Ясінський А.М., Соловей Л.Я., Близнюк С.В.

МОВА ПРОГРАМУВАННЯ R У НАВЧАЛЬНІЙ ТА ДОСЛІДНИЦЬКІЙ ДІЯЛЬНОСТІ

Навчальний посібник

УДК 004.42:519.6

DOI: 10.5281/zenodo.18788660

Затверджено Вченою радою Приватного вищого навчального закладу «Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука» (протокол № 5 від 26 лютого 2026 р.)

Рецензенти:

Турбал Юрій Васильович, доктор технічних наук, професор, професор кафедри комп'ютерних наук та прикладної математики Національного університету водного господарства та природокористування

Войтович Ігор Станіславович, доктор педагогічних наук, професор, професор кафедри цифрових технологій та методики навчання інформатики, проректор з навчально-виховної роботи Рівненського державного гуманітарного університету

Огірко Ігор Васильович, доктор фізико-математичних наук професор, професор кафедри мультимедійних технологій Інституту поліграфії та медійних технологій Національного університету "Львівська політехніка"

Ясінський А.М., Соловей Л.Я., Близнюк С.В.

Мова програмування R у навчальній та дослідницькій діяльності: Навчальний посібник. – Рівне: ПВНЗ «Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука», 2026. – 275с.

ЗМІСТ

ПЕРЕДМОВА

Сучасний етап розвитку освіти характеризується активним упровадженням цифрових технологій, зростанням обсягів даних та необхідністю формування у здобувачів освіти навичок їх аналізу, інтерпретації та використання для прийняття обґрунтованих рішень. У цих умовах особливої актуальності набуває використання інструментів статистичного аналізу та моделювання, які дозволяють поєднати теоретичні знання з практичною діяльністю. Одним із таких універсальних інструментів є мова програмування R та інтегроване середовище розробки RStudio, що широко застосовуються у наукових дослідженнях, освітній аналітиці, економіці, соціальних та педагогічних науках.

RStudio та мова програмування R надає широкі можливості для обробки, аналізу та візуалізації даних, а також для побудови статистичних моделей різного рівня складності. Його використання в освітньому процесі сприяє формуванню інформаційно-аналітичної компетентності, розвитку критичного мислення та здатності працювати з реальними даними. Особливу цінність становить застосування інтегрованих завдань, які поєднують знання з інформатики, математики, статистики та профільних дисциплін, забезпечуючи міждисциплінарний підхід до навчання.

Навчальний посібник «Мова програмування R у навчальній та дослідницькій діяльності» адресований здобувачам вищої освіти бакалаврського, магістерського та докторського рівнів, науково-педагогічним працівникам, дослідникам і аналітикам, які застосовують методи статистичного аналізу даних, моделювання та візуалізації результатів у професійній діяльності. Видання буде корисним для студентів і викладачів спеціальностей галузей інформаційних технологій, економіки, соціальних і поведінкових наук, педагогіки, а також для всіх, хто використовує середовище R та інтегровану систему розробки RStudio з метою організації навчального процесу, проведення

емпіричних досліджень, обробки великих масивів даних і підготовки наукових публікацій.

Особливістю даного посібника є використання інтегрованих завдань, які моделюють реальні професійні та навчальні ситуації. Такі завдання спрямовані не лише на відпрацювання технічних навичок роботи в RStudio, а й на формування здатності самостійно ставити дослідницькі запитання, обирати адекватні методи аналізу, критично оцінювати отримані результати та представляти їх у науково обґрунтованій формі. Значна увага приділяється аналізу освітніх даних, що дозволяє використовувати посібник у процесі підготовки майбутніх педагогів та дослідників у сфері освіти.

Метою посібника є забезпечення методичної підтримки навчального процесу шляхом формування у здобувачів освіти практичних умінь і навичок застосування RStudio для аналізу та візуалізації даних, а також для розв’язування прикладних статистичних задач. Для досягнення поставленої мети в посібнику передбачено систему інтегрованих завдань різного рівня складності, що сприяють поетапному розвитку аналітичних компетентностей.

Посібник може бути використаний у навчальному процесі закладів вищої освіти, фахових коледжів, а також у системі підвищення кваліфікації педагогічних і науково-педагогічних працівників. Матеріали посібника також становлять практичний інтерес для всіх, хто прагне опанувати сучасні інструменти аналізу даних та застосовувати їх у науковій і професійній діяльності.

РОЗДІЛ 1. ОСНОВИ РОБОТИ В СЕРЕДОВИЩІ RSTUDIO

1.1.Робоче середовище мови програмування R

Мова програмування R є високорівневим інструментом для статистичних обчислень, аналізу даних і візуалізації, який широко використовується в науковій, освітній та прикладній діяльності. Робоче середовище мови R охоплює сукупність програмних засобів, інтерфейсів і ресурсів, що забезпечують створення, виконання та аналіз програмного коду, а також обробку результатів обчислень.

Основою робочого середовища є ядро R – інтерпретатор мови, який відповідає за виконання команд, управління об’єктами в оперативній пам’яті та реалізацію вбудованих функцій і бібліотек. Ядро R є кросплатформним і підтримується на основних операційних системах (Windows, macOS, Linux), що забезпечує універсальність і доступність середовища для користувачів з різним технічним оснащенням.

Взаємодія користувача з мовою R здійснюється через інтерфейс командного рядка (Console) або за допомогою інтегрованих середовищ розробки, найпоширенішим з яких є RStudio. Консоль R дозволяє вводити окремі команди та негайно отримувати результати їх виконання, що є зручним для експериментів, перевірки гіпотез і навчальних цілей. Водночас для виконання комплексних завдань доцільно використовувати R-скрипти, які містять послідовність команд і можуть бути повторно використані та модифіковані.

Важливою складовою робочого середовища R є об’єктно-орієнтована модель даних, відповідно до якої всі результати обчислень зберігаються у вигляді об’єктів (вектори, матриці, списки, дата-фрейми, фактори тощо). Такий підхід забезпечує зручність маніпулювання даними, їх аналізу та візуалізації. Користувач має можливість переглядати, змінювати та видаляти об’єкти під час роботи сеансу, а також зберігати їх для подальшого використання.

Робоче середовище мови R підтримує модульну архітектуру, що реалізується через систему пакетів (packages). Пакети містять набори функцій, даних і документації, які розширюють базові можливості мови. На сьогодні екосистема R налічує тисячі пакетів, доступних через репозиторії CRAN, Bioconductor та GitHub, що дозволяє ефективно розв'язувати завдання з аналізу даних, машинного навчання, статистичного моделювання та освітньої аналітики.

Окрему увагу в робочому середовищі R приділено засобам візуалізації та документування результатів. Мова R надає потужні інструменти для побудови графіків і діаграм, а також підтримує інтеграцію з системами підготовки звітів.

Мова R базується на мові програмування S, яка є основою статистичного програмування. Мова S виникла в Bell Laboratories у 1976 році. R була створена як вільний аналог мови S-PLUS – комерційної реалізації мови S. Таким чином, R є результатом розвитку та еволюції S і доступна як вільне програмне забезпечення.

RStudio – це інструмент, створений для спрощення роботи з мовою програмування R. Це інтегроване середовище розробки, яке надає зручний інтерфейс для написання коду, проведення аналізу даних, візуалізації результатів та управління проєктами. RStudio дозволяє працювати з окремими пакетами, що реалізують різноманітні технології обробки даних. Важливо зазначити, що RStudio є некомерційним і вільно розповсюджується на умовах ліцензії GNU General Public License. Основні ідеї та концепції мови S продовжують розвиватися в межах системи R.

RStudio надає інтерфейс для зручного керування пакетами R, що дозволяє легко встановлювати та оновлювати бібліотеки й додаткові функції. Крім того, у цьому середовищі можна створювати високоякісні графіки та візуалізації для ефективного аналізу даних. Усе це робить процес роботи з даними зручнішим і продуктивнішим.

У RStudio можна створювати та керувати проєктами, що сприяє кращій організації робочого процесу. Це включає управління даними, кодом і

результатами аналізу. Також користувач може налаштовувати середовище під власні потреби, змінюючи тему оформлення, гарячі клавіші та інші параметри.

Узагальнюючи, RStudio є потужним інструментом для аналітиків і дослідників, який використовується для розробки, тестування та реалізації аналітичних моделей і алгоритмів мовою R. Завдяки зручному текстовому редактору з підсвічуванням синтаксису та автодоповненням, RStudio значно спрощує написання коду.

1.1.1.Завантаження та інсталяція R

Код, написаний на R компілюється та запускається на різних платформах: UNIX, Windows, MacOS. Для завантаження актуальної версії R варто перейти на офіційний сайт https://cran.r-project.org/. На сайті необхідно обрати необхідну операційну систему і завантажити актуальну версію дистрибутива R (Рис.1.1.).

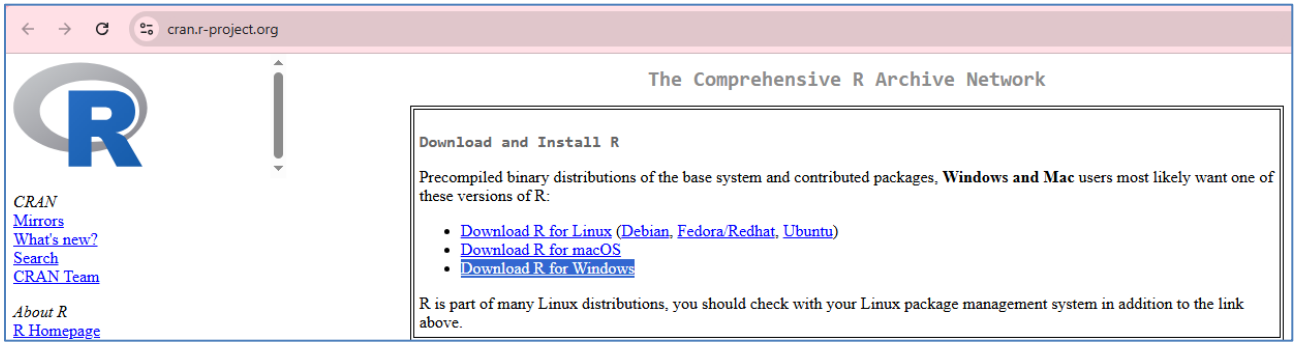


Рис.1.1. Вибір ОС та завантаження R

Запустити інсталяційний файл та виконати покрокові інструкції майстра встановлення: прийняти ліцензійну угоду; обрати шлях встановлення; залишити стандартні налаштування (рекомендовано). Після завершення інсталяції R готовий до використання.

Під час запуску програми відкривається робоче вікно RGui (GUI (Graphical User Interface) - набір візуальних компонентів для інтерактивної взаємодії користувача з програмним забезпеченням) (Рис.1.2.).

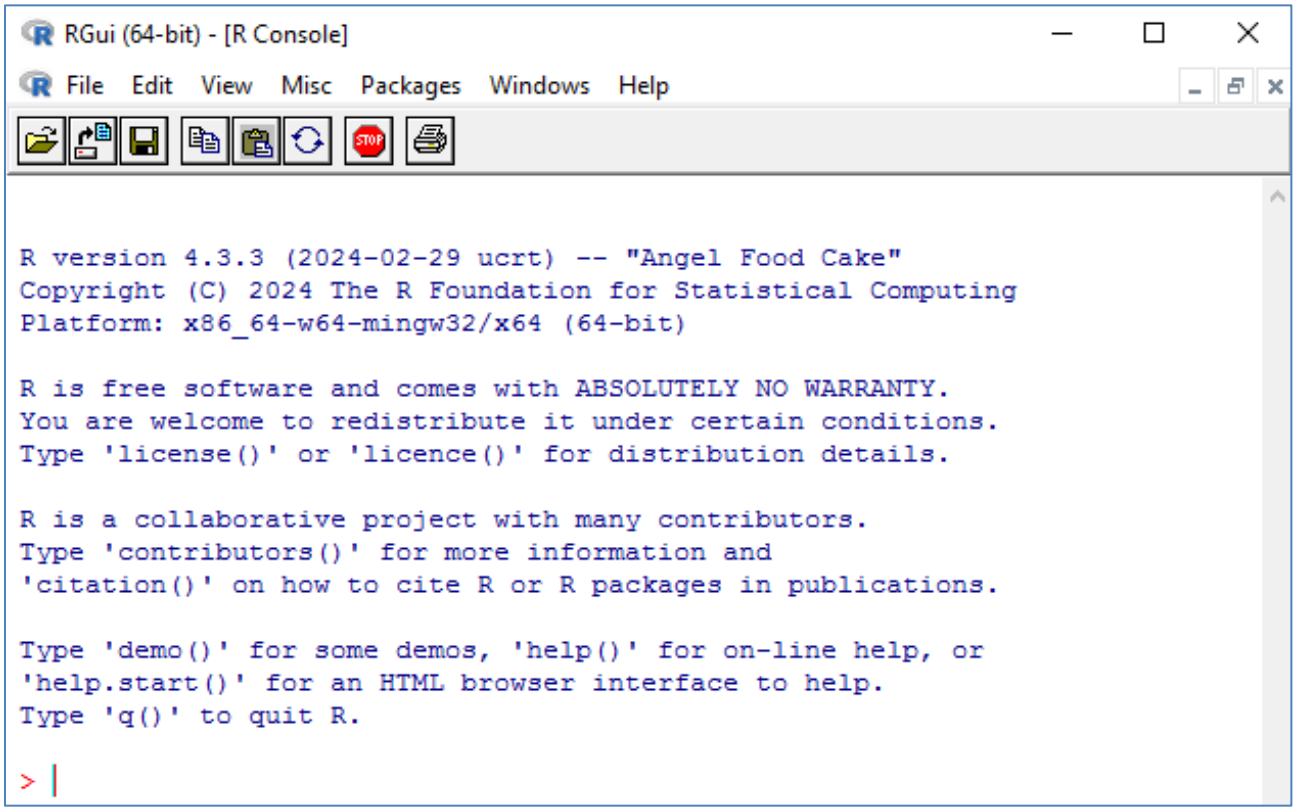


Рис.1.2. Вигляд головного вікна RGui

У вікні R Console можна вводити команди або інструкції, що будуть виконуватися (Рис.1.3.).

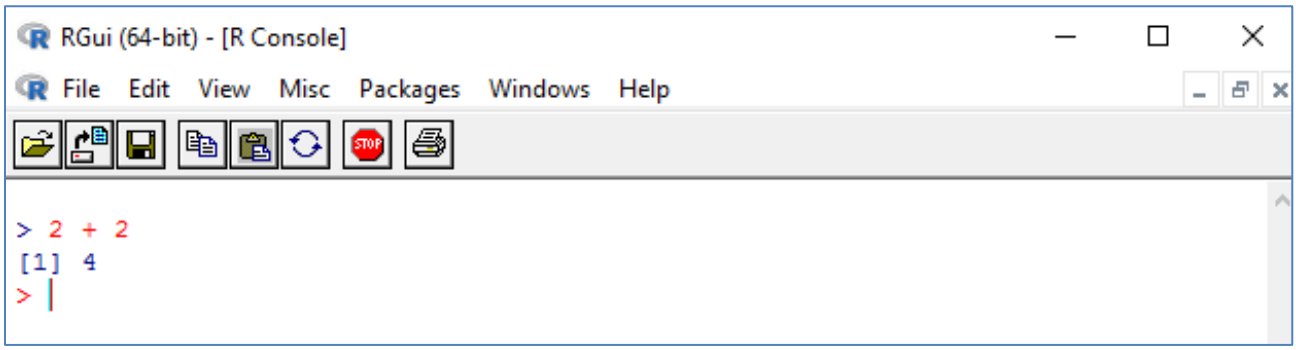


Рис.1.3. Вигляд консолі для команд R

Під час виконання програмного коду всі створені об’єкти, змінні та функції зберігаються в межах поточної сесії R. Мова R надає засоби для перегляду переліку наявних об’єктів, їх повного або вибіркового видалення, а також для збереження стану робочої сесії на диск з подальшим завантаженням. Це дозволяє уникнути повторного виконання всіх етапів обчислень, що є

особливо доцільним у випадках, коли код є складним і потребує значного часу на виконання, а збереження проміжних результатів забезпечує ефективнішу організацію роботи.

1.1.2.Завантаження та інсталяція RStudio Desktop

Вивчення мови програмування R зазвичай починається зі знайомства з RStudio – середовищем, яке включає базовий пакет R та надає користувачеві широкі можливості для програмування, візуалізації результатів обчислень і використання довідкової системи.

RStudio Desktop є безкоштовним інтегрованим середовищем для зручної роботи з мовою R, яке можна завантажити з офіційного вебсайту https://rstudio.com. Середовище підтримує два режими роботи: інтерактивний (робота через консоль) та режим скриптів (створення і виконання програмних файлів). Запустити інсталяційний файл та завершити встановлення. Під час першого запуску RStudio автоматично виявляє встановлену версію R.

Після запуску RStudio зазвичай складається з 3-х або 4-х блоків (Рис.1.4.):

– Файл, з яким працювали останнім (*зліва зверху*).

– Консоль для введення коду та виведення результатів (*зліва знизу*).

– Змінні середовища (Environment); Історія команд (History), Зєднання з зовнішніми ресурсами даних, наприклад, бази даних (Connections), навчальна інструкція (Tutorial) (*справа зверху*).

– Файли каталогу або проекту (Files), Інстальовані пакети (Packages), Допомога (Help), Візуалізація результатів (Plots, Viewer) (*справа знизу*).

Після натискання клавіші Enter введена команда надсилається на виконання. В одному рядку можна вводити кілька команд, розділяючи їх символом «;». Команди можна розміщувати на кількох рядках. У такому разі середовище відображає символ «+», що означає очікування продовження команди.

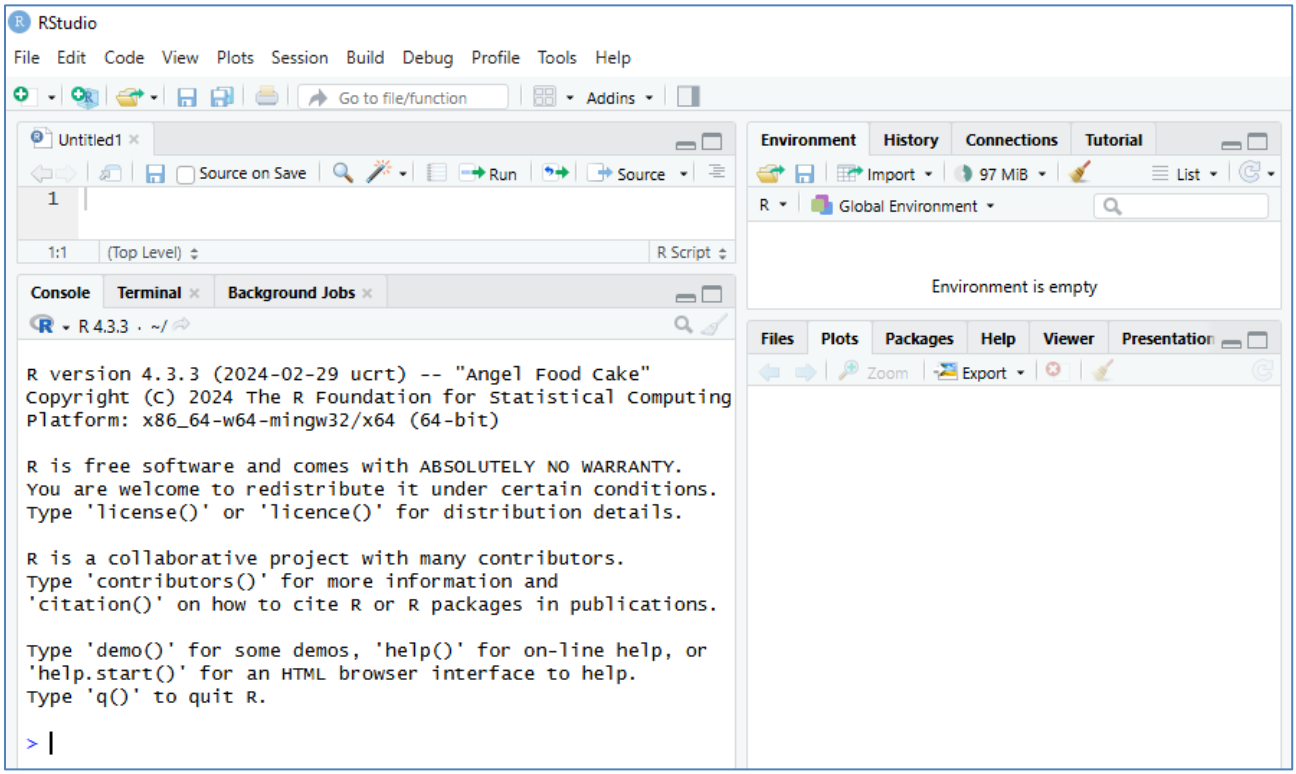


Рис.1.4. Інтерфейс RStudio

Символ «#» використовується для позначення коментарів. Увесь текст після цього символу в одному рядку ігнорується під час виконання програми. Клавіші зі стрілками «вгору» та «вниз» дозволяють переглядати та повторно використовувати раніше введені команди.

Скрипт – це група команд мовою R, що зберігається у файлі з розширенням «.R». У RStudio можна створювати нові скрипти або відкривати існуючі через меню File. Виконання всього скрипту, або його окремих частин здійснюється за допомогою відповідних команд меню або кнопок інтерфейсу.

1.1.3.Створення проєкту в RStudio

Для першої демонстрації роботи виконаємо у консолі рядки коду:

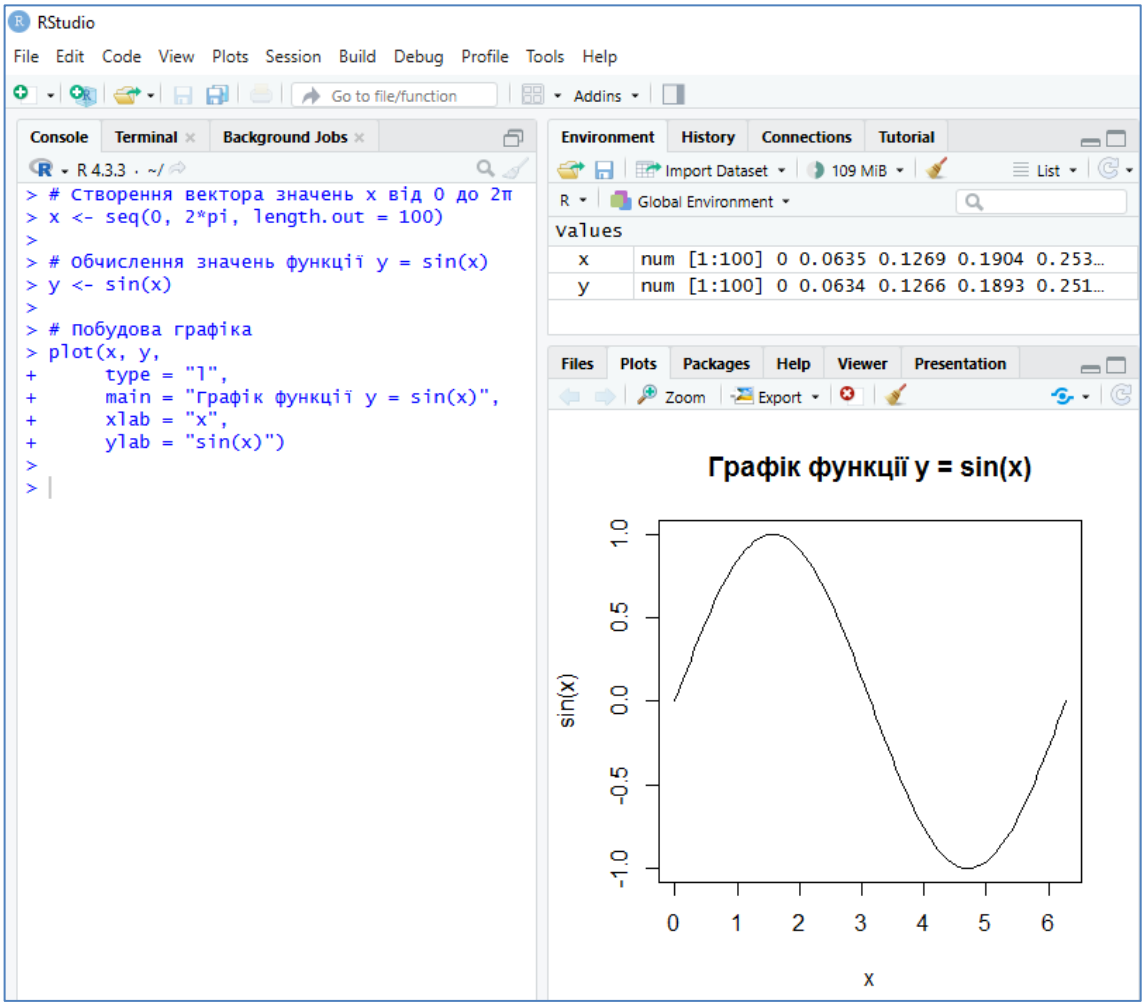


Рис.1.5. Приклад коду в RStudio

На Рис.1.5. показано побудову графіка функції $y = \sin x$ у середовищі RStudio з використанням базових графічних засобів мови R.

В першому рядку формується числовий вектор x, який містить 100 рівновіддалених значень на інтервалі від 0 до 2π.

В другому рядку для кожного значення x обчислюється відповідне значення тригонометричної функції синуса. Обчислення виконуються векторизовано.

В наступних рядках відбувається побудова графіка:

– plot(x, y) – побудова графіка залежності y від x;

– type = "l" – відображення графіка у вигляді ламаної лінії;

– main – заголовок графіка;

– xlab, ylab – підписи осей координат.

У блоці Environments відобраюаться усі змінні, що уснують у поточному робочому просторі. Графіки, що зобржаються як картинки, відображаються у блоці Plots. Якщо ж графік має більш складну візуалізацію з інтерактивними елементами, що використовують уже засоби html/css/js, то він буде відображений у блоці View.

На відміну від R, у середовищі RStudio реалізовано підхід роботи з проєктами, який дає змогу впорядковувати програмний код і об'єднувати різні його складові в межах одного програмного рішення.

Для створення першого проєкту в RStudio необхідно у верхньому меню обрати File → New Project. У вікні вибору способу створення проєкту слід натиснути New Directory (Рис1.6.). Цей варіант використовується у випадку, коли файл проєкту ще не створено, або якщо планується пізніше додати до нього вже підготовлений програмний код.

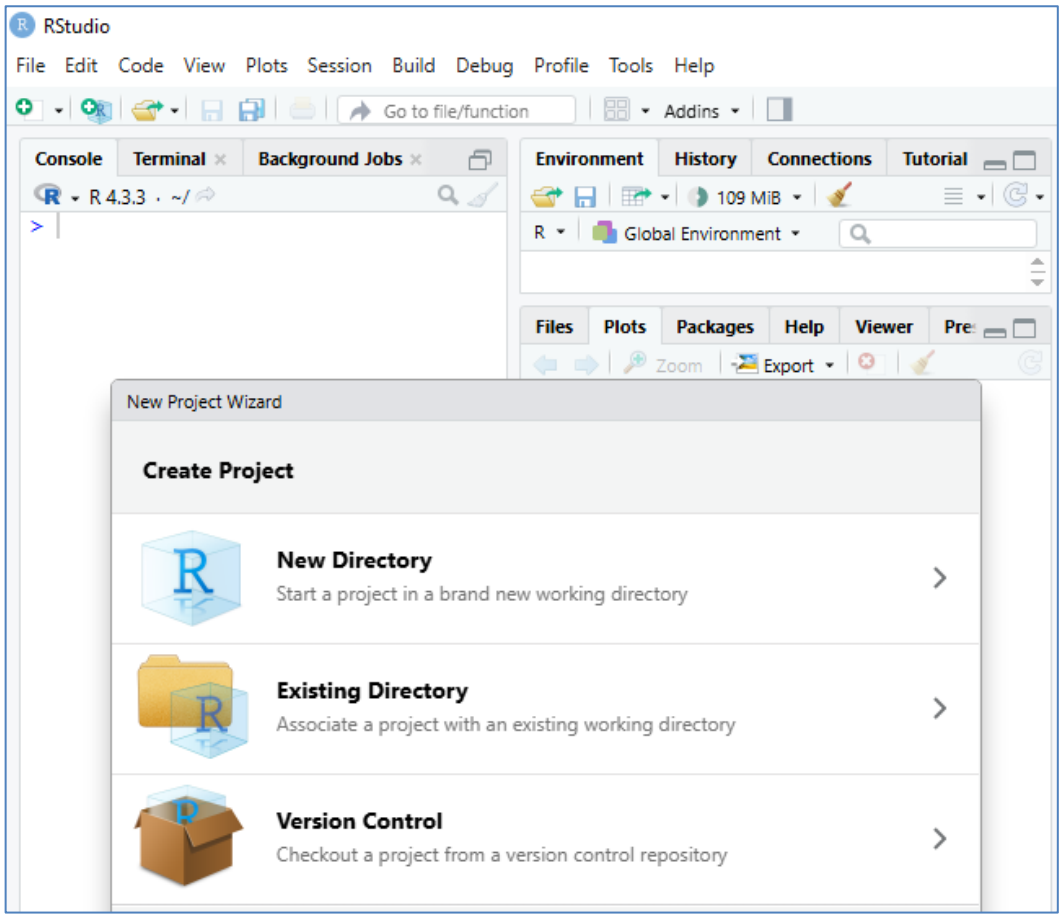


Рис.1.6. Створення нового проєкту в новій папці

Далі обираємо New Project:

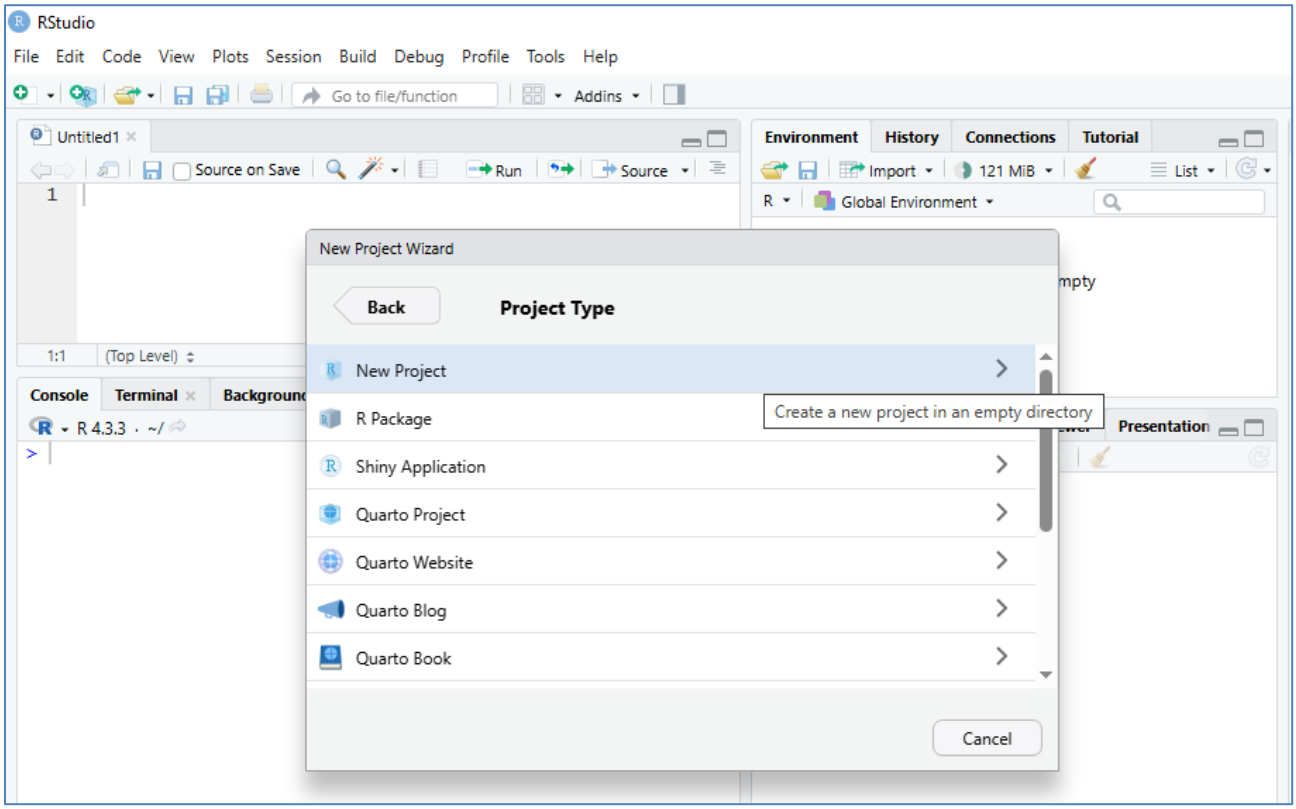


Рис.1.7. New Project для створення порожнього проєкту без попередніх файлів

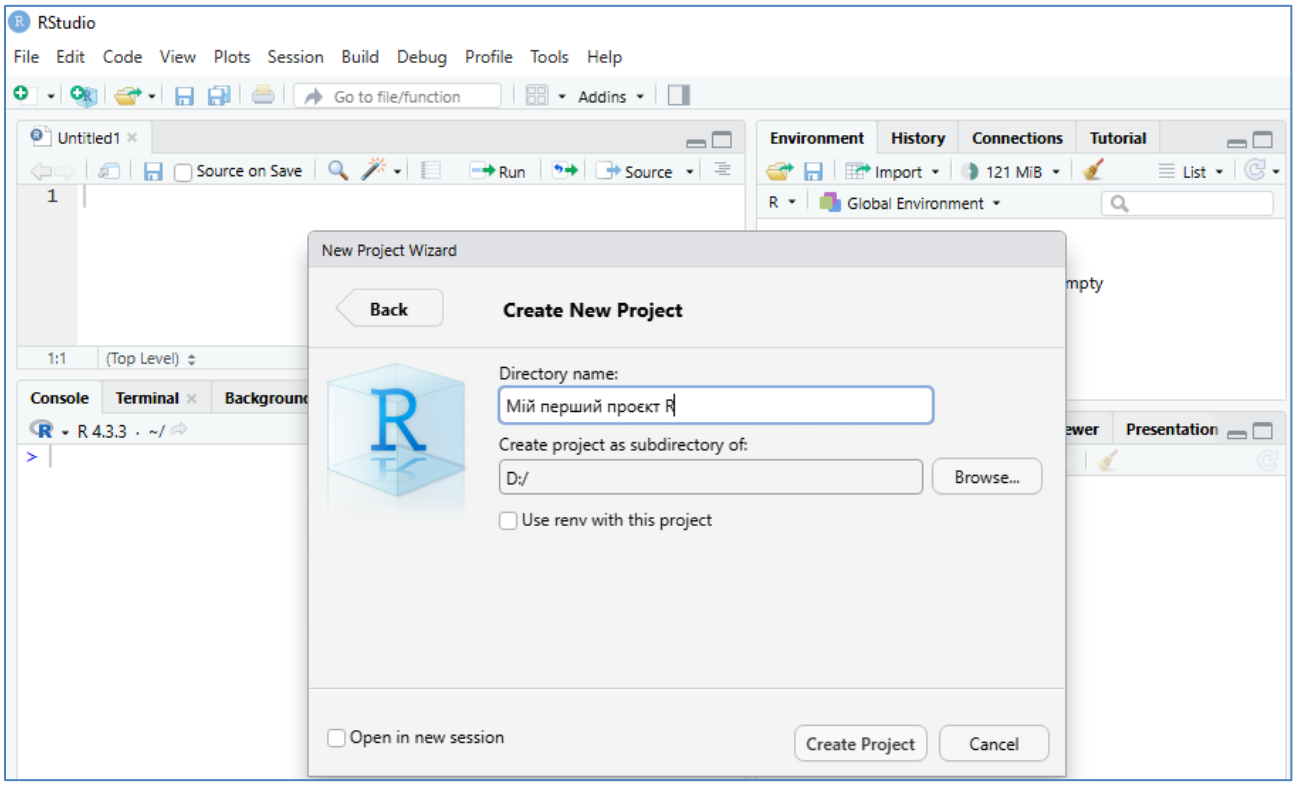


Рис.1.8. Налаштування проекту (назва папки і шлях розміщення)

RStudio автоматично відкриє новостворений проєкт. У робочому каталозі з’явиться файл з розширенням .Rproj, який зберігатиме налаштування проєкту.

1.2.Автоматизована обробка даних у RStudio

Автоматизована обробка даних у середовищі RStudio є сукупністю методів і засобів, спрямованих на формалізоване та відтворюване виконання операцій імпорту, попередньої обробки, трансформації й збереження даних із мінімальним втручанням користувача. Такий підхід забезпечує стабільність результатів аналізу та підвищує ефективність роботи з великими й динамічними масивами інформації.

У RStudio автоматизація реалізується шляхом створення скриптів і функцій мовою R, що дозволяють описати повний цикл обробки даних у вигляді послідовних алгоритмічних процедур. Застосування спеціалізованих пакетів, зокрема tidyverse, data.table, readr, DBI, забезпечує уніфікований доступ до різних джерел даних і виконання типових операцій очищення, агрегації та об’єднання таблиць.

Використання засобів автоматизованої обробки даних у RStudio сприяє зменшенню ймовірності помилок, пов’язаних із ручною обробкою, та створює умови для відтворюваності результатів, що є важливою вимогою навчально-методичних і науково-дослідницьких робіт. Крім того, автоматизація дозволяє інтегрувати процеси аналізу даних із підготовкою аналітичних звітів і навчальних матеріалів, підвищуючи якість і прозорість представлення результатів досліджень.

1.2.1.Робота з даними

Одним із найпоширеніших типів завдань у RStudio є обробка та підготовка даних. У реальних проєктах дані майже завжди потребують очищення, узгодження та трансформації. RStudio надає широкі можливості для імпорту даних із різноманітних форматів і джерел, зокрема текстових файлів CSV і TSV, електронних таблиць Excel, форматів JSON, реляційних баз даних

та великих масивів інформації. Завдяки використанню спеціалізованих пакетів мови R процес завантаження даних може бути автоматизований, що є важливим для виконання регулярних аналітичних процедур і забезпечення відтворюваності результатів.

Важливою складовою роботи з даними є їх очищення та трансформація, які включають обробку пропущених значень, усунення дублікатів, приведення змінних до відповідних типів, стандартизацію текстових даних, а також групування, агрегацію та об’єднання таблиць. Для реалізації зазначених операцій у RStudio широко застосовуються пакети сімейства tidyverse, зокрема dplyr та tidyr, що забезпечують логічну структуру коду та підвищують його читабельність.

Системний підхід до роботи з даними в RStudio дозволяє формувати впорядковані та узгоджені набори даних, придатні для подальшого статистичного аналізу, візуалізації та побудови прогнозних моделей. Таким чином, опанування методів роботи з даними є необхідною передумовою ефективного використання мови R у навчальній і дослідницькій діяльності.

Приклад практичного кейсу. Компанія мобільного зв’язку Lifecell аналізує мільйони записів про навантаження на мережу. Вхідні дані надходять у різних форматах і структурах. У RStudio спеціалісти автоматично імпортують інформацію, об’єднують її в єдину таблицю, очищають та готують до побудови моделей прогнозування навантаження на базові станції.

1.2.2.Статистичний аналіз

Статистичний аналіз є одним із базових напрямів застосування мови програмування R та середовища RStudio, оскільки саме він забезпечує науково обґрунтоване опрацювання даних і отримання узагальнених висновків. Інструментарій RStudio орієнтований на виконання як елементарних статистичних обчислень, так і складних методів аналізу, що робить його ефективним засобом для навчальної та дослідницької діяльності.

Описова статистика дозволяє швидко обчислювати:

- середні значення;

- медіани;

- мінімальні та максимальні значення;

- стандартні відхилення;

- кореляційні коефіцієнти.

Інференційна статистика дає змогу робити висновки про генеральну сукупність на основі вибірки та включає:

- t-тести;

- χ^2-тести;

- дисперсійний аналіз (ANOVA);

- кореляційний аналіз;

- побудову довірчих інтервалів.

RStudio дозволяє створювати:

- лінійні та нелінійні моделі;

- логістичну регресію;

- моделі виживання (Survival Analysis);

- факторний аналіз;

- методи кластеризації.

Приклад практичного кейсу. Мережа супермаркетів АТБ застосовує RStudio для аналізу продажів і прогнозування впливу сезонних факторів. На основі регресійних моделей компанія визначає, як свята, погодні умови або знижки впливають на попит на окремі категорії товарів.

1.2.3.Візуалізація даних

Візуалізація результатів допомагає виявляти структуру, тренди та аномалії у даних. Однією з переваг RStudio є можливість створювати професійні та інформативні графіки.

Засоби мови R надають можливість будувати основні типи графіків, зокрема гістограми, діаграми розсіювання, лінійні та кругові діаграми, а також boxplot-діаграми. Такі візуалізації застосовуються для первинного аналізу

даних і попередньої оцінки їх розподілу, структури та варіативності.

Розширені можливості візуалізації забезпечує пакет ggplot2, який ґрунтується на принципах «граматики графіки». Його використання дозволяє створювати багатошарові графіки, поєднувати різні типи даних, додавати елементи статистичного аналізу та налаштовувати візуальні параметри відповідно до вимог наукових публікацій і навчальних матеріалів. Завдяки цьому результати аналізу набувають високого рівня інформативності та наочності.

Крім статичних зображень, RStudio підтримує створення інтерактивних візуалізацій за допомогою пакетів plotly, highcharter та leaflet, що є особливо актуальним під час розробки аналітичних панелей і дашбордів. Застосування інтерактивних графіків дозволяє користувачам детальніше досліджувати дані та підвищує ефективність представлення результатів у навчальній і дослідницькій діяльності.

Приклад практичного кейсу. Під час проведення педагогічного експерименту здійснюється аналіз результатів навчальних досягнень здобувачів освіти контрольної та експериментальної груп з навчальної дисципліни. Після попередньої обробки даних у середовищі RStudio формується набір показників, що відображає рівень успішності, динаміку результатів та варіативність оцінок.

За допомогою інструментів візуалізації в RStudio будуються гістограми та boxplot-діаграми для порівняння розподілів навчальних досягнень у групах, а також діаграми розсіювання й лінійні графіки для відображення змін показників у часі. Застосування пакета ggplot2 дозволяє представити результати у вигляді наочних і структурованих графічних моделей, що полегшує інтерпретацію отриманих даних.

Отримані візуалізації використовуються для обґрунтування висновків щодо ефективності запропонованої методики навчання, а також для представлення результатів дослідження у звітах і наукових публікаціях, що підвищує їх інформативність та наукову аргументованість.

1.2.4.Машинне навчання та прогнозування

У RStudio реалізовано численні алгоритми машинного навчання для розв'язування задач класифікації та регресії, зокрема лінійну й логістичну регресію, дерева рішень, випадкові ліси, методи градієнтного бустингу та інші ансамблеві підходи. Для зручності побудови, налаштування та оцінювання моделей широко застосовуються спеціалізовані фреймворки, зокрема caret, який забезпечує класичний підхід до машинного навчання, та tidymodels, що реалізує сучасний модульний підхід і сприяє стандартизації аналітичного процесу.

Для аналізу часових рядів і прогнозування, який використовується для дослідження динамічних процесів, застосовуються пакети forecast, prophet та tsibble, що дозволяють моделювати тренди, сезонні коливання та випадкові компоненти, а також оцінювати точність прогнозів. Зазначені інструменти знаходять широке застосування в економічних, соціальних та освітніх дослідженнях.

Приклад практичного кейсу. Компанія «Нова Пошта» використовує моделі прогнозування для оцінювання завантаженості відділень, оптимізації маршрутів кур'єрів і прогнозування часу доставки посилок.

1.2.5.Аналіз великих даних та автоматизація

Аналіз великих даних пов'язаний з опрацюванням значних за обсягом і різнорідних масивів інформації, що характеризуються високою швидкістю надходження та складною структурою. Середовище RStudio забезпечує умови для виконання таких завдань завдяки наявності спеціалізованих засобів обробки та інтеграції даних.

Для роботи з великими наборами даних у RStudio застосовуються пакети data.table та arrow, які орієнтовані на оптимізацію обчислювальних операцій і раціональне використання пам'яті. За допомогою пакета sparklyr можливе підключення до інфраструктури Apache Spark, що дозволяє здійснювати розподілені обчислення та виконувати аналіз даних у кластерному середовищі.

Автоматизація аналітичних процесів у RStudio реалізується шляхом створення відтворюваних сценаріїв обробки даних, які охоплюють етапи імпорту, трансформації, аналізу та збереження результатів. Використання скриптів і користувацьких функцій забезпечує формалізацію аналітичних процедур, зменшення залежності від ручного втручання та підвищення узгодженості результатів. Зазначений підхід застосовується під час виконання регулярних аналітичних операцій і формування звітної документації.

Застосування засобів аналізу великих даних і автоматизації в середовищі RStudio створює передумови для масштабування аналітичних рішень, підвищення відтворюваності досліджень і відповідає сучасним тенденціям розвитку цифрових технологій.

Приклад практичного кейсу. У мережі супермаркетів «Сільпо» аналіз великих даних у середовищі RStudio може застосовуватися для обробки масивів транзакційних даних, що накопичуються щоденно у торговельних залах по всій країні. Обсяг такої інформації обчислюється десятками мільйонів записів і включає дані про продажі, часові мітки, товарні позиції, знижки та способи оплати.

На етапі завантаження даних використовується пакет arrow, який дає змогу працювати з колоночними форматами зберігання та виконувати частину обчислень безпосередньо на рівні файлової системи. Для розподіленої обробки дані інтегруються з кластером Apache Spark за допомогою пакета sparklyr, що забезпечує масштабування аналітичних операцій. Агрегація продажів за часовими інтервалами, магазинами та товарними групами виконується без необхідності повного завантаження даних у оперативну пам’ять.

Подальший аналіз у RStudio передбачає побудову узагальнених показників, які використовуються для оцінювання тенденцій споживчого попиту та ефективності асортиментної політики. Усі етапи обробки реалізуються у вигляді автоматизованих сценаріїв, що запускаються за розкладом і дозволяють регулярно оновлювати результати без ручного втручання.

Завершальним етапом є формування стандартизованих аналітичних звітів і таблиць для внутрішніх інформаційних систем компанії. Така організація аналітичного процесу забезпечує узгодженість результатів, зменшує часові витрати на підготовку звітності та сприяє системному використанню великих даних у процесі управління торговельною мережею.

1.2.6.Розробка веб-додатків та документування

Середовище RStudio надає можливості для створення інтерактивних веб-застосунків і підготовки відтворюваних аналітичних матеріалів, що поєднують результати обчислень, візуалізації та текстові пояснення. Такий підхід забезпечує інтеграцію аналітичної діяльності з представленням результатів у зручній для користувачів формі.

Розробка веб-додатків у RStudio реалізується за допомогою фреймворку Shiny, який дозволяє створювати клієнт-серверні застосунки мовою R без необхідності глибокого використання веб-технологій. Інтерфейс користувача формується на основі декларативних описів, а серверна частина забезпечує обробку даних і виконання обчислень у режимі реального часу. Для побудови аналітичних панелей застосовуються пакети shinydashboard та flexdashboard, що дають змогу структурувати інформацію у вигляді інтерактивних панелей з фільтрами, таблицями та графіками.

Документування результатів аналізу в RStudio здійснюється з використанням інструментів RMarkdown та Quarto, які підтримують поєднання програмного коду, текстових коментарів і графічних матеріалів в одному документі. Це дозволяє формувати наукові статті, звіти, презентації та навчальні матеріали з можливістю автоматичного оновлення результатів при зміні вихідних даних. Такий формат сприяє прозорості аналітичних процедур, відтворюваності обчислень і стандартизації представлення результатів.

Приклад практичного кейсу. У закладі вищої освіти середовище RStudio використовується для створення інтерактивного веб-застосунку з аналізу результатів навчальної успішності студентів. Вихідні дані формуються на

основі електронного журналу та містять відомості про поточні й підсумкові оцінки, відвідуваність і результати тестового контролю з окремих дисциплін.

За допомогою фреймворку Shiny у RStudio розробляється веб-додаток, який надає користувачам можливість обирати навчальний курс, групу студентів та часовий період аналізу. Серверна частина застосунку забезпечує обробку запитів, виконання статистичних обчислень і формування узагальнених показників у режимі реального часу. Результати аналізу відображаються у вигляді таблиць і графіків, що оновлюються відповідно до заданих параметрів.

Паралельно з розробкою веб-застосунку здійснюється документування аналітичних процедур із використанням RMarkdown або Quarto. На основі підготовлених шаблонів автоматично формуються звіти у форматах HTML або PDF, які містять опис методики аналізу, результати обчислень і відповідні візуалізації. Такий підхід дозволяє поєднати інтерактивне представлення даних із відтворюваною документацією та сприяє систематичному використанню результатів аналізу в освітньому процесі.

1.3.Керування пакетами, файлами та робочими каталогами в RStudio

Розпочинають роботи в середовищі RStudio із інсталяції необхідних пакетів. Необхідні пакети можна завантажити з Інтернету у вигляді zip-файлів, виконати розархівацію у визначену папку.

В режимі консолі RStudio викликати функцію інсталяції install.packages. Команда install.packages(‘raster’) викличе звертання комп’ютера до стандартного депозитарію (як правило, це cran.us.r-project.org). В RStudio для цього можна використати пункти головного меню Tools->Install.

Розпочинаючи робочу сесію в середовищі RStudio необхідно підключити пакет у робочу область пам’яті (Workspase) для цього використовують функцію library(). Наприклад, library(readr) підключає пакет readr та дозволяє використовувати всі функції під час роботи сесії.

При певних умовах виникає необхідність відключити пакет для цього використовують функцію detach(). Так для відключення пакет dplyr в режимі

консолі вводять команду detach(“package: dplyr”). Функції пакета перестануть бути доступними у цій робочі сесії.

Для повного видалення пакета з Workspase необхідно використати функцію remove(), наприклад, remove.packages(“ggplot2”).

Підключення пакета sos (library(sos)) дозволяє отримувати довідкову інформацію із всіх доступних депозитаріїв. Доступна команда ??? <тема>.

У командному рядку можна писати декілька команд, але їх треба розділяти знаком ";". Команди в RStudio виконуються після натиснення клавіши Enter. В якості оператора присвоювання як правило застосовують знак "<-", для якого в R можна застосовувати сполучення: "Alt + "-" ".

У процесі роботи з мовою програмування R та середовищем RStudio використовуються різні типи файлів, кожен з яких має своє призначення. Файл з розширенням .R містить програмний код мовою R. Призначення цих файлів: написання скриптів; реалізація алгоритмів обробки даних; виконання статистичних розрахунків; створення функцій тощо. Саме ці файли призначені для виконання в середовищі RStudio. Їх особливістю є те, що вони виконуються по рядках або повністю, не містять форматування чи текстових пояснень та використовується для автоматизації обчислень.

Файли .Rmd (R Markdown) поєднують код R, текстові пояснення та результати виконання. Ці файли призначені для створення звітів лабораторних робіт, наукових статей, презентацій. Ці файли підтримують Markdown-розмітку, можуть експортуватися у HTML, PDF, DOCX. Файли .csv (Comma-Separated Values) це текстовий формат для зберігання табличних даних. Саме через ці файли здійснюється зберігання та обмін наборами даних, імпорт даних у R та експорт результатів аналізу. Зручно, легко зчитується функціями read.csv(), read_csv().

Файл .RData зберігає об’єкти робочого середовища R у бінарному форматі. Їх використання дозволяє зберігати результати обчислень, швидке відновлення сесії та збереження великих об’єктів (data frame, списків, моделей). В середовище R завантажується функцією load().

При роботі в dRStudio важливим є робочий каталог (Working Directory). В мові R hобочий каталог – це папка, з якою R працює за замовчуванням при зчитуванні та збереженні файлів.

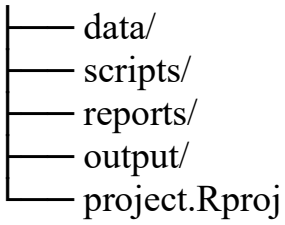
Використовують дві команди для роботи з каталогом. Встановлює новий робочий каталог setwd(). Команда getwd() # показує поточний робочий каталог

Робочий каталог спрощує доступ до файлів, зменшує кількість помилок у шляхах, забезпечує коректне виконання скриптів.

Професійний підхід до використання мови R полягає у використанні проєктів RStudio. Прийнято що це організований простір для роботи над конкретним завданням або дослідженням. Переваги проєктів в тому, що автоматично встановлюють робочий каталог, зберігають історії команд, підтримують контроль версій (Git), покращують структуру коду та даних.

Типова структура проєкту:

project/



Типи файлів .R, .Rmd, .csv, .RData забезпечують гнучку роботу з кодом, даними та результатами аналізу, а використання робочих каталогів і проєктів RStudio є основою для організованої, відтворюваної та професійної роботи в середовищі R.

Мова програмування R має потужну вбудовану довідкову систему, яка дозволяє користувачеві швидко отримати інформацію про функції, пакети, аргументи та приклади їх використання. У середовищі RStudio ця система інтегрована з графічним інтерфейсом і значно спрощує навчання та професійну роботу.

Основний спосіб отримання довідки в R є функція help(), яка використовується для отримання документації щодо конкретної функції, об’єкта або пакета.

Команда help(mean) відкриває довідкову сторінку у вкладці Help в RStudio. Подана інформація описує призначення функції, подає перелік аргументів та наведено приклади використання. Також як параметри функції можна явно вказувати пакет:

help(mean, package = "base")

Типова довідкова сторінка функції в R містить такі розділи:

1. Description – загальний опис функції
2. Usage – синтаксис виклику
3. Arguments – перелік аргументів і їх значення
4. Details – детальні пояснення
5. Value – результат, який повертає функція
6. Examples – приклади використання (дуже важливі!)

Рекомендують завжди переглядати розділ Examples для практичного розуміння роботи функції.

Якщо точна назва функції невідома, можна скористатися пошуком за ключовими словами. help.search("regression"). Це дозволяє знайти всі функції та пакети, пов’язані з темою регресії.

В середовищі RStudio підтримується режим перегляду довідки пакета.

help(package = "dplyr")

Ця довідка відображає загальний опис пакета, перелік доступних функцій, документація до кожної функції. Команда example(mean) автоматично виконує всі приклади з довідкової сторінки функції.

Безпосередньо в RStudio доступні автодоповнення і підказки. RStudio виконує автодоповнення функцій (клавіша Tab), організовує діалогову підказку аргументів при введенні функції, підсвічує помилки синтаксису. Це значно прискорює написання коду та зменшує кількість помилок.

Хоча вбудована довідка є основною, RStudio дозволяє легко працювати з офіційною документацією CRAN та віньєтками пакетів (vignette()).

Наприклад vignette("dplyr")

Правильна робота з help() дозволяє краще розуміти функції, уникати помилок, писати більш якісний код.

Довідкова система R (help, ?, ??, example, vignette) є невід'ємною частиною професійної роботи в RStudio. Вона забезпечує швидкий доступ до документації, підтримує самонавчання та підвищує ефективність програмування і аналізу даних.

RStudio – охоплює всі етапи роботи з даними: від імпорту й очищення до статистичного аналізу, візуалізації, моделювання, машинного навчання, прогнозування та створення інтерактивних аналітичних рішень.

RStudio – це універсальна платформа для аналітики, що охоплює всі етапи роботи з даними:

1. Імпорт і очищення.
2. Статистичний аналіз.
3. Візуалізація.
4. Моделювання й машинне навчання.
5. Прогнозування.
6. Робота з великими даними.
7. Автоматизація та ETL.
8. Розробка інтерактивних аналітичних додатків.
9. Документування та створення звітів.

Завдяки гнучкості та відкритому коду RStudio використовується як у наукових дослідженнях, так і в бізнесі, медицині, фінансовому секторі, освіті та державному управлінні.

