перфильева. – Ульяновск : УлГТУ, 2010. – 320 с.

Додаткова:

1. Корнелл П. Анализ данных в Excel / Корнелл П. – М. : Эксмо, 2007. – 224 с.
2. Нейронные сети. Statistica Neural Networks: Методология и технология современного анализа данных / Под редакцией В.П. Боровикова. – 2-е изд., перераб. и доп. – М. : Горячая линия – Телеком, 2008. – 392 с.
3. Низаметдинов Ш.У. Анализ данных: учебное пособие / Ш.У. Низаметдинов. – М. : МИФИ, 2006. – 248 с.

4. Шитиков В.К. Рандомизация и бутстреп: статистический анализ в биологии и экологии с использованием R / В.К. Шитиков, Г.С. Розенберг. – Тольятти : Кассандра, 2013. – 305 с.
5. Adelchi Azzalini. Data analysis and data mining / Adelchi Azzalini, Bruno Scarpa. – Oxford University Press, 2012. – 278 p.
6. Siegmund Brandt. Data Analysis / Siegmund Brandt. – Springer, 2014. – 523 p.
7. Yanchang Zhao. Data mining applications with R / Yanchang Zhao, Yonghua Cen. – Elsevier, 2014. – 471 p.
8. Stephane Tuffery. Data mining and statistics for decision making / Stephane Tuffery. – Wiley, 2011. – 689 p.
9. Gisele L. Pappa. Automating the Design of Data Mining Algorithms / Gisele L. Pappa, Alex A. Freitas. – Springer, 2010. – 187 p.
10. John K. Kruschke. Doing Bayesian Data Analysis / John K. Kruschke. – Elsevier, 2015. – 759 p.

Інформаційні ресурси:

1. http://statsoft.ru/
2. http://cran.r-project.org/
3. http://www.inside-r.org/
4. https://www.coursera.org
5. http://www.rstudio.com/

Додаток А. Базові відомості з теорії ймовірностей

Випадковою подією називається подія, що може відбутися чи не відбутися при здійсненні сукупності умов.

Ймовірністю події A називають відношення числа сприятливих випадків настання події A до числа можливих випадків

$$P(A)=\frac{m}{n},$$

де m – число сприятливих події A випробувань, n – загальне число можливих випадків.

Для ймовірності випадкової події виконується наступне співвідношення

$$0 \le P(A) \le 1.$$

Випадкова подія A називається **неможливою**, якщо $P(\mathrm{A})=0$.

Випадкова подія A називається **достовірною**, якщо $P(\mathrm{A})=1$.

Випадковою величиною називається величина, що при кожному здійсненні відповідного ймовірнісного експерименту набуває того чи іншого числового значення, наперед невідомого та залежного від випадкових причин, які наперед не можуть бути враховані.

Випадкова величина називається **дискретною**, якщо множина її значень скінченна або зчисленна.

Законом розподілу дискретної випадкової величини називається перелік її можливих значень і відповідних їм ймовірностей. Може задаватися у вигляді таблиці, аналітично (у вигляді формули) та графічно.

Математичним сподіванням дискретної випадкової величини X називається сума добутку всіх її значень на відповідні їм ймовірності:

$$M(X)=x_1p_1+x_2p_2+...+x_np_n=\sum_{i=1}^{n}x_ip_i\ .$$

Математичне сподівання приблизно рівняється (тим точніше, чим більше число випробовувань) середньому арифметичному значень випадкової величини.

Дисперсією дискретної випадкової величини X називають математичне сподівання квадрату відхилень випадкової величини від її математичного сподівання:

$$D(X)=M[X-M(X)]^2=M(X^2)-[M(X)]^2.$$

Середнім квадратичним відхиленням дискретної випадкової величини X називається арифметичне значення квадратного кореня від дисперсії, тобто:

$$\sigma(X)=\sqrt{D(X)}\ .$$

Функцією розподілу випадкової величини називають функцію $F(x)$, яка визначає ймовірність того, що випадкова величина X в результаті випробовувань прийме значення менше за x:

$$F(x)=P(X<x).$$

Ймовірність того, що випадкова величина X прийме значення, що належить відрізку $[x_1, x_2]$, дорівнює прирощенню функції розподілу на цьому відрізку:

$$P(x_1 \le x_2) = F(x_2) - F(x_1).$$

Випадкова величина називається неперервною, якщо функція розподілу її скрізь безперервна, а похідна функції безперервна в усіх точках, за винятком зліченного числа точок на будь-якому скінченному інтервалі.