Лабораторні роботи до розділу 1

Лабораторна робота 1. Довідкова система RStudio та керування робочим середовищем

Метою лабораторної роботи є формування практичних навичок:

- використання довідкової системи середовища RStudio;
- пошуку інформації про функції R;
- роботи з історією команд;
- керування робочою директорією та файлами.

Теоретичні відомості

Середовище RStudio має потужну довідкову систему, яка дозволяє швидко отримувати інформацію про будь-яку функцію, її параметри та приклади використання. Основними інструментами довідки є функції help(), ?, example(), help.search() та RSiteSearch().

Робоча директорія (working directory) – це каталог, у якому R за замовчуванням зчитує та зберігає файли. Коректна робота з директоріями є обов'язковою умовою для організації проєктів аналізу даних.

Хід виконання роботи

1. Використовуючи довідкову систему R, отримайте інформацію про призначення та параметри функцій для статистичних обчислень. Перегляньте документацію та приклади використання відповідних функцій.

help(mean)

?sd

example(rnorm)

Пояснення: Команди help() та ? викликають опис функції, її параметрів і прикладів. Функція example() автоматично виконує приклади з документації.

2. Здійсніть пошук інформації за термінами, пов'язаними з аналізом даних, у локальній довідковій системі та зовнішніх джерелах.

help.search("correlation")

RSiteSearch("linear regression")

Пояснення: help.search() шукає терміни в локальній довідці, RSiteSearch() – у зовнішніх джерелах в Інтернеті.

3. Визначте поточну робочу директорію, змініть її на вказану папку та перегляньте вміст каталогу.

getwd()

setwd("C:/R_projects")

dir()

Пояснення: Команда getwd() повертає поточну робочу папку, setwd() змінює її, dir() показує файли у каталозі.

4. Перегляньте історію раніше виконаних команд у середовищі R.

history(10)

Результат виконання:

Після виконання завдання студент повинен уміти користуватися довідковою системою R, знаходити необхідну інформацію про функції, керувати робочою директорією та переглядати історію команд.

Інтегровані завдання до розділу 1

Завдання 1: Робота з консоллю та багаторядковими командами

Використовуючи консоль RStudio (нижнє ліве вікно), введіть команду для будь-якого математичного обчислення (наприклад, 10 + 20), але розділіть її на два рядки.

Дія: Напишіть першу частину команди, натисніть Enter, переконайтеся, що з'явився символ запрошення +, і завершіть команду на другому рядку.

Мета: Навчитися використовувати символ продовження команди та розуміти різницю між знаками > та +.

Завдання 2: Використання коментарів та навігація в історії

Напишіть у консолі команду, яка містить обчислення та текстовий коментар після нього (наприклад, x <- 5 # присвоєння значення змінній).

Дія: Після виконання команди скористайтеся клавішами стрілок "вгору" та "вниз" на клавіатурі, щоб знайти цю команду в історії та змінити значення змінної.

Мета: Закріпити навичку документування коду за допомогою символу # та швидкого редагування попередніх команд.

Завдання 3: Створення та збереження скрипту

Створіть новий файл скрипту через головне меню RStudio.

Дія: Перейдіть у File -> New script, введіть у верхньому лівому вікні три різні команди (кожна з нового рядка) і збережіть файл під назвою my_first_analysis.r.

Мета: Отримати практичний досвід роботи в режимі скрипта та правильного збереження робочих файлів.

Завдання 4: Вибіркове виконання коду

У створеному в попередньому завданні скрипті виконайте лише одну конкретну команду, не запускаючи весь файл повністю.

Дія: Виділіть мишкою другий рядок вашого скрипту та натисніть кнопку Run line or selection. Перевірте результат виконання в консолі.

Мета: Навчитися тестувати окремі частини програми без необхідності перезапуску всього коду.

Завдання 5: Дослідження інтерфейсу та візуалізації

Знайдіть вікно, яке, згідно з матеріалом, є "найбільш навантаженим" (нижнє праворуч).

Дія: Переключіться між вкладками Files, Plots, Packages та Help. Спробуйте знайти будь-яку інформацію у вкладці Help, ввівши назву базової функції (наприклад, print) у рядок пошуку довідки.

Мета: Ознайомитися з допоміжними інструментами RStudio для керування файлами, пакетами та отримання довідкової інформації.

Завдання 6: Виконання декількох команд в одному рядку

Згідно з текстом, в одному рядку можна вводити кілька команд одночасно.

Дія: Введіть у консолі три прості арифметичні операції (наприклад, 2+2, 5*5 та 10/2), розділивши їх спеціальним символом ;. Натисніть Enter та проаналізуйте, як консоль виводить результати.

Мета: Навчитися компактно записувати послідовність дій у консолі.

Завдання 7: Моніторинг активних змінних

Використовуйте праве верхнє вікно інтерфейсу для відстеження стану програми.

Дія: Створіть у консолі три змінні з різними значеннями (наприклад, a <- 10, b <- 20, c <- a + b). Подивіться у вікно Environment (праве верхнє) і переконайтеся, що ці змінні там з'явилися.

Мета: Навчитися контролювати об'єкти, що знаходяться в пам'яті поточної сесії.

Завдання 8: Робота з історією попередніх сесій

Текст зазначає, що RStudio дозволяє бачити історію роботи не лише поточної сесії.

Дія: У правому верхньому вікні перейдіть на вкладку History. Спробуйте знайти команди, які ви вводили на самому початку роботи, та за допомогою кнопки To Console перемістіть одну з них у вікно консолі для повторного виконання.

Мета: Опанувати інструменти повторного використання коду з історії.

Завдання 9: Примусове завершення введення

Іноді виникає потреба скасувати введення команди, яка була розпочата помилково.

Дія: Почніть вводити довгу команду в консолі, натисніть Enter, щоб з'явився символ +, а потім натисніть клавішу Esc.

Мета: Відпрацювати навичку швидкого виходу з режиму очікування продовження команди (символ +) та повернення до стандартного запрошення >.

Завдання 10: Налаштування персонального робочого простору

Дія: Зайдіть у налаштування середовища (зазвичай Tools -> Global Options). Знайдіть розділ Appearance та спробуйте змінити тему оформлення редактора (Editor theme) на темну або іншу, яка вам подобається.

Мета: Навчитися адаптувати інтерфейс IDE для комфортної тривалої роботи.

Контрольні питання до розділу 1

1. За якими умовами ліцензування розповсюджується RStudio?
2. На якій мові програмування базується R і яка організація була розробником її попередниці?
3. Опишіть призначення чотирьох основних робочих площин (вікон) в інтерфейсі RStudio.
4. Що таке RStudio і яка його основна мета як інструменту для розробника?
5. Що таке RStudio і які основні компоненти воно об'єднує для зручної роботи з мовою програмування R?
6. Що таке фреймворк Shiny і які можливості він відкриває для розробки аналітичних веб-додатків?
7. Як компанії Lifecell та АТБ використовують RStudio у своїй діяльності для оптимізації бізнес-процесів?
8. Як середовище RStudio сигналізує про те, що команда не завершена і потребує продовження на наступному рядку?
9. Яке розширення мають файли, у яких зберігаються скрипти мови R?
10. Який символ використовується для розділення кількох команд, записаних в одному рядку?
11. Який символ використовується для створення коментарів у коді R, і як програма реагує на текст після цього символу?
12. Які два основні режими роботи підтримує RStudio як інтерпретатор мови R?

13. Які команди головного меню або кнопки інтерфейсу використовуються для виконання всього скрипту або лише його виділеної частини?

14. Які типи статистичного аналізу (описовий та інференційний) дозволяє виконувати RStudio?

15. Які формати файлів та джерела даних можна імпортувати в RStudio і за допомогою яких конкретних пакетів?

16. Які фреймворки та методи машинного навчання доступні в RStudio для розв'язання завдань класифікації, регресії та кластеризації?

РОЗДІЛ 2. РОБОТА З ДАНИМИ

2.1.Типи даних і структури даних у R

Мова програмування R надає набір типів даних і структур даних, які забезпечують представлення, зберігання та обробку інформації під час виконання статистичних і аналітичних завдань. Розуміння особливостей типів і структур даних є необхідною умовою коректної реалізації алгоритмів аналізу, побудови моделей та інтерпретації результатів.

До базових (атомарних) типів даних у R належать:

– numeric – числові значення з плаваючою комою, які використовуються для подання кількісних показників та результатів обчислень;

– integer – цілі числа, що застосовуються у випадках, коли потрібна дискретна числова інформація;

– logical – логічні значення TRUE та FALSE, які використовуються в умовних операторах і фільтрації даних;

– character – текстові рядки, що містять символьну інформацію;

– complex – комплексні числа, які використовуються у спеціалізованих математичних обчисленнях.

Кожне значення в R має чітко визначений тип, що впливає на допустимі операції та результат їх виконання.

Для організації та обробки наборів даних у R застосовуються такі структури:

– vector – впорядкована послідовність елементів одного типу даних; є базовою структурою, на основі якої будуються інші об’єкти;

– matrix – двовимірна структура, елементи якої мають однаковий тип; використовується для подання табличних числових даних;

– array – багатовимірне узагальнення матриць, що застосовується для зберігання даних з більш ніж двома вимірами;

– list – структура, яка може містити елементи різних типів і розмірів; широко використовується для збереження результатів обчислень і моделей;

– data frame – таблична структура, де кожен стовпець є вектором одного типу, а рядки відповідають окремим спостереженням; є основним форматом зберігання даних у прикладних задачах;

– factor – спеціальний тип даних для подання категоріальних змінних, що використовується в статистичному аналізі та моделюванні.

Вибір відповідного типу та структури даних визначає коректність обчислень, ефективність обробки та зручність подальшого аналізу. Структури даних у R забезпечують можливість виконання векторизованих операцій, що підвищує продуктивність обчислень і спрощує реалізацію аналітичних алгоритмів. Раціональне використання типів і структур даних є передумовою побудови відтворюваних та масштабованих рішень у середовищі RStudio.

Мова програмування R є об’єктно-орієнтованою, тобто всі дані, з якими працює користувач, зберігаються у вигляді об’єктів. Об’єкт – це іменована область пам’яті, що містить значення або структуру даних. Кожен об’єкт має ім’я, тип даних та значення.

В R об’єктами можуть бути:

- окремі значення;
- вектори;
- таблиці (data frame);
- функції;
- результати обчислень.

Для створення об’єкта та присвоєння йому значення використовується оператор <–. Стандарт кодування R рекомендує використовувати <–, оскільки він чітко відрізняється від оператора порівняння =.

$$x < -10$$

Також можливе присвоєння через $x = 10$ або $10 -> x$. Однак у професійній практиці прийнято використовувати саме <–.

При написанні коду мовою R доцільно застосовувати функції перегляду та керування об’єктами, які забезпечують контроль за вмістом робочого середовища та спрощують процес аналізу даних. Такі функції дозволяють

отримувати інформацію про створені об’єкти, їхню структуру, типи даних і розмір, а також виконувати операції з очищення та організації робочої області пам’яті.

ls() - список об’єктів у середовищі

rm(x) - видалення об’єкта

class(x) - тип (клас) об’єкта

RStudio відображає всі створені об’єкти у вкладці Environment.

Для введення інформації із клавіатури виористовують функцією readline(prompt = "Text"):

a<-readline(prompt = "Введіть a")

b<-readline(prompt = "Введіть b")

Переведення в числовий формат

a<-as.numeric(a)

b<-as.numeric(b)

2.2.Формування виразів у мові R

Мова програмування R використовує багатий набір операторів, за допомогою яких створюються вирази – основні елементи будь-якої програми. Оператори дозволяють виконувати арифметичні обчислення, порівняння, логічні перевірки, присвоєння значень, а також керувати потоком виконання програми.

Оператор – це спеціальний символ або ключове слово, яке виконує певну дію над одним або кількома операндами.

Приклади операторів:

- +, -, *, /
- <-, =
- ==, !=
- &, |
- %in%
- <-

Вираз – це комбінація операторів, об’єктів та значень, який після

обчислення повертає результат.

У R кожен вираз має значення, навіть оператор присвоєння. Оператори присвоєння використовуються для створення об’єктів та надання їм значень.

Основні оператори:

x<- 5 (рекомендований стиль)

x = 5

5 -> x

Таблиця 2.1.

Базові арифметичні операції та вбудовані константи

Операція	Символ	Приклад	Результат
Додавання	+	5 + 3	8
Віднімання	-	5 - 3	2
Множення	*	5 * 3	15
Ділення	/	5 / 3	1.666667
Піднесення до степеня	^ або **	5^3	125
Цілочисельне ділення	%/%	5 %/% 3	1
Остача від ділення	%%	5 %% 3	2

Вбудовані константи:

– pi: Число π (3.141593...)

– Inf: Нескінченність

– NaN: Not a Number (наприклад, результат ділення 0/0)

У R арифметичні операції векторизовані, тобто виконуються над усіма елементами вектора.

2.2.1.Оператори порівняння

У мові програмування R оператори порівняння використовуються для зіставлення значень та формування логічних виразів, результатом яких є значення типу TRUE або FALSE. Такі оператори застосовуються під час фільтрації даних, побудови умовних конструкцій, статистичного аналізу та

моделювання.

Таблиця 2.2.

Базові оператори порівняння

Оператор	Значення
==	дорівнює
!=	не дорівнює
<	менше
>	більше
<=	менше або дорівнює
>=	більше або дорівнює

Результатом застосування кожного з операторів є логічне значення або логічний вектор, якщо порівняння виконується над векторними об’єктами.

Оператори порівняння в R є векторизованими, тобто застосовуються поелементно до всіх компонентів векторів, матриць або стовпців таблиць даних. Це дозволяє формувати логічні маски, які широко використовуються для відбору підмножин даних.

Для символьних даних оператори порівняння виконуються на основі лексикографічного порядку. Під час роботи з факторними змінними порівняння здійснюється з урахуванням їх внутрішнього кодування та рівнів, що потребує коректного задання структури факторів.

Оператори порівняння є основою для формування умов у конструкціях if, ifelse, while та інших керуючих структурах. Логічні результати порівняння визначають подальший хід виконання програми.

2.2.2.Логічні оператори

Логічні оператори в мові програмування R використовуються для поєднання, заперечення та оцінювання логічних виразів. Вони забезпечують побудову складних умов, що застосовуються під час фільтрації даних, керування виконанням програм і реалізації аналітичних алгоритмів.

Таблиця 2.3.

Базові логічні оператори

Оператор	Опис
&	логічне І (векторне)
!	логічне НЕ
&&	логічне І (одне значення)

Практичне застосування:

x <- c(5, 10, 15)

x > 7 & x < 20

Під час роботи з логічними операціями слід враховувати наявність відсутніх значень NA. У більшості випадків операції з NA призводять до результату NA, що потребує додаткового контролю з використанням функцій перевірки на пропущені значення. Необхідно пам’ятати що & і | працюють поелементно, тоді як && і || – лише з першим елементом.

2.2.3.Оператори роботи з векторами

Мова R підтримує векторизовані операції, що дозволяє виконувати обчислення над усіма елементами вектора одночасно без використання явних циклів.

Арифметичні оператори (+, -, *, /, ^) у R застосовуються до векторів поелементно. Якщо операція виконується між двома векторами однакової довжини, обчислення здійснюються для кожної пари відповідних елементів. У випадку різної довжини векторів використовується механізм повторення елементів коротшого вектора, що потребує уважного контролю коректності результатів.

Оператори порівняння та логічні оператори також є векторизованими. Результатом їх застосування є логічний вектор, який використовується для фільтрації та умовного відбору елементів. Такий підхід забезпечує

компактність коду та спрощує реалізацію аналітичних процедур.

Для доступу до окремих елементів або підмножин векторів використовуються оператори індексації з квадратними дужками. Індексація може здійснюватися за числовими індексами, логічними векторами або символьними іменами. Це дозволяє гнучко керувати структурою даних та виконувати вибіркові операції.

Формування векторів у R здійснюється за допомогою функції c(), яка об’єднує окремі значення або вектори в один об’єкт. Для створення послідовностей використовуються оператори : та функції seq() і rep(), що застосовуються під час генерації даних або формування індексів.

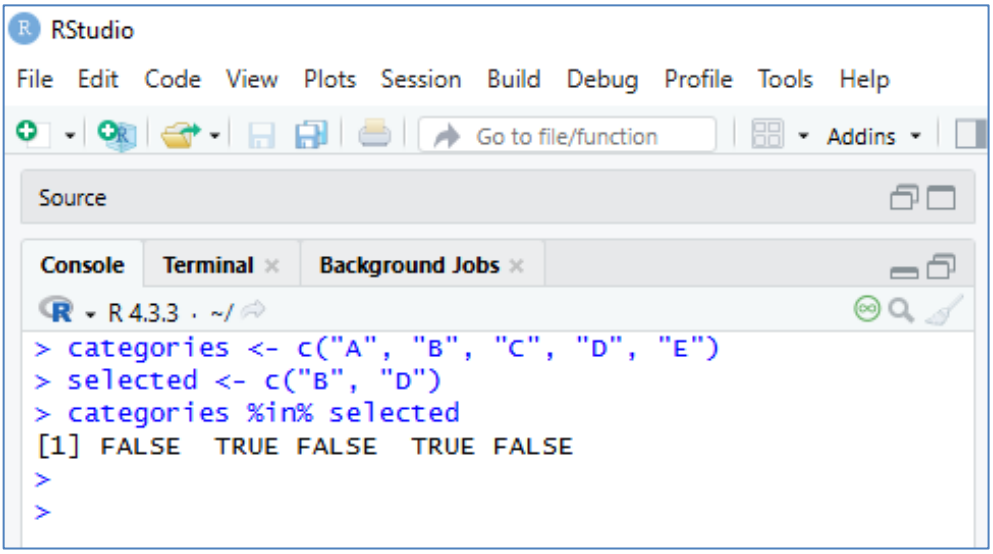
Оператор %in% у мові програмування R використовується для перевірки належності елементів одного вектора до множини значень іншого вектора. Для кожного елемента повертається значення TRUE, якщо він присутній у заданій множині, або FALSE – у протилежному випадку. Такий механізм широко використовується під час фільтрації даних, порівняння списків значень та реалізації умов відбору.

Оператор %in% коректно працює з числовими, символьними та факторними змінними. У разі наявності відсутніх значень NA результат перевірки також може містити NA, що потребує додаткового контролю під час обробки даних. Векторизована природа оператора дозволяє ефективно застосовувати його до великих наборів даних без використання циклів.

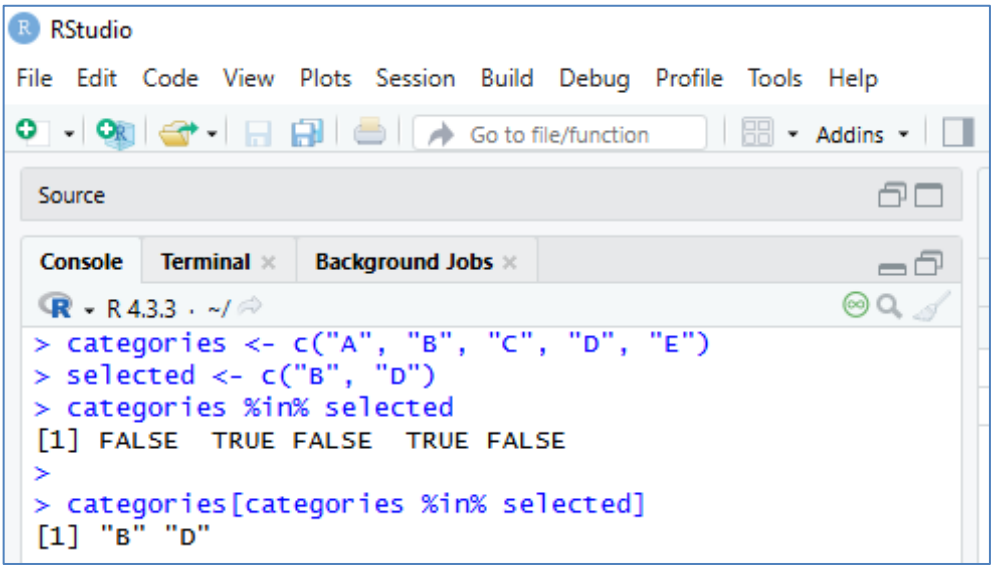
Для прикладу, нехай задано вектор значень, що містить коди товарних категорій, та окремий вектор категорій, які підлягають аналізу.

Застосуємо оператор %in% для перевірки належності кожного елемента вектора categories до множини значень selected:

Результатом виконання виразу буде логічний вектор:



Отриманий логічний вектор може бути використаний для відбору відповідних елементів. У результаті буде сформовано підвектор, що містить лише ті значення, які входять до заданої множини:



2.3.Приклади застосування операторів

Оператори використовуються:

- у фільтрації (filter()),
- у циклах і умовах,
- у статистичних моделях,
- при роботі з великими наборами даних.

Оператори та вирази є фундаментом програмування в R. Глибоке

розуміння арифметичних, логічних, реляційних та спеціалізованих операторів дозволяє ефективно створювати алгоритми, аналізувати дані та будувати статистичні моделі в середовищі RStudio.

Мова R була розроблена з акцентом на статистичні обчислення, але її можливості значно ширші. Вона містить потужний набір вбудованих функцій для виконання будь-яких математичних операцій.

Таблиця 2.4.

Основні математичні функції

Функція	Призначення	Приклад
abs(x)	Абсолютне значення	abs(-10)
sqrt(x)	Квадратний корінь	sqrt(16)
round(x, digits)	Округлення до вказаної кількості знаків	round(3.14159, 2)
ceiling(x)	Найменше ціле число, не менше за x	ceiling(3.14)
floor(x)	Найбільше ціле число, не більше за x	floor(3.14)
trunc(x)	Відкидання дробової частини	trunc(3.14)

Тригонометричні та логарифмічні функції R використовує радіани для всіх тригонометричних функцій.

Таблиця 2.5.

Основні тригонометричні функції

Функція	Призначення	Приклад
sin(x), cos(x), tan(x)	Синус, косинус, тангенс	sin(pi/2)
asin(x), acos(x), atan(x)	Обернені тригонометричні функції	acos(0)
log(x)	Натуральний логарифм ($\ln x$)	log(10)

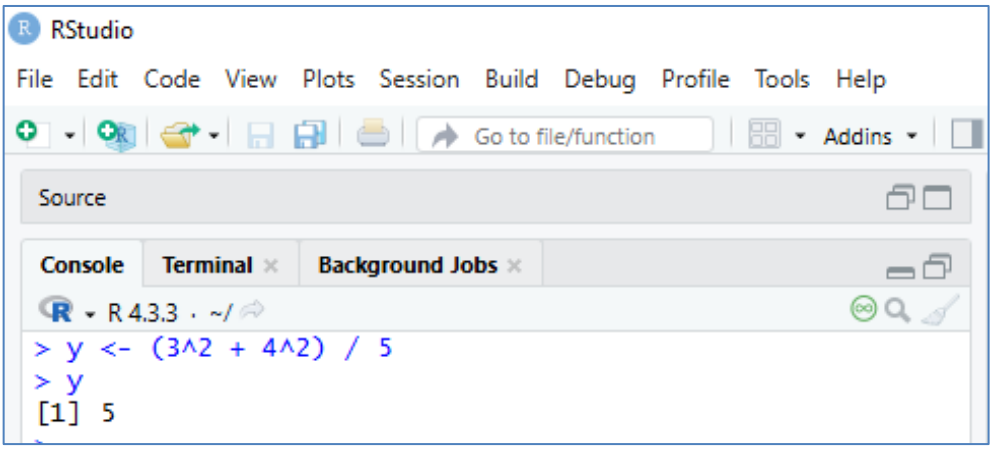
Функція	Призначення	Приклад
log10(x)	Десятковий логарифм ($\log_{10} x$)	log10(100)
log(x, base)	Логарифм за довільною основою	log(8, base = 2)
exp(x)	Експонента (e^x)	exp(1)

Приклад 1. Найпростіші обчислення.

1. Обчислити значення виразу $y = \frac{3^2+4^2}{5}$

Виконання: $y = \frac{3^2+4^2}{5} = \frac{9+16}{5} = \frac{25}{5} = 5$

У середовищі R цей вираз обчислюється так:



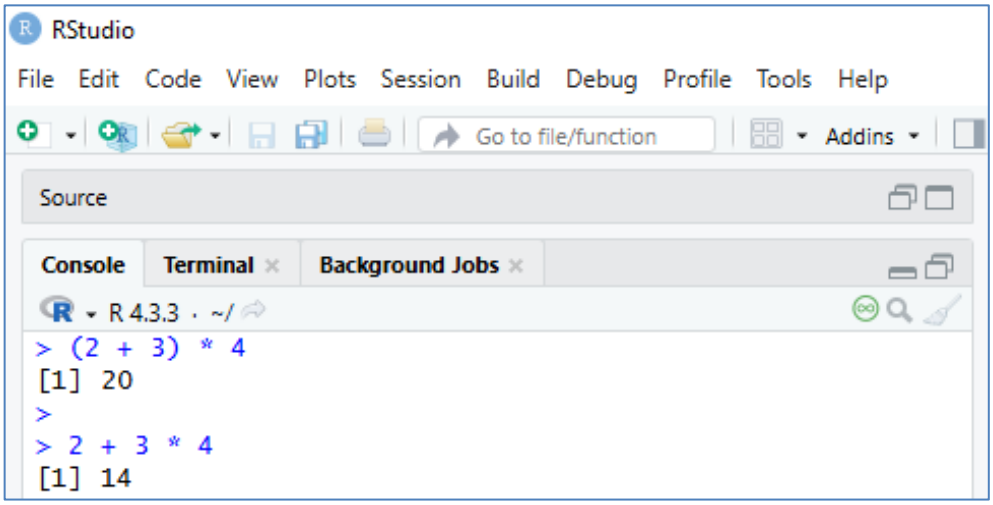
2. Перевірити порядок виконання операцій $(2 + 3) * 4$ та $2 + 3 * 4$

Виконання:

$$(2 + 3) * 4 = 5 * 4 = 20$$

$$2 + 3 * 4 = 2 + 12 = 14$$

R дотримується стандартного пріоритету математичних операцій.



Приклад 2. Використання математичних констант.

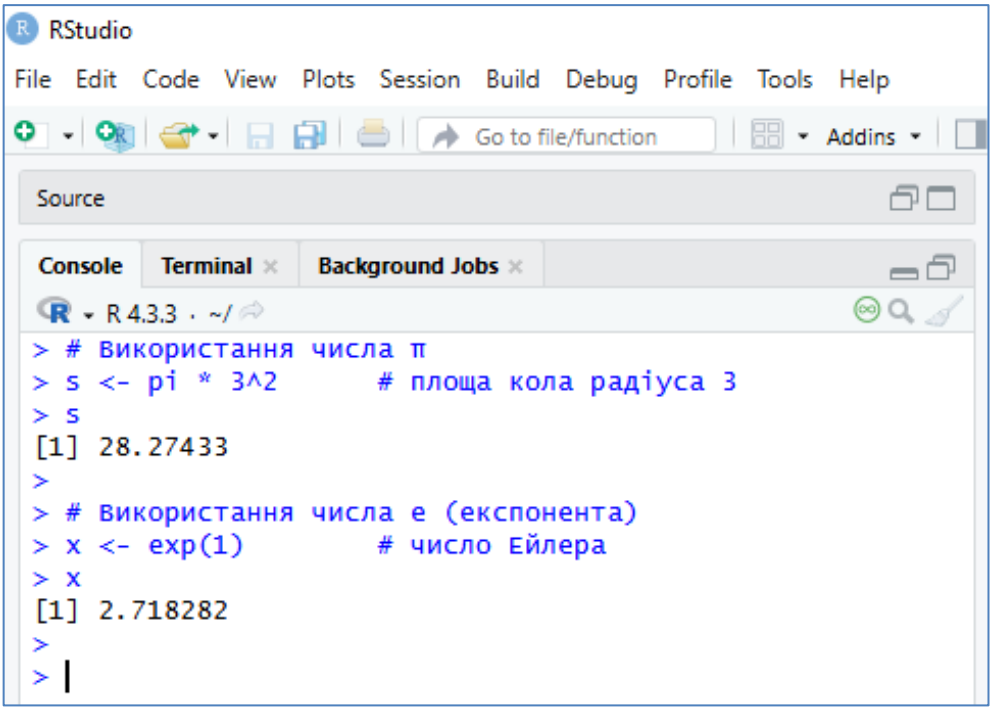
1. Знайти площу кола з радіусом 3
2. Знайти числове значення x, якщо $x = e$

Виконання: Формула площі кола з радіусом 3 матиме вигляд

$$S = \pi * 3^2 = 9\pi = 9 * 3.14 \approx 28.27$$

$$x = e \approx 2.72$$

У мові R доступні вбудовані математичні константи, зокрема число π та число Ейлера.



Приклад 3. Тригонометричні функції.

1. Знайти значення: $\sin\frac{\pi}{6}, \cos\frac{\pi}{6}, \mathrm{tg}\frac{\pi}{6}$

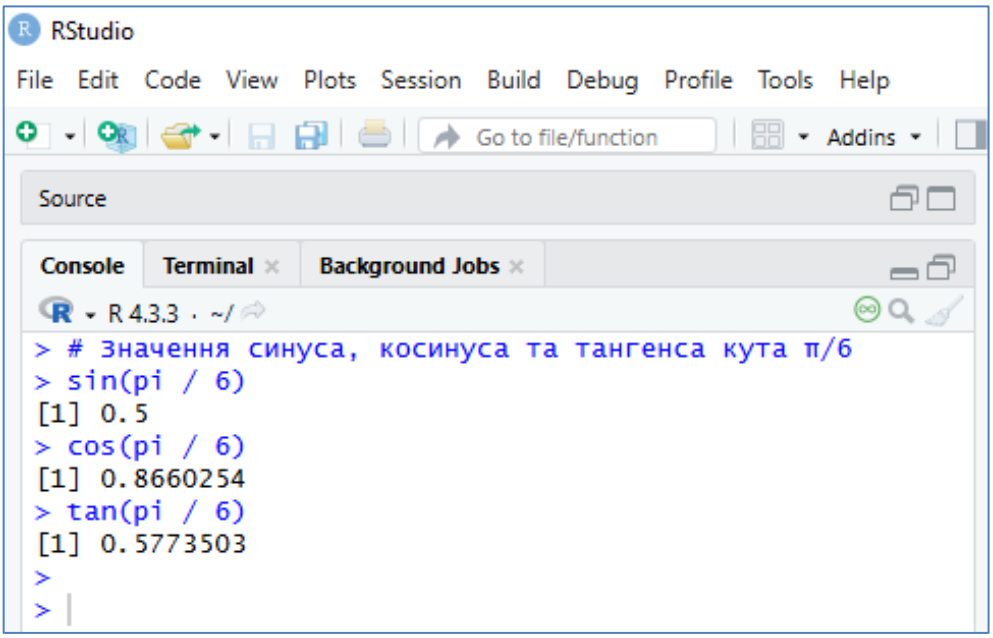
Виконання:

$$\sin\frac{\pi}{6} = \sin 30^\circ = \frac{1}{2} = 0.5$$

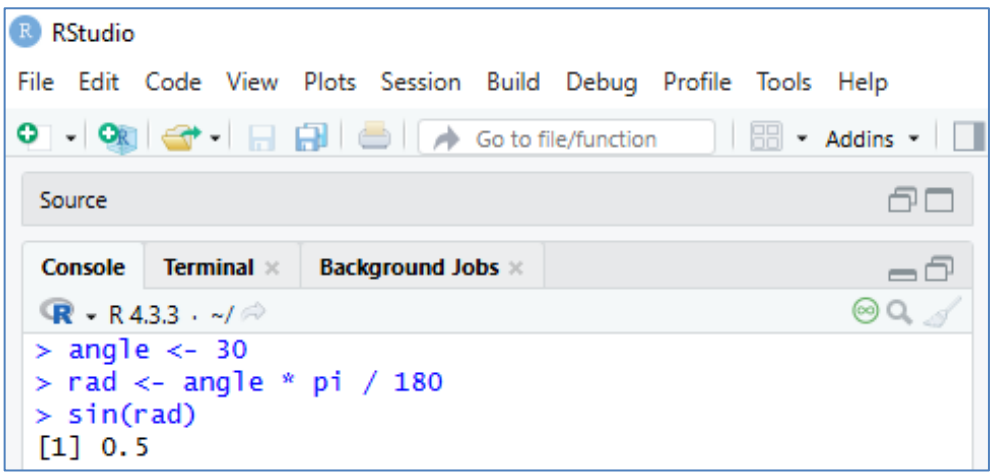
$$\cos\frac{\pi}{6} = \cos 30^\circ = \frac{\sqrt{3}}{2} \approx 0.87$$

$$\mathrm{tg}\frac{\pi}{6} = \mathrm{tg}\, 30^\circ = \frac{\sqrt{3}}{3} \approx 0.58$$

У мові R тригонометричні функції працюють з радіанами, а не з градусами.



Додатковий приклад для $\sin\frac{\pi}{6}$ (переведення градусів у радіани):



Приклад 4. Логарифмічні рівняння.

1. Розв'язати рівняння $log_2 x + log_2(x-2) = 3$

Виконання:

Використаємо властивість логарифмів:

$$log_2(x(x-2)) = 3$$
$$x(x-2) = 2^3$$
$$x^2 - 2x - 8 = 0$$

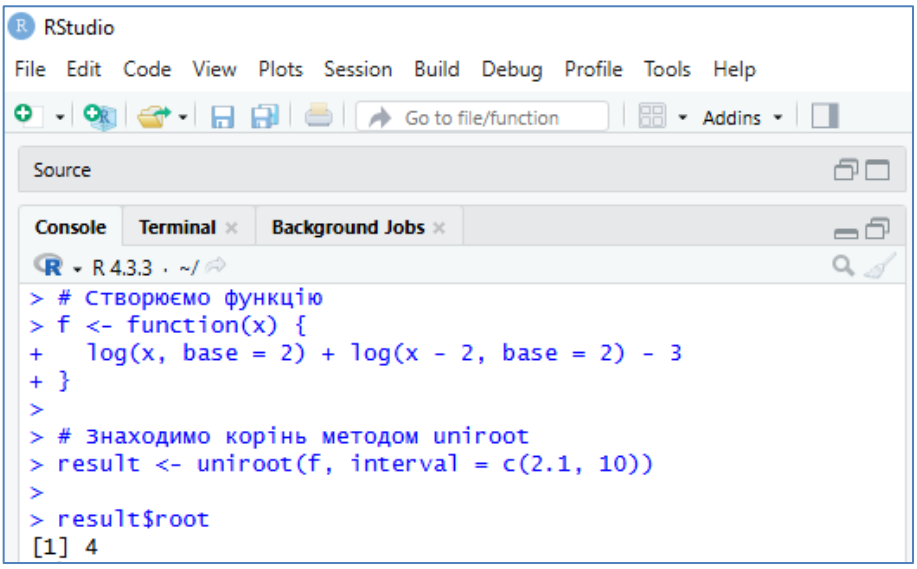
Розв'язок квадратного рівняння $x_1 = 4,\ x_2 = -2$

З урахуванням області допустимих значень:

$$x > 0$$
$$x - 2 > 0 \rightarrow x > 2$$

Отже, правильний розв'язок: $x = 4$

У R логарифм за основою 2 записується як log(x, base = 2)



interval = c(2.1, 10) – вибираємо проміжок з урахуванням ОДЗ (x > 2)

uniroot() – чисельний метод знаходження кореня

result$root – значення розв'язку

Приклад 5. Обчислення з векторами.

1. Для вектора $X = (1,2,3,4)$ знайти вектор Y, елементи якого обчислюються за формулою $Y = X^2 + 5\sin X$

Виконання:

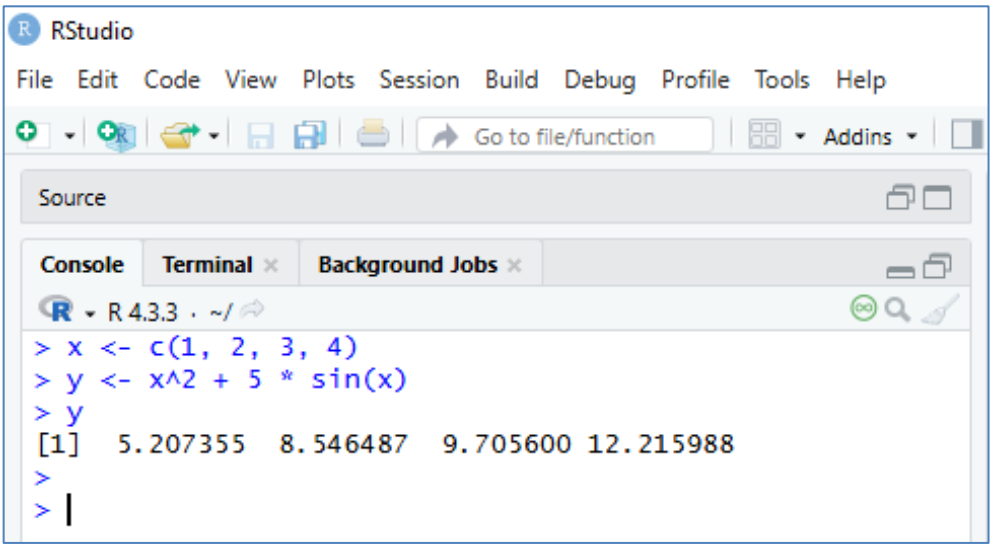
при $x = 1$ отримуємо $y = 1^2 + 5 * \sin(1) = 1 + 5 * 0.84 \approx 5.2$

при $x = 2$ отримуємо $y = 2^2 + 5 * \sin(1) = 4 + 5 * 0.9 \approx 8.5$

при $x = 3$ отримуємо $y = 3^2 + 5 * \sin(3) = 9 + 5 * 0.14 \approx 9.7$

при $x = 4$ отримуємо $y = 4^2 + 5 * \sin(4) = 16 + 5 * (-0.76) \approx 12.2$

У мові R арифметичні та математичні операції виконуються векторизовано, тобто застосовуються до кожного елемента вектора автоматично.



Приклад 6. Розв'язання систем лінійних рівнянь

1. Використовуючи матричні операції та функцію solve(), знайти розв'язок системи лінійних рівнянь $\begin{cases} 2x + 3y = 8 \\ 4x - y = 2 \end{cases}$

Виконання:

Запишемо систему у матричному вигляді $A * X = b$, де $A = \begin{pmatrix} 2 & 3 \\ 4 & -1 \end{pmatrix}$, $X = \begin{pmatrix} x \\ y \end{pmatrix}$, $b = \begin{pmatrix} 8 \\ 2 \end{pmatrix}$

Знайдемо визначник матриці A

$$detA = \begin{vmatrix} 2 & 3 \\ 4 & -1 \end{vmatrix} = 2 * (-1) - 3 * 4 = -14$$

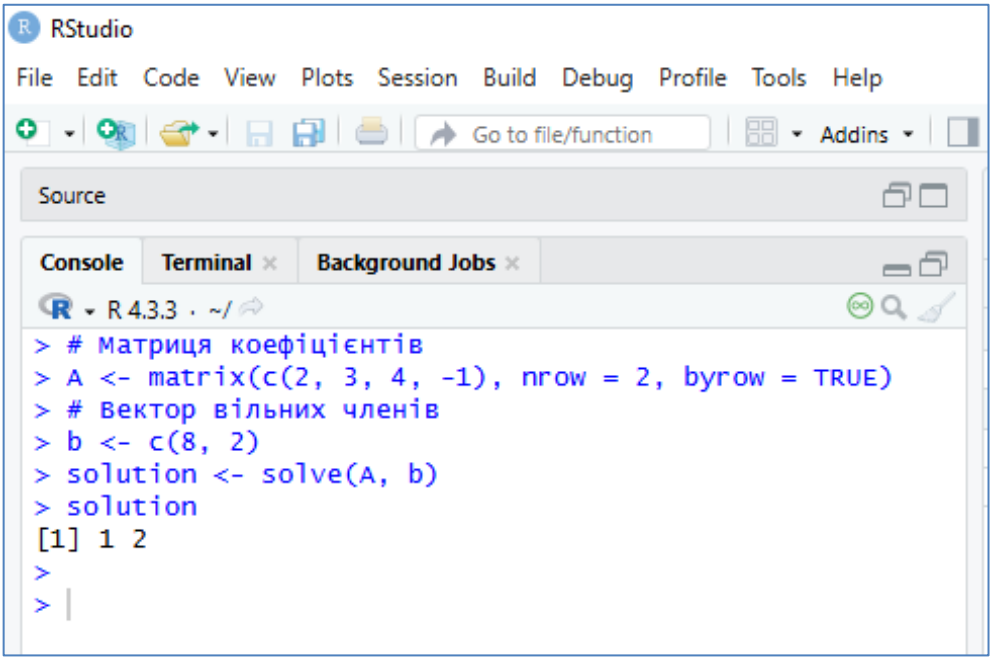
Оскільки $detA \neq 0$ система має єдиний розв'язок.

Обчислимо обернену матрицю $A^{-1} = \frac{1}{detA} \begin{pmatrix} -1 & -3 \\ -4 & 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{14} & \frac{3}{14} \\ \frac{4}{14} & -\frac{2}{14} \end{pmatrix}$

Знайдемо вектор розв'язку $X = A^{-1} * b$

$$X = \begin{pmatrix} \frac{1}{14} & \frac{3}{14} \\ \frac{4}{14} & -\frac{2}{14} \end{pmatrix} * \begin{pmatrix} 8 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{8}{14} + \frac{6}{14} \\ \frac{32}{14} - \frac{4}{14} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Функція solve() у мові R дозволяє ефективно розв'язувати системи лінійних рівнянь, подані в матричній формі, що є зручним для задач лінійної алгебри та статистичного моделювання.



Приклад 7. Розв'язання нелінійних рівнянь

1. Знайти розв'язок рівняння $x^3 - 2x - 5 = 0$ на інтервалі $(2; 3)$.

Виконання:

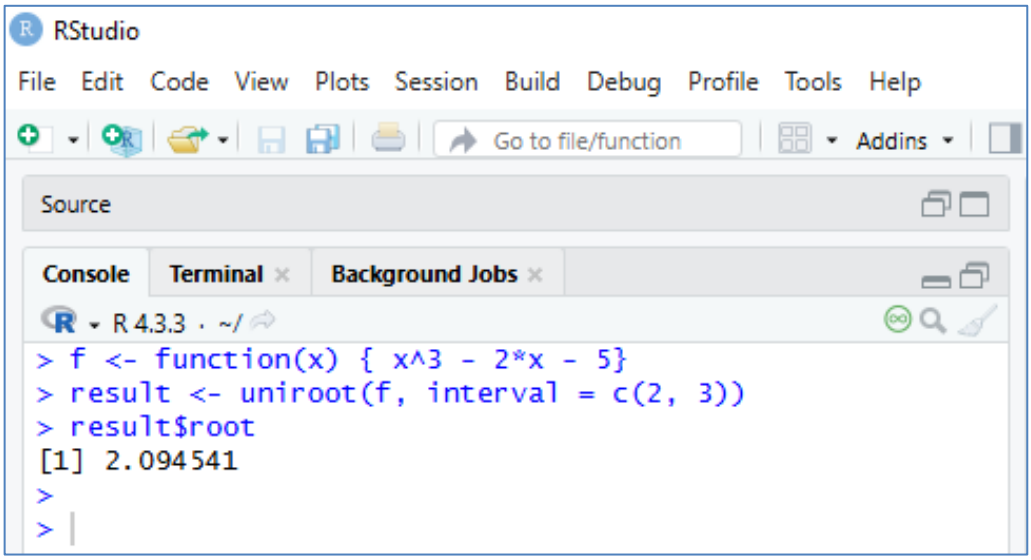
Перевіримо значення функції в точках:

при $x = 2$ отримаємо $f(2) = 2^3 - 2 * 2 - 5 = -1$

при $x = 3$ отримаємо $f(3) = 3^3 - 2 * 3 - 5 = 16$

Оскільки $f(2) < 0$, а $f(3) > 0$, то за теоремою про неперервність існує дійсний корінь на інтервалі $(2; 3)$.

Для розв'язання нелінійних рівнянь в R використовується функція uniroot(), яка шукає корінь (точку, де функція дорівнює нулю) в заданому інтервалі. Отримане значення root є наближеним коренем рівняння. Значення f.root показує, наскільки близько результат до нуля, а estim.prec – оцінює точність обчислень.



Цей метод є ефективним для розв'язання нелінійних рівнянь, які не мають простого аналітичного розв'язку.

Інтегровані завдання до розділу 2

Завдання 1. Базові обчислення та константи

1. Обчисліть значення виразу: $E = \frac{15^3+\sqrt{256}}{1.5*\pi}$
2. Створіть змінну radius зі значенням 5. Обчисліть площу кола $S = \pi r^2$ та збережіть результат у змінну area.
3. Визначте остачу від ділення числа 100 на 7.
4. Обчисліть цілу частину від ділення числа 100 на 7.

Завдання 2. Тригонометричні функції

1. Обчисліть значення $\sin\left(\frac{\pi}{4}\right), \cos\left(\frac{\pi}{3}\right), \tan\left(\frac{\pi}{6}\right)$
2. Створіть вектор angles_deg зі значеннями кутів у градусах $\left(0, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}\right)$
3. Перетворіть ці кути в радіани, використовуючи формулу: радіани = градуси $* \frac{\pi}{180}$. Збережіть результат у вектор angles_rad.
4. Обчисліть синус кожного кута з вектора angles_rad за допомогою векторизації.

Завдання 3. Логарифмічні та експоненційні функції

1. Обчисліть $ln(e^2)$ (використовуйте функцію exp()).
2. Обчисліть $\lg(1000)$.
3. Обчисліть $log_4(64)$.
4. Обчисліть значення виразу: $Z = ln(100) + log_{10}(50) - e^3$.
5. Створіть вектор $X = (1,2,3,4,5)$. Обчисліть вектор $Y = e^{2X} - lg(X)$.

Завдання 4. Розв'язання нелінійних рівнянь

1. Розв’яжіть рівняння $x^3 - 5x + 1 = 0$ на інтервалі $(-3; 3)$., використавши функцію uniroot().

Завдання 5. Розв'язання систем лінійних рівнянь

2. Використовуючи матричні операції та функцію solve(), знайдіть розв'язок системи лінійних рівнянь

$$\begin{cases} 2x + y = 5 \\ x - y = 1 \end{cases}$$

Завдання 6. Розв'язання нелінійних систем рівнянь

Знайти розв’язок нелінійної системи рівнянь:

$$\begin{cases} x^2 + y^2 = 8 \\ x - y = 0 \end{cases}$$

Контрольні питання до розділу 2

1. Для чого використовується тип даних character?
2. У чому перевага матричного методу розв’язування систем рівнянь?
3. У чому полягає відмінність між числовими типами numeric та integer?
4. У чому полягає принцип векторизації обчислень у мові R? Наведіть приклад, як векторизація спрощує обчислення порівняно з традиційним програмуванням (наприклад, у Python або C++).
5. У чому різниця між операторами == та !=?
6. У яких одиницях R очікує аргументи для тригонометричних функцій (sin, cos)? Як перевести градуси в ці одиниці?
7. Що таке вираз у мові R та з яких елементів він складається?

8. Як оператори порівняння та логічні оператори застосовуються до векторів у R?

9. Як у R здійснюється присвоєння значень змінним?

10. Яка функція в R використовується для пошуку коренів нелінійних рівнянь? Чому для цієї функції необхідно вказувати інтервал пошуку?

11. Який порядок виконання арифметичних операцій у виразах мови R?

12. Який результат повертають оператори порівняння в мові R?

13. Які арифметичні операції можна виконувати безпосередньо над векторами?

14. Які логічні оператори застосовуються в R та їх призначення?

15. Які оператори порівняння використовуються в R?

16. Які основні типи даних використовуються в мові R та їх призначення?

17. Які особливості логічного типу даних logical у R?

18. Яку роль відіграють дужки під час формування складних виразів?

РОЗДІЛ 3. ТИПИ ТА СТРУКТУРА ДАНИХ У R

3.1.Поняття даних та їх роль у мові програмування R

Дані є основним об’єктом опрацювання в будь-якій мові програмування, зокрема в мові R, яка орієнтована на статистичний аналіз, обробку та візуалізацію даних. Під даними розуміють формалізовану інформацію, подану у вигляді значень, символів або структур, придатних для зберігання, обробки та аналізу за допомогою комп’ютерних засобів.

У середовищі R всі дані представлені у вигляді об’єктів. Кожен об’єкт має визначений тип і структуру, що зумовлює спосіб його збереження в пам’яті та набір допустимих операцій над ним. Коректний вибір типу та структури даних є важливою умовою ефективного виконання обчислень, точності результатів і оптимального використання ресурсів.

Мова програмування R підтримує різноманітні типи даних, які дозволяють працювати з числовою, логічною, текстовою та іншою інформацією. На основі базових типів формуються складніші структури даних, такі як вектори, матриці, списки та таблиці даних (data frame), що забезпечують гнучке представлення реальних наборів даних.

3.2.Базові типи даних у R

Базові типи даних у мові програмування R визначають формат збереження інформації та характер операцій, які можуть бути виконані над нею. Вони є фундаментом для побудови складніших структур даних і широко використовуються під час виконання статистичних обчислень та аналізу даних. Перед початком детального розгляду окремих типів даних доцільно зазначити, що мова програмування R оперує чітко визначеною системою базових типів, які визначають спосіб подання, зберігання та обробки інформації. Тип даних задає характер значень, які може набувати змінна, а також перелік допустимих операцій над цими значеннями. Розуміння базових типів даних є необхідним для коректного створення об’єктів, уникнення помилок під час обчислень і

ефективної роботи з різними структурами даних. Саме на основі базових типів у R формуються складніші об’єкти, що використовуються для аналізу та інтерпретації даних у статистичних і прикладних задачах.

Спеціальні значення у мові R також відіграють важливу роль при роботі з даними. До них належать NA (відсутнє значення), NaN (нечисловий результат), Inf та -Inf (нескінченність). Ці значення дозволяють коректно обробляти пропущені або некоректні дані під час аналізу.

3.2.1.Числовий тип (numeric)

Тип numeric – об’єкти даного класу діляться на цілочислові (integer) і дійсні (double); complex – об’єкти комплексного типу. За замовчуванням усі числові значення в R належать саме до цього типу. Числові дані застосовуються в арифметичних обчисленнях, статистичних показниках, побудові моделей та графіків.

```
a <- 5
b <- 3.14
```

Характеристики:

- підтримує математичні операції;
- може містити дробові значення;
- використовується у розрахунках, статистиці, аналізі даних.

```
class(a)  # "numeric"
```

3.2.2.Символьний тип (character)

Тип character призначений для зберігання текстових даних (рядків, символьні змінні задаються або в подвійних лапках, або в одинарних).

```
name <- "Тарас"
city <- "Рівне"
```

Характеристики:

- текст береться в лапки;
- не підтримує арифметичні операції;

- використовується для імен, назв, описів.

class(name) # "character"

3.2.3.Логічний тип (logical)

Тип logical використовується для логічних значень:

- TRUE
- FALSE

passed <- TRUE

Характеристики:

- застосовується в умовних конструкціях;
- результат порівнянь;
- основа для фільтрації даних.
- x > 3   # повертає TRUE або FALSE

3.2.4.Категоріальниий тип (factor)

Тип factor використовується для подання категоріальних (якісних) даних, тобто змінних, значення яких належать до обмеженого, наперед визначеного набору категорій або рівнів. Такий тип даних є особливо важливим у статистичному аналізі, оскільки дозволяє коректно працювати з номінальними та порядковими змінними.

Об’єкти типу factor зберігають значення у вигляді рівнів (levels), кожному з яких відповідає певний код. Це забезпечує ефективне використання пам’яті та дає змогу застосовувати спеціальні статистичні методи, зокрема під час побудови регресійних моделей, аналізу дисперсії та візуалізації даних.

Фактори можуть бути невпорядкованими (nominal), коли між категоріями відсутній природний порядок, та впорядкованими (ordered), коли значення мають логічну послідовність. Використання типу factor дозволяє уникнути помилок інтерпретації даних і підвищує точність статистичних висновків.

gender <- factor(c("чоловік", "жінка", "жінка"))

Характеристики:

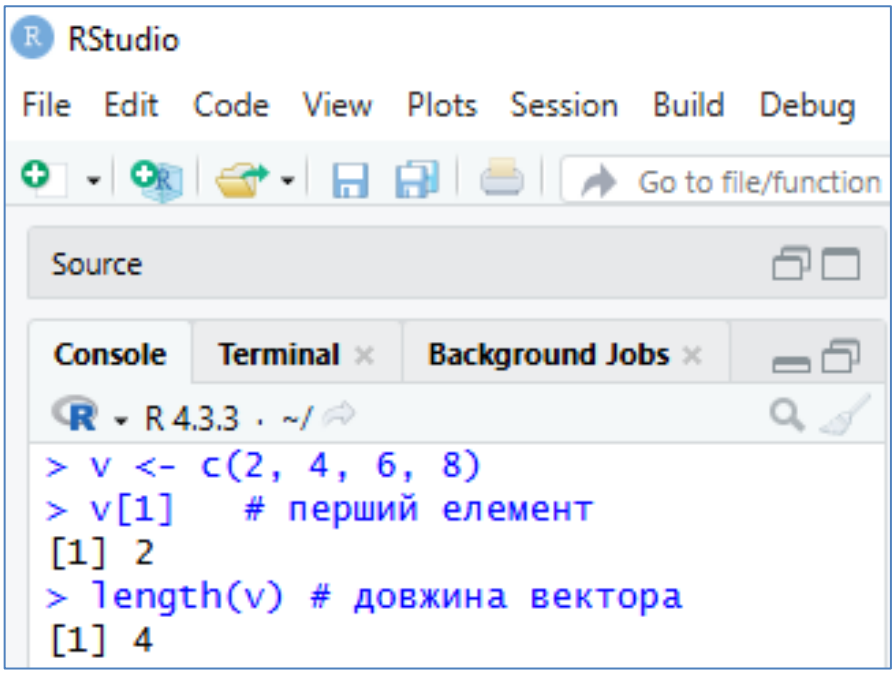
- має рівні (levels);
- оптимізує зберігання повторюваних значень;
- використовується в регресіях, групуваннях, ANOVA.

3.3.Структура даних в R.

У мові програмування R дані організовуються у спеціальні структури даних, які визначають спосіб зберігання, доступу та обробки інформації. Вибір правильної структури даних є важливим поцесом для ефективного аналізу, статистичних обчислень і програмування. До основних структур даних у R належать: вектори, матриці, списки, data.frame, tibble.

3.3.1.Вектори як базова структура даних у R

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



Основні властивості:

- усі елементи одного типу (numeric, character, logical);
- індексація починається з 1;
- базова структура для більш складних об’єктів.

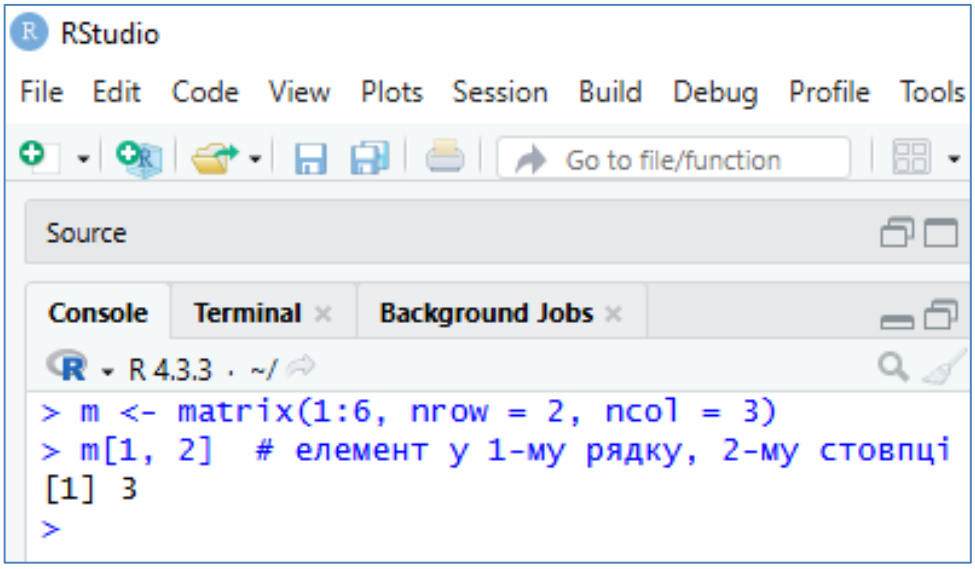
v[1] # перший елемент

length(v) # довжина вектора

3.3.2.Матриці та масиви даних

Матриця – це двовимірна структура даних, що містить елементи одного типу, розміщені у вигляді рядків і стовпців.

Створення матриці:



Основні властивості:

- усі значення одного типу;
- має розміри dim(m);
- підтримує матричні операції.

m[1, 2] # елемент у 1-му рядку, 2-му стовпці

Матриці активно використовуються у лінійній алгебрі та статистиці**.**

У R багатовимірний масив (array) – це базова структура даних для зберігання однотипних елементів (одного типу: numeric, logical тощо), організованих у двох і більше вимірах. Це природне узагальнення вектора та матриці.

Array - це вектор з атрибутом dim; може мати 2, 3, 4 і більше вимірів;

всі елементи мають однаковий тип; тензор, що зберігає числові дані у багатьох вимірах і підтримує ефективні векторизовані обчислення.

Порівняння:

vector → 1 вимір

matrix → 2 виміри

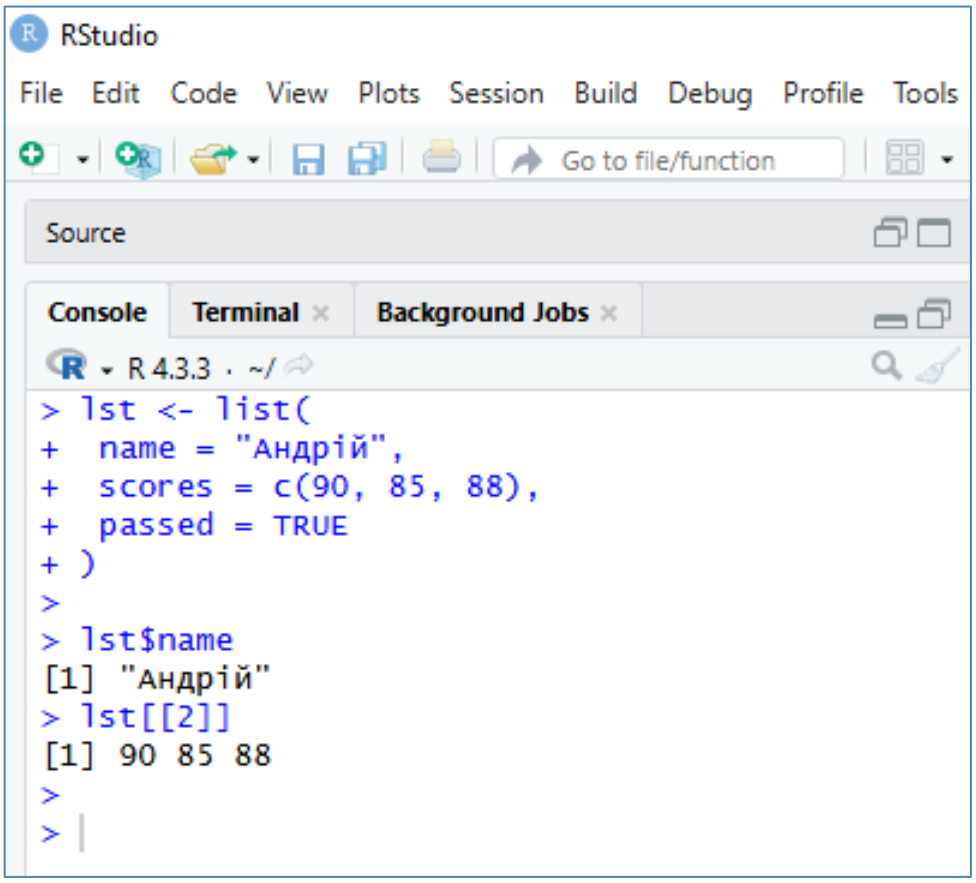
array → 2+ виміри

a <- array(1:12, dim = c(3, 4))

3.3.3.Списки (list) та їх призначення

Список (list) – це колекція об’єктів, доступ до яких можна здійснити по номеру або імені. Список – це структура даних, яка може містити елементи різних типів і різних структур.

Створення списку:



Основні властивості:

- може містити вектори, матриці, data.frame, функції;
- доступ до елементів за ім’ям або індексом.

lst$name

lst[[2]]

Списки широко застосовуються для збереження результатів моделей і складних обчислень.

3.3.4.Таблиці даних (data frame) та їх особливості

Таблиці даних (data frame) призначені для зберігання та обробки табличної інформації, яка за своєю структурою відповідає класичним таблицям баз даних або електронних таблиць.

Таблиця даних складається з рядків і стовпців, де кожен стовпець представляє окрему змінну, а кожен рядок – окреме спостереження. Важливою особливістю data frame є те, що всі значення в межах одного стовпця мають бути одного типу даних, тоді як різні стовпці можуть містити дані різних типів (числові, логічні, символьні, факторні тощо).

До основних особливостей таблиць даних належать:

– можливість зберігання різнорідних типів даних у межах однієї структури;

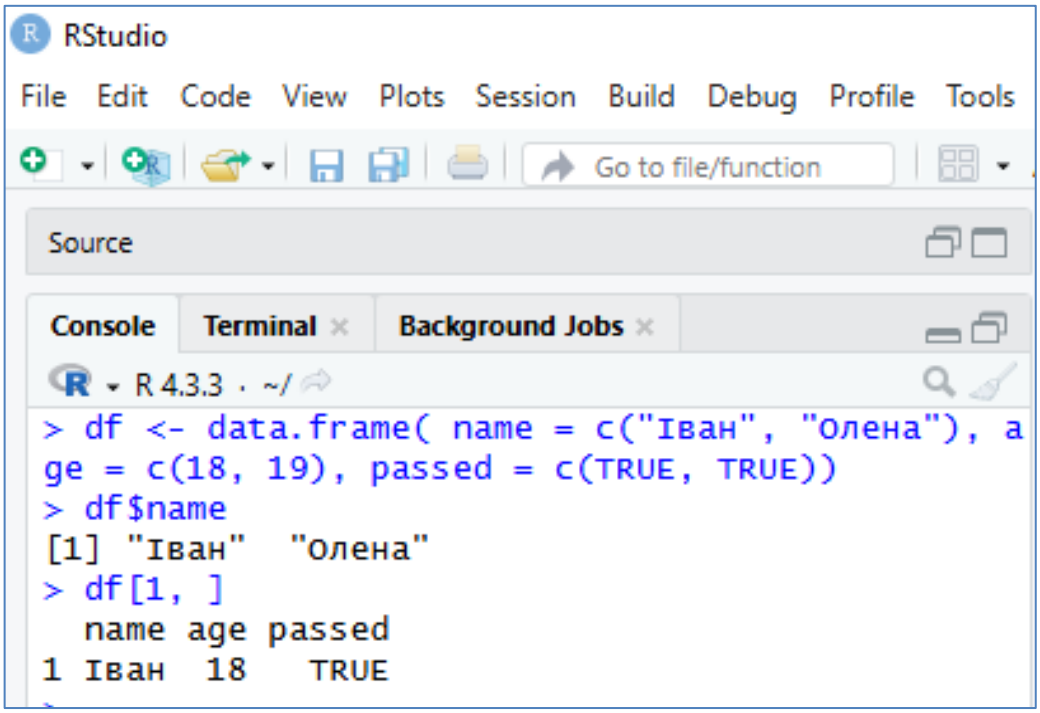
– наявність імен стовпців та, за потреби, імен рядків, що спрощує доступ до даних;

– зручний механізм вибірки, фільтрації та сортування даних;

– тісна інтеграція з функціями статистичного аналізу та візуалізації.

Таблиці даних широко використовуються для імпорту інформації з зовнішніх джерел (CSV-файлів, електронних таблиць, баз даних), підготовки даних до аналізу, виконання статистичних обчислень і побудови аналітичних моделей. Завдяки своїй універсальності data frame є базовою структурою для роботи з реальними наборами даних у середовищі R.

Створення data.frame:



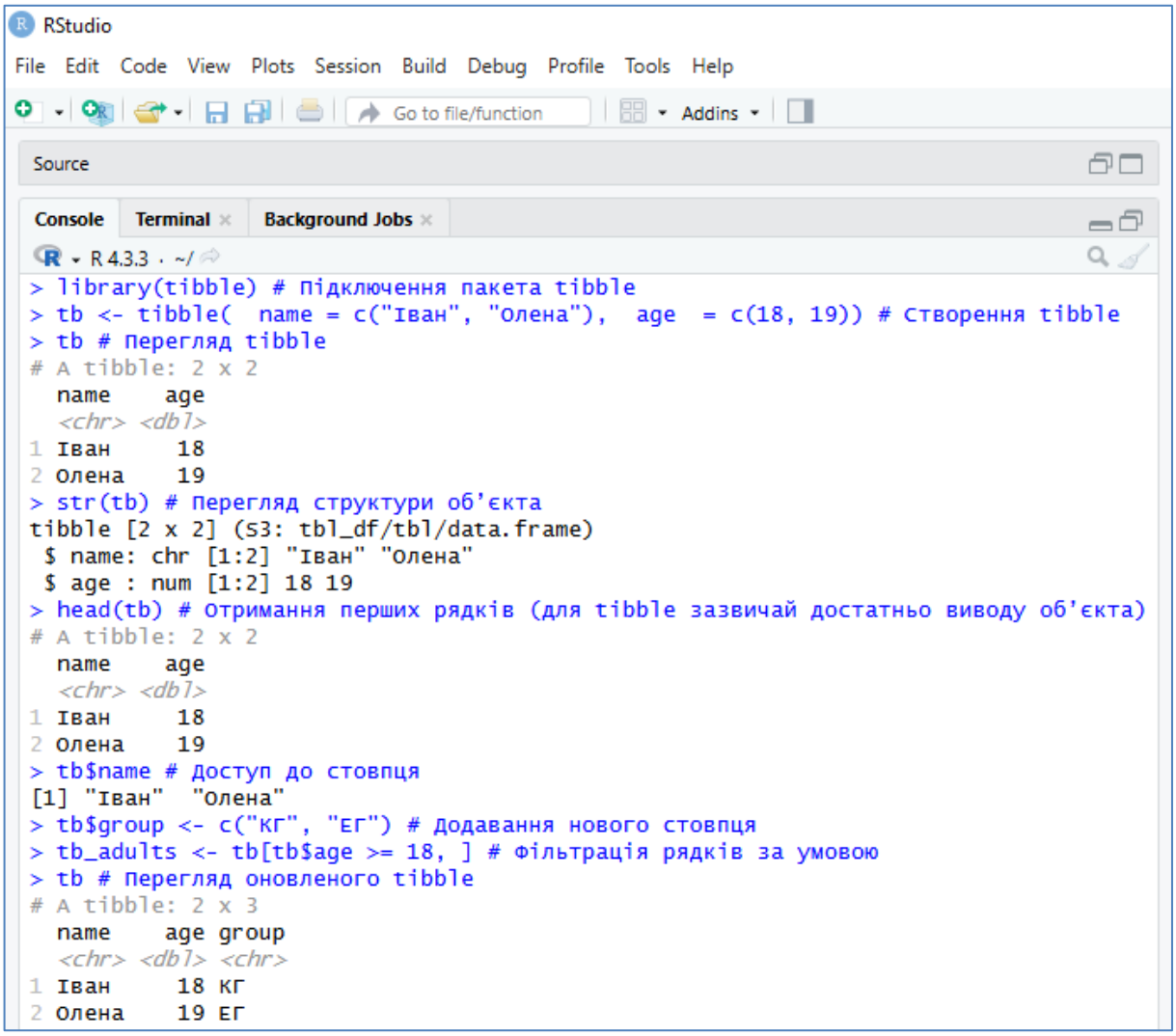
Сучасна вдосконалена форма таблиці даних (data frame) – це tibble, яка використовується у пакеті tidyverse мови програмування R. Вона зберігає всі основні властивості класичного data frame, але має низку додаткових особливостей, що роблять роботу з даними зручнішою та безпечнішою.

Tibble особливо ефективний при роботі з великими наборами даних, у поєднанні з пакетами dplyr, ggplot2 та іншими інструментами tidyverse. Завдяки своїй простоті та прозорості, tibble широко застосовується в сучасному аналізі даних і є рекомендованою формою подання табличної інформації в R.

Основні переваги tibble:

- не змінює типи даних автоматично (наприклад, текст не перетворюється у фактор без явного зазначення);
- під час виведення на екран показує лише перші рядки та стовпці, що запобігає перевантаженню інформацією;
- підтримує зручну роботу з довгими іменами стовпців;
- не використовує імена рядків, що сприяє кращій організації даних;
- забезпечує більш передбачувану поведінку під час вибірки та маніпуляцій з даними.

Створення tibble:



3.3.5.Спеціальні змінні в R

Спеціальні змінні в R – це вбудовані об’єкти та значення, які мають фіксоване призначення в мові R та використовуються для представлення відсутніх, невизначених або нескінченних даних, а також для доступу до службової інформації під час виконання програм.

Найпоширеніші спеціальні змінні:

Inf – нескінченність: додатна (Inf) і від’ємна (-Inf).

NA – відсутнє значення (Not Available).

NaN – не число (Not a Number).

NULL – ніщо.

is.finite(ім’я_об’єкта), is.nan(ім’я_об’єкта), is.na(ім’я_об’єкта), is.null(ім’я_об’єкта). функції для перевірки об’єктів на приналежність до конкретного типу даних.

3.4.Механізм індексації в R

Механізм індексації в мові програмування R визначає способи доступу до окремих елементів, підмножин або структурних частин об’єктів даних. Індексація є базовим інструментом роботи з векторами, матрицями, списками та таблицями даних і забезпечує гнучке керування інформацією.

У мові R використовується індексація, що починається з одиниці, тобто перший елемент будь-якого об’єкта має індекс 1. Це є однією з принципових відмінностей R від багатьох інших мов програмування.

Основними способами індексації в R є:

– Числова індексація, за якої доступ до елементів здійснюється за їх позицією. Можна вказувати один або декілька індексів, а також використовувати від’ємні значення для виключення певних елементів.

– Логічна індексація, що базується на логічних векторах (TRUE/FALSE) і дозволяє відбирати лише ті елементи, які задовольняють задану умову.

– Індексація за іменами, яка застосовується до об’єктів, що мають імена (наприклад, вектори, списки, data frame), і забезпечує зручний та наочний доступ до даних.

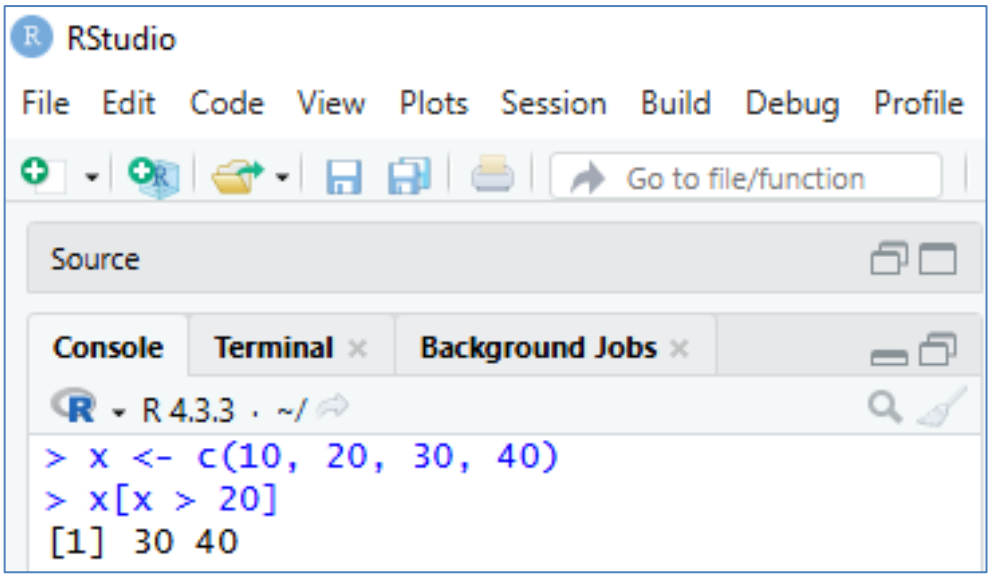
Для роботи зі складними структурами даних у R використовуються спеціальні оператори індексації. Квадратні дужки [] застосовуються для вибірки підмножин даних, подвійні квадратні дужки [[]] – для доступу до окремих елементів списків, а оператор $ – для звернення до іменованих компонентів об’єктів.

Механізм індексації в R є універсальним і потужним засобом, що дозволяє ефективно виконувати вибірку, фільтрацію та модифікацію даних. Його правильне застосування є важливою умовою коректної обробки інформації та реалізації алгоритмів аналізу даних.

3.4.1.Оператори доступу та індексації

Оператор [] – вибір підмножини елементів. Використовується для доступу до одного або кількох елементів об’єкта.

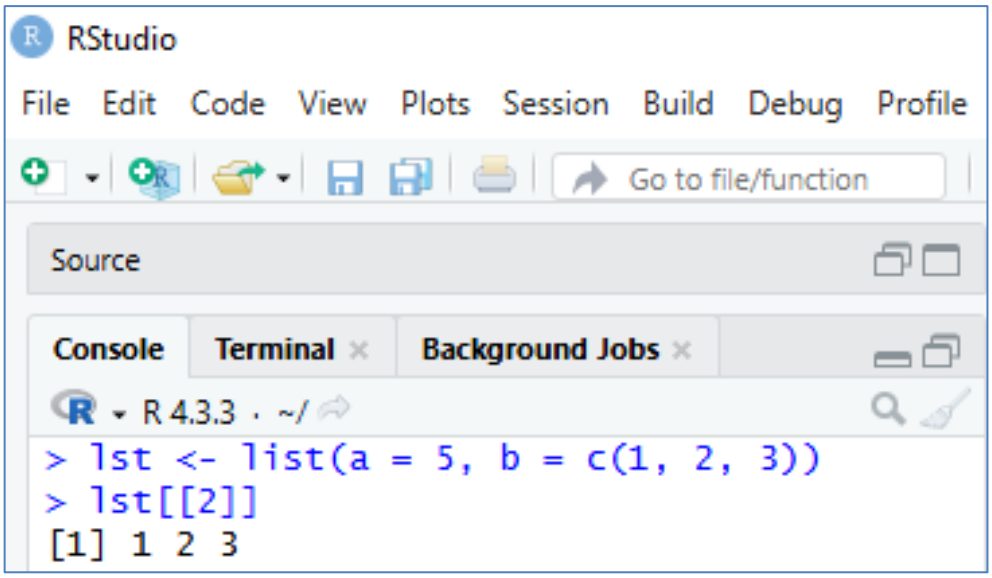
Логічна індексація за допомогою []:



Логічна індексація за допомогою [] – спосіб вибірки елементів в об’єктах R (векторах, матрицях, data frame) на основі логічних умов, за якого повертаються лише ті елементи, для яких відповідний логічний вираз має значення TRUE.

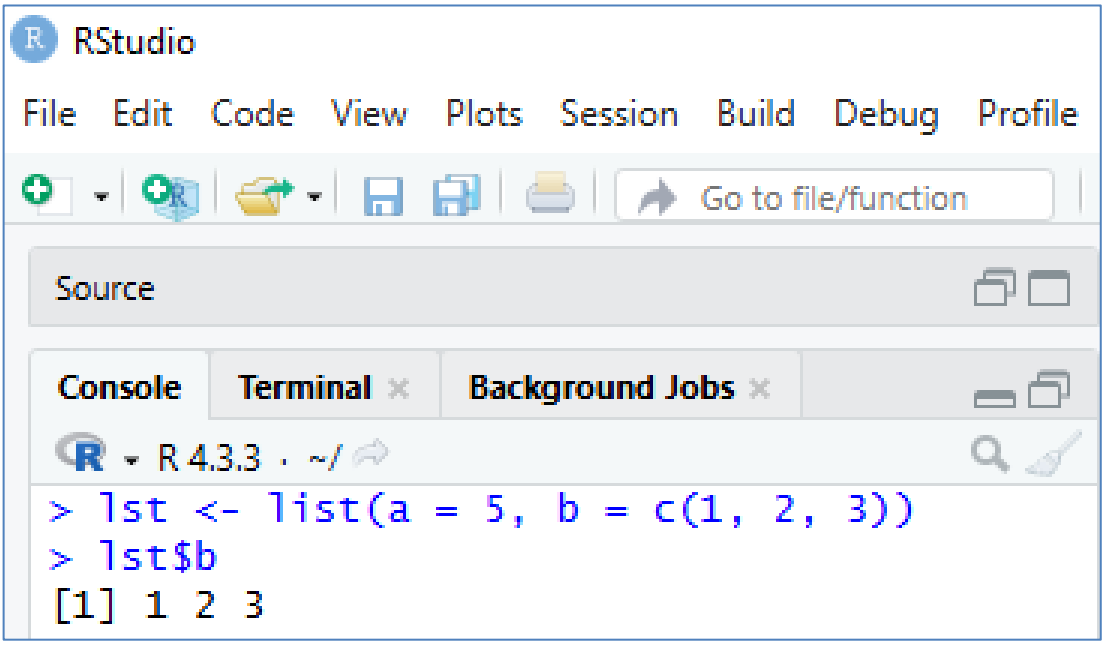
При логічній індексації використовується логічний вектор тієї ж довжини, що й об’єкт, або вираз порівняння, який автоматично формує такий вектор; елементи з FALSE або NA не включаються до результату.

Оператор [[]] – доступ до окремого елемента списку:



Оператор [[ ]] – засіб доступу до окремого елемента списку або іншого складеного об’єкта в R, який повертає сам елемент, а не підсписок; використовується для отримання значень за індексом або ім’ям без збереження структури контейнера.

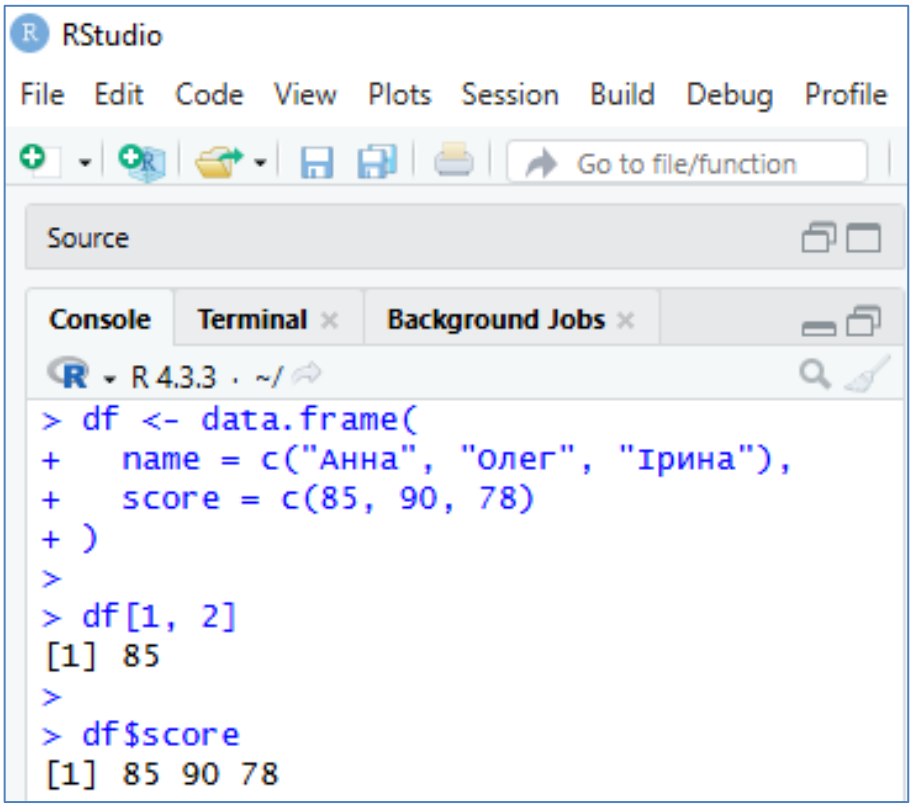
Оператор $ – доступ до іменованих компонентів:



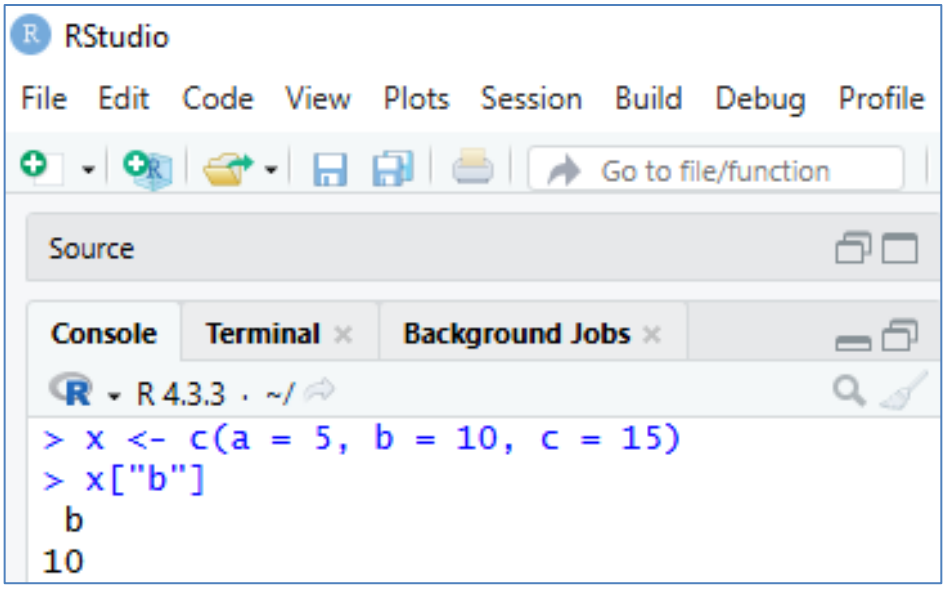
Оператор $ – засіб доступу до іменованих компонентів списків, data frame та інших складених об’єктів у R, який дозволяє звертатися до елемента за його ім’ям у зручній скороченій формі та повертає відповідне значення без збереження структури контейнера.

Індексація таблиці даних (data frame) – спосіб доступу, вибірки та зміни рядків і стовпців таблиці даних у R за допомогою операторів індексації, який ґрунтується на використанні числових, логічних або іменованих індексів і забезпечує гнучке опрацювання табличних структур.

Індексація таблиці даних (data frame):

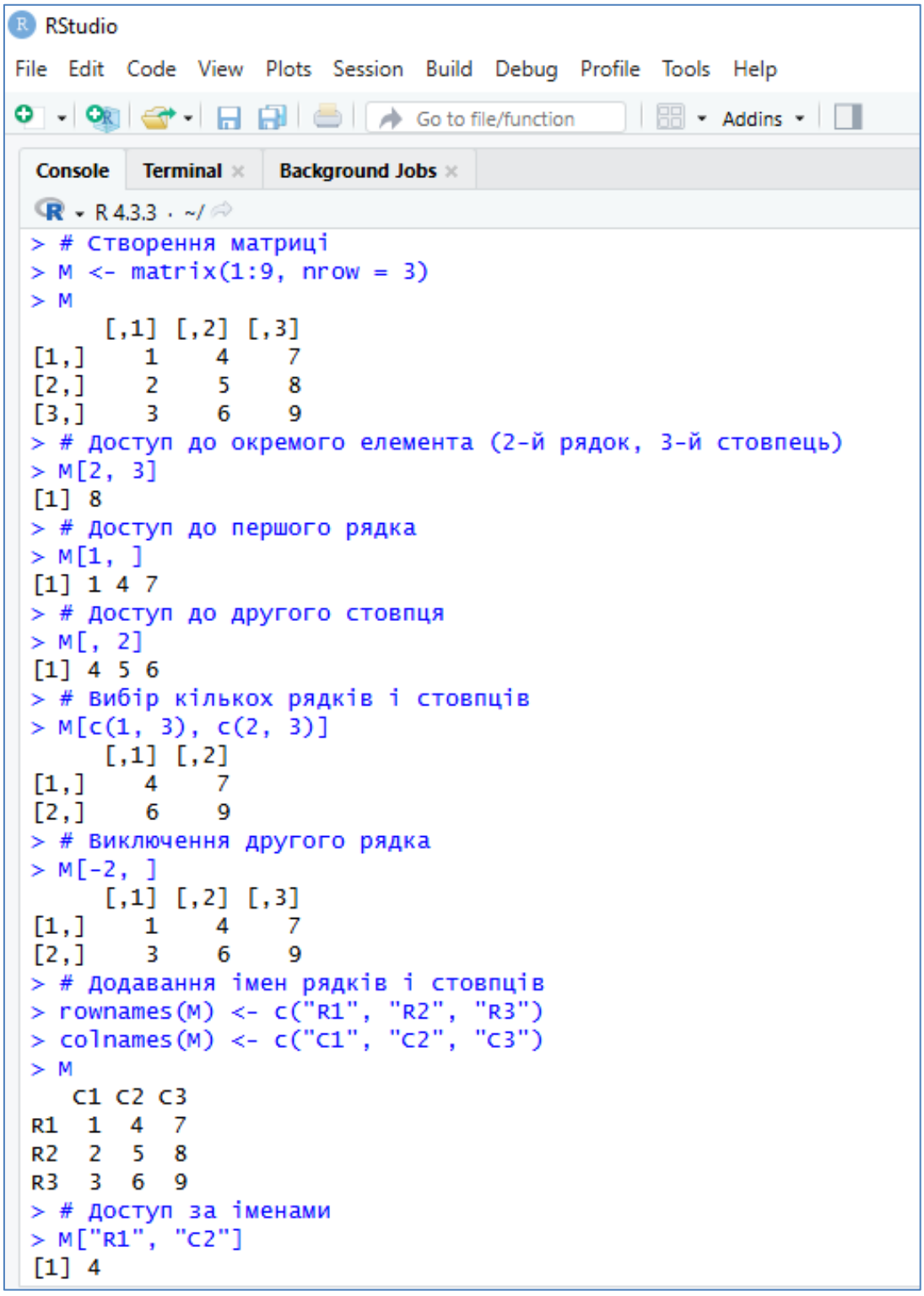


Індексація за іменами:



Індексація за іменами – спосіб доступу до елементів об’єктів R (векторів, списків, data frame) шляхом звернення до них за іменами компонентів або стовпців, що підвищує читабельність коду та зменшує залежність від порядку розміщення даних.

Індексація матриць:



Індексація матриць у мові програмування R використовується для доступу до окремих елементів, рядків, стовпців або їх підмножин. Матриця розглядається як двовимірна структура даних, у якій кожне значення визначається позицією рядка та стовпця.

Доступ до елементів матриці здійснюється шляхом зазначення двох індексів: першого для рядка та другого для стовпця. Це дозволяє вибирати як окремі значення, так і цілі фрагменти матриці. Можливий також вибір кількох рядків або стовпців одночасно, а також виключення окремих елементів з аналізу.

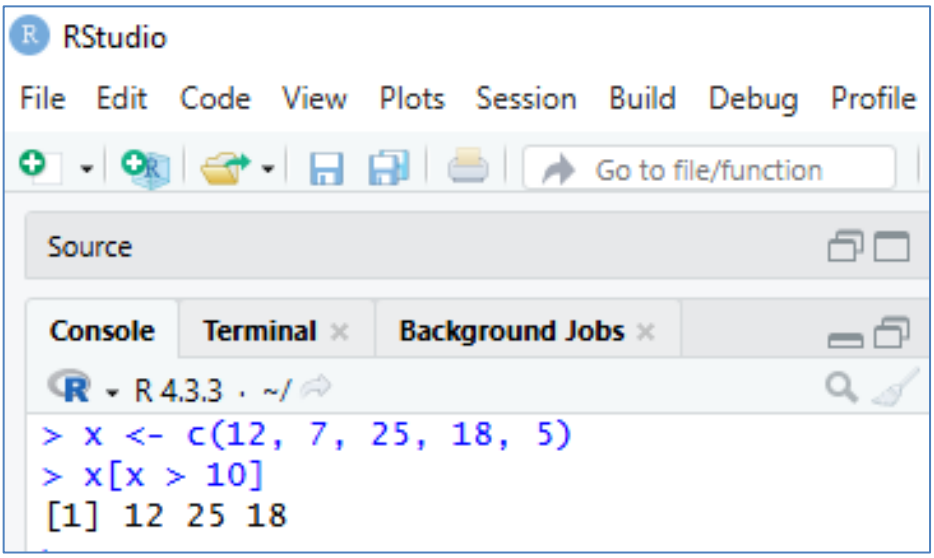
Індексація може виконуватися за позиціями або за іменами, якщо вони задані для рядків чи стовпців. Крім того, можна працювати з цілими рядками або стовпцями, що спрощує аналіз і трансформацію даних у табличній формі.

3.4.2.Логічна фільтрація

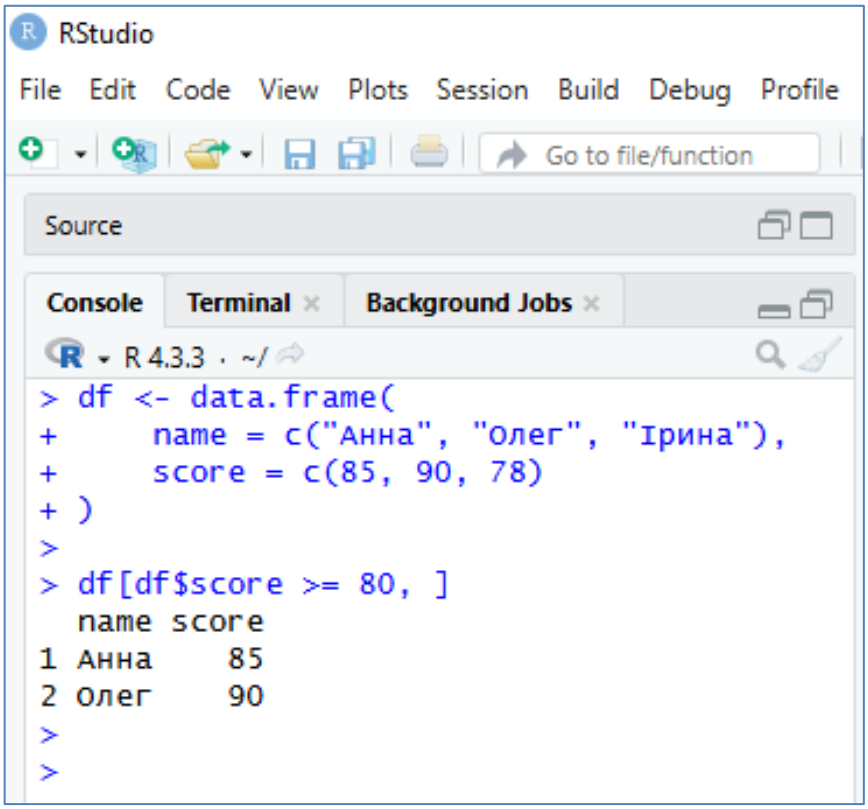
Логічна фільтрація в мові програмування R – це спосіб відбору елементів даних на основі логічних умов, результатом яких є значення TRUE або FALSE. Вона використовується під час аналізу даних для вибірки лише тих спостережень, що відповідають заданим критеріям.

Механізм логічної фільтрації ґрунтується на застосуванні логічних операторів (>, <, ==, !=, >=, <=) та логічних зв'язок (&, |, !). У результаті виконання умови формується логічний вектор, який використовується для індексації даних. Аналог фільтрації у SQL.

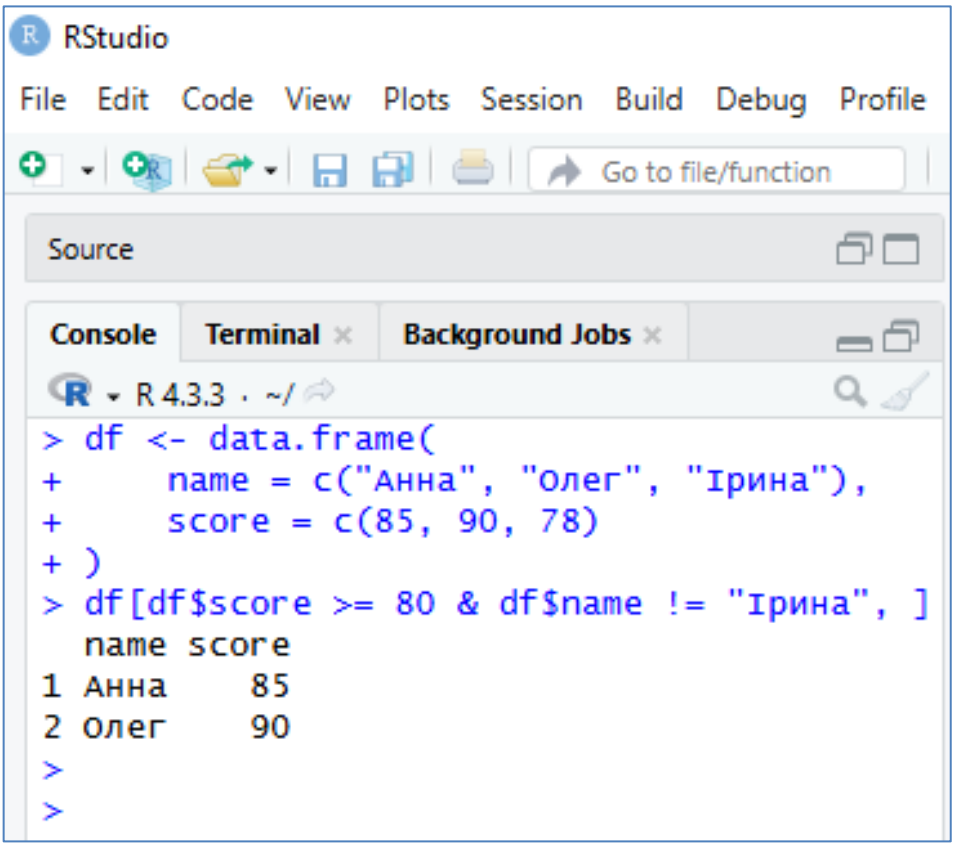
Логічна фільтрація вектора:



Логічна фільтрація таблиці даних (data frame):



Використання кількох умов:



Логічна фільтрація ґрунтується на перевірці значень і формуванні набору істинних та хибних результатів, які визначають, які елементи залишаються в підсумковому наборі даних, а які відкидаються. Такий підхід дає змогу працювати лише з тими даними, що відповідають певним вимогам, наприклад за величиною показника або поєднанням кількох умов. Логічна фільтрація є особливо корисною під час аналізу великих масивів інформації, оскільки дозволяє зосередитися на релевантних спостереженнях і спрощує подальшу обробку результатів.

Лабораторні роботи до розділу 3

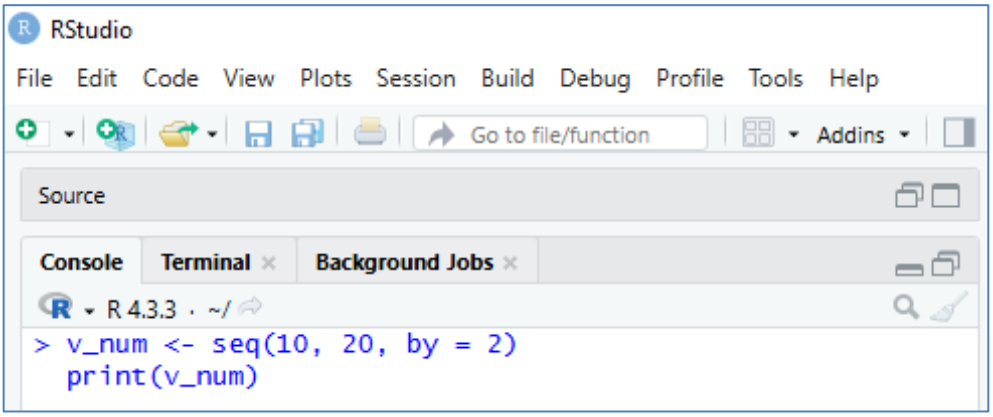
Лабораторна робота 1. Основні структури даних в R: вектори, матриці та списки.

Мета роботи: Набути практичних навичок створення, маніпулювання та доступу до елементів базових одновимірних (вектори), двовимірних (матриці) та гетерогенних (списки) структур даних в R.

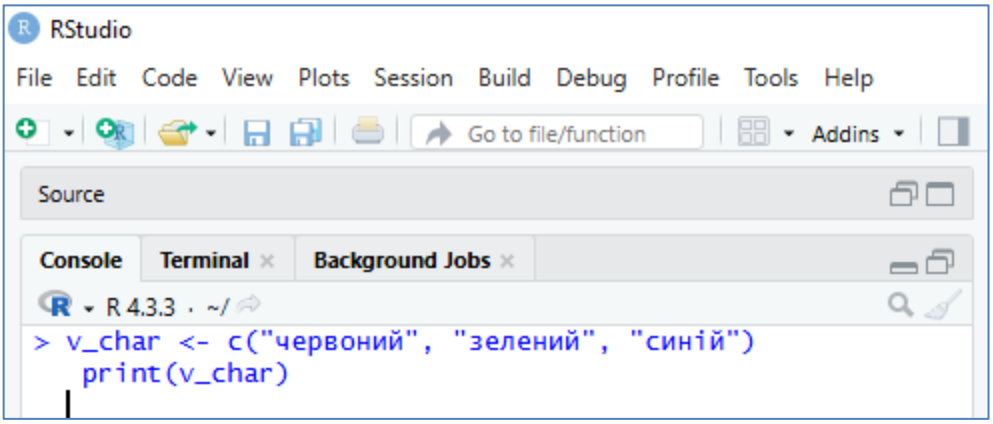
Хід виконання роботи

Завдання 1. Робота з векторами

1. Створити числовий вектор v_num, що містить числа від 10 до 20 з кроком 2.



2. Створити символьний вектор v_char, що містить назви трьох кольорів.



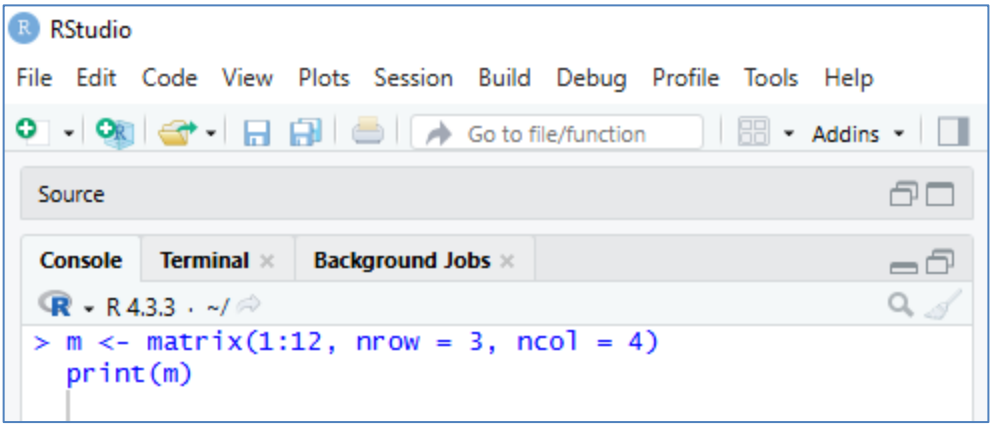
3. Визначити тип даних та клас вектора v_num за допомогою функцій typeof() та class().

4. Отримати третій елемент вектора v_num та всі елементи, крім першого.

5. Створити логічний вектор, який перевіряє, чи елементи v_num більші за 15. Використати цей логічний вектор для фільтрації v_num.

Завдання 2. Робота з матрицями

1. Створити матрицю m розміром 3x4 (3 рядки, 4 стовпці), заповнену числами від 1 до 12.



2. Визначити розміри матриці m за допомогою функції dim().

3. Отримати елемент, що знаходиться на перетині 2-го рядка та 3-го стовпця.

4. Отримати весь 1-й рядок матриці.

5. Виконати транспонування матриці m та зберегти результат у нову змінну m_t.

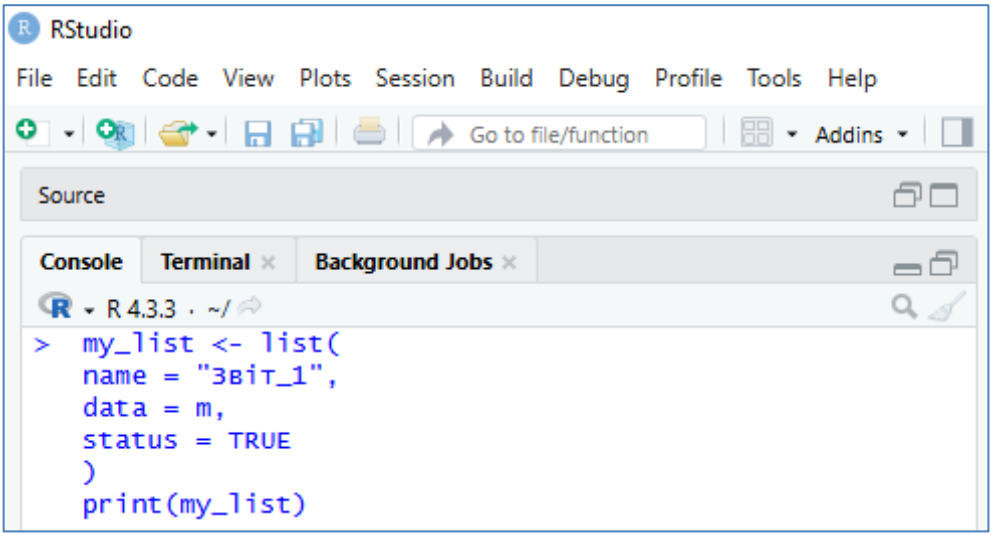
Завдання 3. Робота зі списками

1. Створити список my_list, який містить:

name: символьний вектор "Звіт_1"

data: матриця m (створена у Завданні 2)

status: логічне значення TRUE

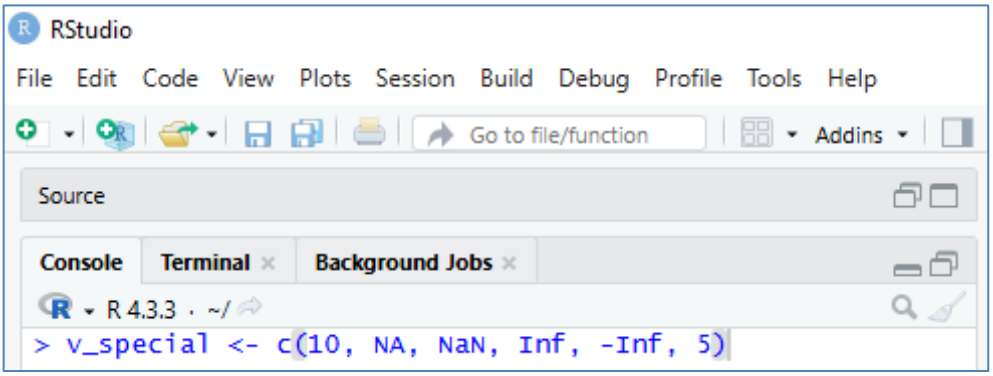


2. Отримати елемент name зі списку, використовуючи оператор $ та оператор [[]].

3. Отримати елемент data зі списку та вивести його розміри.

Завдання 4. Робота зі спеціальними змінними

1. Створити числовий вектор v_special, що містить NA, NaN, Inf та звичайне число.



2. Використати функції is.na(), is.nan(), is.infinite() для перевірки елементів вектора v_special.

3. Перевірити, чи вектор v_special є NULL за допомогою функції is.null().

Лабораторна робота 2. Табличні та категоріальні структури даних в R: Data Frame, Tibble, Factor.

Мета роботи: опанувати роботу з табличними структурами даних (data.frame, tibble), які є основними для статистичного

аналізу в мові R, а також навчитися працювати з категоріальними даними за допомогою структури factor.

Хід виконання роботи

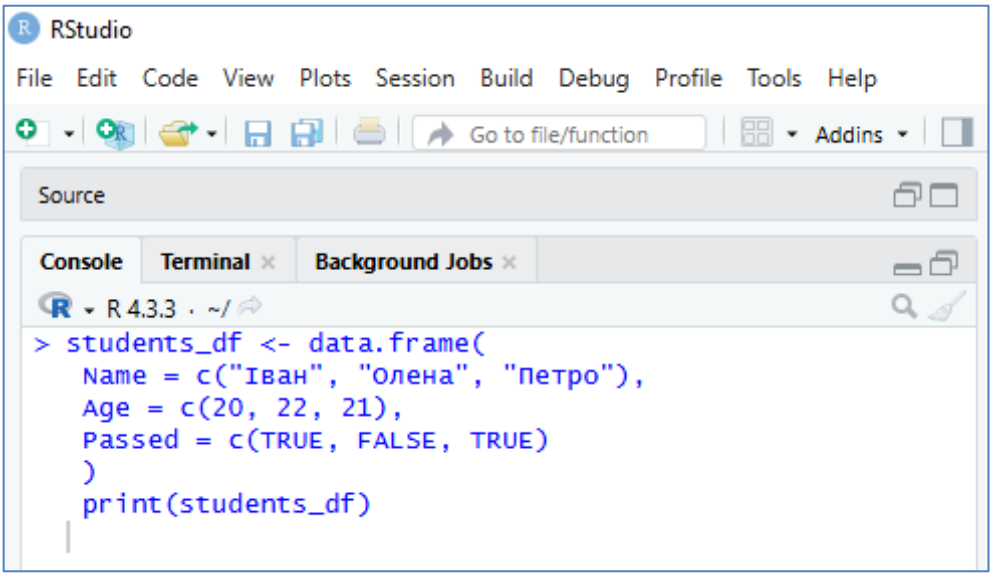
Завдання 1. Робота з Data Frame

1. Створити data.frame під назвою students_df, що містить три стовпці:

Name: символьний вектор (3 імені).

Age: числовий вектор (3 віки).

Passed: логічний вектор (TRUE/FALSE).



2. Отримати доступ до стовпця Name за допомогою оператора $ та вивести його.

3. Отримати доступ до 2-го рядка таблиці.

4. Виконати фільтрацію: вивести лише тих студентів, у яких Passed має значення TRUE.

5. Додати новий стовпець Course (наприклад, "Математика", "Фізика", "Хімія") до students_df використавши одну із трьох технологій поданих як приклад:

students_df$Surname <- c("Коваль", "Шевченко", "Іваненко")

students_df <- cbind(students_df, Group = c("A", "B", "A"))

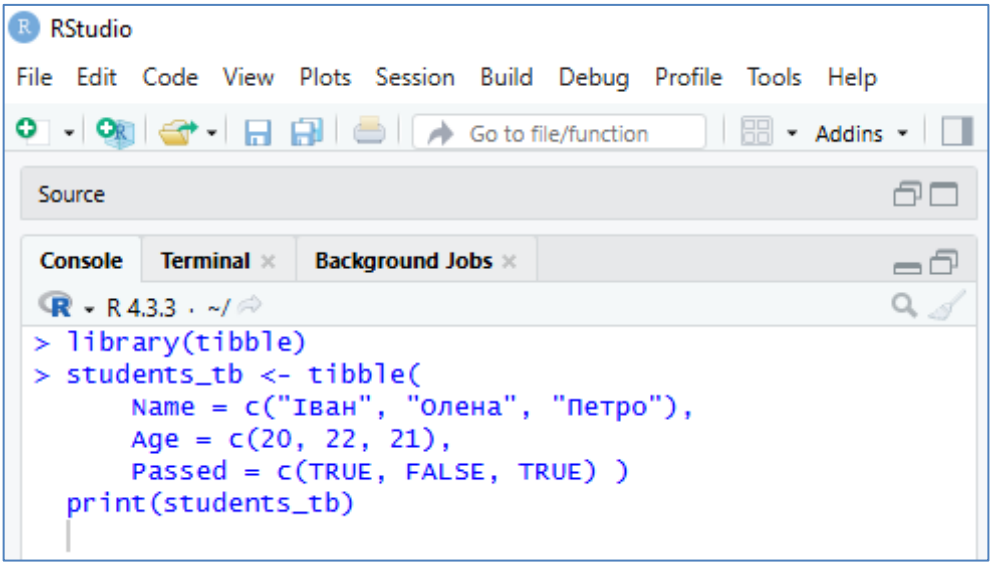
students_df$Status <- ifelse(students_df$Passed, "Зараховано", "Не зараховано")

Завдання 2. Робота з Tibble

1. Встановити та завантажити пакет tibble (якщо він ще не встановлений).

```
install.packages("tibble") # якщо пакет не встановлено
library(tibble)
```

2. Створити tibble під назвою students_tb з тими ж даними, що й у students_df.

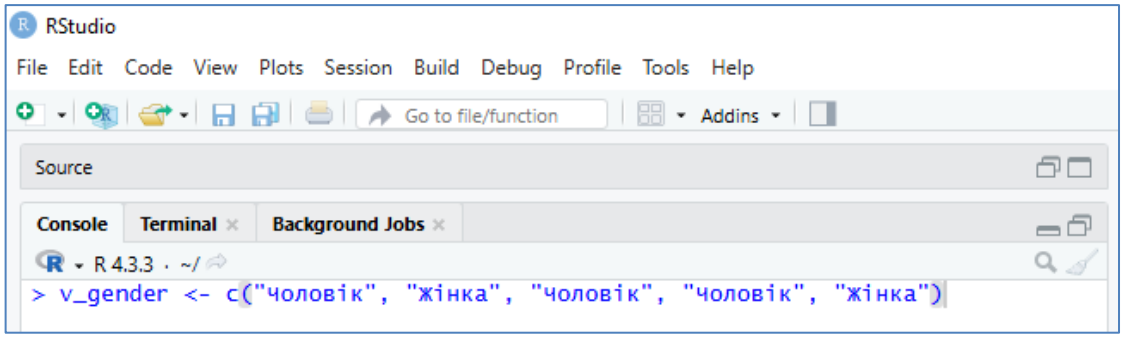


3. Порівняти вивід результатів з students_df та students_tb. Описати відмінності.

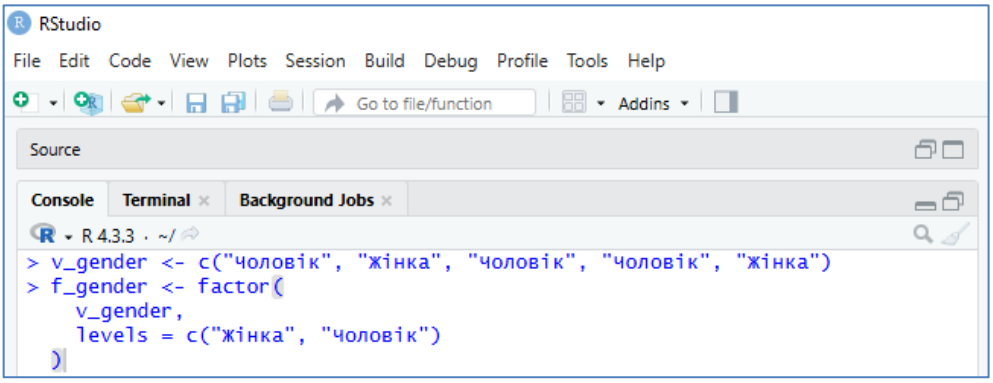
4. Визначити клас об'єкта students_tb за допомогою функції class().

Завдання 3. Робота з факторами

1. Створити символьний вектор v_gender, що містить повторювані значення "Чоловік" та "Жінка".