Можливо дати інше, менш формальне визначення неперервної випадкової величини. Неперервною називають випадкову величину, яка може приймати всі значення з деякого скінченного або нескінченного інтервалу. Число значень неперервної випадкової величини нескінченно.

Щільністю розподілу ймовірностей неперервної випадкової величини X називають функцію $f(x)$ – першу похідну від функції розподілу $F(x)$.

Ймовірність того, що неперервна випадкова величина X прийме значення з інтервалу (a,b) обчислюється за формулою:

$$P(a < X < b) = \int_a^b f(x)\,\mathrm{d}x.$$

Знаючи щільність розподілу, можна знайти функцію розподілу $F(x)$ за формулою

$$F(x) = \int_{-\infty}^{x} f(x)dx.$$

Математичним сподіванням неперервної випадкової величини X, значення якої належать відрізку $[a,b]$, називають визначений інтеграл:

$$M(\mathrm{X}) = \int_a^b xf(x)dx.$$

Якщо можливі значення належать всій осі Ox, тоді

$$M(\mathrm{X}) = \int_{-\infty}^{\infty} xf(x)dx.$$

Дисперсія неперервної випадкової величини X, значення якої належать відрізку $[a,b]$ визначається інтегралом:

$$D(X) = \int_a^b [X - M(X)]^2 f(x)dx,$$

або

$$D(X) = \int_{-\infty}^{\infty} [X - M(X)]^2 f(x)dx.$$

Середнє квадратичне відхилення неперервної випадкової величини визначається аналогічно випадку дискретної величини.

Додаток Б. Базові відомості з математичної статистики

Вибіркова сукупність або **вибірка** – сукупність випадково відібраних об'єктів.

Генеральною сукупністю називають сукупність об'єктів з яких проводиться вибірка.

Повторною називають вибірку, при якій відібраний об'єкт повертається в генеральну сукупність.

Безповторною називають вибірку, при якій відібраний об'єкт не повертається в генеральну сукупність.

Для того, щоб по даних вибірки можна було робити висновки про генеральну сукупність необхідно, щоб вибірка правильно відображала пропорції генеральної сукупності. Коротко ця умова називається **репрезентативністю вибірки**.

Нехай з генеральної сукупності вилучена вибірка, причому x_1 спостерігалося n_1 раз, x_2 – n_2 раз, x_k – n_k раз, $\sum n_i = n$ – об'єм вибірки. Значення, що спастерігаються називаються **варіантами**, а послідовність варіант, що записуються у зростаючому порядку – **варіаційний ряд**. Числа спостережень $n_1, n_2, ..., n_k$ називають **частотами**, а їх відношення до об'єму вибірки $\frac{n_i}{n} = W_i$ – **відносними частотами**.

Статистичним розподілом вибірки називають перелік варіант і відповідних їм частот або відносних частот. Якщо випадкова величина X неперервна або число n досить велике, то використовують **інтервальну таблицю частот** – статистичний розподіл записується в вигляді інтервалів і відповідних їм частот. В якості частоти, що відповідає інтервалу, приймають суму частот, які попадають в даний інтервал.

Емпіричною функцією розподілу (функцією розподілу вибірки) – називають функцію $F^*(x)$, яка визначає для кожного значення x відносну частоту події $X < x$.

На відміну від емпіричної функції розподілу вибірки, функцію розподілу генеральної сукупності називають **теоретичною функцією розподілу**. Різниця між емпіричною та теоретичною функцією полягає в тому, що теоретична функція $F(x)$ визначає ймовірність події $X < x$, а емпірична функція $F^*(x)$ визначає відносну частоту тієї ж події.

Полігоном частот називають ламану, відрізки якої поєднують точки (x_1, n_1), $(x_2, n_2), \ldots, (x_k, n_k)$. Для побудови полігону на осі абсцис відкладають варіанти x_i, а на осі ординат – відповідні їм частоти n_i.