2. Перетворити вектор v_gender на фактор f_gender за допомогою функції factor() з розширеним параметром levels.



3. Вивести фактор f_gender та його рівні (levels()).

4. Створити таблицю частот для фактора f_gender за допомогою функції table().

Лабораторна робота 3. Механізм індексації в мові R

Мета роботи: Опанувати гнучку та потужну систему індексації в R, навчитися отримувати підмножини даних з різних структур (векторів, матриць, списків, data.frame) за допомогою числових, логічних та іменованих індексів.

Теоретичні відомості

Індексація дозволяє вибирати один або кілька елементів вектора.

Тип індексації	Опис	Приклад
Числова (позитивна)	Вибір елементів за їхньою позицією. Можна використовувати вектор позицій	x[2], x[c(1, 4)]
Числова (негативна)	Виключення елементів за їхньою позицією.	x[-1], x[-c(2, 3)]
Логічна	Вибір елементів на основі логічного вектора (TRUE/FALSE). Дуже потужний механізм фільтрації.	x[x > 20]
Іменована	Вибір елементів за їхніми іменами (якщо вони присвоєні	x["b"]

Матриці та Data Frames є двовимірними структурами, тому індексація вимагає вказівки позиції рядка та стовпця у форматі [рядок, стовпець].

Структура	Синтаксис	Опис
Матриця	m[row, column]	Доступ до елемента на перетині рядка та стовпця.
Data Frame	df[row, column]	Таблична індексація.
Вибір рядка	m[1,] або df[1,]	Вибір першого рядка (залишаємо поле стовпця порожнім
Вибір стовпця	m[, 2] або df[, 2]	Вибір другого стовпця (залишаємо поле рядка порожнім).
Доступ до стовпця Data Frame	df$name або df[["name"]]	Зручний доступ до стовпця за його іменем, або віком .

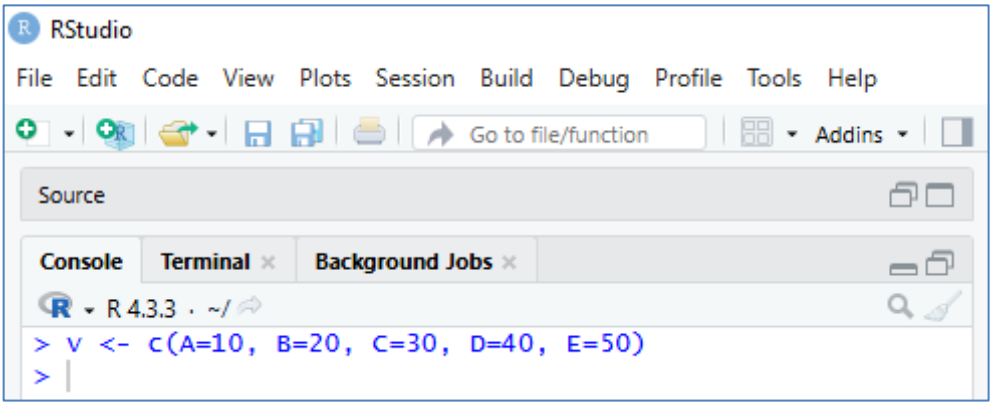
Списки можуть містити елементи різних типів, що обумовлює наявність двох основних операторів доступу.

Оператор	Назва	Призначення	Тип результату
[]	Одинарні дужки	Повертає підсписок, який містить вибрані елементи.	list
[[]]	Подвійні дужки	Повертає сам елемент, витягуючи його зі списку	Тип елемента
$	Знак долара	Доступ до елемента за його іменем.	Тип елемента

Хід виконання роботи

Завдання 1. Індексація векторів

1. Створити іменований числовий вектор v:



2. Написати програмний код який використає елементи числової індексації:

- Вибрати третій елемент.
- Вибрати перший, другий та п'ятий елементи за допомогою вектора індексів.
- Виключити четвертий елемент.

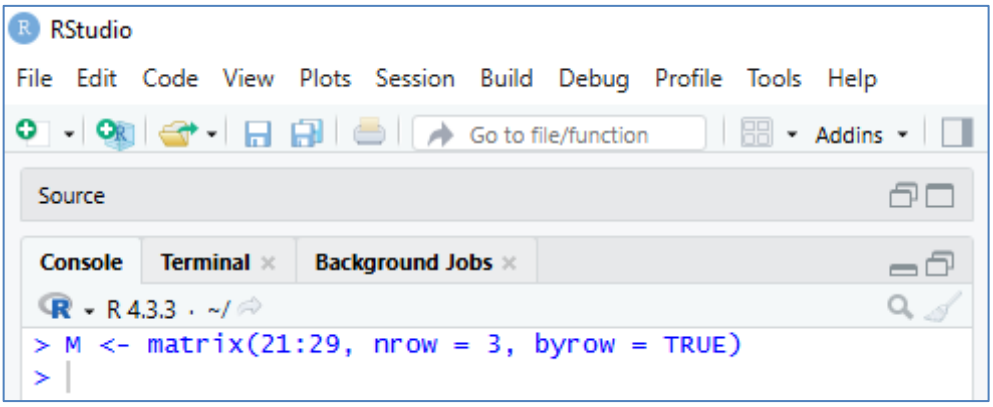
3. Логічна індексація:

- Вибрати усі елементи, значення яких більше за 25.
- Вибрати елементи, які не дорівнюють 10.

4. Іменована індексація: виберіть елементи з іменами "A" та "D".

Завдання 2. Індексація матриць

1. Створити матрицю M розміром 3x3:



2. Двовимірна індексація: вибрати елемент у другому рядку та третьому стовпці.

3. Вибір підмножин:

- Вибрати другий рядок повністю.

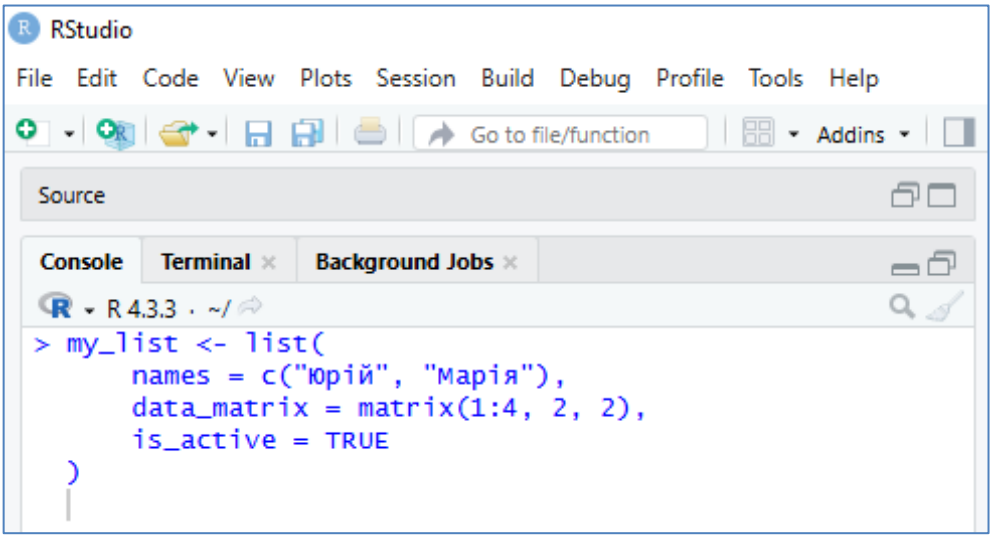
– Вибрати перший стовпець повністю.

– Вибрати підматрицю, що складається з перших двох рядків та перших двох стовпців.

4. Збереження розмірності: вибрати другий стовпець та другий стовпець використовуючи аргумент drop = FALSE. Вивести результат та описати різницю.

Завдання 3. Індексація списків (Lists)

1. Створити список my_list, що містить вектор, матрицю та логічне значення:



2. Доступ до підсписку ([]): вибрати перший елемент як підсписок.

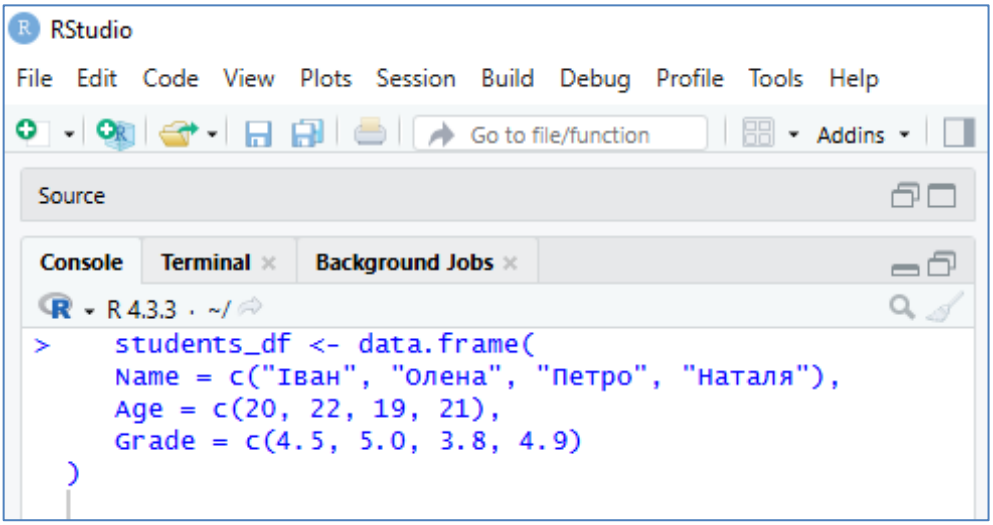
3. Доступ до елемента ([[]]): витягнути сам елемент (вектор імен) за його позицією.

4. Доступ за іменем ($): витягнути елемент is_active за його іменем.

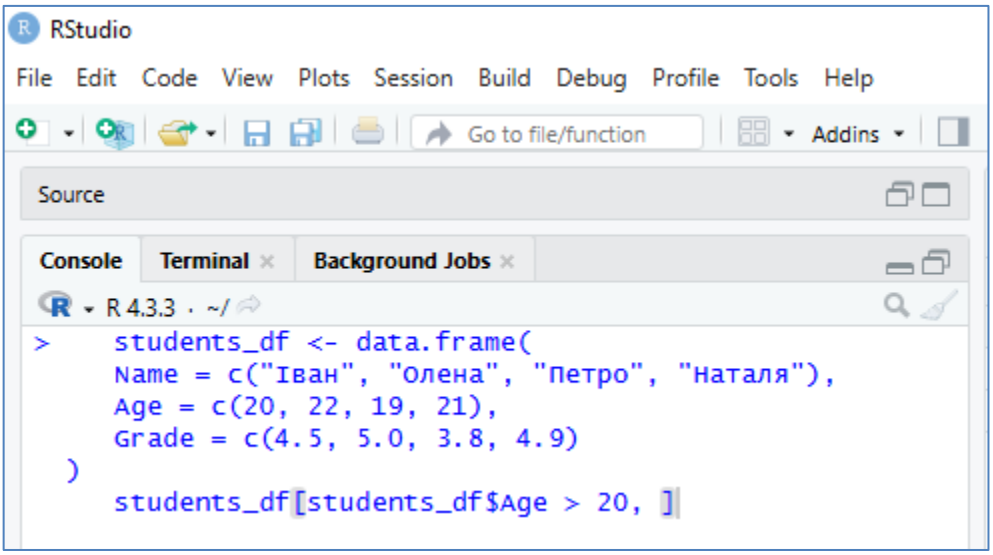
5. Комбінована індексація: витягнути вектор імен, а потім вибрати другий елемент цього вектора.

Завдання 4. Індексація Data Frames (Логічна фільтрація)

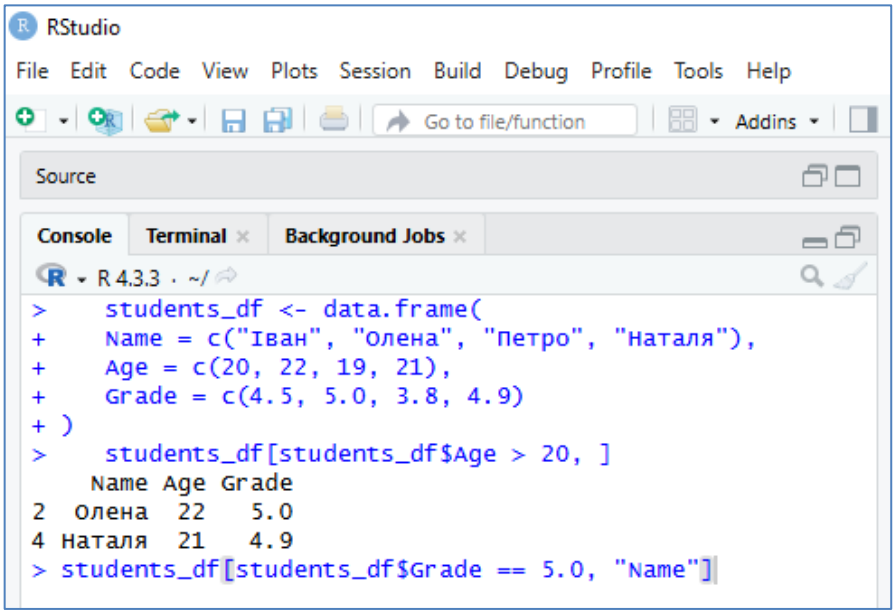
1. Створити Data Frame students_df:



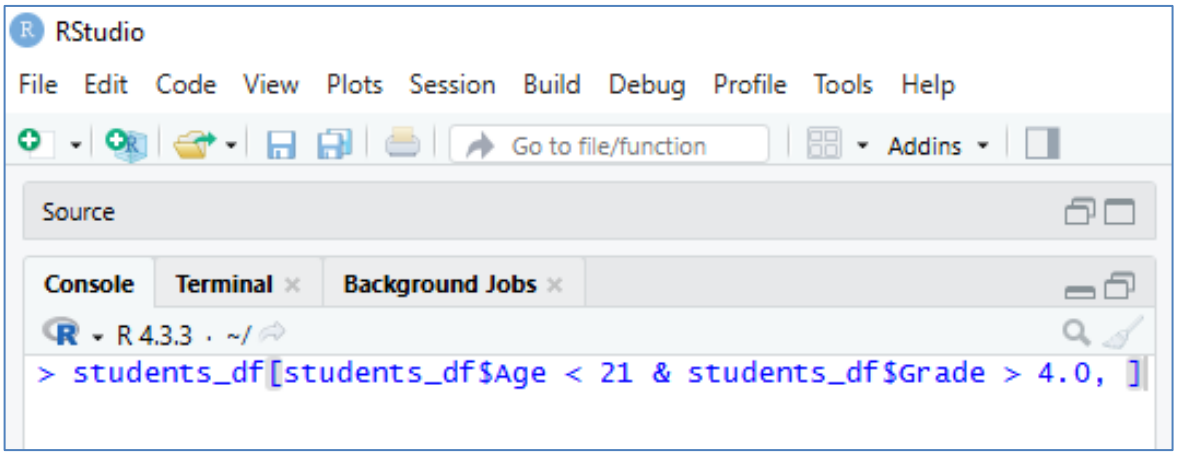
2. Логічна фільтрація рядків: вибрати усі рядки, де вік студента більший за 20 років.



3. Логічна фільтрація з вибором стовпців: вибрати лише імена тих студентів, чий середній бал (Grade) дорівнює 5.0.



4. Складна логічна фільтрація: вибрати студентів, які молодші за 21 рік і мають середній бал вище 4.0.



Інтегровані завдання до розділу 3

Завдання 1. Створення та маніпуляція списком.

1. Створіть список student_profile, що містить:

   Name: Ваше ім'я (символьний вектор).

   Scores: Числовий вектор з 5 оцінками.

   Passed_Courses: Логічний вектор, що вказує, які з 5 курсів були успішно складені.

2. Виведіть середнє значення оцінок (Scores) зі списку.

Завдання 2. Обробка відсутніх даних.

1. Створіть числовий вектор data_with_na з 10 елементів, де 3 елементи є NA.
2. Напишіть код, який:
   - визначає кількість відсутніх значень.
   - замінює всі відсутні значення (NA) на середнє арифметичне значення решти (не-NA) елементів вектора.

Завдання 3. Матричні операції.

1. Створіть дві матриці A та B розміром 2x2 з довільними числами.
2. Виконайте операцію матричного множення (%*%).
3. Збережіть та виведіть результат.

Завдання 4. Створення Data Frame з Factor.

1. Створіть data.frame під назвою sales_data з 5 рядків, що містить:

   Product: Фактор з трьома рівнями ("A", "B", "C").

   Region: Фактор з двома рівнями ("Північ", "Південь").

   Revenue: Числовий вектор з довільними значеннями доходу.

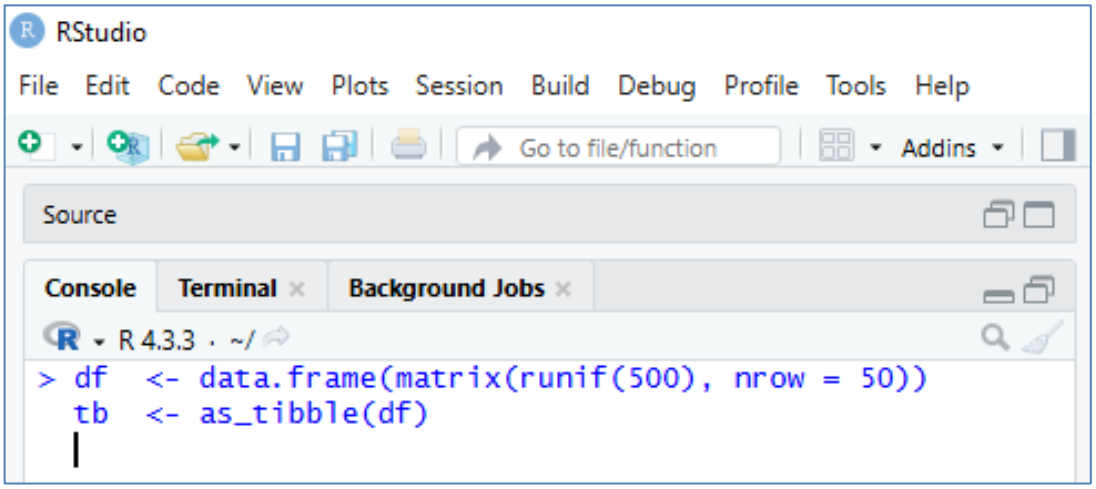
2. Виведіть загальний дохід (Revenue) для кожного регіону, використавши функції aggregate(), tapply(), а також комбінацію table() + sum().

Пояснення: Наприклад with(sales_data, tapply(Revenue, Region, sum)) – ця команда обчислює суму доходу (Revenue) для кожного значення змінної Region.

Завдання 5. Порівняння Data Frame та Tibble.

1. Створіть data.frame з 50 рядків та 10 стовпців.
2. Перетворіть його на tibble.
3. Опишіть, як tibble покращує читабельність великих наборів даних.

Пояснення: Розпочніть із удосконалення запропонованого фрагменту:



Завдання 6. Багатовимірний масив (Array).

1. Створіть тривимірний масив a розміром 2x3x2 (2 рядки, 3 стовпці, 2 шари), заповнений числами.
2. Отримайте доступ до елемента у 1-му шарі, 2-му рядку, 3-му стовпці.
3. Доповніть масив іменами: Row = c("R1", "R2"), Col = c("C1", "C2", "C3"), Layer = c("L1", "L2").
4. Отримайте доступ до елементів масиву використавши імена.

Завдання 7. Індексація для вибірки елементів.

1. Створити data.frame з 5 рядків і виконати фільтрацію за умовою.
2. Використати іменовану індексацію для вибірки елементів.
3. Пояснити різницю між результатами lst[1] та lst[[1]].

Контрольні питання до розділу 3

1. З якого числа починається індексація в R?
2. Наведіть приклад використання негативної числової індексації для вектора.
3. Наведіть приклад використання оператора %in%.
4. Навіщо використовується drop = FALSE?
5. Поясніть концепцію векторизації в R на прикладі арифметичних операцій.
6. Поясніть, чому індексація в R починається з 1.
7. У чому полягає відмінність між логічними операторами & та &&?

8. У чому полягає принципова відмінність між операторами [] та [[]] при роботі зі списками?

9. Що означає векторизація операцій?

10. Що таке вираз у мові R?

11. Як працює логічна індексація? Наведіть приклад її використання для фільтрації Data Frame.

12. Яка принципова різниця між операторами присвоєння <- та =?

13. Який аргумент слід використовувати при індексації матриці, щоб запобігти автоматичному зменшенню розмірності (drop)?

РОЗДІЛ 4. ОСНОВИ РОБОТИ ІЗ ЗОВНІШНІМИ ДАНИМИ

4.1.Поняття імпорту даних та його значення в аналізі даних

Імпорт даних є початковим етапом процесу аналізу даних і полягає у зчитуванні інформації із зовнішніх джерел у середовище програмування для подальшої обробки та дослідження. У контексті мови програмування R імпорт даних забезпечує перенесення структурованих або напівструктурованих наборів даних з файлів різних форматів у вигляді об’єктів, придатних для статистичного аналізу та моделювання.

Значення імпорту даних зумовлюється тим, що переважна більшість емпіричних даних зберігається поза середовищем R – у файлах електронних таблиць, текстових документах або базах даних. Коректне зчитування таких даних є необхідною умовою забезпечення їхньої цілісності, збереження типів змінних та адекватного відображення структури досліджуваного об’єкта.

Процес імпорту даних передбачає не лише завантаження інформації, але й налаштування параметрів зчитування, зокрема вибір роздільників, кодування символів, форматів числових і текстових значень. Невірно виконаний імпорт може призвести до спотворення даних, появи пропущених значень або помилок у подальших обчисленнях, що негативно впливає на результати аналізу. Мова програмування R надає широкий набір інструментів для зчитування даних із різних форматів, зокрема CSV, Excel (XLS/XLSX) та TXT. Правильний імпорт забезпечує коректність подальших обчислень і аналізу.

Перед імпортом даних необхідно:

- визначити робочий каталог;
- перевірити формат файлу;
- врахувати кодування тексту;
- коректно задати роздільники та десяткові символи.

Рекомендується працювати в проєктах RStudio, у межах яких робочий каталог налаштовується автоматично, що забезпечує впорядковану структуру файлів, коректні шляхи доступу до даних і відтворюваність результатів аналізу

незалежно від середовища виконання.

Для роботи з робочим каталогом і проєктами в R / RStudio використовуються такі основні команди:

1. Перегляд поточного робочого каталогу – getwd()
2. Встановлення робочого каталогу вручну –setwd("шлях/до/каталогу")
3. Перегляд файлів у робочому каталозі – list.files()

4. Створення проєкту в RStudio (через інтерфейс) – File → New Project робочий каталог налаштовується автоматично при відкритті проєкту

5. Використання відносних шляхів у проекті – read.csv("data/data.csv")

6. Перевірка існування файлу – file.exists("data/data.csv")

7. Отримання інформації про файл – file.info("data/data.csv")

4.2.Формати файлів для зберігання даних

Формати файлів для зберігання даних визначають спосіб організації, подання та кодування інформації у цифровому вигляді. Вибір формату файлу має суттєве значення для збереження структури даних, коректності їх подальшого зчитування та ефективності аналізу в середовищі програмування R.

Найпоширенішими є текстові формати, які зберігають дані у вигляді символів і є універсальними для обміну інформацією між різними програмними системами. До таких форматів належать CSV, XLS, XLSX та TXT, які легко редагуються та підтримуються більшістю програм для обробки даних.

Формат CSV (Comma-Separated Values) є одним із найбільш поширених форматів для збереження табличних даних. У файлах цього типу значення в рядках розділяються спеціальним символом, зазвичай комою або крапкою з комою, що залежить від регіональних налаштувань. Основною перевагою CSV є його простота та універсальність: такі файли легко відкриваються, редагуються та імпортуються у більшість програмних середовищ. Водночас CSV не підтримує збереження форматування, формул або кількох таблиць в одному файлі, що слід враховувати під час підготовки даних.

Формати Excel (XLS, XLSX) використовуються для збереження даних у

вигляді електронних таблиць з можливістю розміщення інформації на кількох аркушах. Вони підтримують форматування, формули, коментарі та різні типи даних, що робить їх зручними для введення та попередньої обробки інформації. Особливістю файлів Excel є їх закритий або напівзакритий формат, що потребує застосування спеціальних засобів для імпорту в R. Крім того, наявність формул може ускладнювати коректне зчитування фактичних значень.

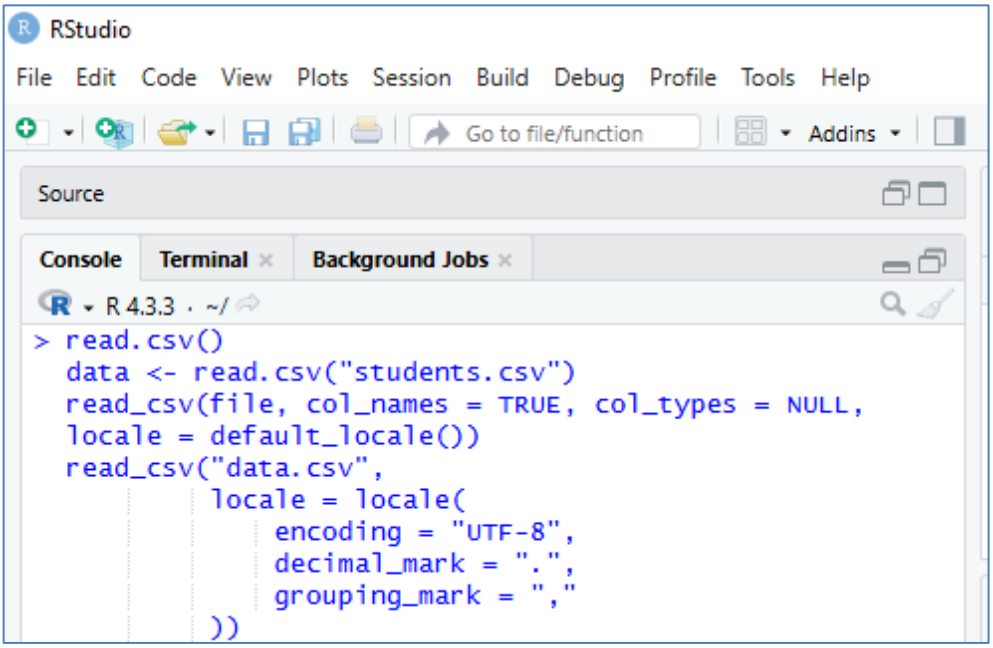
Текстові файли (TXT) є гнучким форматом для збереження даних у символьному вигляді з довільно визначеним роздільником. Вони дозволяють зберігати як прості списки значень, так і складні табличні структури. Основною перевагою TXT-файлів є повна незалежність від конкретного програмного забезпечення та можливість точного контролю над структурою даних. Разом з тим відсутність стандартизованого формату потребує уважного налаштування параметрів зчитування під час імпорту в R.

4.2.1.Імпорт даних з CSV-файлів

Імпорт даних з CSV-файлів – процес завантаження табличних даних у середовище R з файлів формату CSV (Comma-Separated Values) за допомогою спеціалізованих функцій, який забезпечує коректне зчитування структури таблиці, типів змінних та кодування символів для подальшого аналізу й опрацювання.

Формат CSV (Comma-Separated Values) – текстовий формат, у якому дані розділені комами або крапками з комою. Функції read.csv() та read_csv() використовуються в мові програмування R для імпорту табличних даних із файлів формату CSV у середовище аналізу даних. Вони дозволяють зчитувати дані з урахуванням структури файлу, типів змінних і особливостей кодування.

Функція read.csv() є базовим засобом мови R для зчитування CSV-файлів. Вона імпортує дані у вигляді таблиці даних (data frame), де кожен стовпець відповідає окремій змінній, а кожен рядок – спостереженню. За замовчуванням функція вважає, що перший рядок файлу містить назви стовпців, а роздільником значень є кома.



У наведеному прикладі дані з файлу students.csv зчитуються та зберігаються в об’єкті data, після чого можуть бути використані для подальшого аналізу.

Функція read_csv() належить до пакета readr і є сучасною альтернативою read.csv(). Вона забезпечує вищу швидкість зчитування, кращу обробку типів даних та зручніший вивід результатів. За замовчуванням read_csv() повертає об’єкт типу tibble.

Параметр col_names визначає наявність назв стовпців у файлі, col_types дозволяє явно задати типи змінних, а locale використовується для налаштування регіональних параметрів зчитування даних.

Додаткові можливості read_csv() проявляються при роботі з файлами, що мають специфічне кодування або формат числових даних. Наприклад, можна задати параметри locale для коректного зчитування символів і числових роздільників.

У цьому випадку забезпечується правильне зчитування текстових даних у кодуванні UTF-8, а також коректна інтерпретація десяткових і тисячних роздільників у числових значеннях.

4.2.2.Імпорт даних з Excel-файлів

Для імпорту даних з електронних таблиць формату Excel (XLS, XLSX) у

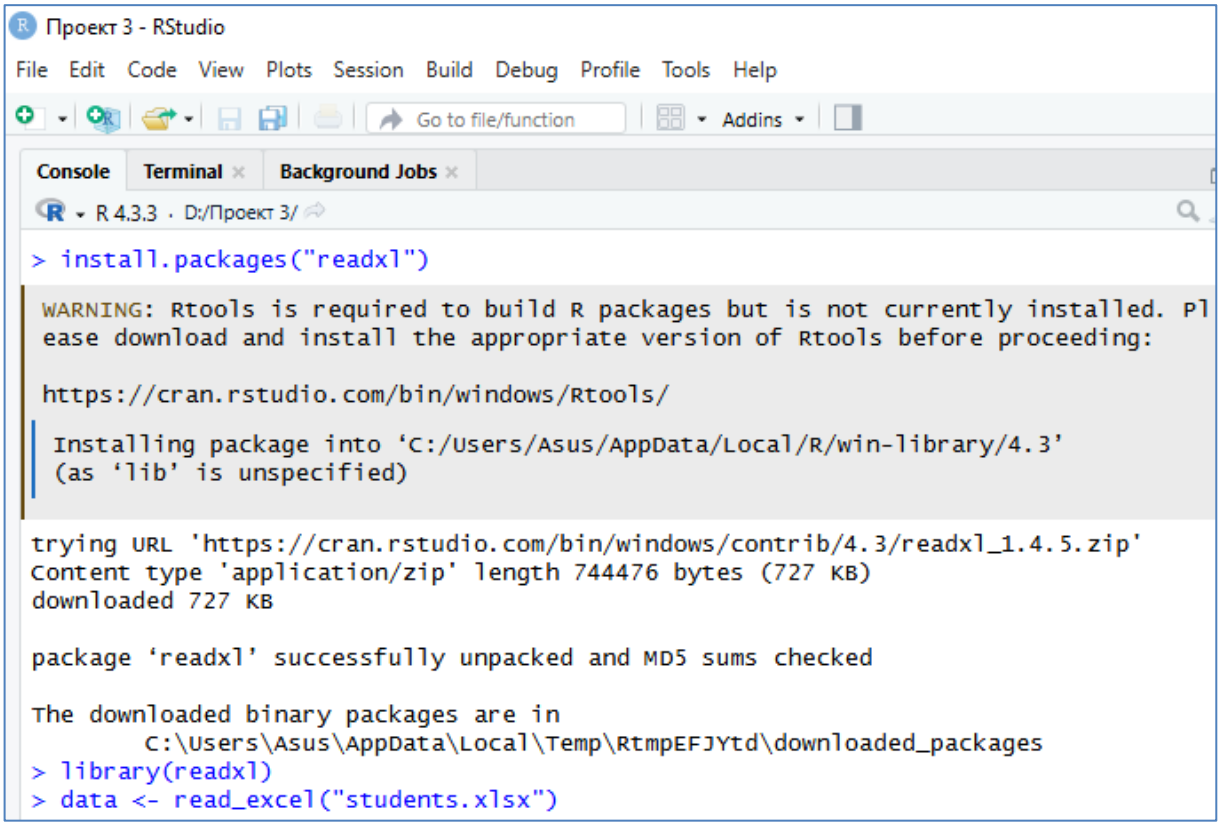
мові програмування R використовується функція read_excel(), яка входить до пакета readxl. Вона дозволяє зчитувати табличні дані з одного або кількох аркушів книги Excel та перетворювати їх у зручну для аналізу форму. Для використання функції read_excel() у R необхідно встановити та підключити пакет readxl, оскільки ця функція не входить до базового набору R.

1. Встановлення пакета (виконується один раз)

install.packages("readxl")

2. Підключення пакета

library(readxl)



Після підключення пакета readxl функція read_excel() стає доступною для зчитування файлів формату .xls та .xlsx без додаткових налаштувань.

Команда data <- read_excel("students.xlsx") зчитує дані з файлу students.xlsx за замовчуванням з першого аркуша книги та зберігає їх в об'єкті data. Результатом виконання є таблиця даних у форматі tibble, де кожен стовпець відповідає змінній, а кожен рядок – окремому спостереженню.

Для зчитування даних з конкретного аркуша використовується параметр sheet, який може задаватися як номером аркуша, так і його назвою:

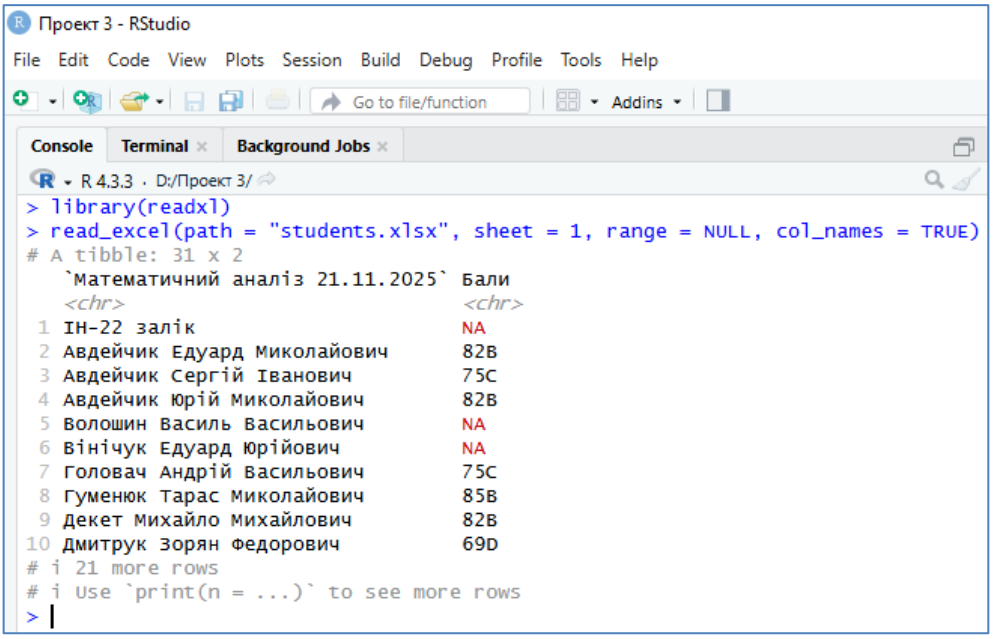
data_xlsx <- read_excel("students.xlsx", sheet = "2025")

У цьому випадку дані будуть зчитані з аркуша з назвою 2025, що є зручним при роботі з файлами, які містять кілька наборів даних.

Загальний вигляд функції має такий формат:

read_excel(path = "students.xlsx", sheet = 1, range = NULL, col_names = TRUE)

Параметр path визначає шлях до файлу, sheet задає номер або назву аркуша, range дозволяє обмежити зчитування певним діапазоном комірок, а col_names визначає, чи використовувати перший рядок як назви стовпців.



Для ознайомлення зі структурою Excel-файлу та перегляду доступних аркушів використовується функція:

excel_sheets("students.xlsx")

Вона повертає список назв усіх аркушів, що містяться у файлі, що полегшує вибір потрібного джерела даних для імпорту.

4.2.3.Імпорт даних з текстових файлів (TXT)

Функція readLines("students.txt", n = 6) використовується для построкового зчитування тексту потрібно переглянути вміст файлу або необхідна попередня діагностика перед імпортом.

У мові R для зчитування текстових файлів використовуються функції

сімейства read.table(). Вона використовується для зчитування файлів з розширенням TXT та інших текстових джерел, у яких дані подані у вигляді рядків і стовпців з певним роздільником.

Команда data <- read.table("students.txt", header = TRUE) зчитує дані з файлу *students.txt* та зберігає їх у вигляді таблиці даних. Параметр header = TRUE означає, що перший рядок файлу містить назви стовпців, які будуть використані як імена змінних.

Основні параметри:

file – шлях до файлу з даними;

header – наявність заголовків стовпців (TRUE або FALSE);

sep – символ-розділювач:

" " або "" – пробіли,

"," – кома,

";" – крапка з комою,

"\t" – табуляція;

stringsAsFactors – забороняє автоматичне перетворення тексту у фактори.

За необхідності роздільник між значеннями можна задати вручну, що особливо важливо для файлів, створених у різних середовищах:

```
data <- read.table("students.txt", sep = "\t")
```

Зчитування з кодуванням UTF-8

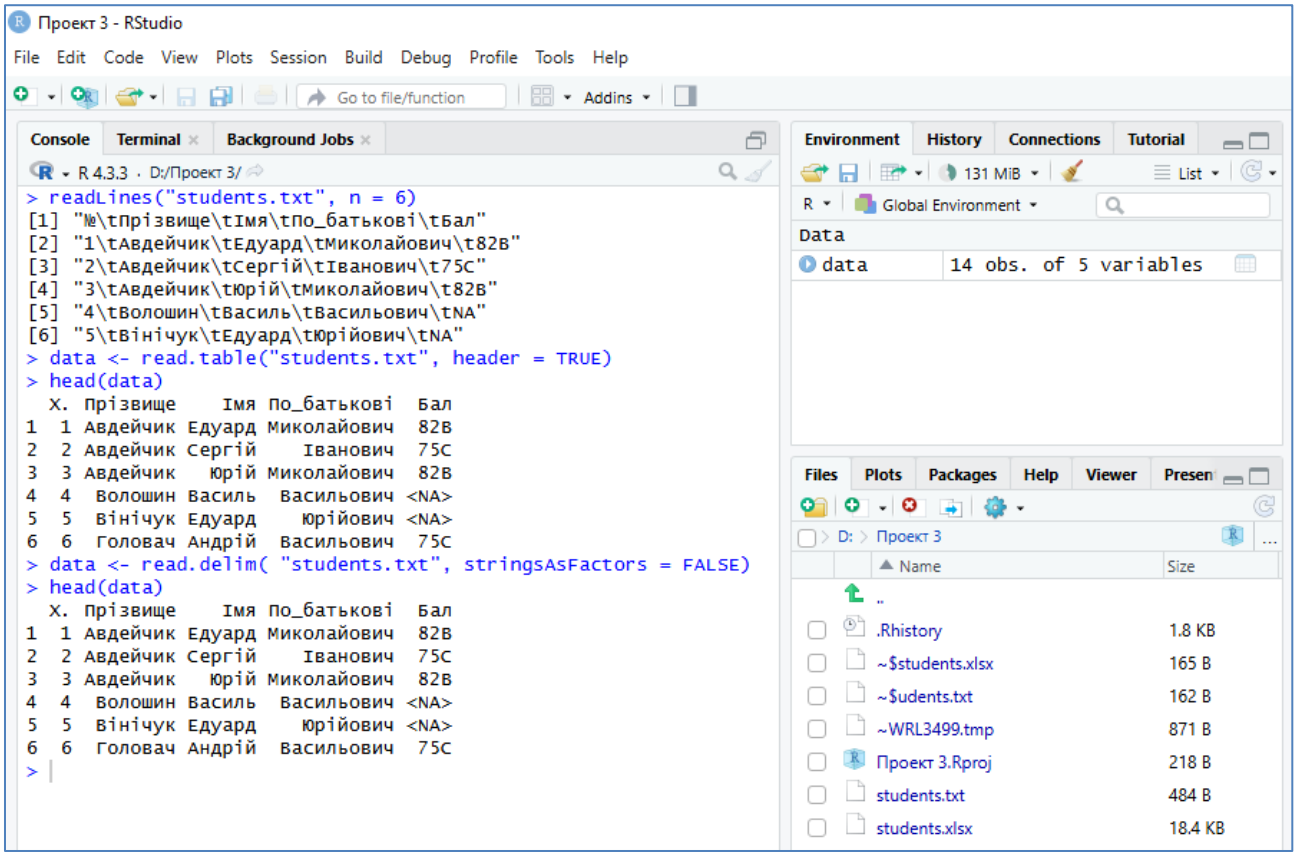
```
data <- read.table(
 "students.txt",
 header = TRUE,
 fileEncoding = "UTF-8")
```

У такому випадку забезпечується правильна інтерпретація символів у кодуванні UTF-8, що є особливо актуальним при роботі з текстами українською мовою.

Функція read.delim() є спеціалізованим варіантом для файлів із табуляцією.

```
data <- read.delim( "students.txt", stringsAsFactors = FALSE)
```

Перевага цієї функції полягає в автоматичному розпізнаванні табуляційного розділювача.



Після зчитування даних доцільно виконати їх первинну перевірку:

head(data)

str(data)

4.2.4.Імпорт даних із файлу бази даних

Для роботи з базами даних у мові програмування R використовується пакет DBI, який забезпечує уніфікований інтерфейс доступу до різних систем керування базами даних. На першому етапі необхідно встановити та підключити відповідний пакет:

install.packages("DBI")

library(DBI)

Далі встановлюється з’єднання з базою даних. У наведеному прикладі використовується база даних SQLite, яка зберігається у вигляді окремого файлу:

con <- dbConnect(RSQLite::SQLite(), "path/to/your/database.sqlite")

Після встановлення з’єднання дані зчитуються за допомогою SQL-запиту. Функція dbGetQuery() виконує запит до бази даних і повертає результат у вигляді таблиці даних, придатної для подальшого аналізу в R:

data <- dbGetQuery(con, "SELECT * FROM your_table")

Після імпорту даних обов’язковим етапом є їх перевірка, що дозволяє оцінити структуру, об’єм і загальну якість отриманого набору. Для цього використовуються стандартні діагностичні функції.

Функція str() надає стислий опис структури даних, типів змінних і кількості спостережень str(data)

Функція head() дозволяє переглянути перші кілька рядків таблиці, що дає уявлення про зміст даних і коректність їх зчитування head(data)

Функція summary() формує узагальнену статистичну характеристику змінних, зокрема мінімальні, максимальні значення, середні показники та наявність пропущених даних summary(data)

4.3.Експорт результатів

Після імпорту та аналізу даних важливим етапом роботи є експорт результатів – збереження отриманих даних, таблиць, статистичних показників, графіків і звітів у зовнішні файли. Мова програмування R надає широкий набір засобів для експорту результатів у різні формати, що дозволяє передавати результати іншим користувачам або використовувати їх у звітах, презентаціях і публікаціях.

Перед експортом результатів необхідно:

1. переконатися у коректності результатів;
2. вибрати відповідний формат файлу;
3. врахувати кодування символів;
4. визначити робочий каталог getwd()

У проєктах RStudio файли зазвичай зберігаються в каталозі проєкту.

Експорт у CSV-файли виконує функція write.csv()

write.csv(results, "results.csv", row.names = FALSE)

Параметр results є посиланням на об’єкт, створений у середовищі R для збереження результатів аналізу. Аргумент row.names = FALSE запобігає запису номерів рядків у файл.

Для використання європейського стандарту форматування застосовується функція:

write.csv2(results, "results.csv", row.names = FALSE)

Особливості формату:

- роздільник полів – крапка з комою (;);
- десятковий роздільник – кома (,).

Файли CSV є зручними для обміну даними між різними програмними засобами. Водночас слід враховувати, що вони не зберігають інформацію про типи змінних та внутрішню структуру об’єктів R.

Експорт у формат Excel (XLSX)

Файли формату XLSX широко використовуються в навчальній і прикладній діяльності, особливо у випадках, коли результати аналізу призначені для користувачів, які не працюють із R.

Для створення Excel-файлів необхідно використати додаткові пакети, наприклад:

```
library(openxlsx)
write.xlsx(results, "results.xlsx")
```

Можливості експорту:

- запис одного або кількох аркушів;
- збереження таблиць із форматуванням.

Запис кількох аркушів

```
write.xlsx(
  list(
    Summary = summary_table,
    Data = results
  ),
  "analysis_results.xlsx")
```

Таким чином можна структурувати результати аналізу в межах одного файлу.

Експорт у текстовий формат (TXT)

Для збереження даних у довільному текстовому форматі використовується універсальна функція write.table():

```
write.table(
  results,
  "results.txt",
  sep = "\t",
  row.names = FALSE
)
```

TXT-файли застосовуються для:

- передачі даних між інформаційними системами;
- архівування результатів;
- інтеграції з іншими програмними продуктами.

Збереження об'єктів у внутрішніх форматах R

Формат .RData

```
save(results, model, file = "session.RData")
load("session.RData")
```

Особливості:

- можливість одночасного збереження кількох об'єктів (наприклад, results, model);
- повне збереження структури об'єктів (data frame, модель, список тощо);
- відновлення об'єктів у робочому середовищі за допомогою функції load().

Формат .rds

```
saveRDS(results, "results.rds")
```

results <- readRDS("results.rds")

Використовується для збереження одного об’єкта з подальшим його явним присвоєнням під час завантаження.

Збереження графічних об’єктів

Процес збереження графіків складається з трьох етапів:

- відкриття графічного пристрою;
- побудова графіка;
- закриття пристрою.

Збереження у форматі PNG

```
png("plot.png")
plot(x, y)
dev.off()
```

Після виклику png() усі графічні команди записуються у файл, а не виводяться на екран. Функція dev.off() закриває графічний пристрій і зберігає файл на диск.

Розширені параметри збереження

```
png("plot.png", width = 800, height = 600, res = 96)
plot(x, y, main = "Залежність y від x", xlab = "x", ylab = "y")
dev.off()
```

Збереження у форматі PDF

```
pdf("plot.pdf")
plot(x, y)
dev.off()
```

Аналогічно працюють функції jpeg() та tiff().

Рекомендації щодо вибору формату:

PNG – для веб-ресурсів;

PDF – для наукових публікацій;

JPG – для презентаційних матеріалів.

Експорт звітів (R Markdown)

Файли з розширенням .Rmd дозволяють формувати повноцінні аналітичні звіти у форматах:

HTML;

PDF;

DOCX.

rmarkdown::render("report.Rmd")

Використання R Markdown забезпечує відтворюваність результатів аналізу та інтеграцію коду, текстових пояснень і графічних матеріалів в одному документі.

Формати CSV, Excel, TXT, RData, RDS, графічні файли та звіти R Markdown забезпечують збереження й поширення результатів аналізу у зручному та відтворюваному вигляді. Обґрунтований вибір формату експорту та його параметрів гарантує коректність представлення даних і професійний рівень роботи в середовищі R та RStudio.

Лабораторні роботи до розділу 4

Лабораторна робота 1. Імпорт даних із різних джерел у середовищі RStudio

Мета роботи. Навчитися імпортувати дані з різних файлових форматів у RStudio; опанувати використання пакетів readr та readxl; перевіряти та аналізувати типи даних змінних; сформувати навички первинного огляду структури даних.

Теоретичні відомості

У реальних аналітичних задачах дані надходять із різних джерел і в різних форматах. Найпоширенішими є: CSV (Comma-Separated Values) – текстовий формат для табличних даних; Excel (XLSX) – електронні таблиці.

У середовищі RStudio пакет readr використовується для швидкого імпорту текстових файлів (CSV, TSV), а пакет readxl дозволяє працювати з файлами Excel без встановлення Microsoft Excel. Коректне розпізнавання типів

змінних (числові, текстові, фактори, дати) є основою для подальшого аналізу, статистики та моделювання.

Вихідні дані. Студент повинен мати:

CSV-файл (наприклад: data_students.csv);

Excel-файл (наприклад: data_scores.xlsx).

Дані не повинні містити персональної інформації. Набір сайтів та порталів, де можна безкоштовно завантажувати реальні набори даних для проведення аналітичних досліджень у R:

https://dataportals.org

https://data.gov.ua/

https://www.kaggle.com/

https://datasetsearch.research.google.com/

https://catalog.data.gov/dataset/

https://www.openml.org/

Хід виконання роботи

1. Створити новий проєкт: File → New Project → New Directory → Empty Project. Задати назву проєкту (наприклад, Lab1_DataImport). Скопіювати Excel файли у папку проєкту. Для зчитування CSV файлів використати сайт https://www.kaggle.com/.

2. Встановлення та підключення необхідних пакетів.

Виконати в консолі або скрипті:

install.packages("readr")

install.packages("readxl")

Підключити пакети:

library(readr)

library(readxl)

Примітка: встановлення виконується лише один раз, підключення – при кожному запуску.

3. Імпорт даних із CSV-файлу (пакет readr). Завантажити CSV-файл у середовище R:

data_csv <- read_csv("data_students.csv")

Переглянути перші рядки таблиці:

head(data_csv)

Перевірити структуру даних:

str(data_csv)

Отримати інформацію про типи колонок:

spec(data_csv)

Проаналізувати:

1. які змінні мають тип dbl (числові);
2. які – chr (текстові);
3. чи є змінні типу date або lgl.

4. Імпорт даних з Excel-файлу (пакет readxl). Завантажити Excel-файл:

data_excel <- read_excel("data_scores.xlsx")

Переглянути перші 10 рядків:

head(data_excel, 10)

Перевірити структуру таблиці:

str(data_excel)

Отримати список назв змінних:

colnames(data_excel)

5. Аналіз правильності розпізнавання типів даних.

Порівняти:

- очікувані типи змінних;
- типи, які визначила R.

Зафіксувати:

- чи не розпізнані числові значення як текст;
- чи правильно визначені дати;
- чи є проблемні змінні;
- чи є поля з пропущеними значеннями.

У разі потреби змінити тип змінної вручну (приклад):

data_csv$score <- as.numeric(data_csv$score)

data_excel$group <- as.factor(data_excel$group)

6. Порівняння двох наборів даних. Визначити розмір кожного набору даних:

dim(data_csv)

dim(data_excel)

Порівняти структури:

str(data_csv)

str(data_excel)

Зробити висновок:

- який формат зручніший;
- у якому випадку типи даних визначаються точніше.

Результати виконання роботи

В ході виконання роботи необхідно успішно імпортувати CSV та Excel файли; перевірити та проаналізувати типи змінних; вміти діагностувати проблеми з типами даних; підготувати дані до подальшого аналізу. Сформуйте текстовий звіт у якому необхідно подати мету роботи; фрагменти коду; скріншоти результатів; короткі висновки.

Лабораторна робота 2. Експорт результатів аналізу даних у R

Мета роботи: Набути практичних навичок збереження результатів обробки та аналізу даних (таблиць, об'єктів R, графіків) у різні зовнішні формати (CSV, XLSX, TXT, RData, RDS, PNG, PDF) для подальшого використання та поширення.

Теоретичні відомості

Важливі принципи:

1. Визначення поточного каталогу за допомогою getwd() та встановлення його за допомогою setwd() (або використання проєктів RStudio).

2. Використання сучасних функцій (наприклад, write_csv з пакету readr) для забезпечення коректного кодування UTF-8.

3. Врахування регіональних стандартів (наприклад, write.csv2() для європейського формату з роздільником ; та десятковим роздільником ,).

Експорт результатів є завершальним і надзвичайно важливим етапом аналізу даних, що забезпечує можливість передачі, архівування та відтворюваності отриманих даних.

Формат файлу	**Основна функція(ї)**	**Призначення**
CSV/TXT	write.csv()write.csv2() write_csv() write.table()	Збереження табличних даних для обміну між різними системами та програмами.
XLSX	write.xlsx() (пакет openxlsx)	Збереження таблиць у форматі MS Excel, зручному для кінцевих користувачів та з підтримкою форматування
RData/RDS	save(), load(), saveRDS()png(), pdf(), dev.off() readRDS()	Збереження об'єктів R (змінних, моделей, функцій) для відновлення робочої сесії
Графічні		Збереження візуалізацій у растрових (PNG, JPG) або векторних (PDF) форматах.
Звіти	rmarkdown::render()	Генерація повноцінних звітів (HTML, PDF, DOCX) з інтегрованим кодом R та результатами.

Хід виконання роботи

Підготовчий етап: Створення тестових об'єктів

Створити тестовий набір даних (data.frame) та вектор для подальшого експорту.

1. Створення data.frame

```
results_df <- data.frame(
 ID = 1:5,
```

Name = c("Alpha", "Beta", "Gamma", "Delta", "Epsilon"),

Value = c(10.5, 22.3, 5.1, 18.9, 30.0),

Category = factor(c("A", "B", "A", "C", "B"))

)

2. Створення тестової моделі (для збереження об'єкта R)

test_model <- lm(Value ~ ID, data = results_df)

3. Перевірка робочого каталогу

print(getwd())

Завдання 1. Експорт табличних даних (CSV, TXT)

1. Експорт у стандартний CSV (США/Міжнародний формат): Використати функцію write.csv() для збереження results_df у файл results_std.csv.

Встановити параметр row.names = FALSE.

write.csv(results_df, "results_std.csv", row.names = FALSE)

Перевірити вміст створеного файлу.

2. Експорт у Європейський CSV-формат:

Використати функцію write.csv2() для збереження results_df у файл results_eu.csv.

Звернути увагу на роздільник стовпців (;) та десятковий роздільник (,).

write.csv2(results_df, "results_eu.csv", row.names = FALSE)

3. Експорт у TXT-файл з табуляцією:

Використати функцію write.table() для збереження даних у файл results.txt. Встановити роздільник sep = "\t".

write.table(results_df, "results.txt", sep = "\t", row.names = FALSE)

Перевірити вміст створеного файлу.

Завдання 2. Експорт у формат Excel (XLSX)

1. Встановлення та завантаження пакету openxlsx: Виконати команди для встановлення та завантаження пакету (якщо він ще не встановлений).

install.packages("openxlsx")

library(openxlsx)

2. Запис data.frame в один аркуш Excel:

Використати функцію write.xlsx() для збереження results_df у файл analysis_results.xlsx.

write.xlsx(results_df, "analysis_results.xlsx", sheetName = "Основні_Дані")

3. Запис кількох об'єктів на різні аркуші:

Створити додаткову таблицю summary_table (наприклад, агреговані дані).

Використати write.xlsx() для запису results_df та summary_table на різні аркуші одного файлу.

summary_table <- aggregate(Value ~ Category, data = results_df, FUN = mean)

write.xlsx(

list(Data = results_df, Summary = summary_table),

"multi_sheet_results.xlsx")

Завдання 3. Збереження об'єктів R (RData, RDS)

1. Збереження кількох об'єктів у форматі .RData:

Використати функцію save() для збереження results_df та test_model у файл session.RData.

Видалити ці об'єкти з робочого середовища (rm(results_df, test_model)).

Відновити об'єкти за допомогою функції load().

save(results_df, test_model, file = "session.RData")

rm(results_df, test_model) # Видалення об'єктів

load("session.RData") # Відновлення об'єктів

2. Збереження одного об'єкта у форматі .rds:

Використати функцію saveRDS() для збереження results_df у файл results.rds. Відновити об'єкт за допомогою функції readRDS() та присвоїти його новій змінній restored_df.

saveRDS(results_df, "results.rds")

restored_df <- readRDS("results.rds")

Завдання 4. Експорт графіків.

1. Створення тестового графіку: Створити простий графік розсіювання використавши data.frame створений на підготовчому етапі.

plot(results_df$ID, results_df$Value, main = "Графік залежності Value від ID", xlab = "ID", ylab = "Value")

2. Збереження графіку у форматі PNG (для вебу/презентацій**):** Використати функції png() та dev.off() для збереження графіку у файл plot.png.

png("plot.png", width = 800, height = 600)

plot(results_df$ID, results_df$Value, main = "Графік PNG", xlab = "ID", ylab = "Value")

dev.off()

3. Збереження графіку у форматі PDF (для публікацій): Використати функції pdf() та dev.off() для збереження графіку у файл plot.pdf.

pdf("plot.pdf")

plot(results_df$ID, results_df$Value, main = "Графік PDF", xlab = "ID", ylab = "Value")

dev.off()

Згенерувати звіт у форматі HTML або PDF. Додати згенерований файл до звіту лабораторної роботи.

Інтегровані завдання до розділу 4

Завдання 1. Робота з власним набором даних.

1. Створіть власний data.frame щонайменше з: 10 рядків; 4–5 змінних різних типів (numeric, character, factor, logical).

2. Виконайте елементарний аналіз: обчисліть середнє значення для однієї числової змінної; побудуйте агреговану таблицю за факторною змінною.

3. Збережіть: повний data.frame у форматі CSV; агреговану таблицю у форматі TXT з роздільником «;».

Завдання 2. Експорт у Excel з кількома аркушами.

1. Створіть Excel-файл у форматі XLSX, який містить: аркуш Data – початкові дані; аркуш Summary – агреговані результати; аркуш Info – короткий

текстовий опис набору даних.

2. Використайте пакет openxlsx.

3. Перевірте відкриття файлу у MS Excel або LibreOffice.

Завдання 3. Збереження та відновлення об’єктів R.

1. Побудуйте лінійну модель або іншу статистичну модель на власних даних.

2. Збережіть: модель та data.frame у файл .RData; лише модель у файл .rds.

3. Видаліть об’єкти з робочого середовища та відновіть їх.

4. Поясніть різницю між load() і readRDS() у письмовому звіті.

Завдання 4. Експорт графіків.

1. Побудуйте не менше двох графіків (наприклад, plot, boxplot, hist).

2. Збережіть: один графік у форматі PNG (вкажіть розмір зображення); один графік у форматі PDF.

3. Поясніть, у яких випадках доцільно використовувати: растрові формати; векторні формати.

Завдання 5. Організація робочого середовища.

1. Перевірте поточний робочий каталог за допомогою getwd().

2. Створіть підкаталог export_results.

3. Збережіть усі файли лабораторної роботи саме в цьому каталозі.

4. Опишіть структуру каталогів у звіті.

Завдання 6. (Додаткове, підвищеної складності)

Створіть простий R Markdown-документ (.Rmd), який містить: короткий опис даних; код аналізу; таблицю результатів; графік.

1. Експорт звітних даних:

Створіть два тестові data.frame: sales_2023 (5 рядків, 3 стовпці) та sales_2024 (5 рядків, 3 стовпці).

Збережіть обидва data.frame в один файл Excel під назвою annual_sales.xlsx, розмістивши їх на різних аркушах з відповідними назвами.

2. Обробка та архівування об'єктів:

Створіть вектор v_data з 10 випадкових чисел.

Створіть лінійну модель model_v на основі цих даних.

Збережіть лише об'єкт model_v у файл final_model.rds.

Переконайтеся, що ви можете відновити модель з файлу та вивести її резюме (summary()).

3. Створення графіку з високою роздільною здатністю:

Створіть графік гістограми для 100 випадкових чисел, згенерованих функцією rnorm(). Збережіть цей графік у форматі PNG з роздільною здатністю 1200x800 пікселів. Збережіть цей самий графік у форматі PDF.

4. Використання write_csv:

Встановіть та завантажте пакет readr. Збережіть results_df у файл results_readr.csv за допомогою функції write_csv(). Порівняйте отриманий файл з файлом, створеним за допомогою write.csv().

5. Робочий каталог:

Напишіть команду, яка тимчасово змінить робочий каталог на /home/ubuntu/temp_exports, виконає експорт тестового файлу, а потім повернеться до попереднього каталогу.

Примітка: Для цього може знадобитися зберегти поточний каталог перед зміною.

Приклад для завдання 5:

```
current_dir <- getwd()
setwd("/home/ubuntu/temp_exports")
write.csv(results_df, "temp_file.csv")
setwd(current_dir)
```

Контрольні питання до розділу 4

1. Назвіть основну перевагу формату XLSX для експорту результатів. Який пакет R використовується для роботи з XLSX-файлами?

2. Поясніть різницю між форматами .RData та .rds з точки зору кількості об'єктів, які можна зберегти.

3. У чому полягає відмінність між функціями write.csv() та write.csv2()? Для яких регіональних стандартів використовується кожна з них?

4. Чому важливо правильно визначати типи змінних?

5. Що таке R Markdown і як він забезпечує відтворюваність результатів?

6. Який пакет використовується для імпорту Excel-файлів у R?

7. Які загальні принципи необхідно врахувати перед експортом результатів у R?

8. Які основні типи даних використовуються в R?

9. Які переваги має використання пакету readr (функція write_csv) порівняно з базовою функцією write.csv?

10. Які переваги формату CSV у порівнянні з Excel?

11. Які функції використовуються для відкриття графічного пристрою та його закриття при збереженні графіків у файли (наприклад, PNG або PDF)?

12. Які функції застосовуються для перегляду структури даних?

РОЗДІЛ 5. АНАЛІЗ, ОЧИЩЕННЯ ТА МАНІПУЛЯЦІЯ ДАНИМИ

5.1.Перегляд та опис структури імпортованих даних

Після імпорту даних у середовище RStudio першим і обов'язковим кроком є їх перегляд та опис. Метою цього етапу є формування уявлення про структуру набору даних, правильність зчитування, типи змінних та загальний характер інформації.

Перегляд даних дозволяє:

- перевірити коректність імпорту з файлу;
- виявити помилки форматування;
- визначити числові та категоріальні змінні;
- оцінити обсяг і повноту даних.

Ігнорування цього етапу часто призводить до логічних і статистичних помилок у подальшому аналізі. Для швидкого ознайомлення з даними використовують функції:

head(data)

tail(data)

Ці функції відображають перші або останні рядки таблиці та дають змогу:

- побачити назви змінних;
- оцінити формат значень;
- виявити очевидні помилки (наприклад, текст у числовому полі).

Для детального візуального аналізу в RStudio використовується команда View(data). Вона відкриває таблицю в інтерактивному режимі, подібному до електронних таблиць.

Аналіз структури даних є однією з найважливіших функцій описового аналізу для цього використовують str(data).

Функція str() показує:

- тип об'єкта (наприклад, data.frame);
- кількість рядків і стовпців;
- тип кожної змінної (numeric, character, factor).

Саме на етапі первинного аналізу імпортованих даних у середовищі RStudio найчастіше виявляються невідповідності типів змінних, помилки структури таблиці або некоректне представлення інформації, що зумовлює необхідність їх подальшої корекції. Для оцінювання загальної структури та розмірності набору даних використовуються спеціалізовані функції мови R.

Розмірність набору даних визначається за допомогою функції dim(), яка повертає кількість рядків і стовпців таблиці. Кількість спостережень у наборі даних може бути отримана за допомогою функції nrow(), тоді як функція ncol() дозволяє встановити кількість змінних. Аналіз цих характеристик дає змогу оцінити обсяг даних та перевірити коректність їх завантаження.

Важливим є також аналіз назв змінних та організації доступу до них. Для цього застосовуються функції names() і colnames(), які повертають перелік імен стовпців набору даних. Функція rownames() використовується для перегляду або задання ідентифікаторів рядків, що може бути доцільним під час роботи з впорядкованими або індексованими даними.

Застосування зазначених інструментів дозволяє здійснити первинну перевірку структури даних, виявити можливі помилки імпорту та забезпечити підготовку набору даних до подальших етапів очищення, перетворення і статистичного аналізу.

5.2.Основні статистичні характеристики (середнє, медіана, мода, дисперсія)

Основні статистичні характеристики використовуються для узагальненого опису числових даних та дають змогу кількісно охарактеризувати розподіл змінної. До базових показників описової статистики належать середнє значення, медіана, мода та дисперсія, кожен з яких відображає окремий аспект розподілу даних.

Середнє арифметичне є однією з найпоширеніших характеристик центральної тенденції та обчислюється як сума всіх значень змінної, поділена

на їх кількість. Цей показник є чутливим до наявності екстремальних значень і тому найбільш інформативний у випадках симетричного розподілу даних.

Медіана визначається як значення, що ділить упорядкований ряд даних на дві рівні частини. На відміну від середнього арифметичного, медіана є стійкою до викидів і краще відображає центральну тенденцію у разі асиметричного розподілу або наявності аномальних значень.

Мода характеризує значення, яке трапляється у вибірці найчастіше. Цей показник може бути використаний як для числових, так і для категоріальних даних, а у випадку складних розподілів можливе існування кількох мод або їх відсутність.

Дисперсія є мірою розсіювання значень відносно середнього та визначається як середнє квадратичне відхилення значень змінної від її середнього арифметичного. Вона характеризує ступінь варіативності даних і слугує основою для обчислення інших показників розсіювання, зокрема стандартного відхилення.

У сукупності наведені статистичні характеристики дозволяють отримати цілісне уявлення про розподіл даних, оцінити їх центральну тенденцію та ступінь варіативності, що є необхідним етапом перед проведенням поглибленого статистичного аналізу.

Для отримання узагальнених характеристик змінних у середовищі R застосовується функція summary(), яка забезпечує стислий опис основних статистичних показників набору даних або окремих змінних.

Функція summary() автоматично визначає тип змінної та формує відповідний набір характеристик. Для числових змінних вона повертає мінімальне та максимальне значення, перший і третій квартилі, медіану, а також середнє арифметичне. Для факторних та категоріальних змінних виводиться інформація про кількість спостережень у кожній категорії. У випадку наявності пропущених значень додатково відображається їх кількість.

Наведемо приклад функцій, що використовуються для обчислення основних статистичних характеристик числової змінної age у наборі даних data.

Параметр na.rm = TRUE забезпечує ігнорування пропущених значень під час обчислень, що дозволяє отримати коректні результати.

Функція mean(data$age, na.rm = TRUE) обчислює середнє арифметичне значення змінної age, яке характеризує центральну тенденцію розподілу вікових даних.

Функція sd(data$age, na.rm = TRUE) визначає стандартне відхилення, що відображає ступінь розсіювання значень віку відносно середнього арифметичного. Чим більше значення стандартного відхилення, тим більшою є варіативність даних.

Функція min(data$age, na.rm = TRUE) повертає мінімальне значення змінної age у вибірці, тоді як max(data$age, na.rm = TRUE) визначає її максимальне значення. Зазначені показники дозволяють встановити межі зміни змінної та виявити можливі екстремальні значення.

5.3. Пропущені значення у наборах даних

Пропущені значення є поширеним явищем у реальних наборах даних і виникають унаслідок різних причин, зокрема помилок під час збору інформації, відсутності відповіді респондента, некоректного введення даних або технічних збоїв при імпорті. Наявність пропущених значень може істотно впливати на результати аналізу даних, спотворюючи статистичні характеристики та знижуючи достовірність отриманих висновків.

У середовищі R пропущені значення позначаються спеціальним маркером NA (Not Available). Такі значення не враховуються більшістю статистичних функцій за замовчуванням і потребують явної обробки під час виконання обчислень. Виявлення пропусків у наборі даних є обов’язковим етапом первинного аналізу та передує подальшому очищенню й перетворенню даних.

Для аналізу пропущених значень у R застосовуються спеціальні функції, які дозволяють визначити їх наявність, кількість та розподіл за змінними. Зокрема, використання функції is.na() дає змогу перевірити окремі елементи на

наявність пропусків, тоді як функції sum(is.na()) або colSums(is.na()) дозволяють оцінити їх загальну кількість у наборі даних або в окремих змінних.

Урахування пропущених значень є необхідною умовою коректного статистичного аналізу. Невраховані пропуски можуть призвести до викривлення результатів, тому вибір адекватних методів роботи з ними має методичне та практичне значення.

5.3.1. Методи роботи з пропущеними значеннями

Обробка пропущених значень є важливою на етапы підготовки даних до статистичного аналізу, оскільки наявність пропусків може істотно впливати на точність обчислень і коректність інтерпретації результатів. У практиці аналізу даних застосовують кілька основних підходів до роботи з пропущеними значеннями, вибір яких залежить від обсягу пропусків, їх природи та мети дослідження.

Перед обробкою необхідно визначити наявність та кількість пропусків у даних is.na(data).

Підрахунок кількості пропущених значень у кожній змінній: colSums(is.na(data)).

Загальна кількість пропусків у наборі даних: sum(is.na(data))

1. Видалення спостережень або змінних з пропущеними значеннями.

Найпростішим методом є виключення з аналізу тих спостережень (рядків) або змінних (стовпців), які містять пропущені значення. У середовищі R для цього використовується функція na.omit(), яка видаляє усі рядки, у яких є хоча б одне пропущене значення (NA) та повертає новий датафрейм без пропусків. Зручно використовувати, якщо кількість пропусків мала і їх вилучення не вплине суттєво на результати аналізу, оскільки масове видалення даних може призвести до втрати важливої інформації та зменшення обсягу вибірки.

У разі незначної кількості пропусків можливе повне виключення відповідних спостережень:

data_clean <- na.omit(data).

Також можна видалити лише ті рядки, у яких є пропуски:

```
data_clean <- data[complete.cases(data), ]
```

Функція complete.cases(data) перевіряє кожен рядок, повертає TRUE, якщо в рядку немає жодного NA і FALSE, якщо є хоча б один пропуск.

Конструкція data[complete.cases(data),] залишає лише ті рядки, для яких значення TRUE та фактично також видаляє всі рядки з пропусками.

2. Ігнорування пропущених значень під час обчислень.

Багато статистичних функцій у R дозволяють виконувати обчислення з урахуванням пропусків шляхом використання аргументу na.rm = TRUE. У цьому разі пропущені значення не включаються до розрахунків середнього, стандартного відхилення, мінімального та максимального значень. Такий підхід не змінює структуру набору даних, однак не вирішує проблему пропусків повністю.

Більшість статистичних функцій у R підтримують аргумент na.rm = TRUE, що дозволяє виконувати обчислення без урахування пропусків:

```
mean(data$age, na.rm = TRUE)
sd(data$age, na.rm = TRUE)
min(data$age, na.rm = TRUE)
max(data$age, na.rm = TRUE)
```

3. Заміна пропущених значень статистичними показниками.

Поширеним методом є заповнення пропусків середнім значенням, медіаною або модою відповідної змінної. Цей підхід дозволяє зберегти обсяг вибірки та спростити подальший аналіз, проте може зменшувати варіативність даних і впливати на результати статистичного моделювання.

– Заповнення пропусків середнім значенням змінної:

```
data$age[is.na(data$age)] <- mean(data$age, na.rm = TRUE)
```

– Заміна пропущених значень медіаною (часто використовується для асиметричних розподілів):

```
data$age[is.na(data$age)] <- median(data$age, na.rm = TRUE)
```

– Для факторних змінних пропуски доцільно замінювати найчастішим значенням:

```
mode_value <- names(sort(table(data$gender), decreasing = TRUE))[1]
data$gender[is.na(data$gender)] <- mode_value
```

4. Імпутація пропущених значень.

Більш складним і точним підходом є імпутація, тобто заміна пропущених значень на основі інформації з інших змінних. До таких методів належать регресійна імпутація, метод найближчих сусідів та множинна імпутація. Вони дають змогу зберегти статистичні властивості даних, але потребують додаткових обчислювальних ресурсів і глибшого аналізу.

Простий приклад регресійної імпутації:

```
model <- lm(age ~ score, data = data)
data$age[is.na(data$age)] <- predict(model, data[is.na(data$age), ])
```

У даному прикладі пропущені значення заповнюються прогнозними показниками, отриманими за допомогою лінійної моделі.

5. Аналіз пропущених значень як окремої інформації.

У деяких дослідженнях сам факт наявності пропусків може нести важливу інформацію. У таких випадках пропущені значення не видаляються, а аналізуються як окрема категорія або ознака, що дозволяє виявити приховані закономірності у даних. Видалення пропусків є найпростішим методом підготовки даних na.omit(data).

У деяких випадках доцільно створити додаткову змінну, що фіксує факт пропуску:

```
data$age_missing <- ifelse(is.na(data$age), 1, 0)
```

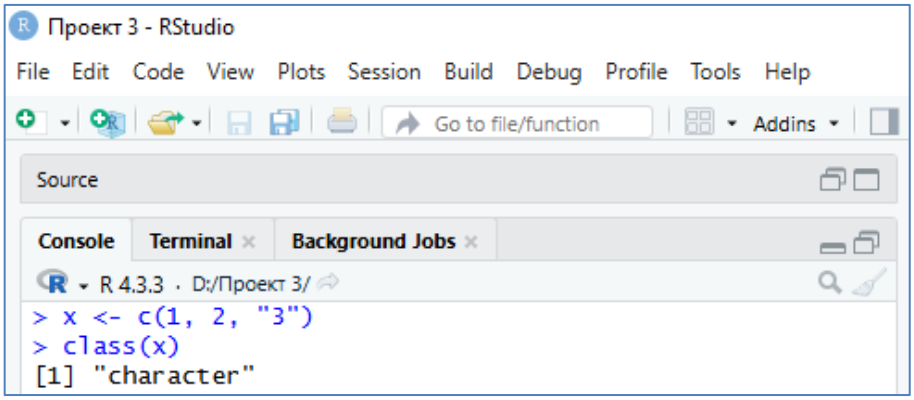
5.3.2. Явне та неявне перетворення типів

Під час аналізу даних у середовищі R часто виникає необхідність змінювати типи змінних з метою забезпечення коректних обчислень, статистичного аналізу та візуалізації результатів. Перетворення типів даних може здійснюватися явно або неявно, залежно від способу виконання операції

та участі користувача.

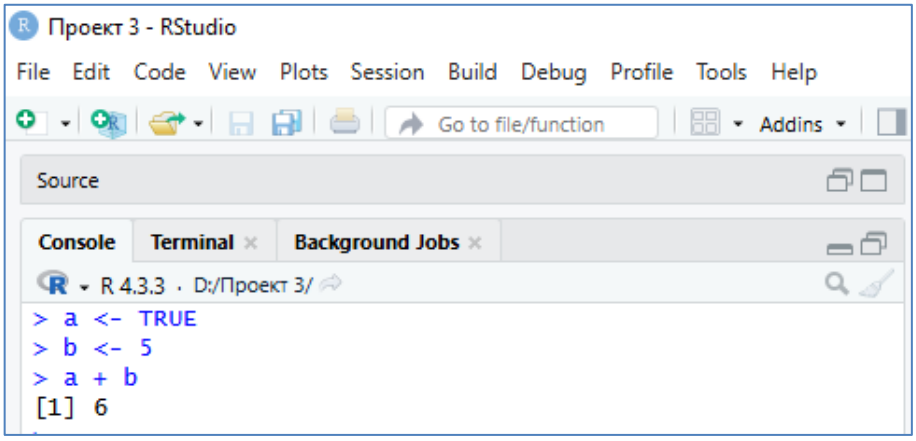
Неявне (автоматичне) перетворення типів відбувається без прямого втручання користувача. R самостійно змінює типи даних для забезпечення виконання операцій, дотримуючись певної ієрархії типів logical → integer → numeric → character.

Приклад неявного перетворення:



У цьому випадку числові значення автоматично перетворюються на символьні, оскільки вектор у R може містити дані лише одного типу.

Приклад арифметичної операції:



Логічне значення TRUE неявно перетворюється на числове 1.

Явне перетворення типів здійснюється за допомогою спеціальних функцій, які явно вказують бажаний тип даних. Такий підхід є рекомендованим у процесі підготовки та очищення даних.

Основні функції явного перетворення:

as.numeric()

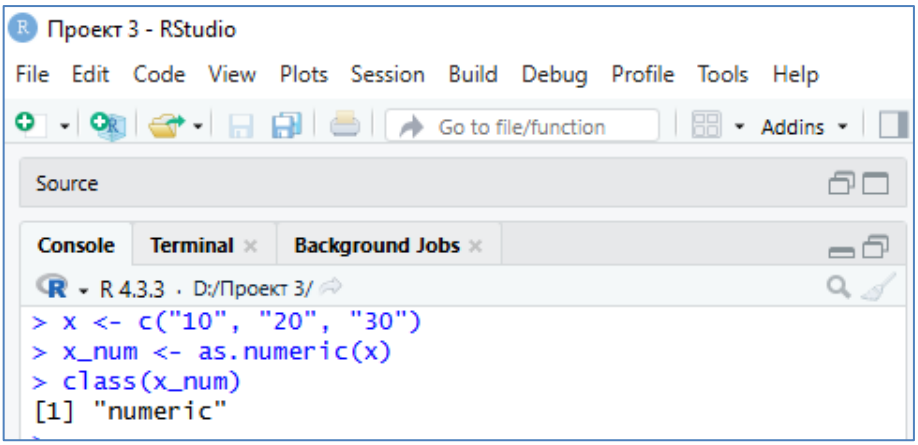
as.integer()

as.character()

as.logical()

as.factor()

Приклад перетворення з символьного типу в числовий:



Перед перетворенням необхідно перевіряти значення, оскільки некоректні дані можуть перетворитися на NA.

5.4. Фактори та їх інтерпретація

У мові програмування R фактори є спеціальним типом даних, призначеним для представлення категоріальних змінних, тобто змінних, значення яких належать до скінченної множини категорій. Фактори широко застосовуються у статистичному аналізі, зокрема в регресійному моделюванні, дисперсійному аналізі та побудові статистичних моделей, оскільки дозволяють коректно враховувати якісні характеристики об’єктів дослідження.

Фактор у R – це категоріальна змінна, яка зберігається як цілий числовий вектор (codes) з атрибутом рівнів (levels), що визначають відповідність між числовими кодами та категоріями.

Наприклад, змінна стать може містити рівні «чоловіча» та «жіноча», які у факторі кодуються відповідними числовими індексами.

gender <- factor(c("чоловіча", "жіноча", "жіноча", "чоловіча"))

levels(gender)

Фактори можуть створюватися безпосередньо за допомогою функції factor() або шляхом перетворення символьних змінних.

data$group <- factor(data$group)

Також можливе явне задання порядку рівнів:

grade <- factor(c("D", "B", "A", "C"), levels = c("D", "C", "B", "A"),

ordered = TRUE)

У цьому випадку створюється упорядкований фактор, що дозволяє виконувати операції порівняння.

У регресійних моделях фактори інтерпретуються через фіктивні (dummy) змінні, де один із рівнів обирається як базовий (референтний), а інші порівнюються з ним.

Наприклад, у лінійній моделі:

lm(score ~ group, data = data)

Для аналізу розподілу категоріальних даних використовується функція table(data$grade).

Відносні частоти можна отримати за допомогою prop.table(table(data$grade)).

Однією з поширених помилок є некоректне перетворення фактора у числовий тип, що приводить до втрати змісту даних.

- Некоректно: as.numeric(data$grade)
- Коректно: as.numeric(as.character(data$grade))

5.5. Очищення та фільтрація даних

Якість вихідних даних безпосередньо впливає на достовірність результатів дослідження, тому на цьому етапі здійснюється виявлення та усунення помилок, пропущених значень, дублювань, а також відбір релевантних спостережень відповідно до мети аналізу.

Очищення даних передбачає комплекс процедур, спрямованих на приведення набору даних до коректного та узгодженого вигляду. Основними завданнями цього етапу є:

- виявлення та обробка пропущених значень;
- усунення або корекція аномальних і некоректних значень;

– приведення змінних до відповідних типів даних;

– видалення дубльованих спостережень.

На етапі очищення часто виконується перетворення значень змінних, зокрема нормалізація числових показників, заміна некоректних значень або приведення даних до єдиного формату.

Приклад:

```
data$age[data$age < 0] <- NA
data$age <- as.numeric(data$age)
```

Пояснення:

Заміна некоректних значень на NA

```
data$age[data$age < 0] <- NA
```

В першому рядку відбувається наступне:

data$age < 0 – створює логічний вектор (TRUE для від’ємних значень).

data$age[...] <- NA – усі значення віку, які менші за 0, замінюються на NA, оскільки вік не може бути від’ємним, тому це помилка введення і відбувається заміна на пропущене значення.

Перетворення у числовий тип data$age <- as.numeric(data$age). Цей рядок перетворює змінну age у числовий тип (numeric).

Фільтрація даних полягає у відборі підмножини спостережень, що відповідають заданим умовам. Вона дозволяє зосередити аналіз на релевантних даних та спростити інтерпретацію результатів.

Фільтрація рядків за логічною умовою:

```
data_adults <- data[data$age >= 18, ]
```

Фільтрація за кількома умовами:

```
data_selected <- data[data$age >= 18 & data$gender == "жіноча", ]
```

Вибір окремих змінних:

```
data_subset <- data[, c("name", "age", "score")]
```

Для відбору спостережень за категоріальними змінними застосовується фільтрація за рівнями факторів:

```
data_B <- data[data$grade == "B", ]
```

Комбінація логічних операторів дозволяє створювати складні критерії відбору.

5.6. Основні способи об’єднання наборів даних

Об’єднання наборів даних використовується під час аналізу інформації, отриманої з різних джерел або сформованої на різних етапах дослідження. У середовищі R існує кілька базових способів об’єднання таблиць даних, вибір яких залежить від структури наборів даних, наявності спільних змінних та поставлених аналітичних завдань.

1. Вертикальне об’єднання (додавання рядків) застосовується у випадках, коли необхідно об’єднати набори даних з однаковою структурою (однакові змінні та їх типи), але з різними спостереженнями.

У базовому R для цього використовується функція rbind():

data_all <- rbind(data1, data2)

Умовою коректного виконання операції є збіг назв і типів стовпців у обох таблицях.

2. Горизонтальне об’єднання (додавання стовпців) передбачає приєднання нових змінних до наявного набору даних. Такий підхід застосовується, коли таблиці містять однакову кількість спостережень та відповідають один одному за порядком рядків.

Для цього використовується функція cbind():

data_extended <- cbind(data1, data2)

Цей спосіб є доцільним лише за умови повної відповідності рядків у таблицях.

5.7.Частотний аналіз категоріальних даних

Частотний аналіз є методом дослідження категоріальних (якісних) даних, який дозволяє оцінити розподіл спостережень за окремими категоріями, виявити домінуючі та рідкісні значення, а також зробити первинні узагальнення щодо структури вибірки. У середовищі R частотний аналіз широко

застосовується на етапі попереднього аналізу даних та підготовки їх до подальших статистичних процедур.

1. Абсолютні частоти

Для обчислення абсолютних частот категоріальної змінної використовується функція table():

table(data$grade)

Результат відображає кількість спостережень, що належать до кожної категорії змінної.

2. Відносні частоти

Для оцінювання частки кожної категорії у загальній сукупності застосовується функція prop.table():

prop.table(table(data$grade))

Отримані значення подаються у вигляді часток від 1. Для представлення у відсотках результати можна помножити на 100:

prop.table(table(data$grade)) * 100

3. Частотний аналіз факторів

Оскільки категоріальні змінні в R зазвичай подаються у вигляді факторів, функція table() автоматично враховує їх рівні:

table(data$gender)

Для впорядкованих факторів частотний аналіз дозволяє також оцінити розподіл спостережень з урахуванням порядку рівнів.

4. Двовимірні таблиці частот

Для аналізу взаємозв’язку між двома категоріальними змінними використовуються контингентні таблиці:

table(data$gender, data$grade)

Такі таблиці дозволяють досліджувати спільний розподіл категорій і є основою для подальших статистичних перевірок, зокрема критерію χ^2.

5.8.Перегляд даних у форматі tibble

У сучасному аналізі даних у середовищі R все більшого поширення

набуває формат tibble, який є розширеною та вдосконаленою версією класичного data.frame. Об’єкти типу tibble забезпечують зручніші засоби перегляду даних, підвищену читабельність та кращу інтеграцію з пакетами екосистеми tidyverse, зокрема dplyr, tidyr та ggplot2.

На відміну від звичайного датафрейму, tibble за замовчуванням:

– відображає лише перші рядки набору даних;

– показує типи даних для кожної змінної;

– не виводить усі рядки, що запобігає перевантаженню консолі;

– не змінює типи даних автоматично (наприклад, рядки не перетворюються на фактори).

Для створення tibble використовується функція tibble() з пакета tibble:

```
library(tibble)
tb <- tibble( name = c("Іван", "Олена"),
  age  = c(18, 19)
)
tb
```

При виведенні об’єкта в консоль RStudio автоматично відображається компактне та структуроване подання даних.

Для ознайомлення зі структурою tibble застосовується функція str():

```
str(tb)
```

Вона дозволяє визначити кількість спостережень, змінних та їх типи.

Перші рядки tibble можна переглянути за допомогою:

```
head(tb)
```

Окремі змінні доступні за їх назвами:

```
tb$age
```

Для детального перегляду даних у табличному вигляді використовується функція View(tb).

Цей режим дозволяє здійснювати зручну навігацію та швидкий огляд великих наборів даних.

На відміну від data.frame, tibble не використовує автоматичне приведення

типів і не змінює назви змінних, що зменшує ризик помилок під час аналізу даних. Це робить tibble більш передбачуваним і зручним форматом для наукових і прикладних досліджень.

5.9.Функції загального використання

У процесі роботи з даними в середовищі R важливу роль відіграють функції загального використання, які застосовуються на різних етапах аналізу незалежно від предметної області. Ці функції забезпечують базові операції зі змінними, об’єктами, наборами даних і середовищем виконання, а також слугують основою для більш складних процедур статистичного аналізу та моделювання.

Таблиця 5.1.

Функції середовища RStudio.

RSiteSearch("термін")	пошук в Інтернеті
example(ім'я_функції)	виводить приклад для обраної функції
library(readr) library(dplyr) library(ggplot2)	підключення бібліотек необхідних для виконання команд
read.delim() і read.delim2()	функції читання файлів із врахуванням формату запису чисел
rm()	видалення змінної
ls() або objects()	виводять створені протягом сесії об’єкти
rm(ім’я_об’єкта)	видаляє із пам’яті об’єкт із зазначеним ім’ям
history(n)	буде виведене нове вікно, у якому перераховані n останніх команд
source('ім’я_файлу.R')	збереження R-коду в поточній директорії у файлі із зазначеним ім’ям і розширенням .R
find("ім'я функції")	пошук функції
sink('ім’я_файлу.розширення')	перенаправляє потік виводу з екрану в зазначений файл; повторний виклик команди sink() закриває файл

print(z, digits=3)	обмежує виведення на екран не більше трьох значущих цифр
%>%	дозволяє застосувати наступну команду до результатів виконання поточної
arrange(n)	сортує за вказаним параметром
filter(параметр == 2)	фільтрує дані при виведенні
rnorm(n,mean,sd)	генерація послідовності n випадкових чисел, що мають нормальний розподіл з математичним очікуванням mean і середнім квадратичним відхиленням sd
ім’я об’єкта$ім’я	атрибуту формат звернення до змінної
atributes(my_data)	повертають структуру та атрибути даних
sapply(data, class)	визначити типи атрибутів
unique(data$ccc)	унікальні значення, які приймає змінна
cor()	кореляція
cov()	Коваріація показує, як дві числові змінні змінюються разом: > 0 – зростають разом; < 0 – одна зростає, інша зменшується; ≈ 0 – зв’язок відсутній
na.omit(data)	якщо хоча б в одному стовпці рядка є NA, увесь рядок буде вилучено

Лабораторні роботи до розділу 5

Лабораторна робота 1. Перегляд та опис структури даних у середовищі RStudio

Мета роботи:

- навчитися переглядати імпортовані дані в RStudio;
- аналізувати структуру набору даних;
- визначати типи змінних;

- виконувати описову статистику числових і категоріальних змінних.

Теоретичні відомості

Після імпорту даних першим етапом аналізу є перегляд і опис структури набору даних. Це дозволяє переконатися в коректності зчитування інформації, визначити типи змінних та виявити помилки форматування.

Для цього в R використовуються функції:

- head(), tail() – швидкий перегляд даних;
- View() – табличний перегляд;
- str() – аналіз структури;
- summary() – описова статистика.
- data – вхідні дані попередньо імпортовані із CSV або Excel-файл у RStudio.

Хід виконання роботи

Завдання 1. Перегляд даних

1. Виведіть перші 6 рядків набору даних head(data)
2. Перегляньте останні 6 рядків tail(data)
3. Відкрийте таблицю в інтерактивному режимі View(data)

Проаналізуйте:

- назви змінних;
- формат значень;
- наявність очевидних помилок.

Завдання 2. Аналіз структури набору даних

1. Виконайте команду str(data)
2. Зафіксуйте:
   - кількість рядків і стовпців;
   - тип кожної змінної (numeric, character, factor).

Зверніть увагу, чи немає числових даних, зчитаних як текст.

Завдання 3. Вивчення показників описової статистики

1. Отримайте загальну статистичну характеристику summary(data)
2. Для числових змінних визначте:
   - мінімум;
   - максимум;
   - середнє;
   - медіану.
3. Для факторних змінних проаналізуйте частоти категорій.

Лабораторна робота 2. Очищення, перетворення та фільтрація даних у RStudio

Мета роботи: виявляти та коректно обробляти пропущені значення у наборах даних, що є необхідною умовою забезпечення достовірності подальшого аналізу.

Теоретичні відомості

Пропущені значення (NA) є поширеною проблемою в реальних даних. Їх некоректна обробка може призвести до хибних результатів. Також важливу роль відіграє правильне задання типів даних, зокрема факторів. data – вхідні дані попередньо імпортовані із CSV або Excel-файл у RStudio.

Хід виконання роботи

Завдання 1. Виявлення пропущених значень

1. Визначте, де є пропуски is.na(data)
2. Розділіть масиву data у R на дві частини

```
set.seed(123)
idx <- sample(seq_along(data), size = floor(length(data) * 0.5))
data1 <- data[idx]
data2 <- data[-idx]
```

3. Підрахуйте кількість пропусків у кожній змінній:

```
colSums(is.na(data1))
colSums(is.na(data1))
```

Зробіть висновок про масштаб проблеми.

Завдання 2. Обробка пропущених значень

1. Створіть набір даних без пропусків data_clean <- na.omit(data1)
2. Заповніть пропуски в числовій змінній медіаною:

data$score[is.na(data2$score)] <- median(data$score, na.rm = TRUE)

Порівняйте розміри наборів даних до і після обробки.

Завдання 3. Перетворення типів даних

1. Перетворіть змінну на числову Data1$score <- as.numeric(data$score)
2. Перетворіть текстову змінну на фактор

Data2$group <- as.factor(data$group)

3. Повторно виконайте аналіз структури:

str(data1); str(data2);

Завдання 4. Фільтрація даних.

1. Відіберіть спостереження за однією умовою:

data_adult <- data[data1$age >= 18,]

2. Виконайте фільтрацію за двома умовами:

data_filtered <- data[data2$age >= 18 & data$score > 80,]

Умови сформуйте відповідно до наявних даних. Проаналізуйте, як змінився обсяг даних.

Завдання 5. Внутрішнє об’єднання таблиць.

1. Об’єднайте два набори даних за спільним ідентифікатором id:

merged_data <- merge(data1, data2, by = "id")

2. Перевірте структуру утвореного об’єкту:

str(merged_data)

Лабораторна робота 3. Робота з об’єктами, бібліотеками та імпорт даних

Мета роботи: навчитися підключати необхідні бібліотеки, що розширюють функціональні можливості середовища R та забезпечують доступ до спеціалізованих інструментів аналізу даних. Також формується здатність створювати, переглядати та видаляти об’єкти, що є основою

ефективної роботи з даними та керування робочим середовищем.

Теоретичні відомості

Усі дані в R зберігаються у вигляді об’єктів. Для розширення функціональності використовуються бібліотеки library(), які підключають додаткові можливості аналізу та візуалізації.

Хід виконання роботи

Завдання 1. Підключіть бібліотеки та використовуючи help, уточніть їх призначення.

```
library(readr)
library(dplyr)
library(ggplot2)
```

Завдання 2. Імпортуйте дані із CSV-файлу в об’єкт data1.

```
data1 <- read.csv("data.csv", sep = ",", header = TRUE)
```

Завдання 3. Імпортуйте дані у табличній формі.

```
data2 <- read.table("data.tsv", sep = "\t", header = TRUE)
```

Завдання 4. Вивчіть структуру та властивостей імпортованих об’єктів:

```
ls()
objects()
str(),class(),summary()
```

та інші відомі вам функції.

Завдання 5. Опрацюйте об’єкти виконавши наступні операції.

1. замініть NA об’єкта data1 на середнє значення або моду.
2. відфільтруйте data2 за обраним параметром у об’єкт data3.
3. rm(data3) видаліть відфільтровані дані.
4. Обчисліть критерії χ^2 між обраними змінними.

Завдання 6. Самостійно сформулюйте та виконайте аналітичні операції із об’єктами які збереглись в проєкті. Перевірте структуру отриманих об’єктів та опишіть їх у висновку.

Аналіз результатів

Сформуйте висновок: які табличні дані було імпортовано, опишіть структуру даних, середовища та відпрацьовано навички керування об’єктами.

Лабораторна робота 4. Маніпуляція даними та функції загального використання

Мета роботи: навчитися самостійно фільтрувати та сортувати дані; використовувати оператор %>%; застосовувати функції apply, sapply, lapply; аналізувати структуру даних.

Теоретичні відомості

Для ефективної роботи з наборами даних у R застосовують функції пакета dplyr, а також функції сімейства apply.

Використовуючи сайт Kaggle підготуйте CSV файл.

Проведіть дослідження даних та підготуйте їх до проведення аналітичних досліджень.

Хід виконання роботи

Завдання 1. Вивчіть структуру даних отриманого файлу. Виконайте фільтрацію та сортування даних за числовим параметром.

```
filtered <- data %>% filter(age > 18)
sorted <- filtered %>% arrange(score)
```

Поля “age” та “score” замініть іменами наявними у ваших даних.

Завдання 2. Вивчіть призначення запропонованої функції, опишіть її необхідність в системі аналізу даних.

```
unique(data$group)
```

Завдання 3. Запропоновано дві функції: вивчіть атрибути та типи даних; виведіть структуру даних перед аналізом; перевірте відсутність даних.

```
attributes(data)
sapply(data, class)
```

Завдання 4. Для того щоб отримати швидкі описові статистики для кількох змінних застосовують функції:

```
lapply(data[,1:3], mean)
```

```
sapply(data[,1:3], sd)
summary(data)
```

Використайте їх із різними параметрами та опишіть результат.

Завдання 5. Розділіть об’єкт data на дві частини.

```
idx <- sample(length(data), 0.7 * length(data))
train <- data[idx]
test <- data[-idx]
```

Вивчіть та опишіть результат такого поділу та їх доцільність.

Аналіз результатів роботи

Опишіть опрацьовані інструменти маніпуляції даними, оцініть відфільтровані та агреговані дані, чи підвищує така підготовка продуктивність аналітичної роботи.

Лабораторна робота 5. Статистичний аналіз, генерація даних та експорт результатів

Мета роботи:

- генерувати випадкові дані;
- обчислювати кореляцію та коваріацію;
- очищати дані від пропусків;
- експортувати результати.

Теоретичні відомості

R широко використовується для статистичного аналізу, зокрема для роботи з випадковими величинами, оцінювання взаємозв’язків між змінними та підготовки результатів для звітів.

Хід виконання роботи

Завдання 1. Згенеруйте дані та дослідіть їх.

```
set.seed(123)
z<-sample(1:10, 10)
x <- rnorm(100, mean = 50, sd = 10)
y <- rnorm(100, mean = 45, sd = 8)
```

Завдання 2. Коваріація (cov) показує напрям спільної зміни змінних, тоді як кореляція (cor) – нормалізовану силу та напрям лінійного зв'язку між ними. Використовуючи запропоновані функції обчисліть кореляцію та ковариацію між об'єктами. Поясніть отримані результати.

```
cor(x, y)
cov(x, y)
cor(x, y, method = "kendall")
cor(x, y, method = "spearman")
```

Завдання 3. Виконайте запропоновану групу команд. Вивчіть результати роботи кожної із команд.

```
set.seed(123)
n <- rnorm(10)
n[sample(1:10, 3)] <- NA
clean_data <- na.omit(n)
```

Завдання 4. Експорт результатів роботи програми можна виконати наступною групою команд. Команди sink() дозволяють перенаправити текстовий вивід функцій R із консолі у файл.

```
sink("results.txt")
summary(clean_data)
sink()
```

*Важливо **завжди закривати sink()**, інакше консоль залишиться «порожньою».*

Завдання 5. Оператор %>% (pipe) використовується для передачі результату попереднього виразу як першого аргументу наступної функції. Виконайте групу команд та вивчіть отримані результати.

```
data.frame(x, y) %>%
 filter(x > y) %>%
 summarise( mean_x = mean(x), mean_y = mean(y) )
```

Сформувати висновок

1. Інтерпретувати отримані числові характеристики та механізм збереження результатів аналізу у файл.
2. Сформувала практичні навички окремих критеріїв статистичного аналізу.
3. Описати технологію експорту результатів у RStudio.

Контрольні питання до розділу 5

1. Для чого використовується library()?
2. Для чого використовується na.omit()?
3. Для чого використовується оператор %>%?
4. Для чого використовується оператор pipe?
5. Для чого використовується функція merge()?
6. Для чого потрібно знати параметри даних?
7. Чим відрізняється head() від View()?
8. Чому важливо правильно визначати типи даних?
9. Чому фактори важливі для аналізу даних?
10. Що показує коефіцієнт кореляції?
11. Що таке NA у мові R?
12. Що таке пропущені значення (NA) і чому вони небезпечні?
13. Як вивчити структуру об’єкта?
14. Як видалити дані та об’єкти із проекту?
15. Як визначити типи змінних?
16. Як відфільтрувати дані об’єкта?
17. Як здійснюється фільтрація даних за умовами?
18. Як очистити пам’ять від об’єктів?
19. Як перенаправити вивід у файл?
20. Яка мета етапу перегляду і опису даних у R?
21. Яка різниця між read.csv() і read.table()?
22. Які технології об’єднання таблиць ви знаєте?
23. Які основні методи обробки пропусків ви знаєте?
24. Які способи обробки пропущених значень існують?
25. Які типи об’єднання таблиць існують і чим вони відрізняються?
26. Яку інформацію надає функція str()?
27. Яку роль відіграють фактори в аналізі даних?

РОЗДІЛ 6. ФУНКЦІОНАЛЬНЕ ПРОГРАМУВАННЯ В R

6.1.Основи функціонального програмування та його роль у середовищі RStudio

Функціональне програмування є парадигмою програмування, на якій ґрунтується мова R. У межах цього підходу процес обчислення розглядається як послідовність застосування функцій до даних, а основна увага зосереджується не на зміні стану програми, а на перетворенні вхідних значень у вихідні результати.

Особливістю функціонального програмування є використання функцій як основних елементів програми. У мові R функції є об’єктами першого класу, тобто вони можуть зберігатися у змінних, передаватися як аргументи іншим функціям та повертатися як результат виконання. Це забезпечує високу гнучкість коду та сприяє побудові модульних і повторно використовуваних програмних рішень.

Важливою складовою функціонального стилю в R є векторизація обчислень. Вона дозволяє виконувати операції одразу над цілими векторами або масивами даних без явного використання циклів. Завдяки цьому код стає компактнішим, зрозумілішим та значно ефективнішим з точки зору швидкодії. Усі базові арифметичні та логічні операції в R підтримують векторизовану обробку даних.

Ще одним інструментом функціонального програмування в R є сімейство функцій apply. Ці функції надають зручний механізм застосування заданої операції до елементів структурованих даних, таких як вектори, списки, матриці та таблиці даних. Використання функцій apply(), lapply(), sapply(), vapply(), tapply() та mapply() дозволяє уникнути громіздких циклів і підвищує читабельність програмного коду.

Середовище RStudio забезпечує зручні інструменти для створення, налагодження та тестування функцій, підсвічування синтаксису, автодоповнення коду та інтерактивний перегляд результатів обчислень.

Інтеграція з пакетами tidyverse, зокрема з бібліотекою purrr, розширює можливості функціонального підходу та сприяє формуванню єдиного стилю роботи з даними.

6.2.Структура функцій, аргументи та механізми області видимості

У мові R функції призначені для інкапсуляції обчислень і повторного використання коду. Кожна функція має чітко визначену структуру, яка включає список аргументів, тіло функції та значення, що повертається в результаті її виконання. Опис функції здійснюється за допомогою слова function, після якого в дужках задаються аргументи, а в фігурних дужках – послідовність інструкцій, що формують логіку обчислень.

Аргументи функції визначають вхідні дані та можуть мати значення за замовчуванням, що забезпечує гнучкість виклику функцій. У мові R підтримується передавання аргументів за позицією та за іменем, що дозволяє змінювати порядок їх задання без порушення коректності виконання. Аргументи можуть набувати будь-якого допустимого типу даних, зокрема векторів, списків, таблиць даних або інших функцій.

Важливим критерієм функціонування програм у R є механізм області видимості змінних. Мова R реалізує лексичну (статичну) область видимості, за якої значення змінних визначається середовищем, у якому функція була створена. Під час виконання функції система послідовно здійснює пошук змінних спочатку в локальному середовищі функції, а за їх відсутності – у зовнішніх середовищах, пов’язаних із моментом оголошення функції.

Застосування лексичної області видимості забезпечує передбачуваність поведінки програмного коду, зменшує ймовірність конфліктів імен змінних та сприяє створенню модульних і надійних програм. Чітке розуміння структури функцій, принципів роботи з аргументами та механізмів області видимості є необхідною умовою ефективного використання функціонального програмування в мові R.

6.3.Розгалуження та цикли в мові R

Розгалуження та циклічні конструкції є елементами алгоритмічної логіки в мові R і використовуються для керування порядком виконання програмних інструкцій залежно від заданих умов або для багаторазового повторення обчислень. Застосування цих конструкцій дає змогу реалізовувати алгоритми різного рівня складності та поєднувати імперативний і функціональний підходи програмування.

Розгалуження в мові R реалізується за допомогою умовного оператора if. Він дозволяє виконувати певний вираз або блок інструкцій лише за умови істинності логічного виразу. Для векторизованої перевірки умов і формування результатів на основі логічних значень застосовується функція ifelse(), яка забезпечує компактний і ефективний спосіб обробки векторних даних.

Циклічні конструкції в мові R призначені для багаторазового виконання одного й того самого блоку коду. Найпоширенішим є цикл for, який використовується для ітерації по елементах вектора, послідовності або списку. Він є зручним у випадках, коли кількість повторень відома заздалегідь. Цикл while виконує блок інструкцій доти, доки задана умова залишається істинною, що доцільно застосовувати в ситуаціях із невизначеною кількістю ітерацій. Цикл repeat створює нескінченний процес виконання, вихід з якого здійснюється за допомогою оператора break у поєднанні з умовною перевіркою.

У мові R цикли можуть доповнюватися керуючими операторами break та next. Оператор break забезпечує негайне завершення виконання циклу, тоді як next пропускає поточну ітерацію та переходить до наступної. Такі механізми дозволяють гнучко керувати ходом обчислень і оптимізувати виконання програм.

Хоча цикли є універсальним інструментом реалізації повторюваних обчислень, у мові R перевага часто надається векторизованим операціям і функціональному підходу. Проте розуміння принципів роботи розгалужень і циклів є необхідним для коректної реалізації алгоритмів, аналізу логіки програм і ефективного поєднання різних стилів програмування.

6.3.1. Умовні оператори if, ifelse

Оператор if використовується для перевірки одиничної логічної умови та виконання відповідного виразу або блоку коду. Він застосовується у випадках, коли умова має скалярне логічне значення (TRUE або FALSE). Стандартна форма оператора if може бути доповнена гілкою else, що дозволяє реалізувати альтернативний сценарій виконання програми. Така конструкція є основним інструментом для побудови розгалужень у алгоритмах. Загальна форма оператора має вигляд if (умова) { вираз }, а у випадку необхідності вибору між двома альтернативами використовується конструкція if (умова) { вираз1 } else { вираз2 }.

Функція ifelse призначена для векторизованої перевірки умов і використовується тоді, коли необхідно одночасно обробити кілька значень. Вона приймає три аргументи: логічну умову, значення для випадку її виконання та значення для випадку невиконання. Результатом роботи ifelse є вектор, у якому кожен елемент формується відповідно до перевірки умови для відповідного елемента вхідних даних. Такий підхід забезпечує компактність коду та високу ефективність обчислень.

Основна відмінність між if та ifelse полягає в тому, що оператор if працює з одиничними логічними значеннями і використовується для керування потоком виконання програми, тоді як ifelse орієнтована на обробку векторів і реалізує функціональний стиль програмування. Коректний вибір між цими конструкціями сприяє підвищенню читабельності коду, його ефективності та відповідності принципам мови R.

6.3.2. Цикли for, while, repeat

Циклічні конструкції в мові R використовуються для організації багаторазового виконання одного й того самого блоку коду та реалізації алгоритмів, у яких необхідно здійснювати повторювані обчислення, та дозволяють контролювати порядок і кількість ітерацій залежно від заданих

умов.

Цикл for призначений для послідовної ітерації за елементами вектора, списку або іншої впорядкованої послідовності. Кількість повторень у цьому випадку визначається наперед структурою даних, по якій здійснюється перебір. Цикл for є зручним для обробки колекцій даних і часто застосовується під час реалізації алгоритмів з відомою кількістю ітерацій. Загальна форма циклу має вигляд for (змінна in послідовність) {вираз}.

Цикл while(умова) {вираз} виконує блок інструкцій до тих пір, доки логічна умова залишається істинною. Кількість ітерацій у цьому випадку не визначається заздалегідь і залежить від результатів обчислень, що виконуються всередині циклу. Така конструкція доцільна для реалізації алгоритмів, у яких завершення процесу пов’язане з досягненням певного критерію або граничного значення.

Цикл repeat {вираз} забезпечує безумовне багаторазове виконання блоку коду і створює нескінченний цикл. Завершення його виконання можливе лише за допомогою оператора break, який зазвичай використовується у поєднанні з умовним оператором. Цикл repeat застосовується у випадках, коли умову завершення неможливо або недоцільно перевіряти на початку кожної ітерації.

6.4. Векторизація як основа ефективного програмування в R

У мові R більшість базових арифметичних, логічних і статистичних операцій реалізовані у векторизованому вигляді. Це означає, що одна інструкція може застосовуватися до кожного елемента структури даних автоматично. Векторизація ґрунтується на внутрішніх оптимізаціях інтерпретатора R, що забезпечують швидше виконання операцій порівняно з покомпонентною обробкою за допомогою циклів.

Векторизація тісно пов’язана з функціональним стилем програмування, який є характерним для мови R. Функції ifelse(), pmin(), pmax(), а також функції сімейства apply реалізують векторизовану обробку даних і слугують

альтернативою імперативним циклам. Їх застосування є доцільним у більшості задач аналізу даних.

Векторизація – це можливість виконання операцій одразу над цілим вектором даних без явного використання циклів for.

У R всі базові арифметичні операції векторизовані.

Приклад:

```
x <- c(1, 2, 3, 4)
y <- x * 2   # множення на скаляр
z <- x + c(10, 20, 30, 40) # вектор + вектор
```

У цьому прикладі множення і додавання відбуваються над усім вектором без жодного циклу.

Переваги векторизації:

- значно швидше, ніж цикли;
- простота коду;
- читабельність та зниження кількості помилок.

Таблиця 6.1.

Порівняння застосування циклів та векторизації

З циклом	**Векторизовано**

res <- c()
for (i in 1:length(x)) {
res[i] <- x[i] * 2
}
	res <- x * 2

Функції, які підтримують векторизацію:

- +, -, *, /
- sqrt(), log(), exp(), abs()
- ifelse(), pmin(), pmax()

Приклад з ifelse():

```
age <- c(17, 20, 15, 19)
status <- ifelse(age >= 18, "Дорослий", "Неповнолітній")
```

6.5. Сімейство функцій apply як альтернатива циклам

Сімейство функцій apply у мові R призначене для обробки структурованих даних без явного використання циклів. Ці функції забезпечують застосування заданої операції до елементів векторів, списків, матриць і таблиць даних, що дозволяє реалізувати алгоритми у компактній та зрозумілій формі.

1. Функція apply() використовується для застосування функції до рядків або стовпців матриці чи об’єкта типу data.frame. Вона приймає як аргументи масив даних, напрям обробки та функцію, яка виконує обчислення. Такий підхід дає змогу швидко отримувати агреговані характеристики, зокрема суму, середнє значення або стандартне відхилення, без необхідності реалізації вкладених циклів.

Формат:

apply(X, MARGIN, FUN, ...)

- X – матриця або data.frame;
- MARGIN = 1 – по рядках;
- MARGIN = 2 – по стовпцях;
- FUN – функція (наприклад, mean, sum, sd, min).

Приклад:

```
m <- matrix(1:9, nrow = 3)
Сума по рядках
apply(m, 1, sum)
Середнє по стовпцях
apply(m, 2, mean)
```

apply() повертає вектор або матрицю результатів.

2. Функція lapply() призначена для роботи зі списками та векторами і завжди повертає результат у вигляді списку. Вона є універсальним засобом ітеративної обробки елементів складних структур даних, зберігаючи їхню початкову організацію.

Формат:

lapply(X, FUN)

– X – список або вектор;

– Повертає список результатів.

Приклад:

lst <- list(a = 1:3, b = 10:15)

lapply(lst, mean)

Якщо ви хочете залишити структуру списку – використовуйте lapply().

3. Функція sapply() є спрощеним варіантом lapply() і автоматично намагається перетворити результат у вектор або матрицю, якщо це можливо.

Формат:

sapply(X, FUN)

Спроба спростити результат: перетворити список у вектор або матрицю. У серйозних пакетах і функціях (краще vapply())

Приклад:

sapply(lst, mean)

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

Часто використовується, коли всі елементи дають один тип результату (наприклад, число).

4. Функція vapply(), яка вимагає явного задання шаблону значення повернення, використовується для забезпечення контролю типу та розміру результатів використовується ня. Такий механізм підвищує надійність програмного коду та зменшує ризик помилок під час обробки даних.

Формат:

vapply(X, FUN, FUN.VALUE)

FUN.VALUE – шаблон результату (наприклад, numeric(1)).

5. Функція tapply() застосовується для групової обробки векторів на основі факторів і використовується для обчислення статистичних характеристик у межах окремих груп.

Формат:

tapply(X, INDEX, FUN)

– X – числовий вектор;

– INDEX – фактор або вектор групування;

– FUN – функція.

Приклад:

```
age <- c(18, 19, 17, 21, 20)
group <- c("A", "B", "A", "B", "B")
tapply(age, group, mean)
  A    B
17.5 20.0
```

6. Функція mapply() дозволяє виконувати паралельне застосування функції до кількох аргументів одночасно.

Формат:

```
mapply(FUN, ..., MoreArgs = NULL)
```

Приклад:

```
mapply(sum, c(1, 2), c(10, 20)) # 1+10, 2+20
```

Використання функцій сімейства apply забезпечує лаконічність і виразність коду, зменшує його складність і підвищує читабельність. У порівнянні з класичними циклами for, цей підхід краще узгоджується з концепцією векторизації та функціонального програмування, що робить його доцільним для більшості задач аналізу даних у мові R.

Вибір конкретної функції сімейства apply визначається структурою вхідних даних, вимогами до формату результату та необхідним рівнем контролю над типами даних.

Таблиця 6.3.

Порівняльна характеристика apply-функцій

Функція	**Для чого призначена**	**Вхідні дані**	**Результат**
apply()	по рядках або стовпцях	матриця / data.frame	вектор/матриця
lapply()	для списків /	список / вектор	список

	векторів		
sapply()	як lapply(), але з спрощенням	список / вектор	вектор / матриця
vapply()	як sapply(), але з контролем	список / вектор	визначений тип
tapply()	групування та агрегація	вектор + група	таблиця
mapply()	багатовимірна обробка	кілька векторів	вектор / список

Приклад на data.frame:

```
df <- data.frame(
 math = c(80, 90, 85),
 physics = c(70, 75, 80),
 chemistry = c(90, 95, 100)
)
```

Середнє по рядках (студентам)

```
apply(df, 1, mean)
```

Середнє по стовпцях (предметам)

```
apply(df, 2, mean)
```

Сімейство apply – набір потужних функцій, які дозволяють обробляти таблиці, списки та вектори без зайвого кодування.