Гістограмою частот називають ступінчасту фігуру, яка складається з прямокутників, основами яких служать часткові інтервали довжиною h, а

висоти дорівнюють відношенню $\frac{n_i}{h}$. Площа гістограми частот дорівнює об'єму вибірки.

Статистичною оцінкою невідомого параметру теоретичного розподілу називають деяку функцію від спостережуваних значень вибірки.

Для того, щоб статистичні оцінки давали задовільні наближення параметрів генеральної сукупності, вони повинні задовільнити наступним вимогам.

Незсуненою називають статистичну оцінку Θ^* математичне сподівання якої дорівнює параметру, що оцінюється Θ при довільному об'ємі вибірки.

$$M(\Theta^*) = \Theta.$$

Ефективною називаються статистичну оцінку, яка при заданому об'ємі вибірки має найменшу можливу дисперсію.

Конзистентною називають статистичну оцінку, яка при зростанні об'єму вибірки $n \to \infty$ прагне по ймовірності до параметру, що оцінюється.

Вибірковим середнім $\overline{x}_в$ називають середнє арифметичне значень признаку вибіркової сукупності:

$$\overline{x}_в = \frac{\left(x_1 + x_2 + ... + x_n\right)}{n}.$$

Вибіркове середнє – незміщена, ефективна та конзистентна статистична оцінка.

Вибірковою дисперсією $D_в$ називають середнє арифметичне квадратів відхилень спостережуваних значень признаку від їх середнього значення $\overline{x}_в$:

$$D_в = \frac{\sum_{i=1}^{k} n_i \left(x_i - \overline{x}_в\right)^2}{n}.$$

Вибіркове середнє квадратичне відхилення обчислюється наступним чином:

$$\sigma_в = \sqrt{D_в}.$$

Мода варіаційного ряду – значення варіанти, в якої найбільша частота.

Медіана – це число, яке ділить варіаційний ряд на дві частини, що мають однакову кількість елементів. Якщо об'єм сукупності n непарне, то медіаною є елемент варіаційного ряду із середнім номером. Якщо n – парне, тобто $n = 2k$, то $x_M = \frac{x_k + x_{k+1}}{2}$.

Квантиль – одна з числових характеристик випадкових величин. Квантилі відсікають в межах ряду певну частину його членів. Тобто, квантиль розподілення ймовірностей значень – це таке число x_p, що значення p-ї частини сукупності менше або рівне x_p. Наприклад, квантиль 0,25 (також називається 25-м процентилем або нижнім квартилем) змінної – це таке значення (x_p), що 25% (p) значень змінної попадають нижче даного значення.

Найпоширеніші види квантилей такі:

- **перший** або **нижній квартиль**, $p = 0,25$;
- **медіана**, або **другий квартиль**, $p = 0,5$;
- **третій**, або **верхній квартиль**, $p = 0,75$.

Точковою називають оцінки параметрів розподілення ймовірностей, які визначаються одним числом.

Інтервальною називають оцінку, яка визначається двома числами – кінцями інтервалу. Нехай за даними вибірки знайдено оцінку Θ^* деякого параметру Θ. **Точністю оцінки** будемо називати таке $\delta > 0$, що $|\Theta - \Theta^*| < \delta$. Чим менше δ, тим точніша оцінка.

Надійністю (**довірчою ймовірністю**) оцінки Θ за Θ^* називають ймовірність γ, з якою виконується нерівність $|\Theta - \Theta^*| < \delta$. Як правило, надійність оцінки задають наперед, причому в якості γ беруть число біля одиниці. Найбільш часто задають надійність, рівну 0,95; 0,99.

Навчально-методичне видання
(українською мовою)

Кудін Олексій Володимирович

ЕМПІРИЧНІ МЕТОДИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Методичні рекомендації до лабораторних робіт
для студентів освітнього ступеня «бакалавр»
напряму підготовки «Програмна інженерія»

Рецензент *С.М. Гребенюк*
Відповідальний за випуск *О.В. Кудін*
Коректор *В.О. Шевченко*

89