Кожна функція з apply має свою специфіку. Важливо знати, яку структуру ви обробляєте**,** щоб обрати правильну функцію.

Лабораторні роботи до розділу 6

Лабораторна робота 1. Функціональне програмування в R. Цикли та розгалуження.

Мета роботи: сформувати уявлення про функціональне програмування в R, навчитися створювати власні функції, застосовувати

умовні оператори if та ifelse, використовувати цикли for і while у поєднанні з функціями, а також порівнювати імперативний і функціональний підходи до програмування.

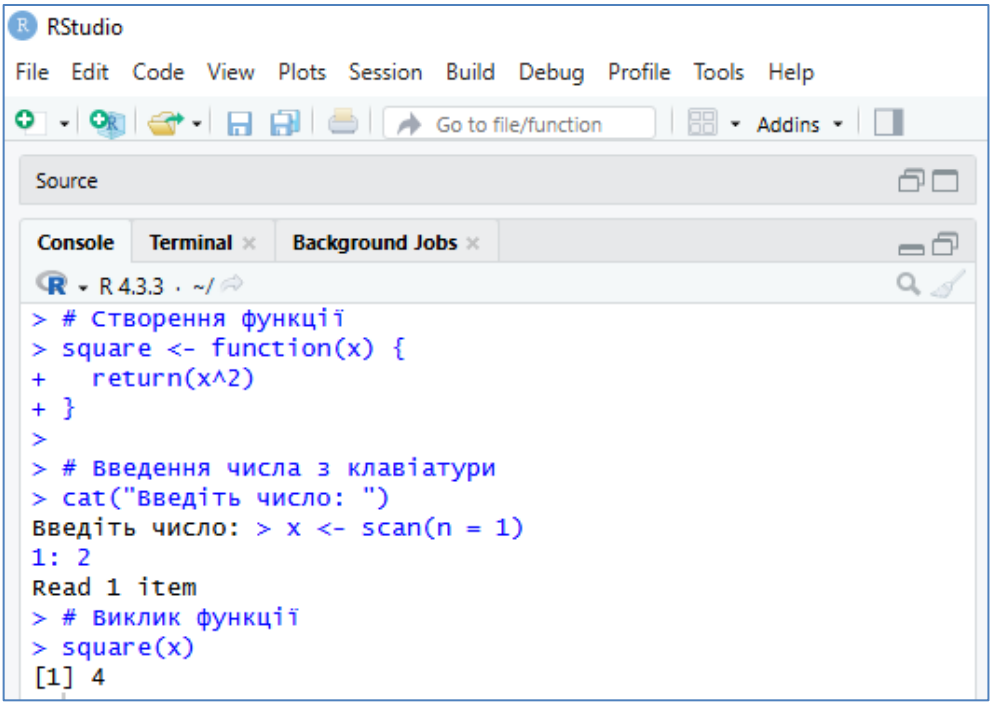
Теоретичні відомості

Функціональне програмування в R – це підхід, за якого:

- основною одиницею є функція;
- функції можуть приймати інші функції як аргументи;
- мінімізується використання глобальних змінних;
- код стає більш модульним, зрозумілим і повторно використовуваним.

Хід виконання роботи

Завдання1. Створення простої функції. Створити функцію, яка обчислює квадрат числа.



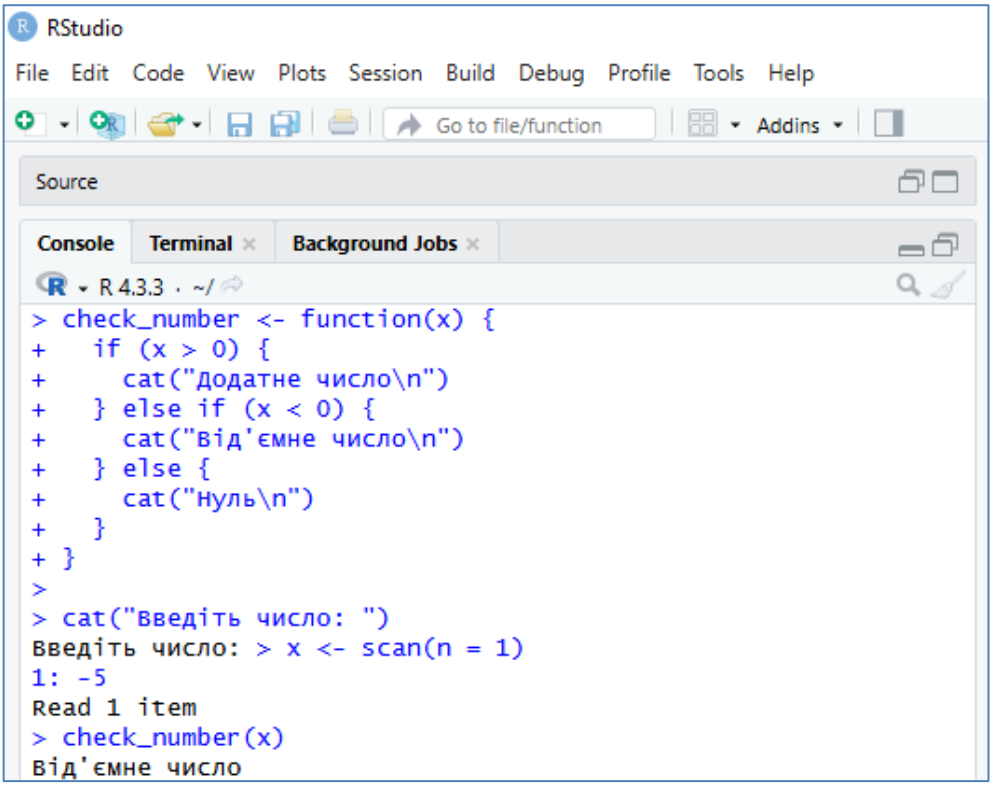
Пояснення:

scan(n = 1) – зчитує одне число з клавіатури

x <- scan(n = 1) – зберігає введене значення у змінну x

square(x) – обчислює квадрат введеного числа.

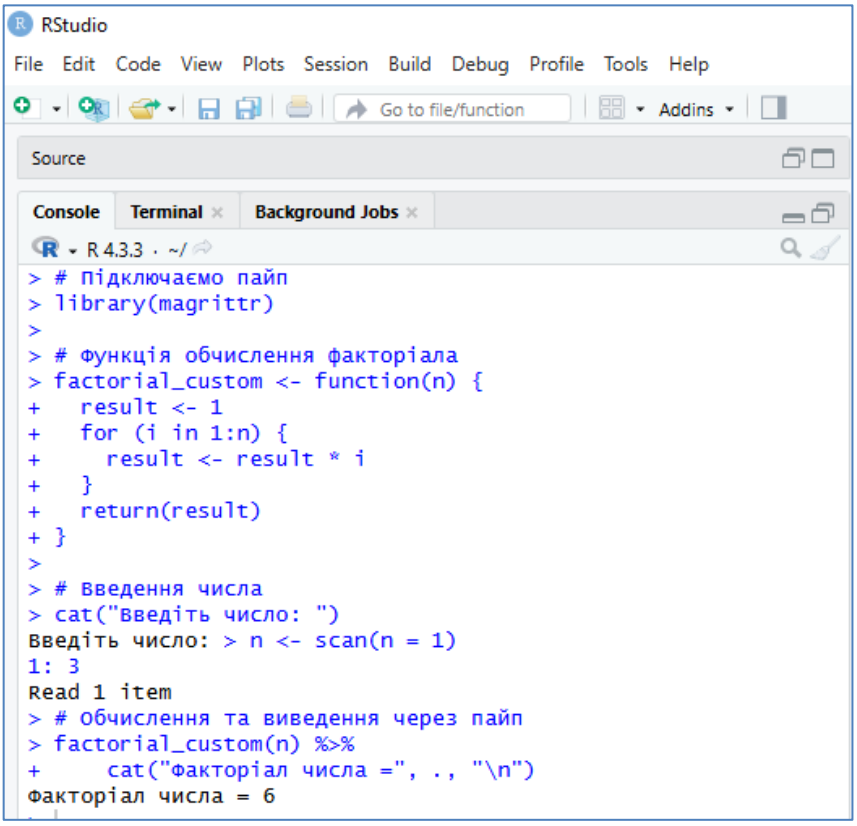
Завдання 2. Функція з розгалуженням (if). Створити функцію, яка визначає, чи число додатне, від'ємне або нуль.



Пояснення: Оператор if ... else if … else дозволяє реалізувати логіку вибору.

Завдання 3. Функція з циклом for. Створити функцію, яка обчислює факторіал числа.

– цикл for використовується для повторення обчислень;
– функція поєднує функціональний та імперативний стиль.



Пояснення:

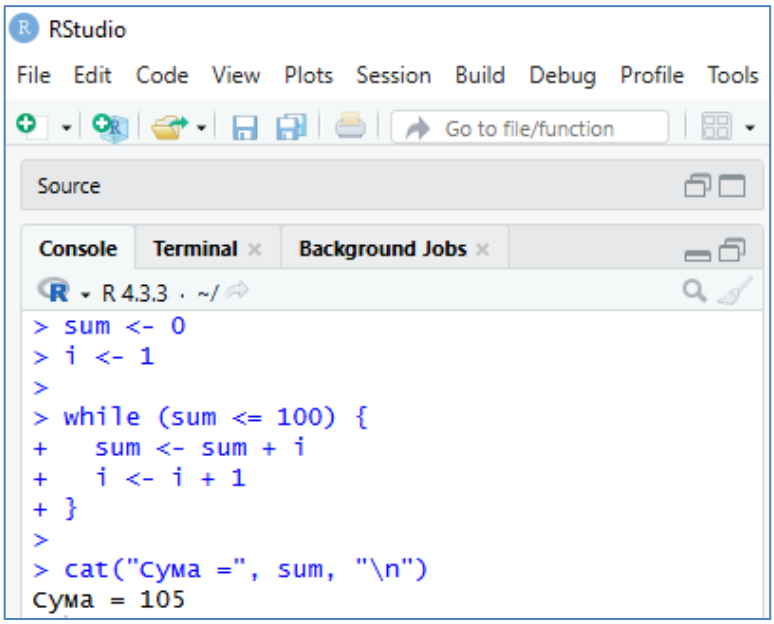
scan(n = 1) – зчитує одне число з клавіатури

%>% – передає результат функції далі

. – означає передане значення

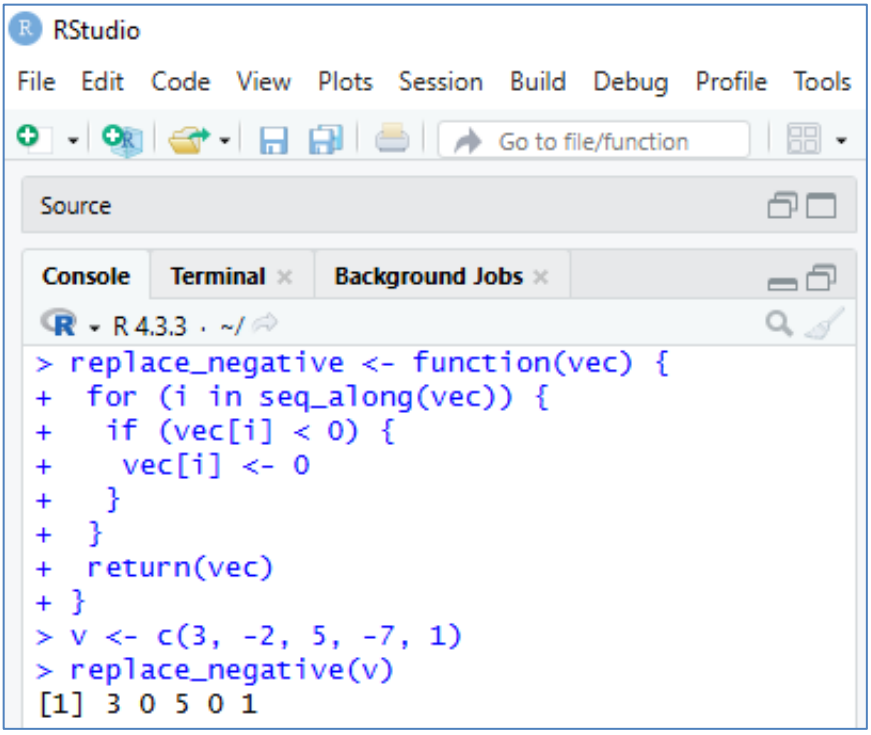
cat() – виводить текст у консоль

Завдання 4. Використовуючи функцію з циклом while знайти суму чисел, поки вона не перевищить 100.



Пояснення: Цикл while виконується, поки умова істинна.

Завдання 5. Функція з вектором та умовами. Створити функцію, яка замінює всі від'ємні значення вектора на 0.



Пояснення:

seq_along() – безпечний спосіб ітерації;

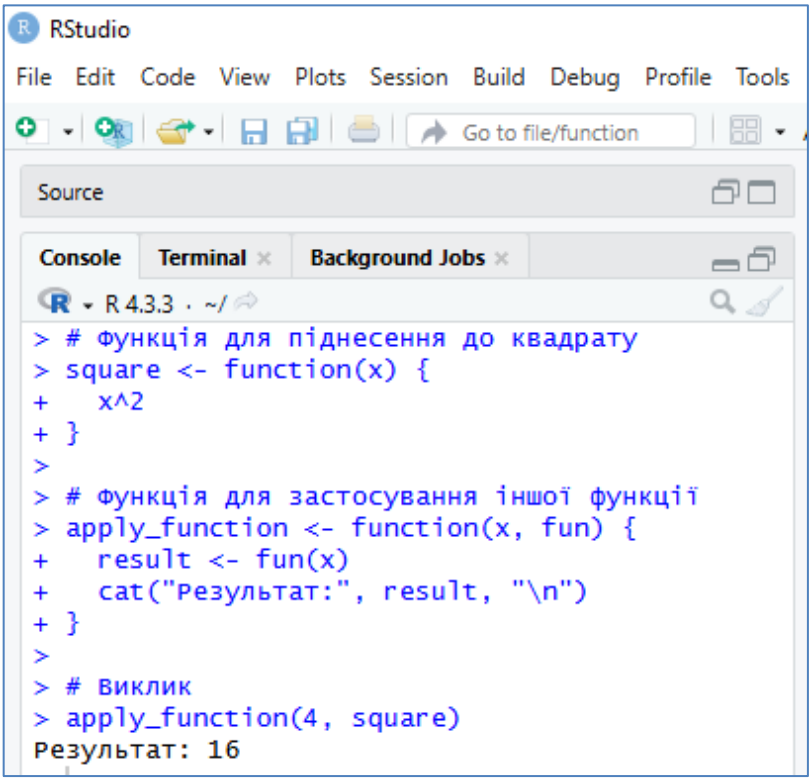
відбувається покомпонентна перевірка.

Завдання 6. Альтернатива: умовний оператор ifelse()

```
replace_negative2 <- function(vec) {
 return(ifelse(vec < 0, 0, vec))
}
replace_negative2(v)
```

Пояснення: ifelse() – векторизований умовний оператор (типово функціональний стиль).

Завдання 7. Функція, що приймає іншу функцію. Передати функцію як аргумент.



Пояснення: Це основна ідея функціонального програмування.

fun(x) обчислює результат (у цьому випадку $4^2 = 16$).

cat("Результат:", result, "\n") виводить його у зручному форматі.

Завдання 8. Порівняння: обчислення із використанням циклу та функціонального підходу.

Цикл

result <- numeric(length(v))

for (i in seq_along(v)) {

result[i] <- v[i]^2

}

result

Функціонально

sapply(v, function(x) x^2)

Лабораторна робота 2. Використання функцій apply(), lapply(), sapply(), vapply(), tapply(), mapply() у мові R

Мета роботи: набути практичних навичок використання функцій сімейства apply() для обробки даних у середовищі RStudio, навчитися обирати доцільну функцію залежно від типу даних і задачі аналізу.

Теоретичні відомості

Функція	Призначення
apply()	Застосування функції до рядків або стовпців матриці
lapply()	Застосування функції до елементів списку, результат – список
sapply()	Спрощений варіант lapply(), повертає вектор або матрицю
vapply()	Строго типізований варіант sapply()
tapply()	Груповий аналіз за фактором
mapply()	Паралельне застосування функції до кількох аргументів

Хід виконання роботи

Завдання 1. Функція apply(). Виконайте завдання, оцініть отримані результати.

Створіть матрицю розміром 5×4 з випадкових чисел:

```
set.seed(123)
mat <- matrix(sample(1:100, 20), nrow = 5)
```

Обчисліть середнє значення:

для кожного стовпця;

для кожного рядка.

```
apply(mat, 2, mean)  # по стовпцях
apply(mat, 1, mean)  # по рядках
```

Завдання 2. Функція lapply().Виконайте завдання, оцініть отримані результати.

Створіть список з трьох числових векторів:

```
lst <- list(a = 1:5, b = 6:10, c = c(2, 4, 6))
```

Для кожного елемента списку:

знайдіть суму;

знайдіть довжину.

lapply(lst, sum)

lapply(lst, length)

Зверніть увагу: результат – список.

Завдання 3. Функція sapply().Виконайте завдання, оцініть отримані результати.

Застосуйте sapply() до того ж списку:

sapply(lst, sum)

Порівняйте результат lapply() та sapply().

Створіть data.frame та визначте тип кожного стовпця:

```
df <- data.frame(
  Name = c("A", "B", "C"),
  Score = c(80, 90, 85),
  Passed = c(TRUE, TRUE, FALSE)
)
sapply(df, class)
```

Завдання 4. Функція vapply(). Виконайте завдання та дайте відповіді на питання.

Повторіть попереднє завдання, використовуючи vapply():

vapply(df, class, character(1))

Поясніть, чому vapply() вважається безпечнішою функцією.

Завдання 5. Функція tapply(). Поясніть результати отримані після виконання пакета команд.

Створіть вектори:

```
scores <- c(80, 85, 90, 75, 88, 92)
group <- factor(c("A", "A", "B", "B", "A", "B"))
```

Знайдіть:

середній бал у кожній групі;

кількість спостережень у групах.

tapply(scores, group, mean)

tapply(scores, group, length)

Завдання 6. Функція mapply(). Оцініть ефективність використання функцій.

Створіть два вектори:

x <- c(1, 2, 3, 4)

y <- c(10, 20, 30, 40)

Обчисліть суму відповідних елементів:

mapply(sum, x, y)

Використайте власну функцію:

f <- function(a, b) a^2 + b

mapply(f, x, y)

Інтегровані завдання до розділу 6

Завдання 1. Створити data.frame з оцінками трьох студентів з трьох предметів.

1. Вивести:
   - середнє по кожному предмету;
   - середній бал кожного студента;
   - кількість предметів, де оцінка > 85;
2. Застосувати tapply() для обчислення середнього віку студентів у групі.
3. Створити функцію, яка знаходить максимум у заданому векторі (через цикл).

Завдання 2. Створити матрицю розмірністю 10×5.

1. Знайти:
   - мінімум у кожному рядку;
   - максимум у кожному стовпці.
2. Для data.frame з 5 числових стовпців:
   - обчислити стандартне відхилення кожного стовпця за допомогою sapply();

– повторити з vapply().

3. Створити фактор з трьома рівнями та застосуйте tapply() для обчислення медіани.

4. Проаналізувати отримані результати та зробити узагальнений висновок щодо переваг застосування функцій сімейства apply порівняно з використанням явних циклів у мові R.

Контрольні питання до розділу 6

1. Що таке функціональне програмування та які його переваги в мові R?

2. Які переваги використання функцій і яку роль вони відіграють у модульності програмного коду в RStudio?

3. Чим функціональний стиль програмування відрізняється від імперативного підходу?

4. Опишіть структуру функції в мові R та призначення її аргументів.

5. Що таке область видимості змінних і як вона реалізується в R?

6. У чому різниця між умовними операторами *if* та *ifelse*?

7. Коли доцільно використовувати цикл *while*?

8. Охарактеризуйте принципи векторизації та її значення для ефективного програмування.

9. Для яких типів даних застосовується функція *apply()* і які результати вона повертає?

10. У чому полягає різниця між функціями *apply()* та *lapply()*?

11. Які відмінності між функціями *lapply()*, *sapply()* та *vapply()*?

12. Чому функція *sapply()* не рекомендується для розробки пакетів?

13. Яку роль відіграє аргумент *FUN.VALUE* у функції *vapply()*?

14. Для яких задач використовується функція *tapply()*?

15. Поясніть принцип роботи функції *mapply()*.

16. Які переваги мають функції сімейства *apply* порівняно з використанням циклу *for*?

РОЗДІЛ 7. ФУНКЦІЇ ДЛЯ РОБОТИ З ВЕКТОРАМИ

7.1. Вектори як базова структура даних у мові R

Однією з особливостей мови програмування R є те, що вона спочатку проєктувалася як векторно-орієнтоване обчислювальне середовище. На відміну від багатьох традиційних мов програмування, де базовою одиницею даних є окреме скалярне значення, в R вектор є фундаментальним типом даних, з яким виконуються практично всі обчислення. До векторів в R можна застосовувати як арифметичні, так і функціональні оператори. Функціональні оператори застосовуються до кожної компоненти вектора окремо.

Вектори в R використовуються для зберігання числових, логічних та символьних даних і є основою для побудови більш складних структур – матриць, факторів, таблиць даних (data frame) та масивів. Саме тому ефективна робота з векторами є необхідною умовою для опанування аналізу даних, статистичних обчислень і математичного моделювання в середовищі RStudio. Компоненти вектора мають містити дані однакових типів.

Функції для роботи з векторами дозволяють:

– створювати та ініціалізувати вектори різними способами;

– виконувати поелементні арифметичні й логічні операції;

– застосовувати математичні функції до всіх елементів вектора одночасно (векторизація);

– здійснювати відбір, сортування та перетворення даних;

– отримувати узагальнені характеристики (сума, середнє, мінімум, максимум тощо).

7.2.Створення та ініціалізація векторів

Використання вбудованих векторних функцій дає змогу значно підвищити продуктивність обчислень, зменшити обсяг програмного коду та зробити його більш наочним і читабельним. Завдяки механізму автоматичної векторизації, R виконує багато обчислювальних операцій без явного

використання циклів, що є однією з головних переваг мови.

Найпоширенішим способом створення векторів є використання функції c() (combine), яка об'єднує окремі значення в єдину послідовність. Тип елементів вектора визначається автоматично відповідно до правил приведення типів у мові R.

c() функція що формує вектор в системі R:

y <- c("Й","Ц","У","К","Е").

Звернення до координат вектора x позначається як *X25 <- x[2:5]* – дані від 2 до 5.

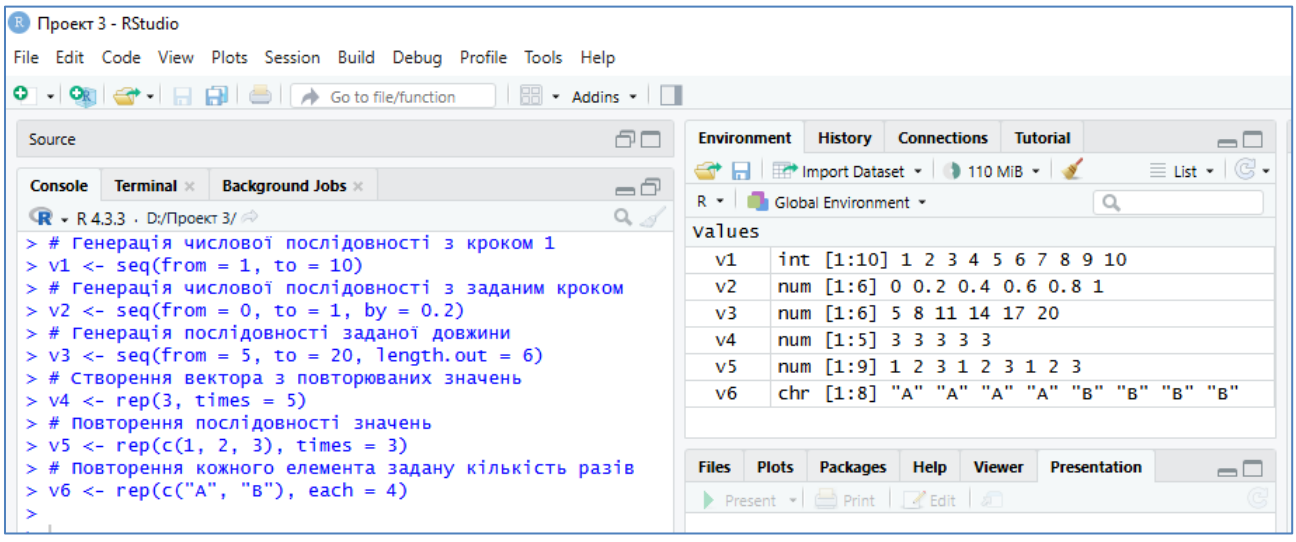
Для створення векторів заданого типу та розміру застосовують функцію vector(), а також спеціалізовані функції numeric(), character() і logical(). Такі функції дають змогу ініціалізувати вектори з наперед визначеною довжиною та значеннями за замовчуванням, що є доцільним під час підготовки структур для подальшого заповнення.

Повний формат функції:

vector(mode = "тип даних", довжина): j <- vector("character", 10)

Генерація числових послідовностей здійснюється за допомогою функції seq(), яка дозволяє задавати початкове та кінцеве значення, а також крок зміни елементів. Для створення повторюваних значень використовується функція rep(), що дає можливість формувати вектори шляхом багаторазового відтворення окремих елементів або цілих послідовностей.

Приклад використання функцій seq() та rep() у мові R:



У наведених прикладах функція seq() використовується для формування числових послідовностей з різними параметрами, тоді як функція rep() забезпечує створення векторів шляхом повторення окремих елементів або цілих наборів значень.

У середовищі R створення векторів може здійснюватися як детермінованим способом (через c(), seq(), rep()), так і стохастичним – шляхом генерації випадкових чисел (runif(), rnorm(), sample() тощо).

Генерація випадкових чисел базується на псевдовипадковому алгоритмі. Це означає, що послідовність чисел є детермінованою функцією початкового стану генератора. Саме цей початковий стан задається функцією set.seed(seed), де seed – ціле число, що визначає стартову точку генератора.

Якщо вектор ініціалізується випадковими значеннями, його елементи змінюватимуться при кожному запуску програми. Використання set.seed() гарантує отримання однакової послідовності чисел при повторному виконанні коду.

Приклад без set.seed()

```
v <- runif(5)
```

При кожному запуску результат буде різним.

Приклад із set.seed()

```
set.seed(123)
v <- runif(5)
```

Повторний запуск цього коду дасть ідентичний результат.

Приклад рівномірного розподілу:

```
set.seed(100)
v1 <- runif(10, min = 0, max = 1)
```

Створюється вектор довжини 10 з рівномірно розподіленими значеннями на інтервалі [0,1].

Приклад нормального розподілу:

```
set.seed(100)
v2 <- rnorm(10, mean = 5, sd = 2)
```

Вектор ініціалізується значеннями з нормального розподілу з параметрами:

математичне сподівання = 5

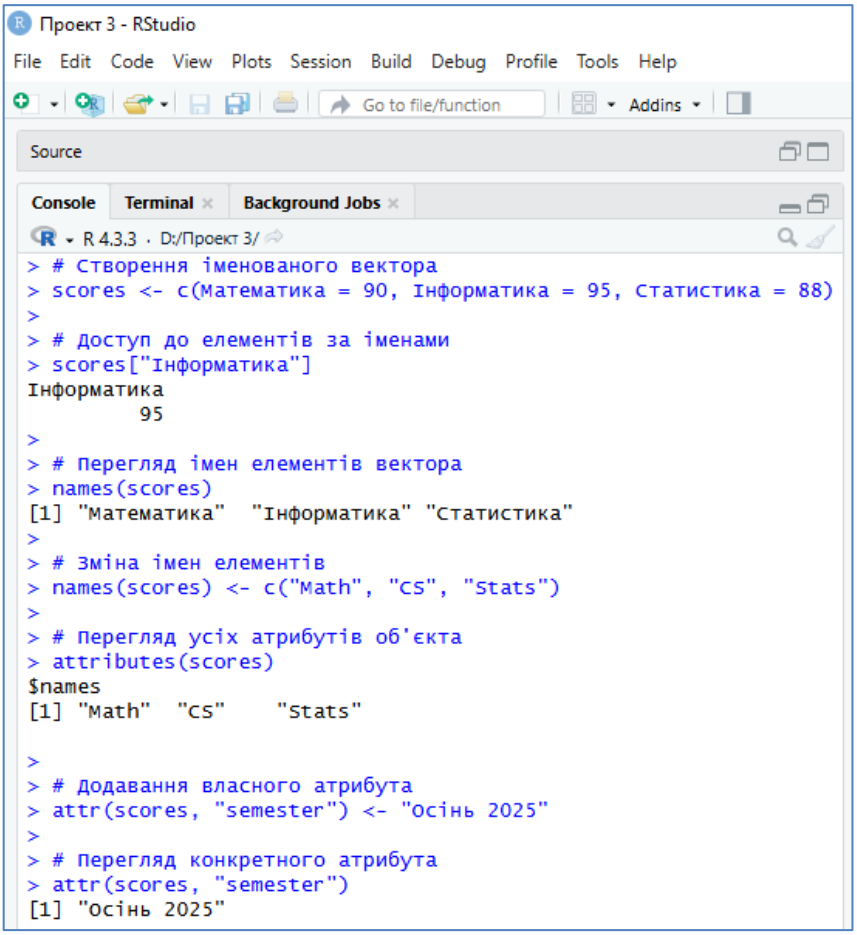
стандартне відхилення = 2

Приклад випадкової вибірки з наявного вектора:

```
set.seed(50)
v3 <- sample(1:20, 5)
```

Формується випадкова підвибірка з чисел від 1 до 20.

Приклад використання іменованих векторів та атрибутів у мові R:



Іменовані вектори є різновидом векторів у мові R, елементам яких присвоєні унікальні імена. Використання імен підвищує зрозумілість коду та полегшує доступ до окремих значень без необхідності звернення за числовими індексами. Призначення імен елементам вектора здійснюється за допомогою функції names(), а також безпосередньо під час створення вектора. Імена зберігаються як атрибут об’єкта і не впливають на його тип або довжину.

Атрибути в R використовуються для зберігання додаткової службової

інформації про об’єкт, зокрема імен елементів, розмірів або класу. Робота з атрибутами здійснюється за допомогою функцій attributes(), attr() та class(). Застосування атрибутів дає змогу розширювати можливості базових структур даних без зміни їхньої внутрішньої організації.

7.3.Основні характеристики та перевірки векторів

Основні характеристики та перевірки векторів – набір функцій та операцій у R, що дозволяє визначати властивості векторів, їх типи та структуру, а також перевіряти відповідність певним умовам перед виконанням обчислень чи аналізу даних.

Для ефективної роботи з векторами в мові R важливо мати можливість швидко визначати їхні основні характеристики та виконувати перевірки коректності даних. Такі операції є необхідними для попереднього аналізу, очищення та підготовки даних.

Однією з характеристик вектора є його довжина, яка визначається за допомогою функції length(). Тип даних вектора можна перевірити за допомогою функцій typeof() та class(), що дає змогу коректно обирати подальші методи обробки.

Перевірка на наявність пропущених значень є важливим етапом аналізу даних. Для цього використовуються функції is.na(), anyNA() та complete.cases(), які дозволяють виявляти, локалізувати або виключати пропуски у векторах. Окремо слід виділяти перевірку на спеціальні числові значення, зокрема NaN та Inf, за допомогою функцій is.nan() та is.infinite().

Додатково у процесі аналізу можуть застосовуватися функції логічних перевірок, такі як any() і all(), які дозволяють оцінювати виконання умов для всіх або окремих елементів вектора. Для з’ясування унікальності значень використовуються функції unique() та duplicated().

До основних характеристик і перевірок належать:

typeof(x) – визначає базовий тип елементів (integer, double, character, logical);

mode(x) – режим зберігання даних;

class(x) – клас об’єкта (numeric, character, factor і т.д.).

length(x) – кількість елементів у векторі.

names(x) – імена елементів вектора (якщо присвоєні);

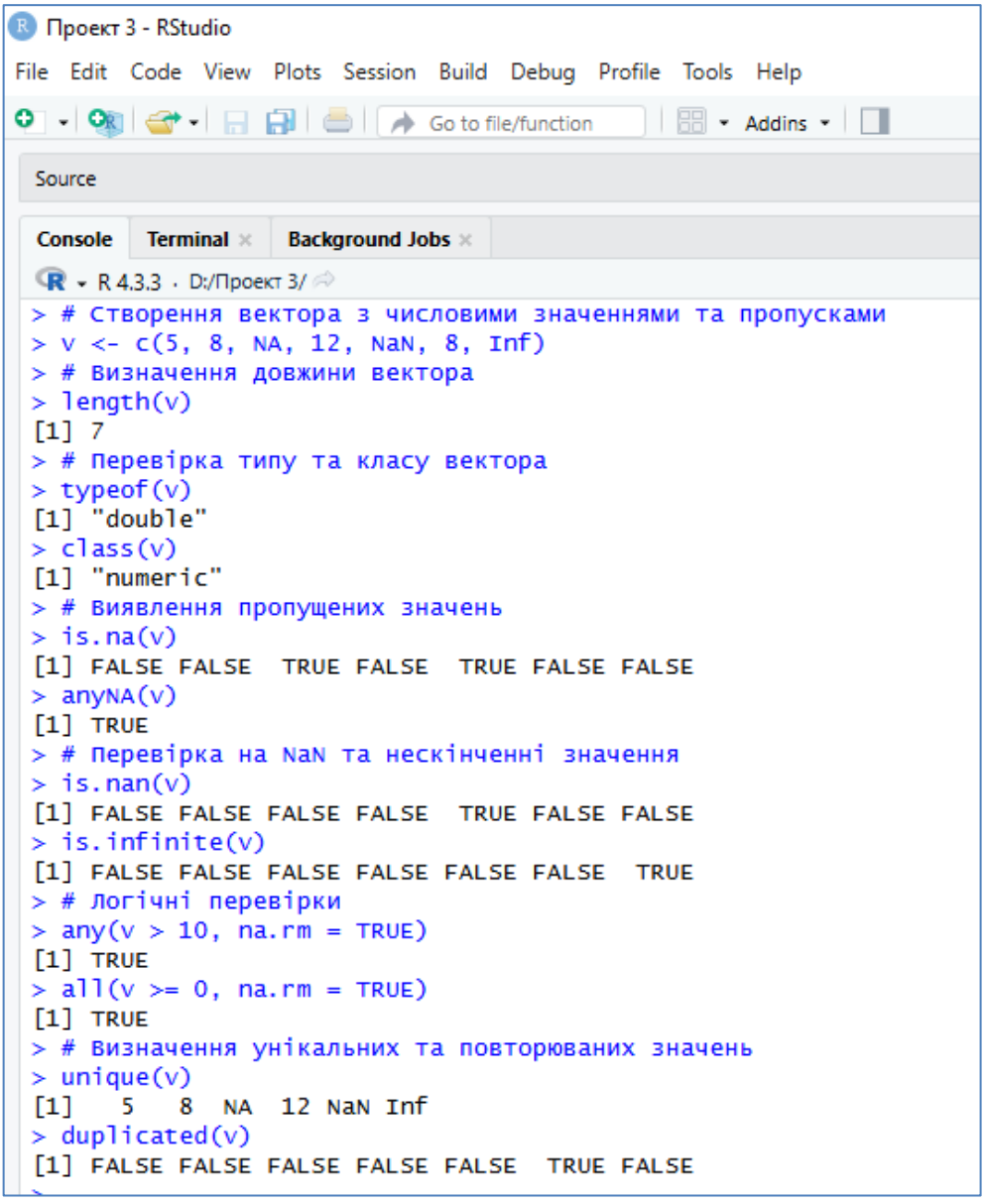
str(x) – відображає структуру об’єкта.

is.numeric(x), is.character(x), is.logical(x) – перевірка типу даних;

is.na(x) – перевірка на відсутні значення;

anyDuplicated(x) – перевірка на повтори;

unique(x) – отримання унікальних значень.



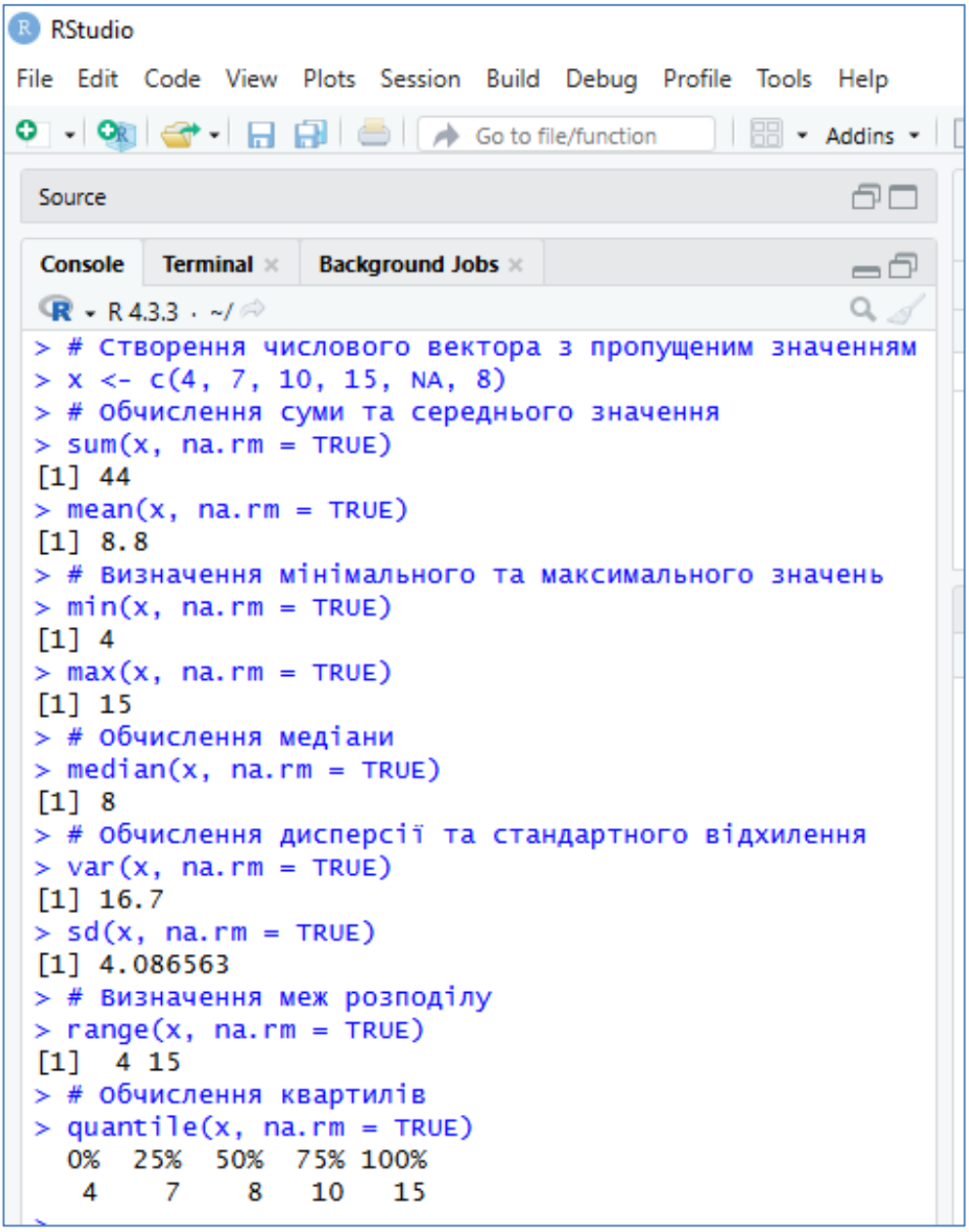
У наведеному прикладі продемонстровано визначення основних характеристик вектора, перевірку на наявність пропущених і спеціальних

числових значень, а також використання логічних і структурних перевірок під час аналізу даних у мові R.

7.4. Статистичні функції для векторів

Мова R містить набір вбудованих статистичних функцій для аналізу векторних даних, що дає змогу виконувати базові та розширені числові обчислення. Застосування цих функцій забезпечує ефективну обробку даних для статистичного аналізу.

Приклад використання статистичних функцій для векторів у мові R:



До основних статистичних функцій належать sum(), mean() та prod(), які використовуються для обчислення суми, середнього арифметичного та добутку

елементів вектора відповідно. Для визначення мінімального та максимального значень застосовуються функції min() і max(), а функція range() дозволяє отримати межі зміни даних.

Для оцінювання розсіювання даних використовуються функції var() та sd(), які обчислюють дисперсію і стандартне відхилення. Центральні тенденції, окрім середнього значення, можуть бути проаналізовані за допомогою функції median(). Функція quantile() надає можливість визначати квартилі та інші квантілі розподілу.

Важливою особливістю статистичних функцій у R є підтримка параметра na.rm, який дозволяє ігнорувати пропущені значення під час обчислень. Це забезпечує коректність результатів у разі наявності неповних даних.

7.5.Арифметичні та векторні операції

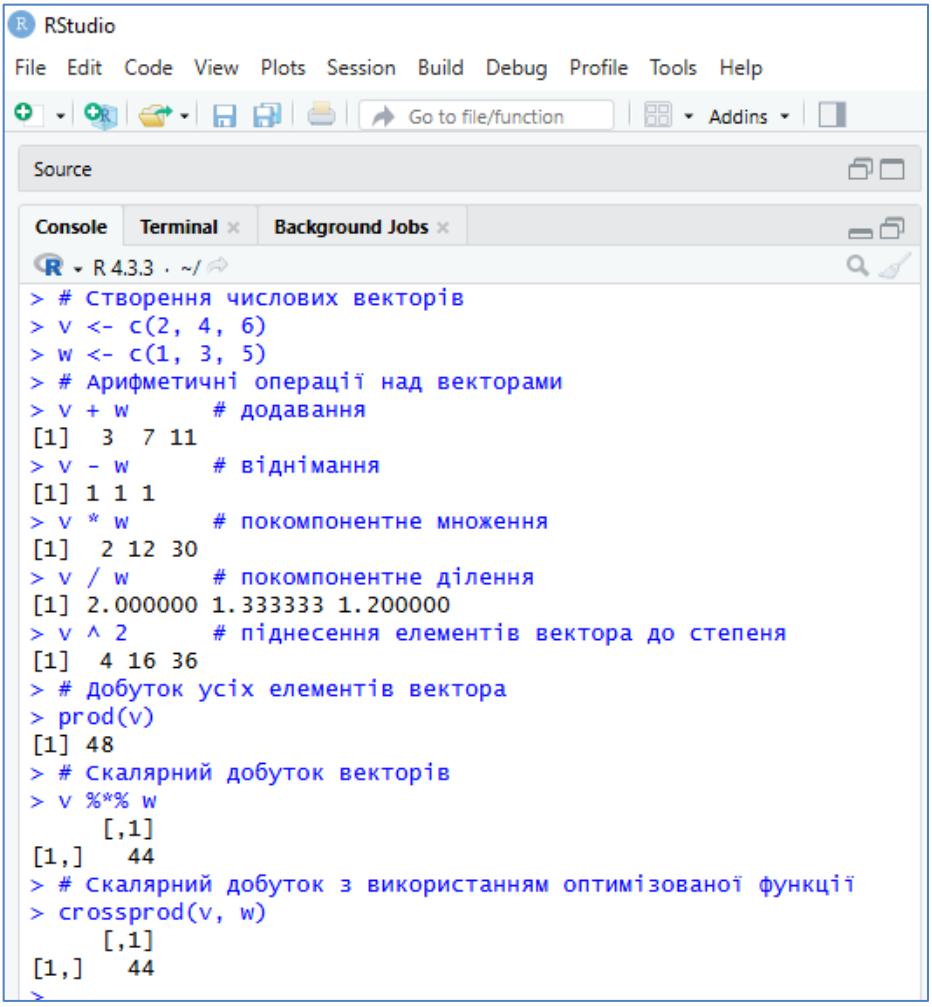
Арифметичні та векторні операції в мові R забезпечують ефективну обробку числових даних на основі векторизованих обчислень. Усі базові математичні дії в R виконуються над цілими векторами без явного використання циклів, що підвищує продуктивність і покращує читабельність програмного коду. Для узагальненого множення елементів вектора використовується функція prod(), яка обчислює добуток усіх компонентів числового вектора та повертає скалярне значення. Дана функція є корисною під час виконання статистичних і математичних розрахунків, де необхідно агрегувати множинні значення.

Оператор %*% реалізує векторно-матричне та матричне множення, зокрема скалярний добуток векторів однакової довжини. Він широко застосовується в задачах лінійної алгебри, статистичного моделювання та аналізу даних, забезпечуючи коректне та ефективне виконання арифметичних операцій над структурованими даними.

Функція crossprod() використовується для обчислення скалярного добутку векторів або виконання відповідних операцій над матрицями в оптимізованій формі. Вона є еквівалентною виразу t(v) %*% w, проте

забезпечує кращу обчислювальну ефективність, що є особливо важливим під час роботи з великими наборами даних.

Приклад використання арифметичних та векторних операцій у мові R:



Таким чином, арифметичні та векторні операції в мові R формують основу для реалізації математичних алгоритмів, підтримують функціональний стиль програмування та сприяють ефективному аналізу даних у середовищі RStudio.

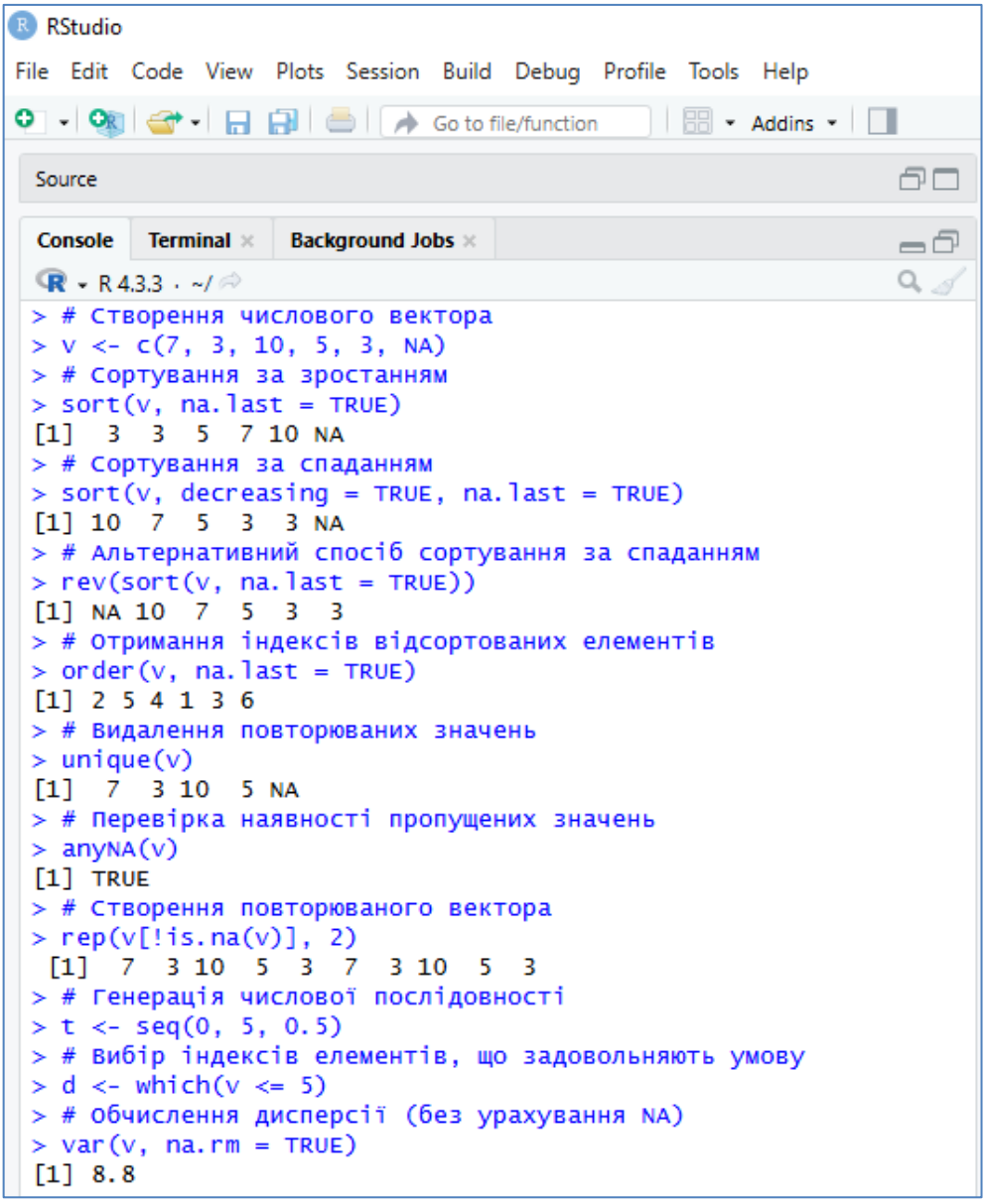
7.6.Сортування та перетворення векторів

Сортування та перетворення векторів є важливими операціями під час попередньої обробки й аналізу даних у мові R. Вони дозволяють впорядковувати елементи, змінювати структуру даних та готувати вектори до подальших статистичних обчислень.

Для впорядкування елементів вектора використовується функція sort(),

яка за замовчуванням виконує сортування за зростанням. За допомогою параметра decreasing = TRUE можна здійснити сортування за спаданням. Функція повертає вектор тієї ж довжини, що й вихідний, при цьому значення NA та NaN автоматично ігноруються під час сортування. Альтернативним способом упорядкування за спаданням є поєднання функцій rev() та sort().

Приклад використання сортування та перетворення векторів у мові R:



Функція order() повертає індекси елементів у відсортованому порядку, що дає змогу виконувати сортування складніших структур даних, зокрема векторів, матриць і таблиць, на основі одного або кількох критеріїв.

Перетворення векторів у R охоплює зміну типів даних і структури. Для приведення типів використовуються функції as.numeric(), as.character() та

as.logical(), які є необхідними під час імпорту даних або інтеграції інформації з різних джерел. Для створення повторюваних значень застосовується функція rep(), яка формує новий вектор шляхом багаторазового відтворення заданого вектора або окремих його елементів.

Окрім цього, функція seq() використовується для генерації числових послідовностей з рівномірним кроком на заданому інтервалі. Функція which() дозволяє вибирати індекси елементів вектора, що задовольняють певну логічну умову, і широко застосовується для фільтрації даних.

Для аналізу та контролю якості даних використовуються допоміжні функції, зокрема anyNA(), яка перевіряє наявність пропущених значень у векторі, а також var(), що обчислює дисперсію елементів вектора. Сукупність зазначених операцій забезпечує гнучке керування векторними даними та підвищує ефективність аналітичних процедур у середовищі RStudio.

Лабораторні роботи до розділу 7

Лабораторна робота 1. Створення векторів та доступ до їх елементів у мові R

Мета роботи: сформувати практичні навички створення векторів різних типів у мові R з використанням функцій c(), vector(), numeric(), character() і logical(), визначення їх довжини, а також доступу до окремих елементів і підвекторів.

Теоретичні відомості

Вектор є базовою структурою даних у мові R. Усі елементи вектора повинні бути одного типу**.** Саме з векторів будуються складніші структури – матриці, data frame, tibble.

Для створення векторів у R використовуються:

- функція c() – об’єднання значень;
- функція vector() – створення порожнього вектора заданого типу й довжини;

– спеціалізовані функції numeric(), character(), logical().

Доступ до елементів вектора здійснюється за допомогою індексації x[i].

Хід виконання роботи

Завдання 1. Створити два вектори задавши їх початкові значення. *Пояснення:* Функція c() об’єднує значення в єдиний вектор

```
a <- c(2, 5, 7, 10)
b <- c("A", "B", "C")
```

Завдання 2. Створити три типізованих вектори.

```
x <- vector("numeric", 5)
y <- character(4)
z <- logical(3)
```

Завдання 3. Визначити довжини створених векторів, перевірити типи та вивести інформацію про структуру.

```
length(a)
```

Завдання 4. Відпрацювати доступ до елементів векторів. Видалити окремий елемент та перевірити результати роботи.

```
a[1]
a[2:4]
a[-2]
```

Пояснення:

a[2:4] – підвектор;

a[-2] – виключення другого елемента.

Завдання 5. Провести сортування компонентів створених векторів. Вивести елемент вектора що відповідає певній умові.

```
sort(x, dec=T)).
w <- which(a<=7)
```

Лабораторна робота 2. Генерація та модифікація векторів. Послідовності та повторення

Мета роботи: навчитися створювати вектори за допомогою функцій rep() і seq(), застосовувати функцію scan(), використовувати іменовані вектори та перевіряти наявність значень у векторі.

Теоретичні відомості

Функції rep(), seq() та rnorm(n, mean, sd) дозволяють створювати регулярні числові послідовності, що широко застосовується в моделюванні та статистиці. Іменовані вектори (names()) полегшують інтерпретацію даних.

Хід виконання роботи

Завдання 1. Створити вектори p та q використавши функцію повторення елементів. Вивчити склад та структуру утворених векторів.

```
p <- rep(0, 15)
q <- rep(c(1, 2), 5)
```

Завдання 2. Створити числову послідовність t за заданими властивостями. Видалити 5 елемент послідовності. Обчислити суму числової послідовності.

```
t <- seq(0, 5, 0.5)
```

Завдання 3. Створити іменований вектор scores. Вивести числове значення за жіночим іменем елемента вектора.

```
scores <- c(90, 75, 88)
names(scores) <- c("Іван", "Марія", "Олег")
```

Завдання 4. Провести перевірку входження конкретного елемента у створені вектори (p, q, t, score).

```
"a" %in% c("a", "b", "c")
```

Завдання 5. Створити вектор f із послідовності 10 випадкових чисел.

1. Відсортувати значення вектора;
2. Знайти медіану та середнє значення;
3. Знайти скалярний добуток векторів f та t.

Лабораторна робота 3 Статистичні характеристики векторів та робота з NA

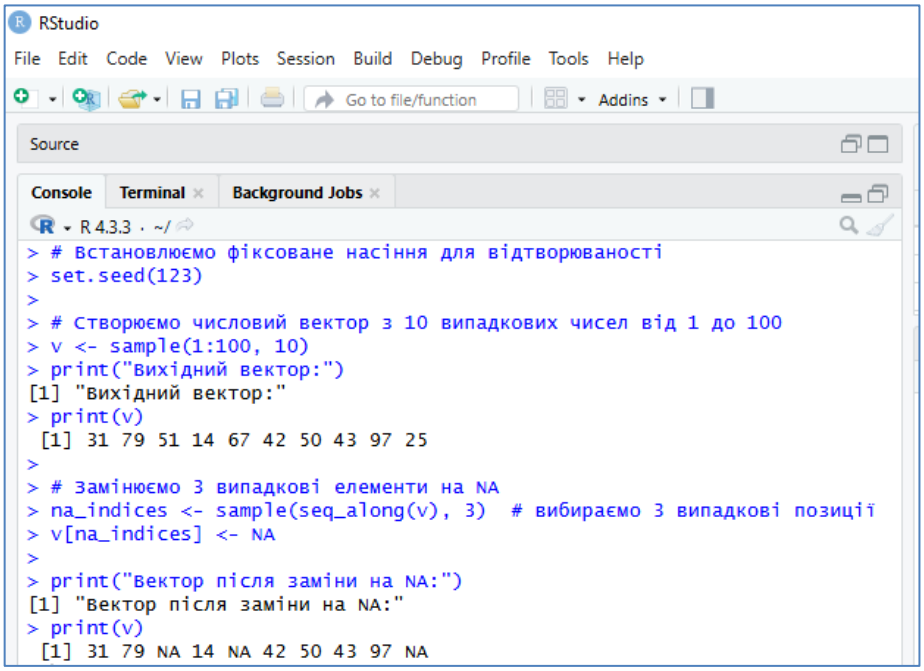
Мета роботи: навчитись обчислювати середнє, суму, добуток; знаходити мінімум, максимум, розмах; працювати з пропущеними значеннями; застосовувати var() та range().

Теоретичні відомості

Статистичні функції R працюють векторизовано, тобто застосовуються до всіх елементів одночасно. Наявність значень NA або NaN потребує спеціальної обробки з параметром na.rm = TRUE.

Хід виконання роботи

Завдання 1. Створити числовий вектор v використавши set.seed() та sample(). Замінити 3 елементи на NA.



Завдання 2. Вивчити основні статистичні функції числового вектора. Які будуть отримані дані при відсутності параметру na.rm = TRUE?

mean(v, na.rm = TRUE)

sum(v, na.rm = TRUE)

prod(v, na.rm = TRUE)

Завдання 3. Вивести межі та варіацію(дисперсію) вектора.

min(v, na.rm = TRUE)

max(v, na.rm = TRUE)

range(v, na.rm = TRUE)

var(v, na.rm = TRUE)

Завдання 4. Перевірти вектор на наявність NA. Використовуючи програмні засоби мови R видаліть із об’єкта v невизначені дані.

anyNA(v)

is.na(v)

????

Завдання 5. Зберегти ортимані результати у файлі v2.CSV.

Лабораторна робота 4. Сортування, індексація та векторні операції

Мета роботи: навчитися сортувати вектори, знаходити індекси елементів за заданими умовами, виконувати векторні та математичні операції, а також обчислювати скалярний добуток.

Теоретичні відомості

R підтримує векторні обчислення, що робить мову надзвичайно ефективною. Арифметичні та функціональні оператори застосовуються до кожного елемента вектора окремо.

Хід виконання роботи

Завдання 1. Створити числовий вектор a використавши для цього функцію зчитування даних з клавіатури (**scan()**. Відпрацюйте методику сортування.

```
# Створення числового вектора з клавіатури
cat("Введіть числа через пробіл і натисніть Enter:\n")
a <- scan()  # користувач вводить числа, наприклад: 5 2 9 1 7

# Вивід введеного вектора
cat("Введений вектор:\n")
print(a)

# Сортування вектора
```

```
# Сортування за зростанням
a_sorted_asc <- sort(a)
cat("Вектор, відсортований за зростанням:\n")
print(a_sorted_asc)

# Сортування за спаданням
a_sorted_desc <- sort(a, decreasing = TRUE)
cat("Вектор, відсортований за спаданням:\n")
print(a_sorted_desc)
```

Завдання 2. Відпрацювати операцію пошуку елементів за вказаною умовою. Створити вектори, використавши функцію which(). Результати зберегти у векторах a1, a2, a3.

```
which(a > 4)
```

Завдання 3. Створити вектор u. Обчислити та вивести на екран значення тригонометричних функцій для створеного вектора.

```
u <- c(0, pi/6, pi/3, pi/2)
sin(u), …
```

Завдання 4. Підготувати два числові вектори за поданим зразком. Обчислити скалярний добуток векторів. Порівняти результати проведених обчислень.

```
v <- c(1, 2, 3)
w <- c(4, 5, 6)
v %*% w
crossprod(v, w)
```

Завдання 5. Виконати завдання 4 використавши підготований текстовий файл dat1.csv для введення даних та файл dat2.csv здля збереження даних.

Інтегровані завдання до розділу 7

Завдання 1.

1. Створити вектор із 10 нульових елементів.

2. Створити символьний вектор із назвами місяців.
3. Отримати підвектор із 3–7 елементів.
4. Створити z вектор із 10 випадкових чисел.
5. Відсортувати елементи вектора z. Виведіть значення 5 елемента.

Завдання 2.

1. Створити вектор з числами від –10 до 10 з кроком 2.
2. Створити вектор R повторивши вектор c(3,5) 4 рази.
3. Додати імена до числового вектора R.

Завдання 3.

1. Створити вектор із 20 випадкових чисел.
2. Обчислити всі статистичні характеристики.
3. Пояснити отримані результати.
4. Зберегти вектор у файлі xlsx.

Завдання 4.

1. Відсортувати вектор довільної довжини двома способами.
2. Знайти індекси елементів, більших за середнє.
3. Обчислити синуси і косинуси кутів за елементами вектора.

Контрольні питання до розділу 7

1. Для чого використовується na.rm = TRUE?
2. Для чого використовується which()?
3. Для чого використовується функція rep()?
4. Для чого використовується функція vector()?
5. Чим відрізняється sort() і rev(sort())?
6. Чому mean() повертає NA?
7. Що показує функція range()?
8. Що таке вектор у мові R?
9. Що таке векторизація?
10. Що таке іменований вектор?
11. Як виключити елемент із вектора?

12. Як обчислюється скалярний добуток?
13. Як перевірити, чи належить елемент вектору?
14. Яка вимога до типів даних у векторі?
15. Яка різниця між is.na() та anyNA()?
16. Яке призначення seq()?

РОЗДІЛ 8. ФУНКЦІЇ ДЛЯ РОБОТИ З МАТРИЦЯМИ

8.1. Поняття матриць та їх місце серед структур даних у R

У RStudio над числовими матрицями можна виконувати широкий спектр математичних операцій – від елементарної арифметики до лінійної алгебри, статистики та матричних перетворень. RStudio дозволяє виконувати повний набір операцій лінійної алгебри над матрицями, що робить його потужним інструментом для аналізу даних, математичного моделювання, машинного навчання та наукових досліджень.

Матриці в мові R є двовимірними однорідними структурами даних, елементи яких мають однаковий тип. Вони широко використовуються для представлення числових таблиць, реалізації алгоритмів лінійної алгебри та виконання математичних обчислень.

Створення матриць у R здійснюється за допомогою функції matrix(), яка дозволяє задати набір елементів, кількість рядків і стовпців, а також порядок заповнення. Матриці можуть формуватися на основі векторів, з автоматичним заповненням за стовпцями або рядками. Для об’єднання кількох векторів у матрицю використовуються функції cbind() та rbind().

Перелік функцій для виконання математичних операції над числовими матрицями в RStudio:

matrix(c, nrow=, ncol= ,) створює матрицю: c- вектор, nrow=m кількість рядків, ncol=m кількість стовпців.

nrow(A) і ncol(A) повертають відповідно кількість рядків і стовпців матриці A.

dim() кількість вимірів матриці.

head() відображає першу частину об'єкта.

tail() відображає останню частину об'єкта.

str() та summary() виводить узагальнену інформацію про таблицю.

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.

crossprod(v, w) – обчислює скалярний добуток (dot product векторів матричний добуток).

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

det(A) – визначник матриці A.

solve(A) – обернена матриця A.

t(A) – транспонування матриці.

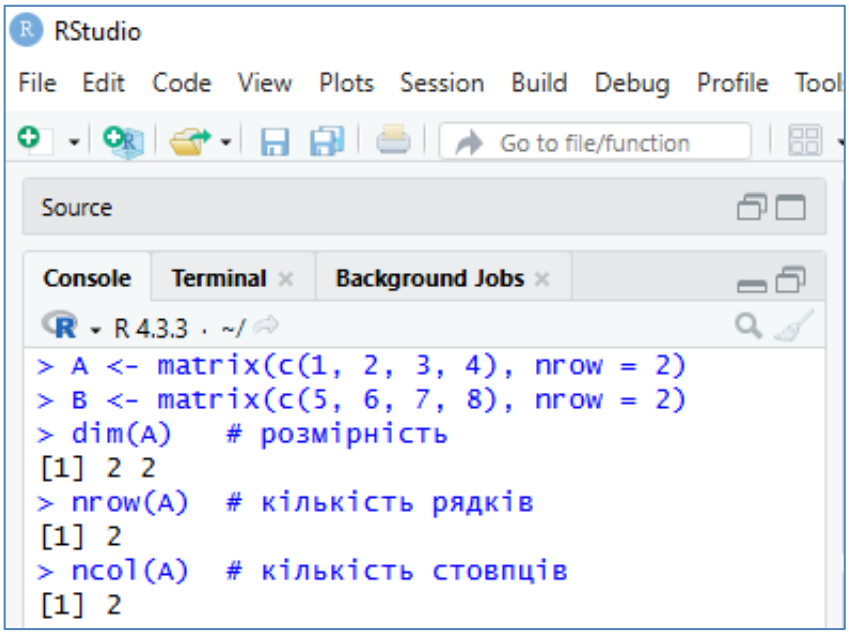
A = cbind(A, B) – створює матрицю, приписуючи до A справа B (для цього кількість рядків у A і B має співпадати).

A = rbind(A, B) – створює матрицю, приписуючи до A знизу B (для цього кількість стовпців у A і B має співпадати).

glimpse() – для швидкого узагальнення таблиць.

8.2.Створення та ініціалізація матриць

До базових властивостей матриць належать розмірність, форма та імена рядків і стовпців. Функція dim() повертає кількість рядків і стовпців матриці, nrow() і ncol() – їх окремо, а rownames() та colnames() дозволяють переглядати або задавати відповідні імена. Перевірка належності об’єкта до матричного типу здійснюється за допомогою функції is.matrix().



У наведеному прикладі створено дві матриці A і B розміром 2×2, після чого за допомогою функцій dim(), nrow() та ncol() визначено їх розмірність, кількість рядків і кількість стовпців відповідно, що ілюструє базові властивості матриць у мові R.

8.3.Матричні обчислення та перетворення

R має вбудовані засоби для створення матриць, виконання базових арифметичних операцій, транспонування, знаходження обернених матриць, визначників, власних значень і власних векторів, а також розв’язання систем лінійних рівнянь.

Розглянемо приклади для двох матриць:

```
A <- matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(c(5, 6, 7, 8), nrow = 2)
```

Поелементні операції.

Операції додавання та множення A + B і A * B виконуються поелементно, тобто над відповідними елементами матриць:

```
A + B
A * B
```

Матричне множення.

Для класичного матричного множення використовується оператор %*%:

```
A %*% B
```

Умова виконання: кількість стовпців матриці A повинна дорівнювати кількості рядків матриці B.

Транспонування матриці.

Транспонування здійснюється за допомогою функції t():

```
image_matrix <- matrix(1:9, nrow = 3)
t(image_matrix)
```

У результаті рядки матриці стають стовпцями, а стовпці – рядками.

Обернена матриця.

Для знаходження оберненої матриці використовується функція solve():

solve(A)

Умова: матриця має бути квадратною та невиродженою (det ≠ 0).

Визначник матриці.

Визначник обчислюється функцією det():

det(A)

Використовується для перевірки можливості обчислення оберненої матриці.

Власні значення та власні вектори.

Функція eigen() дозволяє знайти спектральні характеристики матриці:

```
eig <- eigen(A)
eig$values   # власні значення
eig$vectors  # власні вектори
```

Такі обчислення застосовуються в методі головних компонент (PCA), аналізі стабільності систем та машинному навчанні.

Ранг матриці.

Ранг матриці визначається як кількість лінійно незалежних рядків або стовпців:

```
qr(A)$rank
```

Наведені операції формують основу роботи з матрицями в середовищі RStudio та є необхідними для подальшого вивчення методів аналізу даних і математичного моделювання.

Функції rowSums() і rowMeans() обчислюють суму та середнє арифметичне елементів у кожному рядку матриці відповідно. Аналогічно, colSums() і colMeans() застосовуються до стовпців. Ці операції використовуються для аналізу структури даних, порівняння рядків і стовпців, а також для отримання узагальнених показників у табличних наборах даних.

Під час застосування статистичних функцій mean(), sd() та var() до матриці вона розглядається як один вектор значень. У результаті обчислюється загальне середнє, стандартне відхилення та дисперсія для всіх елементів

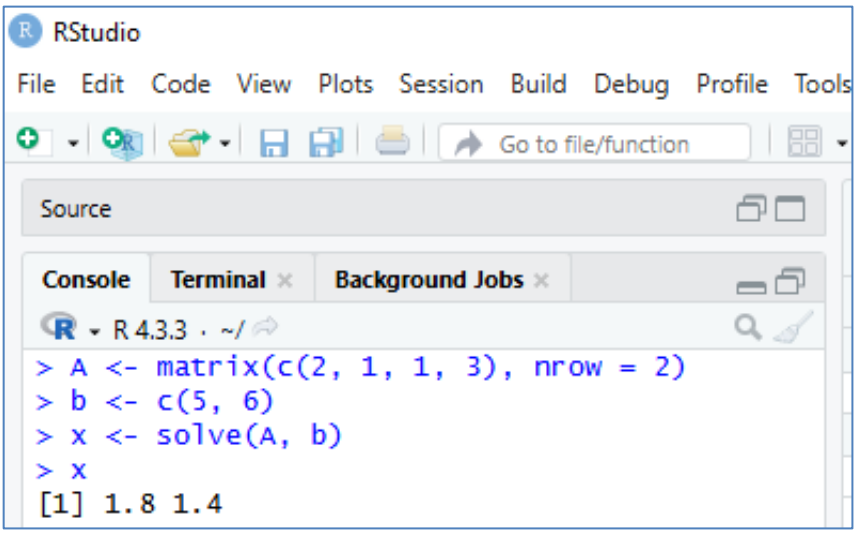
матриці без урахування її двовимірної структури. Такий підхід є корисним для оцінки загальних статистичних характеристик набору даних.

Функція scale() використовується для нормалізації та масштабування даних. Вона приводить значення кожного стовпця матриці до порівнюваного масштабу, зазвичай забезпечуючи нульове середнє відхилення та одиничне стандартне відхилення. Нормалізація є важливим етапом підготовки даних у статистичному аналізі та машинному навчанні, оскільки дозволяє усунути вплив різних одиниць вимірювання.

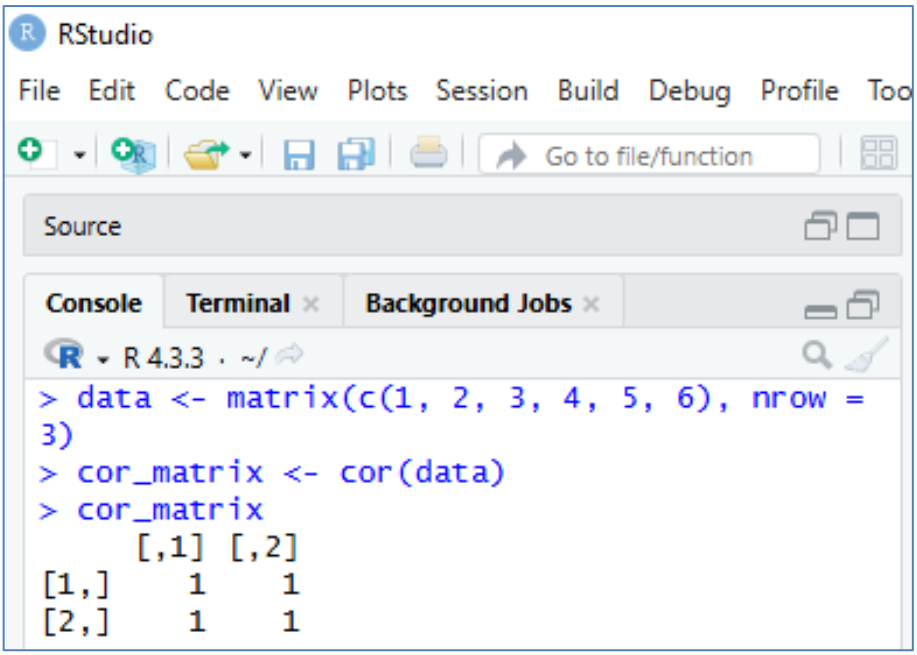
8.4. Приклади використання матриць у практичних задачах

Матриці є універсальним інструментом для розв'язування широкого кола практичних задач у середовищі R. Одним із найпоширеніших напрямів їх використання є розв'язування систем лінійних рівнянь, що виникають у задачах оптимізації, економічного моделювання та технічних розрахунків. Пакет Matrix розширює можливості роботи з матрицями в R, зокрема забезпечує підтримку розріджених матриць, що містять велику кількість нульових елементів. Використання таких матриць дозволяє зменшити обсяг пам'яті та значно прискорити обчислення. Пакет library(Matrix) широко застосовується під час роботи з великими наборами даних, розв'язування великих систем лінійних рівнянь і реалізації алгоритмів машинного навчання.

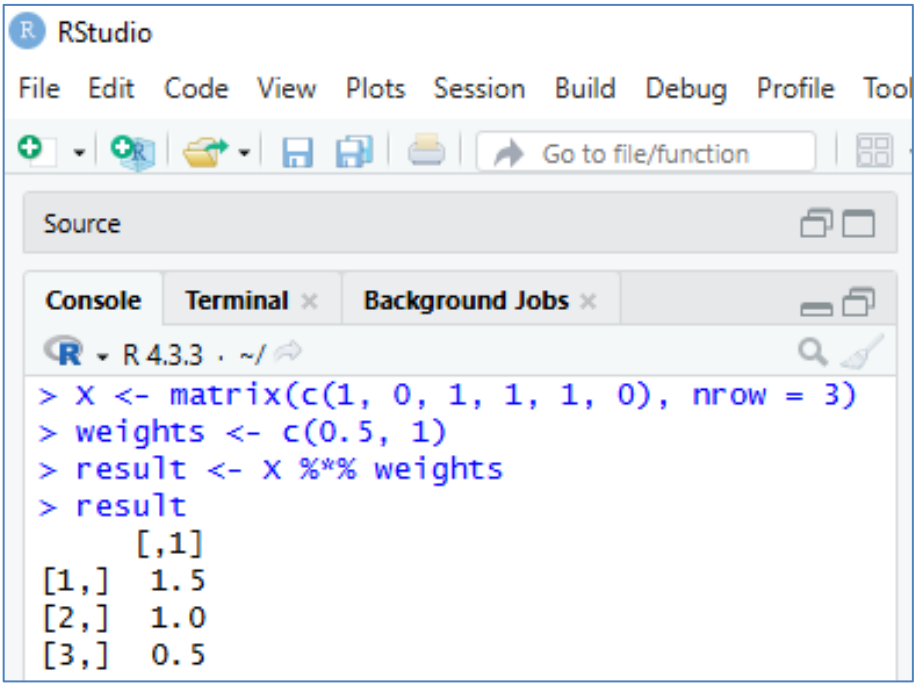
Матричні методи дозволяють компактно подати систему рівнянь і ефективно обчислити шукані значення.



У статистичному аналізі матриці застосовуються для обробки багатовимірних вибірок, обчислення кореляційних і коваріаційних матриць, а також під час побудови регресійних моделей. Таке подання даних забезпечує наочність результатів і спрощує подальший аналіз взаємозв'язків між змінними.



У задачах машинного навчання та аналізу даних матриці використовуються для представлення навчальних наборів даних, де рядки відповідають спостереженням, а стовпці – ознакам. Матричні обчислення лежать в основі багатьох алгоритмів класифікації, кластеризації та зменшення розмірності даних.



Крім того, матриці активно застосовуються під час обробки зображень, моделювання фізичних процесів та аналізу результатів експериментів. Матричне подання даних дає змогу виконувати перетворення та обчислення над великими масивами значень з високою ефективністю.

Лабораторні роботи до розділу 8

Лабораторна робота 1. Математичні операції над числовими матрицями в RStudio

Мета роботи: навчитися створювати числові матриці в R, опанувати покомпонентні та матричні операції, виконують основні операції лінійної алгебри, застосовують статистичні й агрегатні функції до матриць та аналізують отримані результати обчислень.

Теоретичні відомості

У мові R матриця – це двовимірна структура даних, що містить елементи одного типу (у нашому випадку – числового). R підтримує:

- покомпонентні операції (+, *, /);
- матричні операції (%*%);
- операції лінійної алгебри (визначник, обернена матриця);
- статистичний аналіз матриць.

Мова R має вбудовані засоби лінійної алгебри, які широко використовуються в математиці, статистиці, економіці та аналізі даних. У RStudio можна зручно розв’язувати системи лінійних рівнянь.

Хід виконання роботи

Розглянути наведені приклади та виконайте запропоновані завдання. RStudio повністю підтримує розв’язання систем лінійних рівнянь. Можна працювати з:

- точними системами;
- перевизначеними системами;

- статистичними моделями;
- великими матрицями.

Завдання 1. Створити числові матриці та збережіть їх у файлі xlsx.

A <- matrix(c(2, 4, 6, 8), nrow = 2)

B <- matrix(c(1, 3, 5, 7), nrow = 2)

Пояснення: Матриці A і B мають розмір 2×2 та заповнюються по стовпцях.

Завдання 2. Переглянути властивості матриць A та B. Виконати об'єднання матриць.

dim(A)

nrow(A)

ncol(A)

str(B) та інші

Завдання 3. Створити два числові вектори a та b з довільними елементами та однакової розмірності. Об'єднати їх у матриці A та B. Виконати покомпонентні арифметичні операції.

1. Додавання A + B
2. Віднімання A - B
3. Множення A * B
4. Ділення A / B
5. Піднесення до степеня A^2

Пояснення: Кожна операція виконується над відповідними елементами матриць.

Завдання 4. Виконати матричне множення. Пояснити результати.

A %*% B

Пояснення: Оператор %*% виконує класичне матричне множення, а не покомпонентне.

Завдання 5. Транспонування матриці це заміна рядків на стовпці – часто використовується у статистиці та ML. Виконати транспонування матриці A, зберегти результат в об'єкті A2. Порівняти утворену матрицю із оригіналом.

t(A)

Завдання 6. Обчислити визначник матриці та зробити висновок про можливі шляхи її використання. Порівняти результат із визначником матриці A2.

det(A)

Пояснення: Якщо det(A) ≠ 0, матриця є оберненою.

Завдання 7. Обчислити обернені матриці для A та A2. Порівняйте їхні характеристики та значення.

solve(A)

Призначення: Використовується при розв'язуванні систем лінійних рівнянь.

Завдання 8. Обчислити власні значення та власні вектори для матриць A та A2. Порівняти їх.

eig <- eigen(A)

eig$values

eig$vectors

Завдання 9. Визначити ранги матриць які сформовані на даний час у середовищі RStudio. Порівняти їх та зробити висновок.

qr(A)$rank

Пояснення: Показує кількість лінійно незалежних рядків.

Завдання 10. Використовуючи агрегатні операції підрахувати суми і середні по об’єктах та змінних для обох матриць.

rowSums(A)

colSums(A)

rowMeans(A)

colMeans(A)

Призначення: Аналіз значень за рядками та стовпцями.

Завдання 11. Особливість використання статистичних функцій для матриць полягає в тому, що R спочатку перетворює матрицю на вектор, об’єднуючи всі елементи. Обчислити статистичні характеристики та порівняти їх.

mean(A)

sd(A)

var(A)

*Пояснення:*Матриця розглядається як єдиний набір даних.

Завдання 12. Логічні операції

A > 5

A == 4

Результат:Матриця логічних значень (TRUE / FALSE).

Завдання 13. Створити матрицю А розмірністю 2х2. Порівняти значення матриці до нормалізації та після. Обчислити scale(A) змінивши один елемент матриці на «викид».

scale(A)

Контрольні питання до розділу 8

1. Для чого використовується функція scale() під час роботи з матрицями?

2. Для чого використовуються функції rbind() і cbind()?

3. Чим відрізняється поелементне множення від матричного множення?

4. Що таке матриця в мові R та чим вона відрізняється від вектора?

5. Як визначити кількість рядків і стовпців матриці в R?

6. Як виконати транспонування матриці?

7. Як застосовуються статистичні функції до матриці та як при цьому вона інтерпретується?

8. Як здійснюється доступ до окремих елементів, рядків і стовпців матриці?

9. Як обчислити суму та середнє значення елементів рядків і стовпців матриці?

10. Як перетворити вектор або фрейм даних у матрицю?

11. Як створити діагональну матрицю в R?

12. Яке призначення оператора %*% у R?

13. Яке призначення функції matrix() та її основні аргументи?

14. Який результат виконання логічних операцій над матрицею?

15. Які арифметичні операції можуть виконуватися над матрицями поелементно?

16. Які основні способи створення матриць у R?

17. Які переваги надає пакет Matrix для роботи з великими або розрідженими матрицями?

18. Які типи даних можуть міститися в матриці R?

19. Які функції використовуються для обчислення визначника та оберненої матриці?

РОЗДІЛ 9. ВІЗУАЛІЗАЦІЯ ДАНИХ

9.1.Поняття візуалізації даних та її роль в аналізі даних

Візуалізація даних – це процес подання числової, текстової або категоріальної інформації у графічній формі з метою полегшення її сприйняття, аналізу та інтерпретації. Вона дає змогу перетворити складні масиви даних на наочні графіки, діаграми та схеми, що дозволяє швидко виявляти закономірності, тенденції, взаємозв'язки між змінними, а також аномалії у даних.

У процесі аналізу даних візуалізація сприяє глибшому розумінню структури даних і результатів обчислень. Завдяки графічному представленню можна оперативно оцінити розподіл змінних, порівняти групи спостережень, перевірити гіпотези та зробити обґрунтовані висновки. Крім того, візуалізація є інструментом комунікації результатів аналізу, адже дозволяє зрозуміло та переконливо представити отримані дані широкій аудиторії, зокрема у наукових роботах, звітах та навчальних матеріалах.

У середовищі R візуалізація даних поєднує аналітичну точність із гнучкістю налаштувань, що робить її невід'ємною складовою сучасного статистичного аналізу та досліджень в науці та освіті.

Базові графічні можливості мови R представлені вбудованою графічною системою, яка не потребує підключення додаткових пакетів і дозволяє швидко створювати наочні двовимірні та тривимірні графіки. Вона орієнтована на оперативний аналіз даних і широко використовується для попереднього дослідження статистичної інформації.

Основною функцією базової графіки є plot(), яка автоматично підбирає тип візуалізації залежно від типу вхідних даних. За її допомогою можна будувати лінійні графіки, діаграми розсіювання, графіки часових рядів та інші види візуалізацій. Для доповнення графіків застосовуються функції points() і lines(), які дозволяють додавати нові точки та лінії до вже побудованого зображення, а функція text() – розміщувати текстові пояснення.

R також надає спеціалізовані функції для створення різних типів діаграм, зокрема hist() для побудови гістограм, barplot() для стовпчикових діаграм, boxplot() для ящиків з вусами, pie() для кругових діаграм. Керування зовнішнім виглядом графіків здійснюється за допомогою параметрів кольору, типу ліній, маркерів, підписів осей та заголовків.

9.2.Побудова простих графіків: plot(), points(), lines()

У навчальному контексті plot() – базовий інструмент аналізу даних, з якого починають роботу з візуалізацією.

Побудова простих графіків у мові R за допомогою функцій plot(), points() і lines() може бути гнучко налаштована завдяки використанню додаткових параметрів, які керують зовнішнім виглядом графіків та способом відображення даних.

9.2.1.Функція plot()

Окрім базової побудови графіка, plot() підтримує низку параметрів для його оформлення:

main – заголовок графіка;

xlab, ylab – підписи осей;

col – колір точок або ліній;

pch – тип маркера точки;

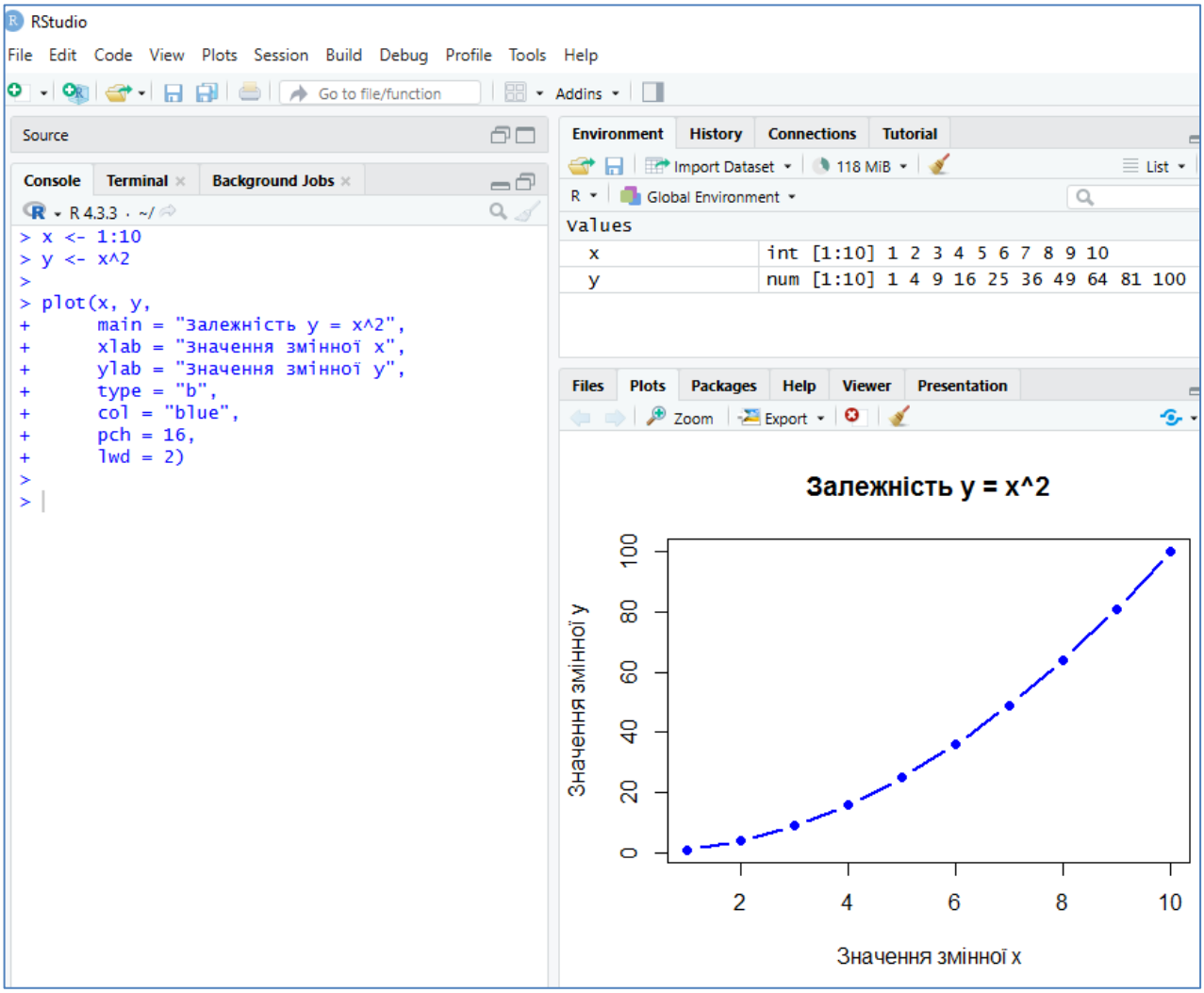
lty – тип лінії;

lwd – товщина лінії;

type – тип графіка ("p" – точки, "l" – лінії, "b" – точки з лініями).

Функція plot() використовується для побудови початкового графіка. Вона автоматично визначає тип візуалізації залежно від вхідних даних і найчастіше застосовується для побудови діаграм розсіювання або лінійних графіків.

Приклад побудови базового графіка:



9.2.2.Функція points()

Функція points() використовується для додавання додаткових точок до вже існуючого графіка. Це дає змогу порівнювати кілька наборів даних, виділяти окремі спостереження або накладати експериментальні дані на теоретичні моделі.

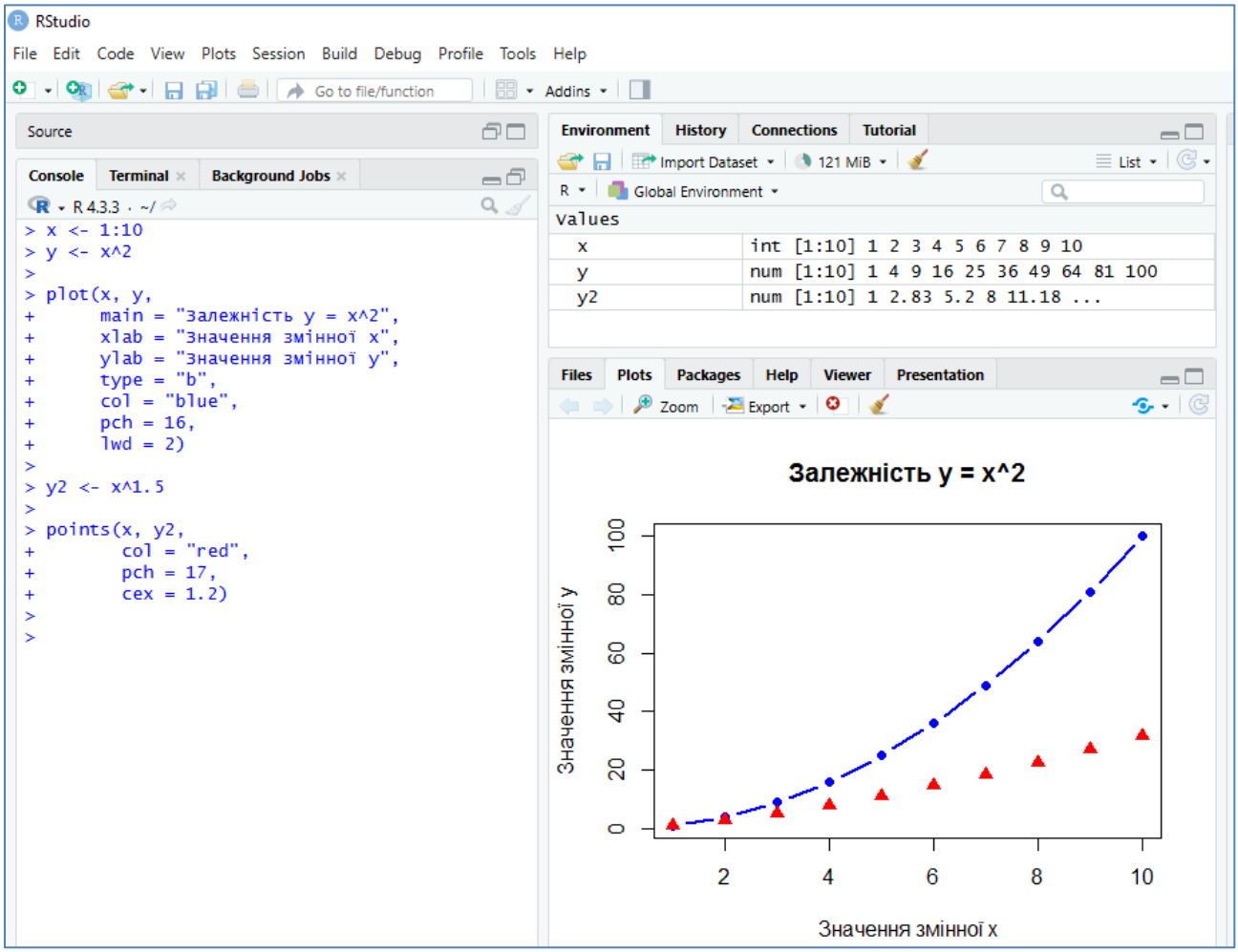
points() підтримує більшість параметрів форматування, доступних у функції plot(), зокрема:

col – колір точок;

pch – форма маркерів;

cex – розмір точок.

Приклад використання points():



У цьому прикладі на графік додається другий набір даних, що дозволяє наочно порівняти дві різні функціональні залежності. Параметр cex задає розмір точок.

9.2.3.Функція lines()

Функція lines() призначена для додавання ліній до вже побудованого графіка. Вона часто використовується для з’єднання точок, відображення трендів або накладання згладжених і теоретичних кривих.

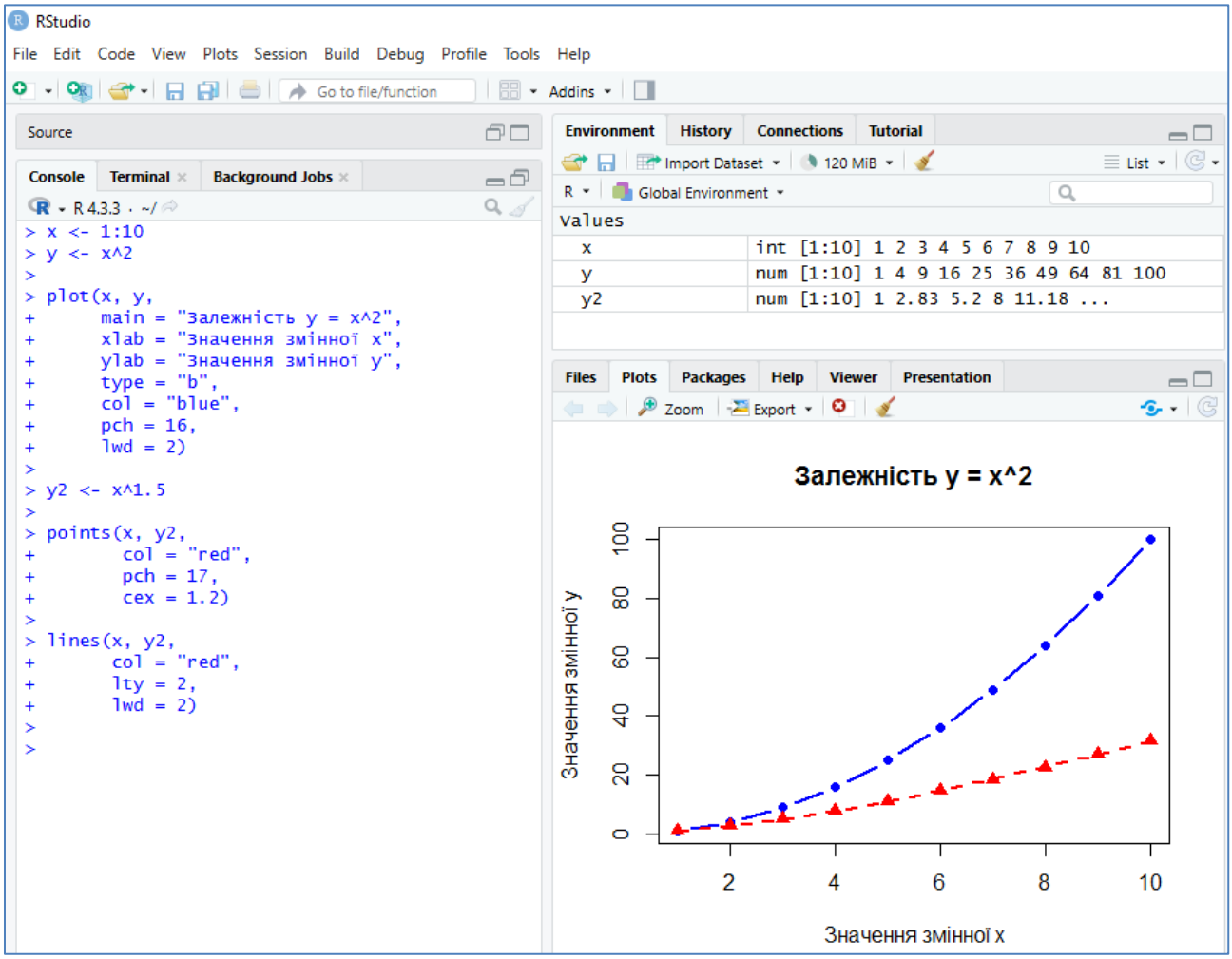
Основні параметри функції lines():

col – колір лінії;

lty – тип лінії (суцільна, пунктирна, штрихова);

lwd – товщина лінії.

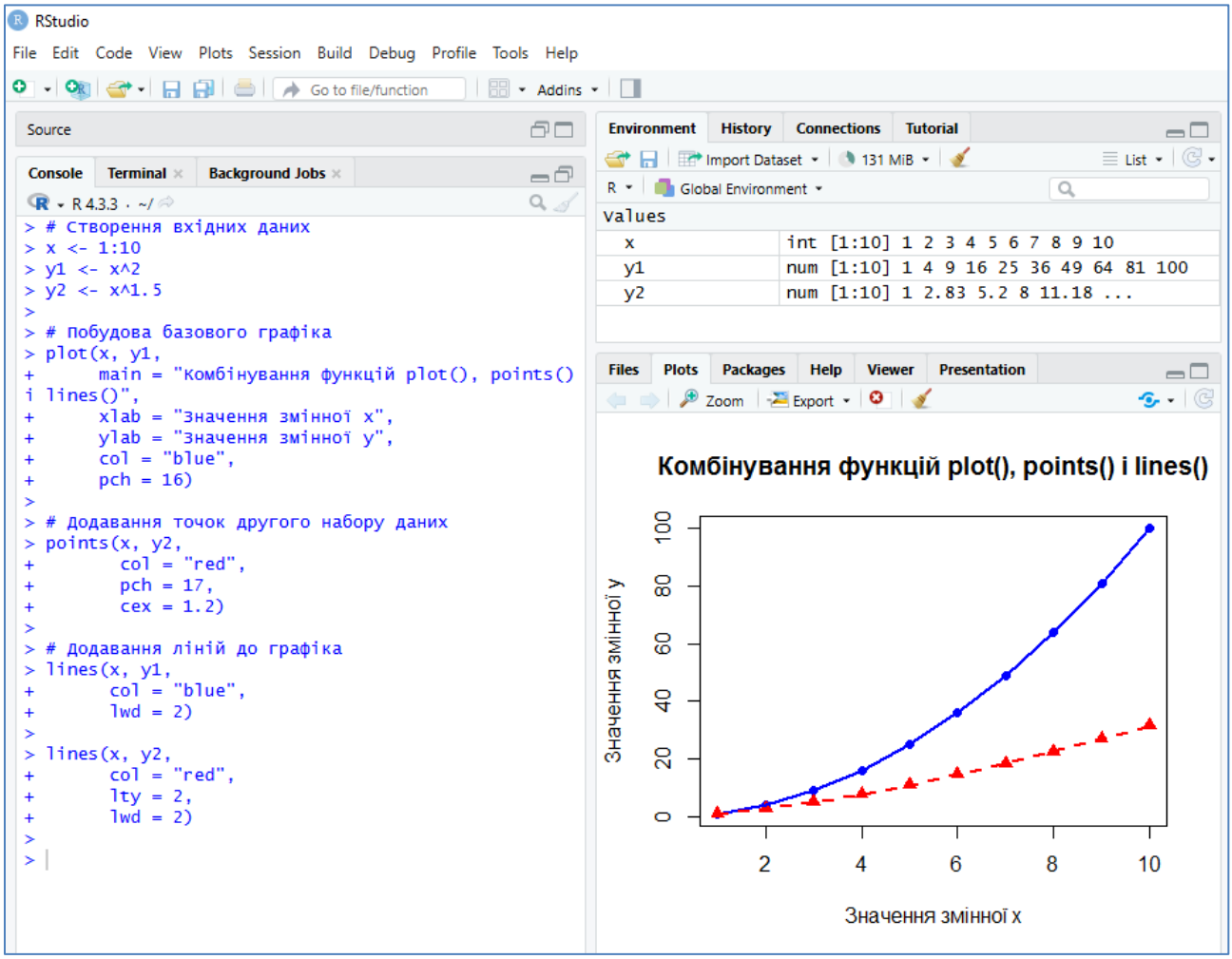
Приклад використання lines():



9.2.4. Комбінування функцій plot(), points() і lines()

Однією з переваг базової графіки R є можливість поєднання кількох функцій у межах одного графіка. Це дозволяє створювати інформативні візуалізації для аналізу та порівняння даних.

На одному графіку можна поєднати кілька наборів даних із різними стилями оформлення.



Функції plot(), points() і lines() формують основу базової візуалізації в R та широко застосовуються на початкових етапах аналізу даних. Їх використання дозволяє швидко отримати уявлення про структуру даних, виявити тенденції та взаємозв'язки, а також підготувати якісні графічні матеріали для навчальних, наукових і прикладних завдань.

9.3.Типи графіків у R: лінійні графіки, гістограми, діаграми розсіювання

Візуалізація даних є невід'ємною складовою процесу статистичного аналізу та дослідження даних. Графічне подання інформації дозволяє швидко оцінити загальну структуру набору даних, виявити закономірності, тенденції та аномалії, які можуть бути непомітними при аналізі числових таблиць. У мові програмування R реалізовано інструментарій для побудови різноманітних

графіків, що дає змогу ефективно поєднувати обчислювальні та візуальні методи аналізу.

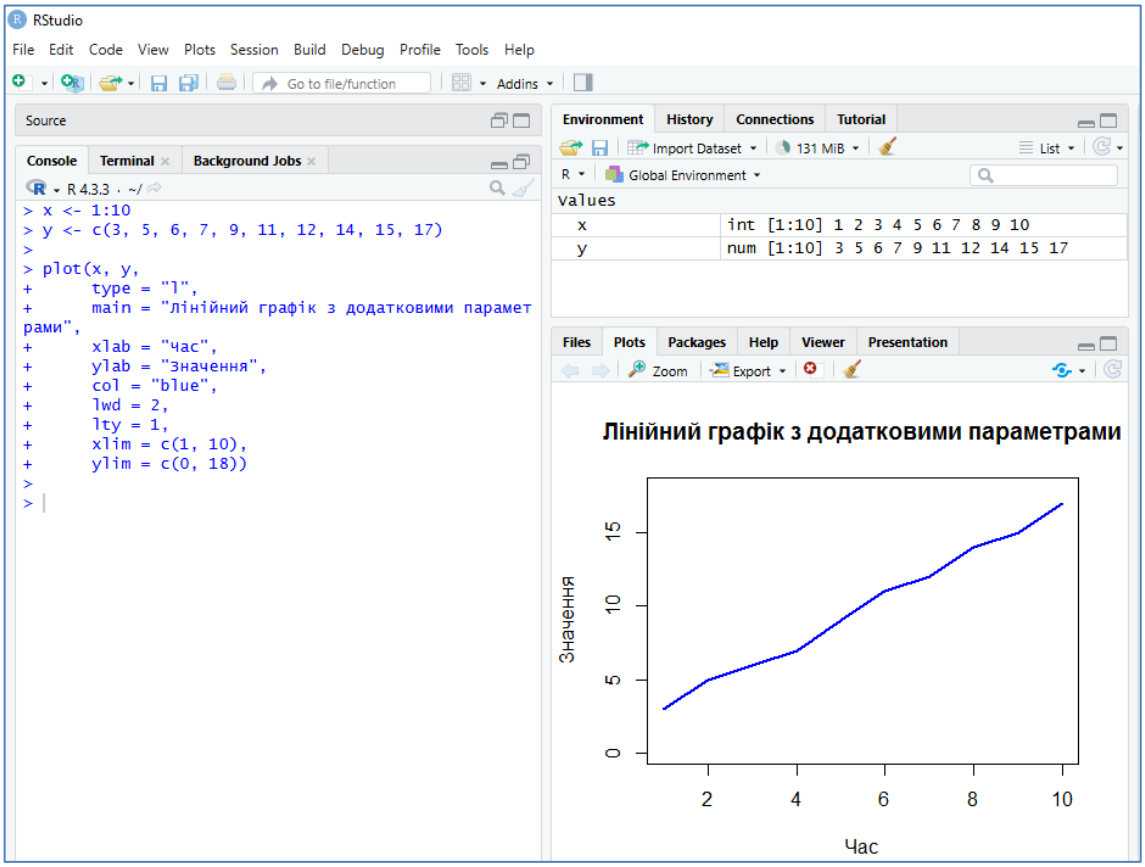
Лінійні графіки, гістограми та діаграми розсіювання є найбільш уживаними у навчальній та науковій практиці. Вони забезпечують наочне відображення змін даних, розподілу значень та взаємозв'язків між змінними, що робить їх незамінними на початкових етапах аналізу даних.

9.3.1.Лінійні графіки

Лінійні графіки використовуються для відображення динаміки зміни однієї змінної залежно від іншої, зокрема часу або впорядкованої числової шкали. Вони дозволяють простежити тенденції розвитку процесів, порівняти значення показників у різні моменти часу та виявити загальний характер змін.

У мові R лінійні графіки зазвичай будуються за допомогою функції plot() з параметром type = "l" або шляхом використання функції lines(). За допомогою додаткових параметрів користувач може налаштовувати колір, товщину та тип ліній, підписи осей і заголовки, що суттєво підвищує інформативність графіка.

Приклад:



Основні додаткові параметри:

type – тип графіка ("l" – лінійний, "b" – точки з лініями);

col – колір лінії;

lwd – товщина лінії;

lty – тип лінії (суцільна, пунктирна тощо);

xlim, ylim – межі осей;

main, xlab, ylab – заголовок і підписи осей.

Лінійні графіки широко застосовуються у статистиці, економіці, освіті та інших галузях для аналізу часових рядів і прогнозування.

9.3.2.Гістограми

Гістограма є одним із основних засобів візуалізації розподілу числових даних. Вона показує, як значення змінної групуються за інтервалами та з якою частотою вони зустрічаються. За допомогою гістограми можна оцінити форму розподілу даних, наявність асиметрії, піків або викидів.

У мові R гістограми створюються функцією hist(), яка дозволяє задавати кількість інтервалів, змінювати кольори стовпців, відображати абсолютні або відносні частоти. Це робить гістограму ефективним інструментом для первинного статистичного аналізу та підготовки даних до подальших досліджень.

Основні додаткові параметри:

breaks – кількість або межі інтервалів;

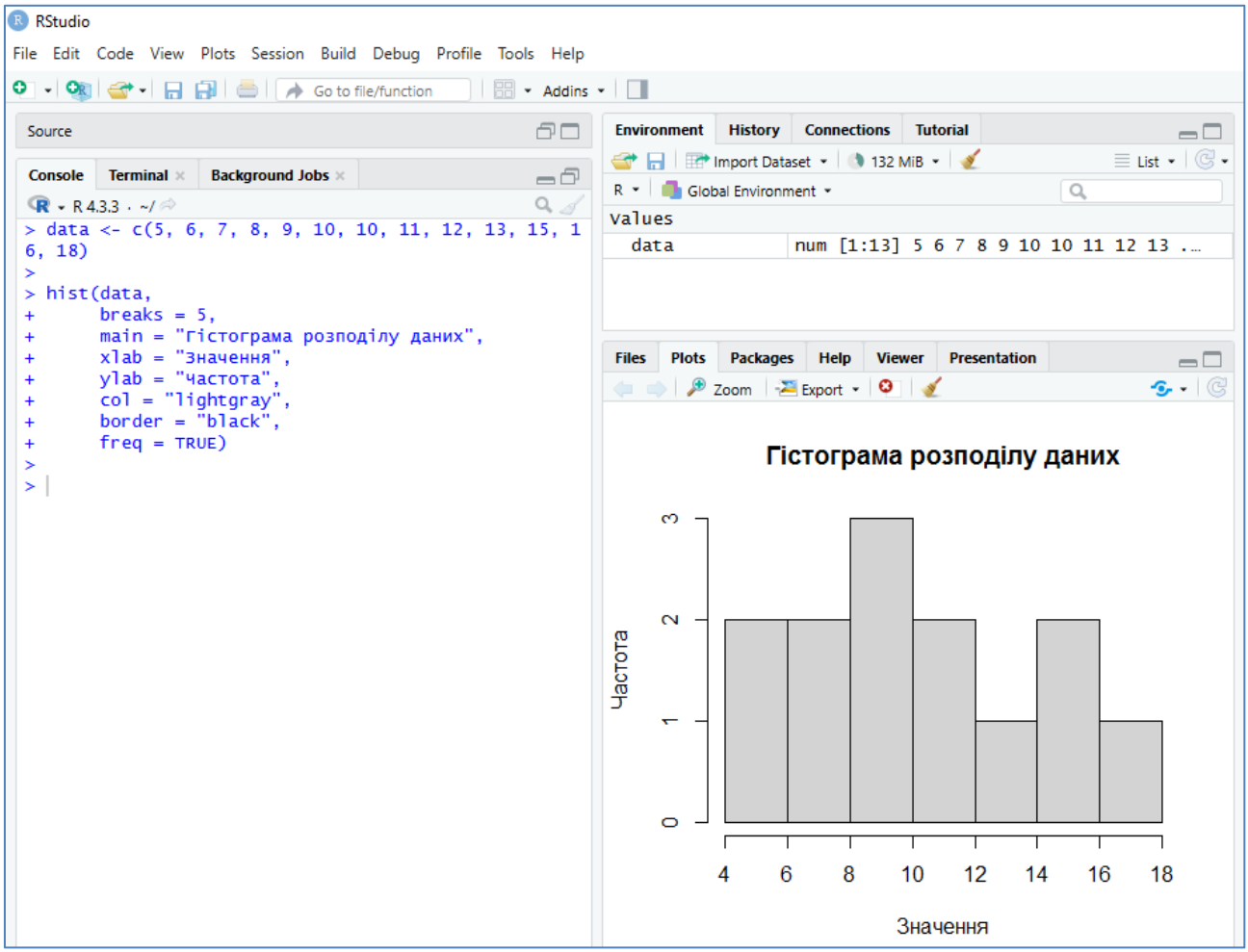
col – колір стовпців;

border – колір меж;

freq – відображення частот (TRUE) або ймовірностей (FALSE);

xlab, ylab, main – підписи та заголовок.

Приклад:



Гістограми є особливо корисними в навчальному процесі, оскільки наочно демонструють основні поняття описової статистики.

9.3.3.Діаграми розсіювання

Діаграми розсіювання застосовуються для аналізу взаємозв'язку між двома числовими змінними. Кожна точка на такому графіку відповідає окремому спостереженню, що дозволяє оцінити характер залежності, силу кореляції та наявність аномальних значень.

У R діаграми розсіювання будуються за допомогою функції plot() у стандартному режимі. Користувач може змінювати форму, розмір і колір точок, а також додавати лінії тренду чи додаткові набори даних. Це сприяє глибшому аналізу та інтерпретації взаємозв'язків між змінними.

Основні додаткові параметри:

pch – форма маркерів точок;

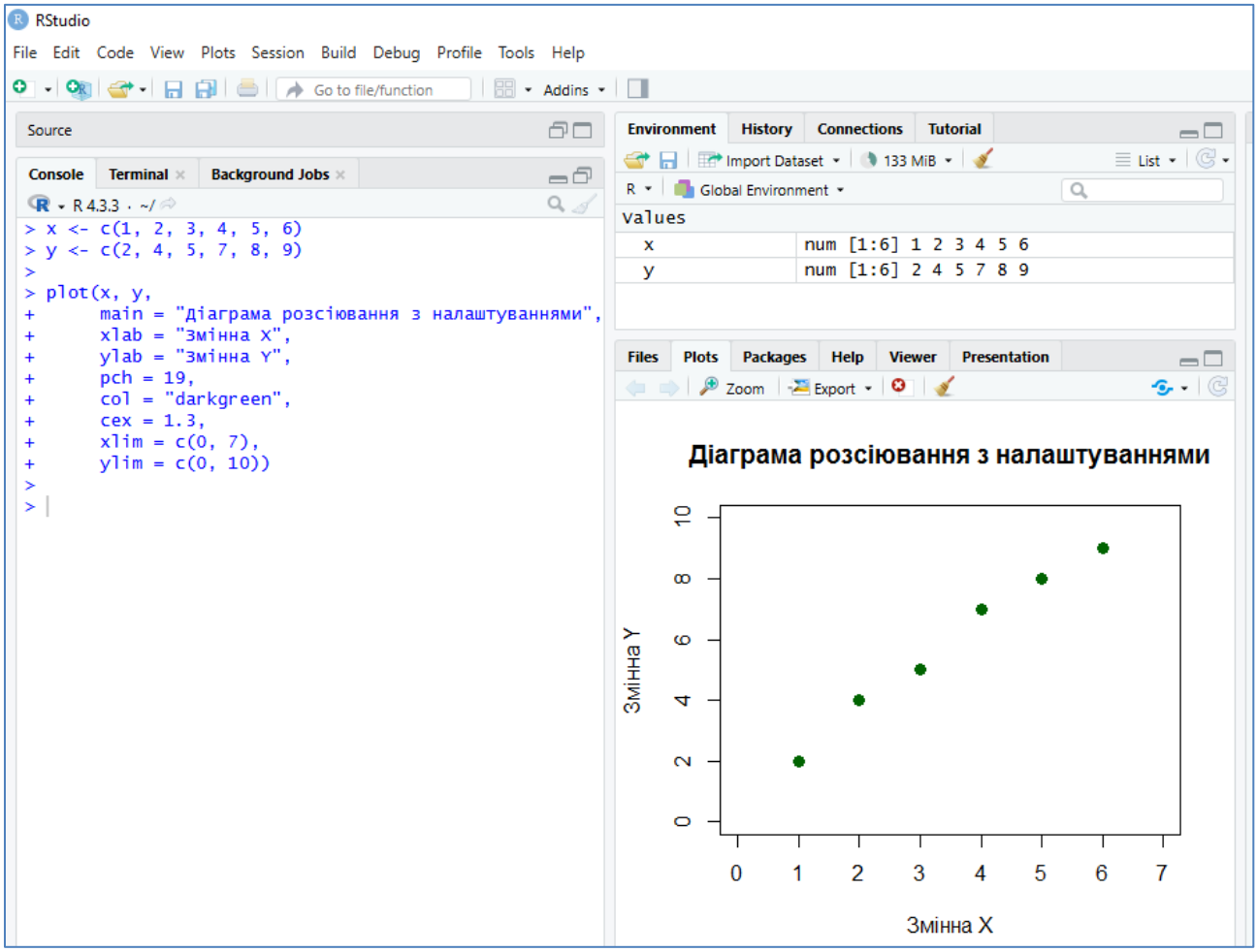
col – колір точок;

cex – розмір точок;

xlim, ylim – межі осей;

main, xlab, ylab – заголовок і підписи осей.

Приклад:



Діаграми розсіювання широко використовуються у наукових дослідженнях для перевірки гіпотез та попередньої оцінки статистичних моделей.

9.4.Візуалізація статистичних характеристик: boxplot(), barplot()

Під час аналізу даних важливим етапом є не лише виявлення загальних тенденцій, а й наочне представлення статистичних характеристик, таких як медіана, квартилі, розмах варіації, частоти та середні значення. Візуалізація цих

показників дозволяє швидко оцінити особливості розподілу даних, порівняти окремі групи спостережень і зробити обґрунтовані висновки. У мові R для цього широко використовуються функції boxplot() та barplot(), які є основними інструментами статистичної графіки.

1.Ящикові діаграми (boxplot())

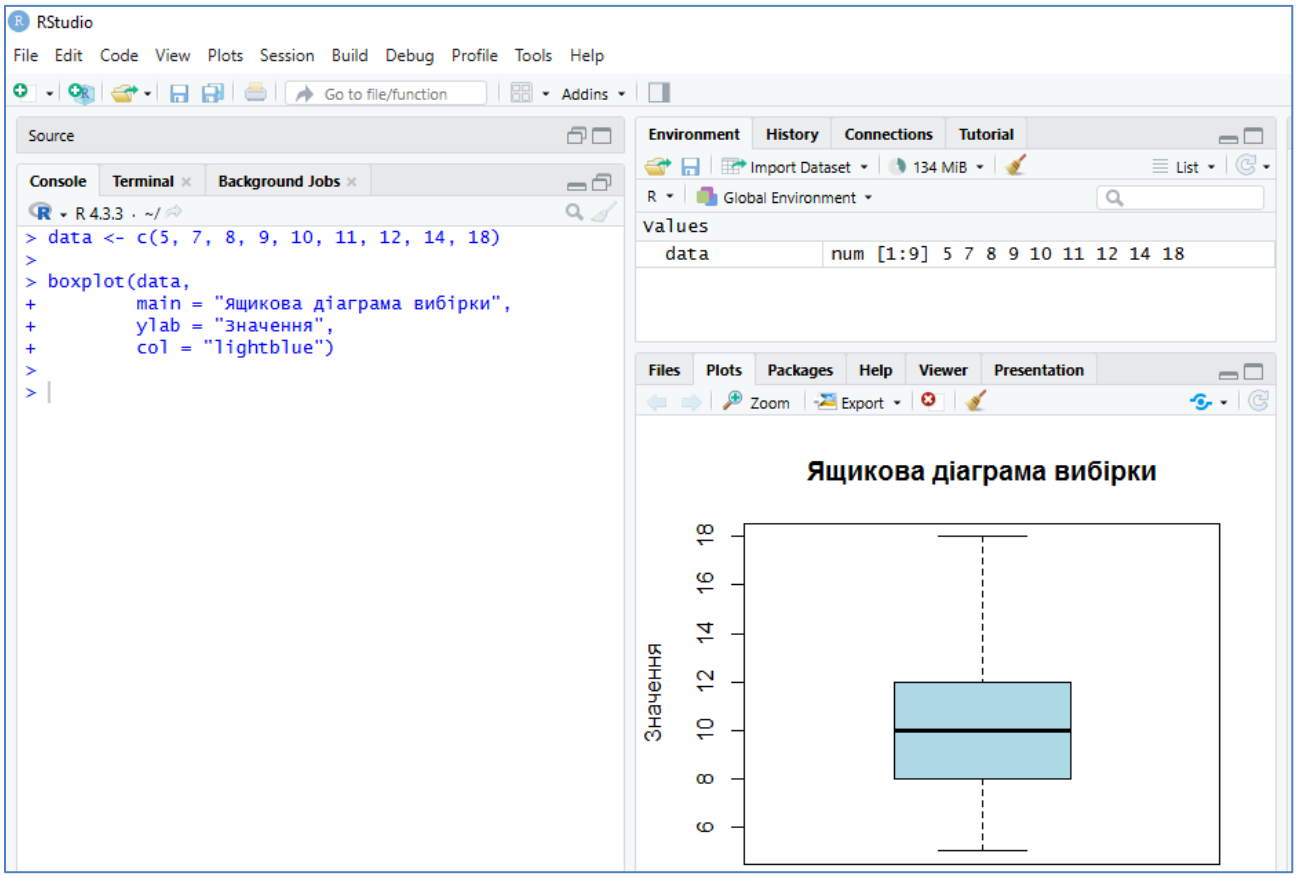
Функція boxplot() використовується для побудови ящикових діаграм (boxplot), які наочно відображають основні статистичні характеристики числових даних. Така діаграма показує медіану, нижній і верхній квартилі, міжквартильний розмах, а також можливі викиди.

Ящикова діаграма є особливо корисною для:

– порівняння розподілів кількох вибірок;

– виявлення асиметрії розподілу;

– знаходження аномальних значень (викидів).



Основні параметри функції boxplot():

main – заголовок діаграми;

xlab, ylab – підписи осей;

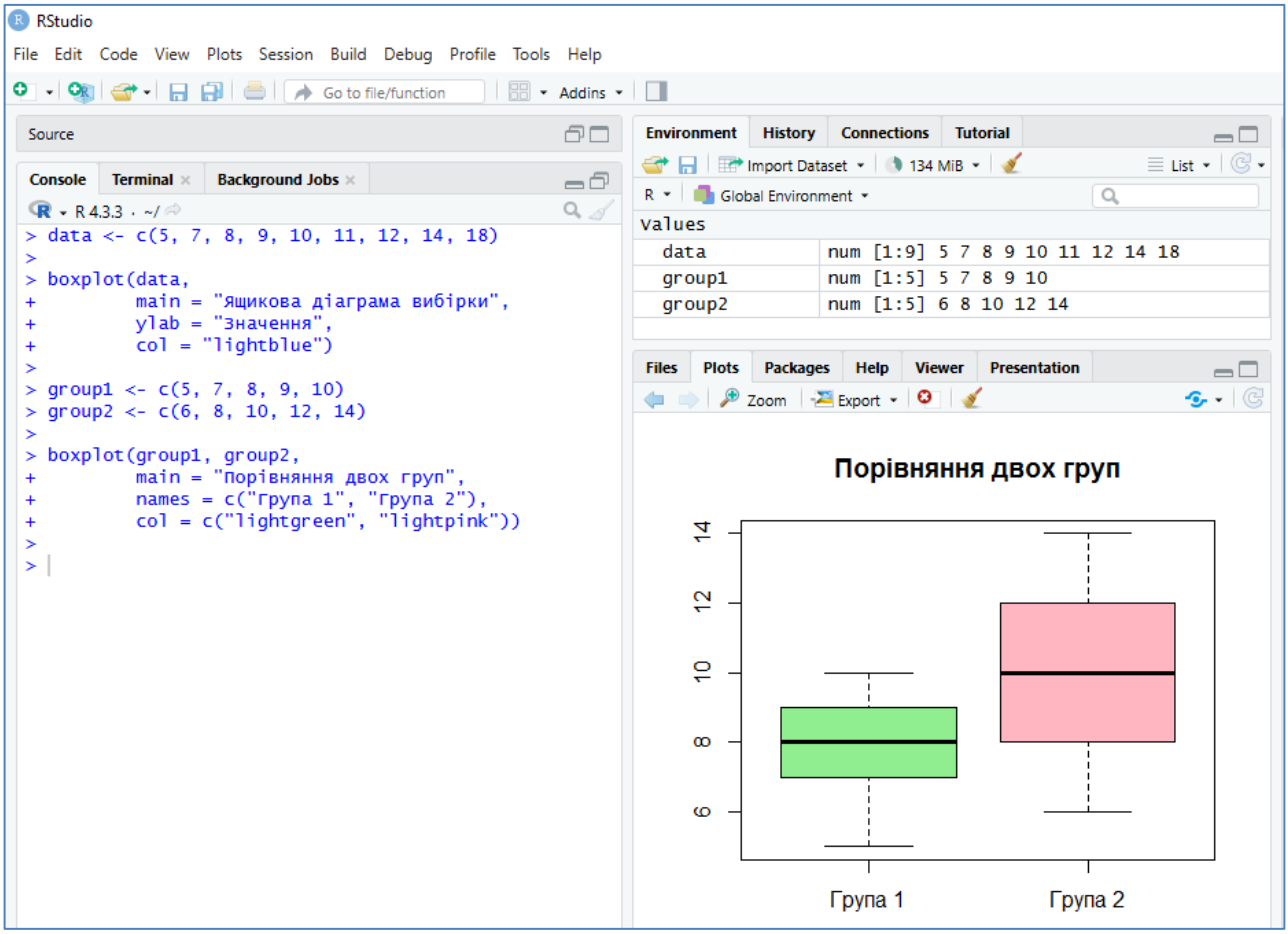
col – колір ящиків;

horizontal – орієнтація діаграми;

notch – відображення довірчого інтервалу медіани.

Приклад побудови ящикової діаграми:

Для порівняння кількох груп можна передати кілька векторів даних.



2.Стовпчикові діаграми (barplot())

Функція barplot() використовується для побудови стовпчикових діаграм, які відображають кількісні або категоріальні статистичні характеристики, зокрема частоти, середні значення або підсумкові показники.

Стовпчикові діаграми зручні для:

- порівняння значень між різними категоріями;
- візуалізації результатів підрахунків і агрегування;
- представлення статистичних даних у зрозумілій формі.

Основні параметри функції barplot():

main – заголовок діаграми;

xlab, ylab – підписи осей;

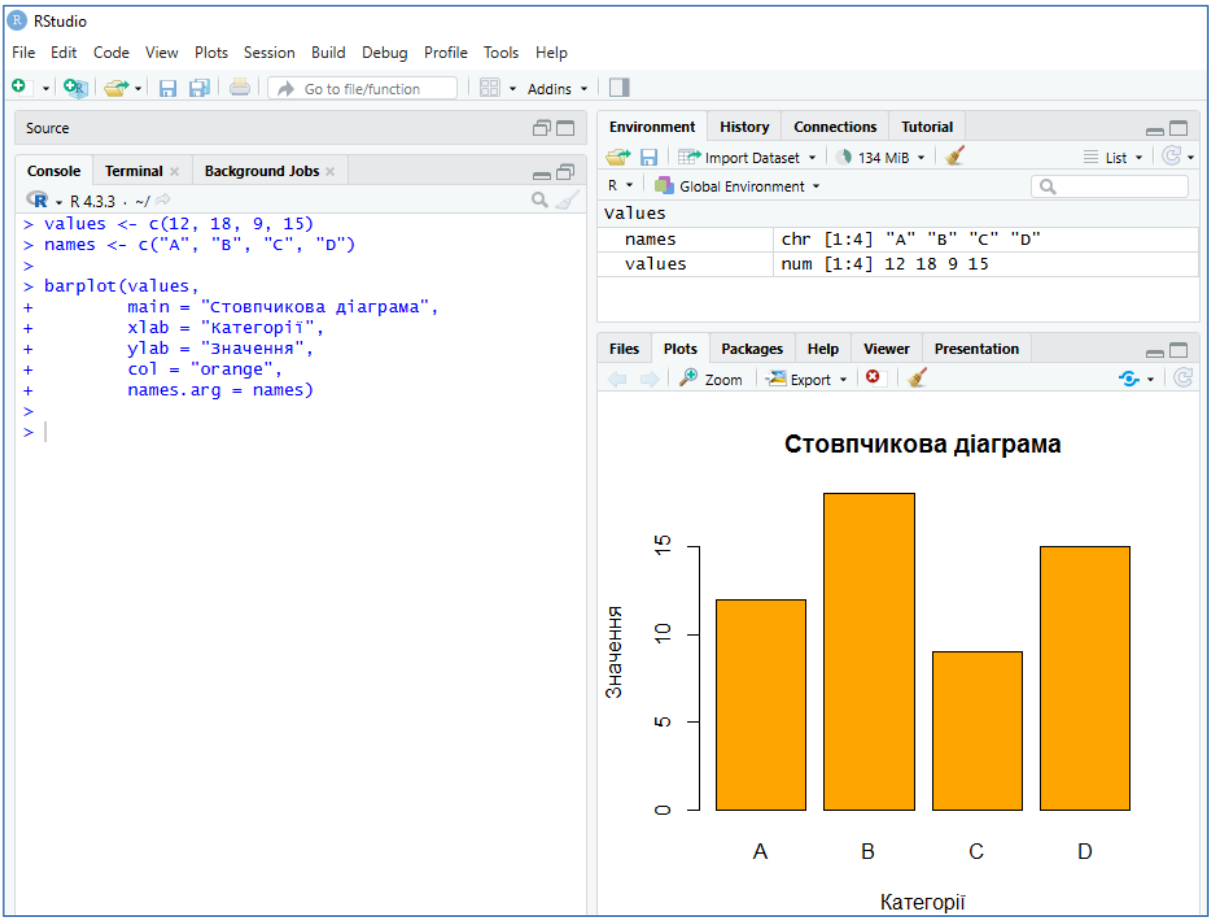
col – колір стовпців;

names.arg – підписи категорій;

beside – розміщення стовпців поруч або один над одним;

horiz – горизонтальна орієнтація діаграми.

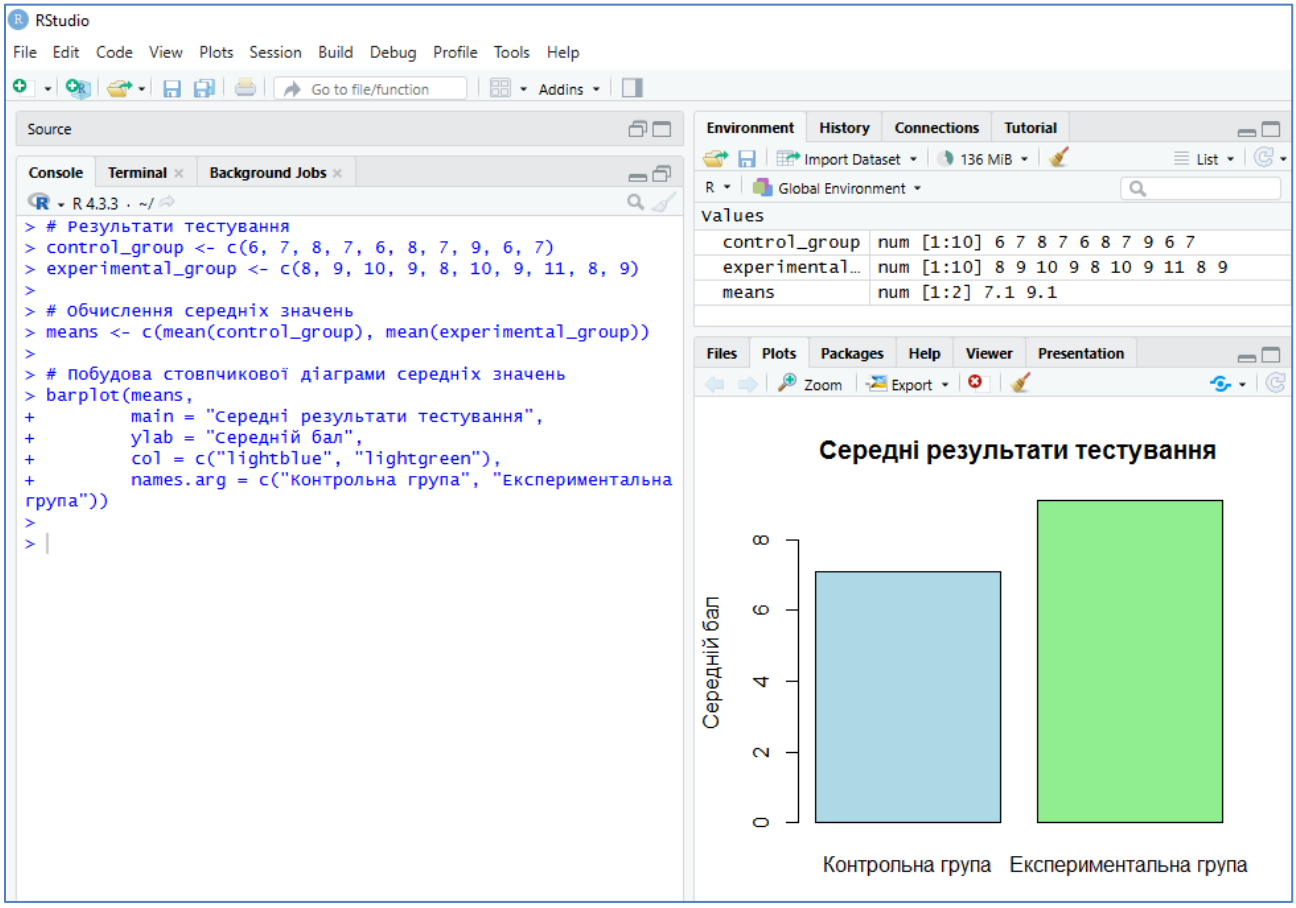
Приклад побудови стовпчикової діаграми:



Стовпчикові та ящикові діаграми часто використовуються разом для комплексного аналізу даних. Якщо barplot() дозволяє оцінити узагальнені показники, то boxplot() дає змогу глибше проаналізувати розподіл значень усередині кожної групи.

Приклад комплексного використання barplot() та boxplot():

Нехай маємо результати тестування учнів з інформатики у двох групах: контрольній та експериментальній. Необхідно порівняти середні бали та розподіл результатів у кожній групі.

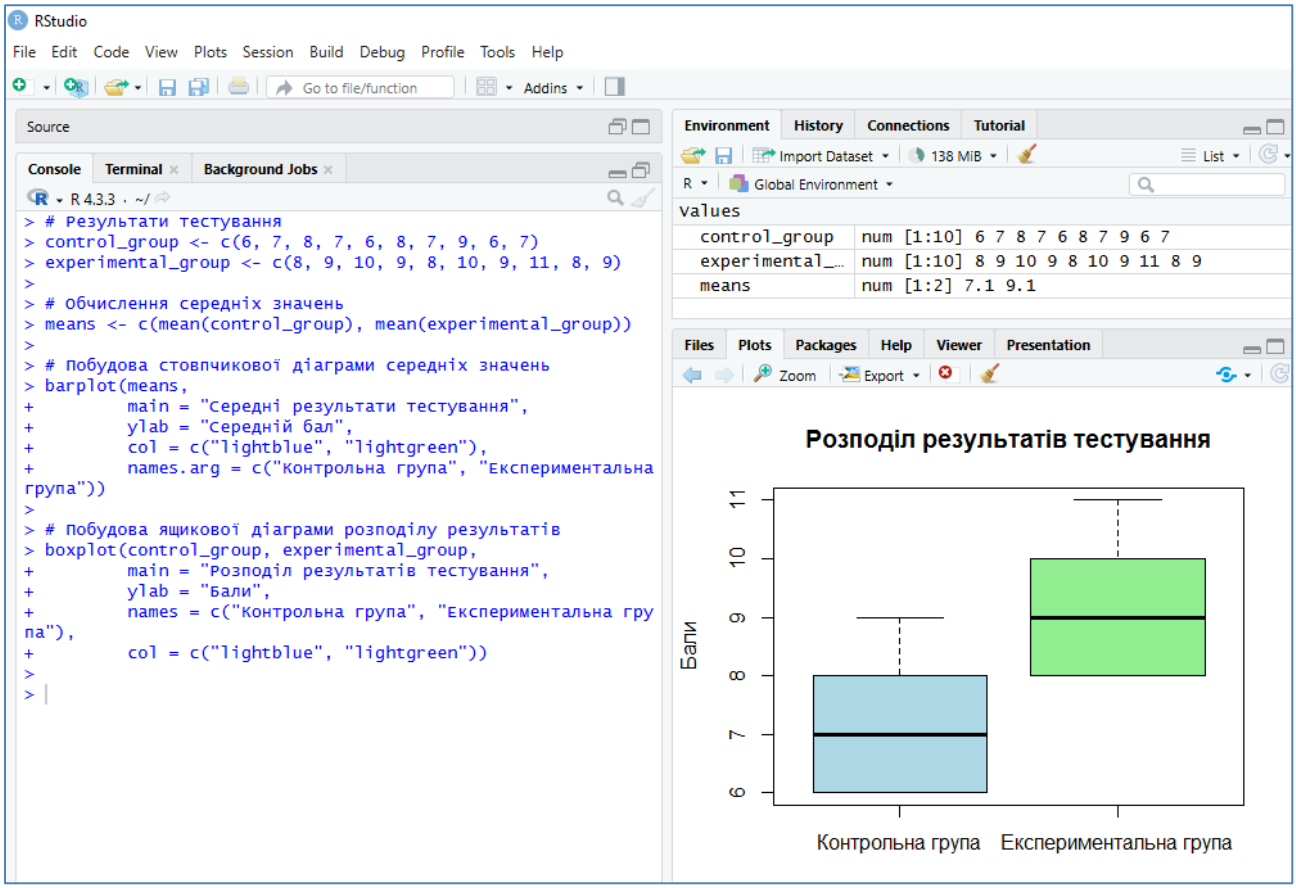


У даному випадку стовпчикова діаграма (barplot()) дозволяє швидко порівняти середні результати навчальних досягнень двох груп та зробити загальний висновок про їх відмінності.

Ящикова діаграма (boxplot()) дає змогу детально проаналізувати розподіл балів у кожній групі, оцінити медіану, варіативність результатів, а також виявити можливі викиди.

Стовпчикова діаграма показує, що експериментальна група має вищий середній бал порівняно з контрольною. Водночас ящикова діаграма демонструє, що результати експериментальної групи є більш стабільними та зосередженими навколо вищих значень, тоді як у контрольній групі спостерігається більша варіативність. Таким чином, спільне використання barplot() і boxplot() забезпечує комплексний підхід до аналізу даних, поєднуючи узагальнену

оцінку показників із детальним дослідженням розподілу значень усередині кожної групи.



9.5.Багатовимірна візуалізація даних

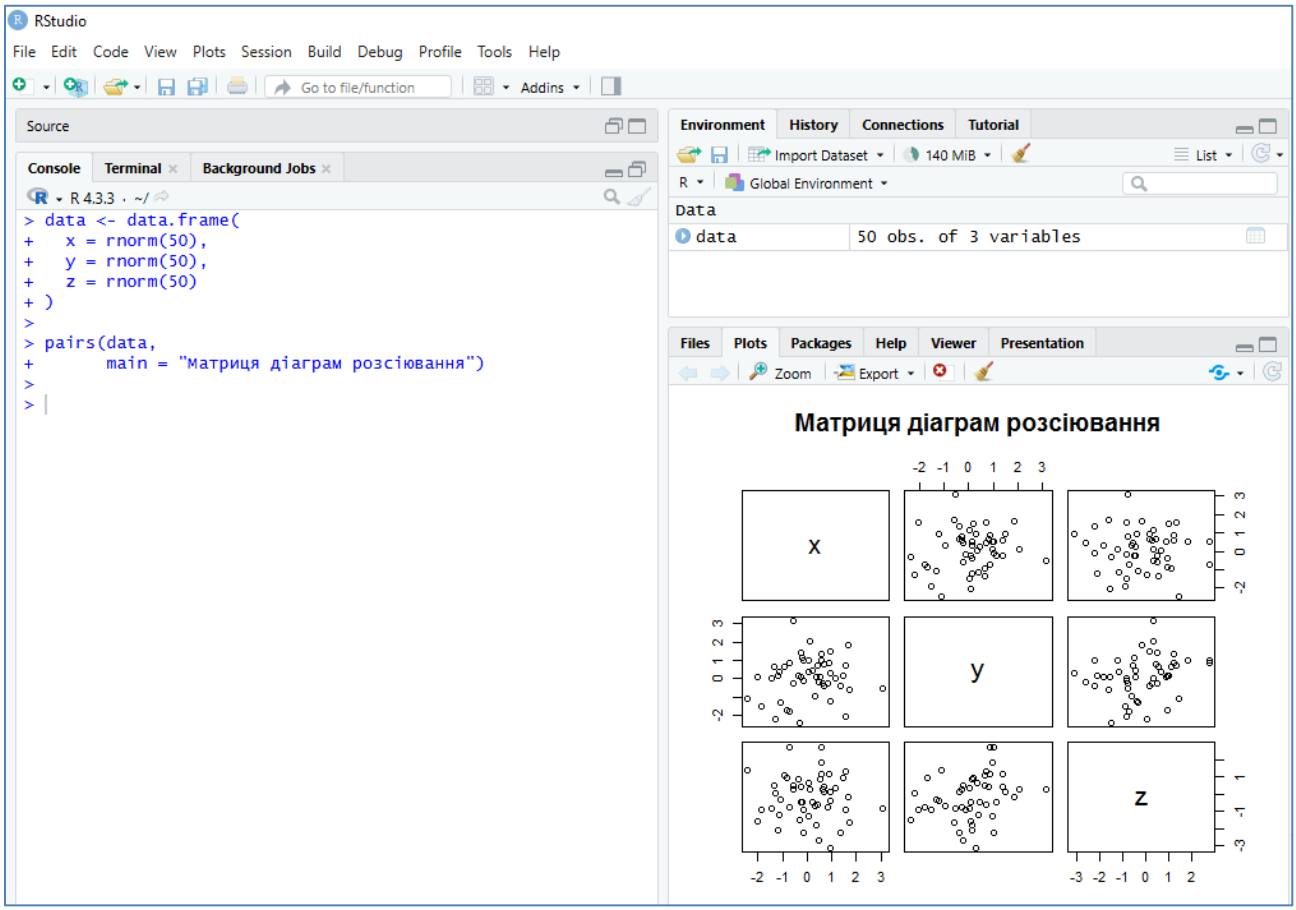
У практиці аналізу даних часто виникає потреба досліджувати декілька змінних одночасно, оскільки реальні процеси зазвичай описуються не двома, а значно більшою кількістю параметрів. Багатовимірна візуалізація даних спрямована на подання інформації, що містить три й більше змінних, у графічній формі, зручній для аналізу та інтерпретації. Такий підхід дозволяє виявляти складні взаємозв'язки, групування спостережень і приховані закономірності, які важко помітити за допомогою одновимірних або двовимірних графіків.

У мові R реалізовано низку засобів для багатовимірної візуалізації, як у базовій графічній системі, так і з використанням додаткових пакетів. Ці інструменти широко застосовуються у статистиці, машинному навчанні, освітніх дослідженнях та інших галузях.

Одним із найпоширеніших способів візуалізації багатовимірних даних є розширення діаграми розсіювання, коли додаткові змінні кодуються за допомогою кольору, розміру або форми точок. Такий підхід дозволяє в межах одного графіка відобразити три–чотири змінні.

Іншим важливим методом є матриці діаграм розсіювання, які дають змогу одночасно проаналізувати попарні взаємозв'язки між кількома змінними. У R для цього використовується функція pairs().

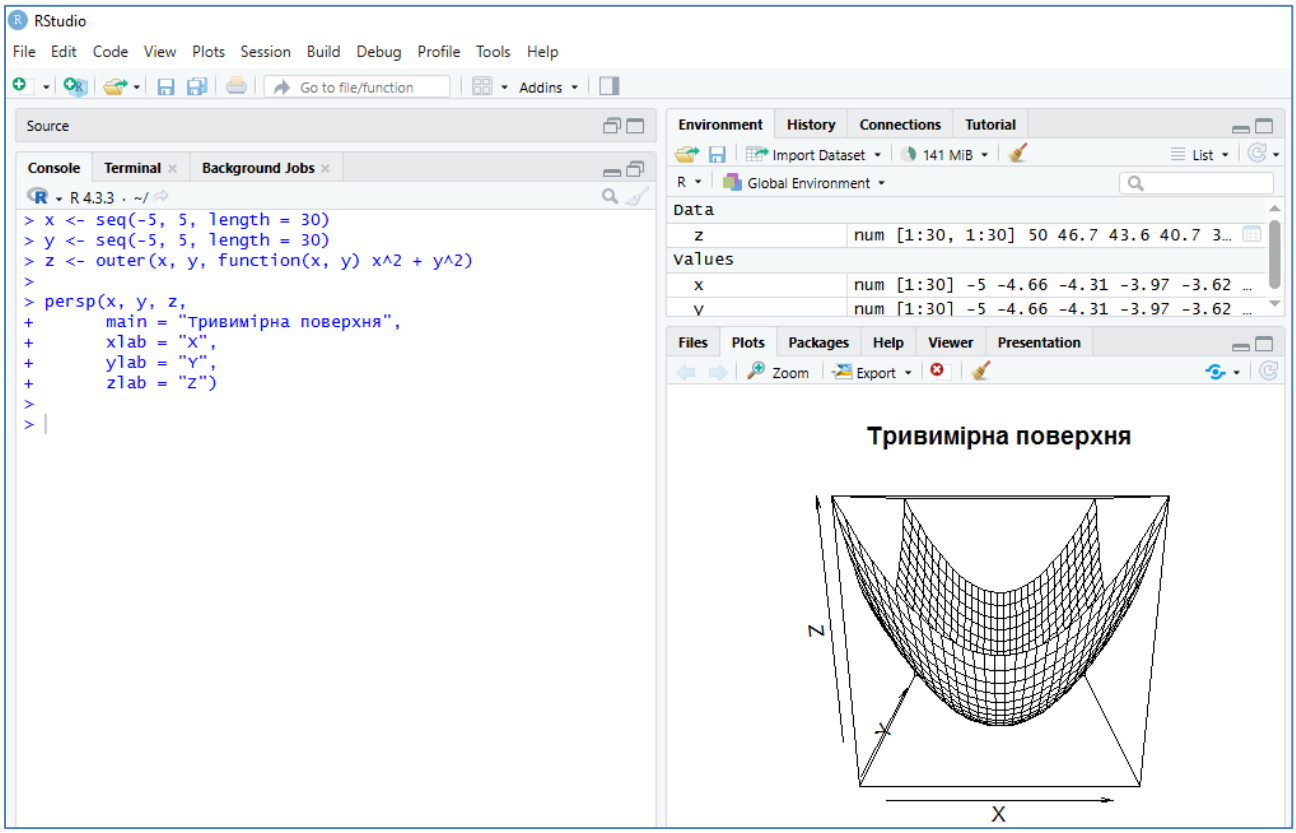
Приклад матриці діаграм розсіювання:



Такий графік дозволяє швидко оцінити наявність кореляцій між змінними та загальний характер їх взаємодії.

Для відображення трьох числових змінних використовуються тривимірні графіки, зокрема просторові діаграми розсіювання та поверхні. У базовій графіці R прикладом є функція persp(), яка дозволяє будувати тривимірні поверхні.

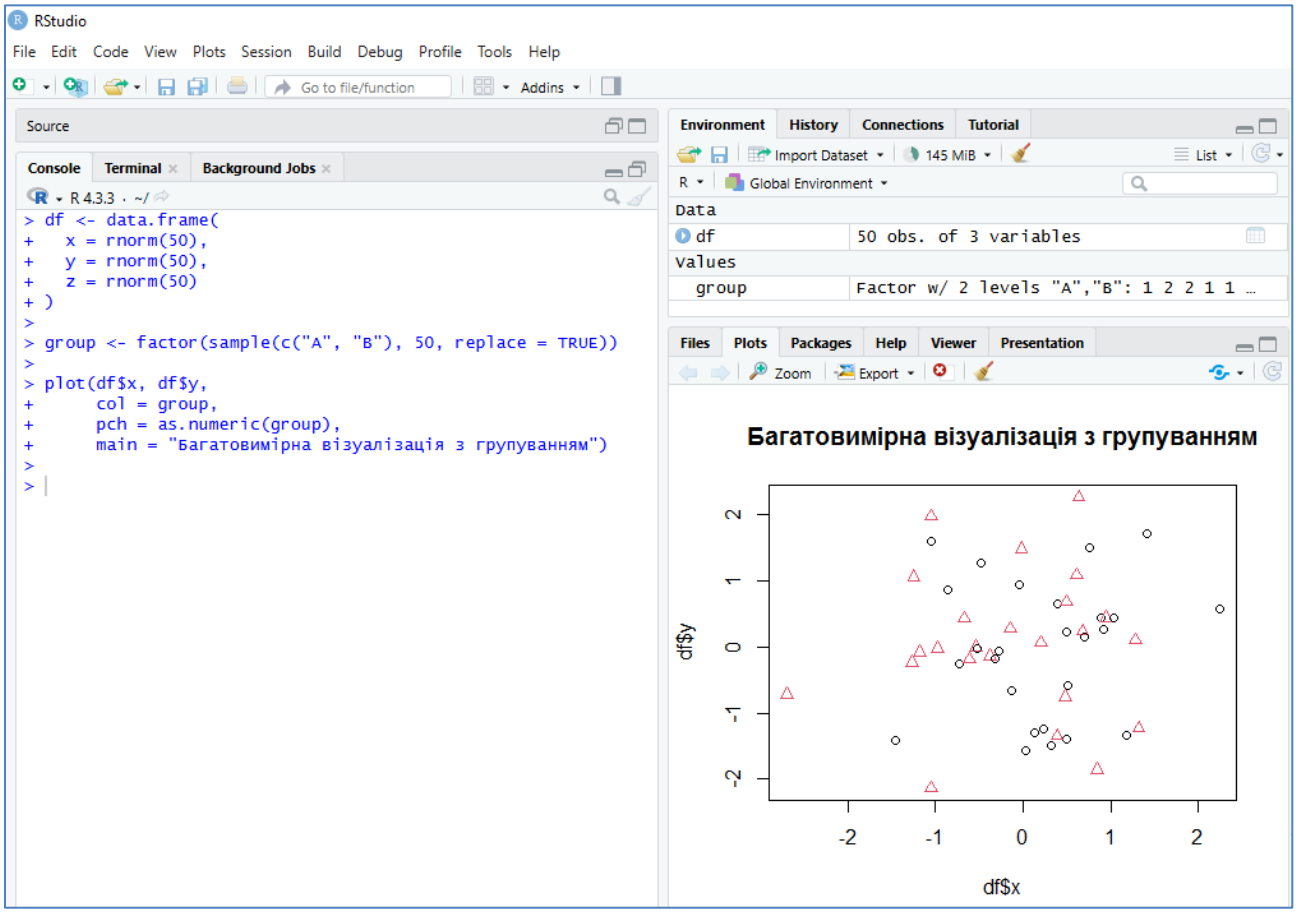
Приклад тривимірного графіка:



У багатовимірному аналізі важливу роль відіграє групування даних за факторними змінними. У R групи можуть відображатися різними кольорами або символами, що дозволяє порівнювати підмножини даних у межах одного графіка.

Багатовимірна візуалізація є важливим інструментом попереднього аналізу даних, оскільки дозволяє:

- виявляти структуру та кластери у даних;
- оцінювати взаємозв’язки між кількома змінними;
- підвищувати якість інтерпретації статистичних моделей.



9.6.Побудова складних графіків за допомогою ggplot2

Пакет ggplot2 входить до складу екосистеми tidyverse та базується на концепції граматики графіки (Grammar of Graphics), запропонованої Л. Вікхемом. Згідно з цією концепцією, графік розглядається як поєднання незалежних компонентів, кожен з яких відповідає за певний аспект візуального представлення даних.

На відміну від базової графіки R, де кожен графік створюється окремою функцією, у ggplot2 графіки будуються шляхом нашарування (layering) різних елементів. Це забезпечує більшу гнучкість, читабельність коду та можливість створення складних багатовимірних візуалізацій.

Основні переваги пакета ggplot2:

- декларативний стиль опису графіків;
- підтримка багатовимірних даних;
- автоматичне масштабування та легенди;
- висока якість графічного оформлення;

– зручність інтеграції з data.frame та tibble.

Пакет ggplot2 широко використовується в статистичному аналізі, машинному навчанні, освітніх дослідженнях та наукових публікаціях.

Підключення пакета здійснюється командою library(ggplot2).

Побудова графіків у ggplot2 здійснюється шляхом визначення естетик (aes) та геометричних об’єктів (geoms).

Естетики визначають, як змінні даних відображаються на візуальні характеристики графіка.

Найуживаніші естетики:

x, y – координати на осях;

color – колір елементів;

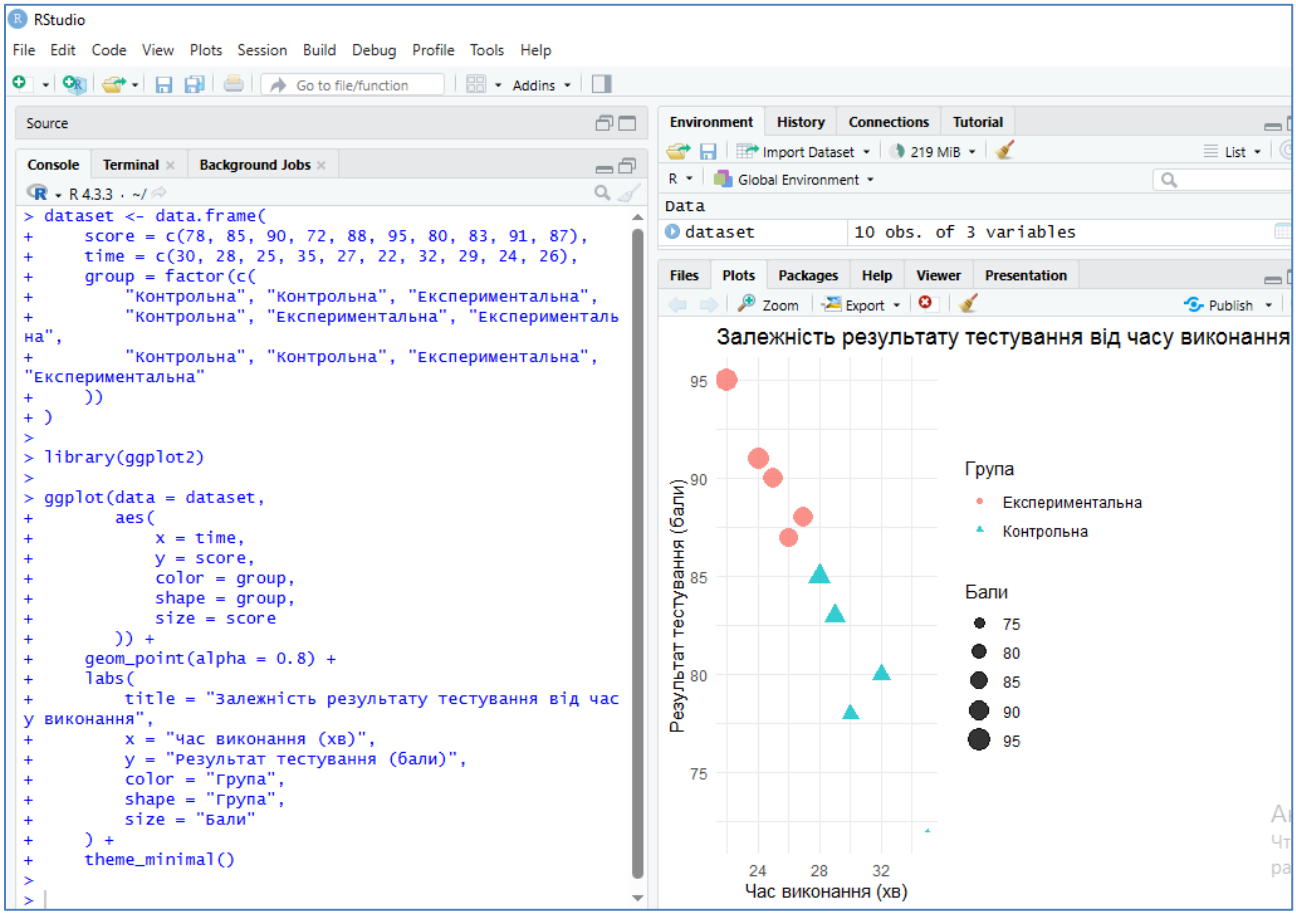
fill – колір заповнення;

size – розмір;

shape – форма маркерів;

alpha – прозорість.

Приклад задання естетик:



Геометричні об’єкти визначають тип графічного представлення даних. Основні геометрії:

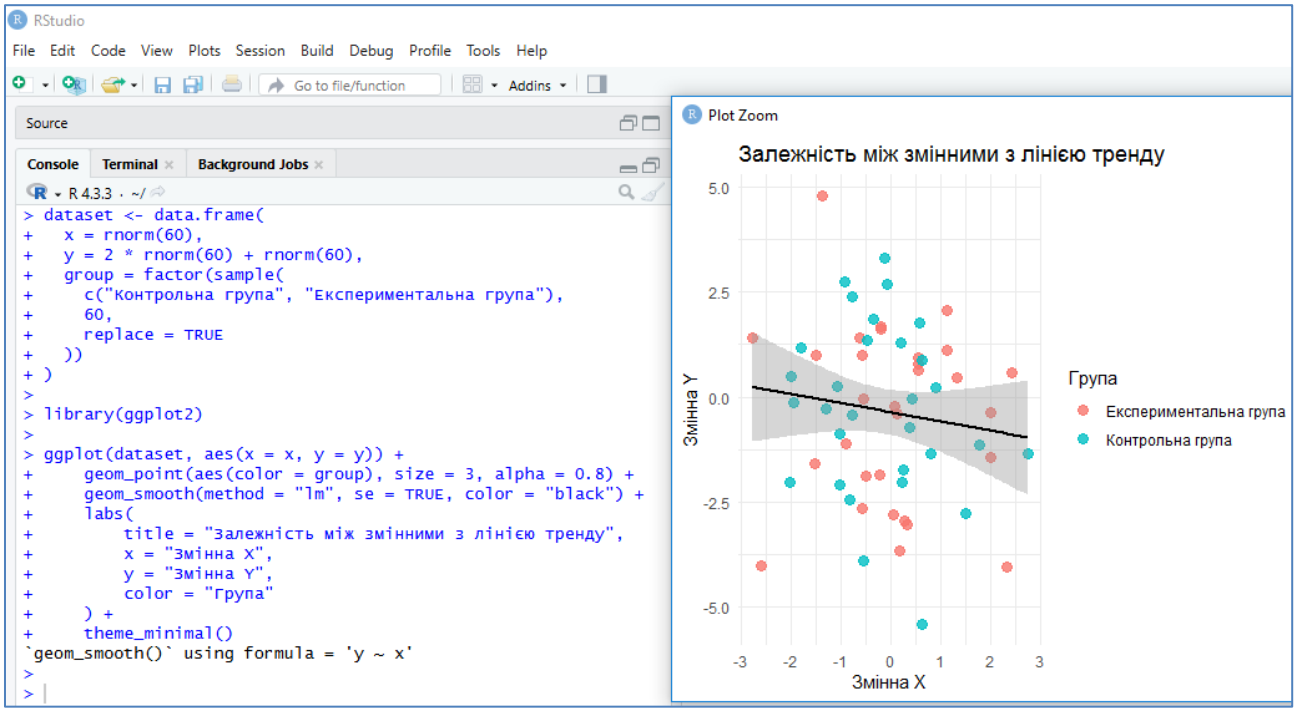
Геометричний об’єкт	Призначення
geom_point()	діаграма розсіювання
geom_line()	лінійний графік
geom_bar()	стовпчикова діаграма
geom_histogram()	гістограма
geom_boxplot()	ящикова діаграма
geom_smooth()	лінія тренду

Приклад:

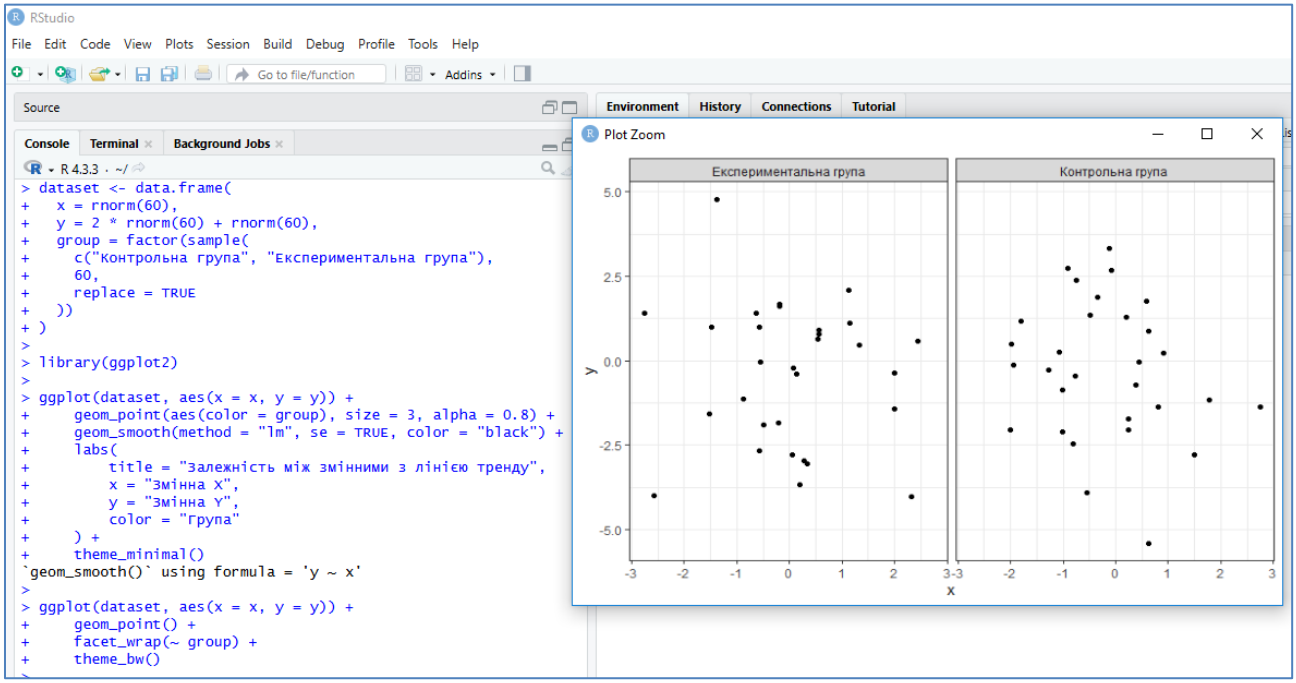
```
ggplot(dataset, aes(x = x, y = y)) +
  geom_point()
```

Однією з переваг ggplot2 є можливість створення складних комбінованих графіків, що дозволяють одночасно аналізувати кілька змінних та їх взаємозв’язки.

Приклад комбінування кількох геометрій:



Фасетування дозволяє розділити дані на підграфіки за значеннями факторної змінної:




```
> dataset <- data.frame(
+   x = rnorm(60),
+   y = 2 * rnorm(60) + rnorm(60),
+   group = factor(sample(
+     c("Контрольна група", "Експериментальна група"),
+     60,
+     replace = TRUE
+   ))
+ )
>
> library(ggplot2)
>
> ggplot(dataset, aes(x = x, y = y)) +
+     geom_point(aes(color = group), size = 3, alpha = 0.8) +
+     geom_smooth(method = "lm", se = TRUE, color = "black") +
+     labs(
+         title = "Залежність між змінними з лінією тренду",
+         x = "Змінна X",
+         y = "Змінна Y",
+         color = "Група"
+     ) +
+     theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
>
> ggplot(dataset, aes(x = x, y = y)) +
+     geom_point() +
+     facet_wrap(~ group) +
+     theme_bw()
```

9.7.Збереження та експорт графіків у різні формати

Після побудови графіків важливим етапом аналізу даних є їх збереження та експорт для подальшого використання у звітах, наукових публікаціях, презентаціях або вебресурсах. Мова R надає кілька способів експорту графічних об’єктів, які відрізняються гнучкістю налаштувань, підтримуваними

У графіці R збереження здійснюється шляхом відкриття графічного пристрою перед побудовою графіка та його закриття після завершення.

1.Збереження у формат PNG

```
png("plot.png", width = 800, height = 600)
plot(x, y, main = "Приклад графіка")
dev.off()
```

2.Збереження у формат PDF

```
pdf("plot.pdf")
plot(x, y)
dev.off()
```

Поширені параметри:

width, height – розміри зображення;

res – роздільна здатність (dpi);

bg – фон.

Для графіків, створених із використанням пакета ggplot2, рекомендовано застосовувати функцію ggsave(), яка автоматично зберігає останній побудований графік або явно заданий об’єкт.

Приклад:

```
ggsave("plot.png")
```

Збереження з указанням об’єкта графіка

```
p <- ggplot(dataset, aes(x = x, y = y)) +
  geom_point()
ggsave("scatter_plot.jpg", plot = p)
```

Налаштування параметрів експорту:

```
ggsave(
  filename = "analysis_plot.png",
  plot = p,
  width = 8,
  height = 6,
  dpi = 300
)
```

Приклад експорту у формат SVG:

```
ggsave("plot.svg", plot = p)
```

Основні параметри функції ggsave():

filename – ім’я файлу;

plot – об’єкт ggplot;

width, height – розміри (за замовчуванням у дюймах);

dpi – роздільна здатність;

units – одиниці вимірювання ("in", "cm", "mm").

Таблиця 9.1.

Основні формати експорту

Формат	Призначення
PNG	веб, презентації
JPG	компактні зображення
PDF	наукові статті
SVG	масштабована векторна графіка
TIFF	поліграфія

Детальну інформацію про графічні функції можна отримати використавши функцію help().

```
help(plot)        # Відкриває документацію для функції plot()
?plot             # Те саме, що і help(plot)
help(package = "ggplot2")      # Перелік функцій і документація пакета ggplot2
```

Лабораторні роботи до розділу 9

Лабораторна робота 1. Двовимірна графіка та статистичні діаграми

Мета роботи: формування практичних навичок:

- побудови двовимірних графіків у середовищі R;
- використання функцій високого рівня plot(), hist(), boxplot(), barplot(), pie();
- візуального аналізу статистичних даних;
- налаштування графічного вікна за допомогою par().

Теоретичні відомості

Графіка високого рівня в R призначена для швидкої побудови завершених графіків на основі числових і категоріальних даних. Такі функції автоматично створюють графічне вікно, масштабують осі та додають базові елементи візуалізації.

Функція plot() є універсальною і залежно від типу аргументів може будувати:

- лінійні графіки;
- діаграми розсіювання;
- графіки для факторів і таблиць даних.

Статистичні функції (hist(), boxplot(), barplot(), pie()) дозволяють аналізувати розподіли, варіацію та структуру даних.

Хід виконання роботи

Завдання 1. Підготовка даних із базових компонентів RStudio.

```
data(iris)
head(iris)
```

Пояснення: Набір даних iris використовується як стандартний приклад для статистичного аналізу.

Завдання 2. Побудувати діаграму розсіювання, що дозволить оцінити взаємозв'язок між двома числовими змінними.

```
plot(iris$Sepal.Length, iris$Petal.Length,
  main = "Діаграма розсіювання",
  xlab = "Довжина чашолистка",
  ylab = "Довжина пелюстки")
```

Завдання 3. Побудувати гістограму використовуючи один із параметрів бази даних iris. Гістограма відображає розподіл числової змінної.

```
hist(iris$Sepal.Length,
  main = "Гістограма довжини чашолистка",
  xlab = "Значення",
  col = "lightblue")
```

Завдання 4. Побудувати ящик Boxplot() для опису розподілу числових даних. Показати: медіану (лінія всередині ящика); квартилі Q1 і Q3 (межі ящика); міжквартильний розмах (IQR = Q3 − Q1); викиди (окремі точки).

```
boxplot(iris$Sepal.Length ~ iris$Species,
  main = "Boxplot за видами",
```

```
   col = c("red", "green", "blue"))
```

Завдання 5. Побудувати кругову та стовпчикову діаграми.

```
counts <- table(iris$Species)
barplot(counts, main = "Стовпчикова діаграма")
pie(counts, main = "Кругова діаграма")
```

Завдання 6. Налаштування графічного вікна

```
par(mfrow = c(2, 2))
plot(iris$Sepal.Length)
hist(iris$Petal.Length)
boxplot(iris$Sepal.Width)
pie(counts)
```

За аналізом отриманих результатів підготувати звіт. Описати, які побудовані графіки, оцінити розподіли, порівняти групи даних та вказати виявлені у них закономірності.

Лабораторна робота 2. Багатовимірна та тривимірна графіка в R

Мета роботи: формування практичних навичок:

- побудови багатовимірних графіків;
- використання функцій pairs(), matplot(), contour(), image();
- створення тривимірних графіків (persp(), scatterplot3d()).

Теоретичні відомості

Для аналізу багатовимірних даних у R використовують спеціалізовані графічні функції. Функція pairs() створює матрицю діаграм розсіювання, що дозволяє одночасно аналізувати кілька змінних. Тривимірна графіка (persp(), scatterplot3d()) використовується для відображення поверхонь та просторових взаємозв’язків між змінними.

Хід виконання роботи

Завдання 1. Побудувати діаграму розсіювання параметрів iris/

```
pairs(iris[1:4],
  pch = 21,
  bg = c("red", "green3", "blue")[unclass(iris$Species)])
```

Пояснення: Кожна пара змінних відображається на окремій діаграмі.

Завдання 2. Побудувати просторовий графік

```
x <- seq(-2, 2, length = 30)
y <- seq(-2, 2, length = 30)
z <- outer(x, y, function(x, y) x^2 + y^2)
contour(x, y, z, main = "Контурний графік")
```

Завдання 3. Вивести зображеннята зберегти йогоу файл з розширенням .png

```
image(x, y, z, main = "Кольорова сітка")
```

Завдання 4. Побудувати тривимірну поверхню та зберегти у форматі .jpg

```
persp(x, y, z,
  theta = 30, phi = 30,
  col = "lightblue",
  main = "3D поверхня")
```

Завдання 5. Побудувати тривимірну діаграму розсіювання, використавши параметри бази даних iris.

```
library(scatterplot3d)
scatterplot3d(iris$Petal.Width,
      iris$Sepal.Length,
      iris$Sepal.Width,
      color = c("red", "green3", "blue")[unclass(iris$Species)])
```

Лабораторна робота 3. Функції низького рівня для візуалізації даних у R

Мета роботи: закріпити теоретичні знання та набути практичних навичок використання функцій низького рівня мови R для детального налаштування, додавання графічних об'єктів та анотацій на існуючий графік.

Теоретичні відомості

Мова R надає потужний набір інструментів для створення графіків. Функції низького рівня (Low-level plotting functions) – це функції, які не створюють новий графік, а додають елементи (точки, лінії, текст, осі тощо) до вже існуючого графічного вікна. Вони дозволяють створювати складні та висококастомізовані візуалізації.

Хід виконання роботи

Створити єдиний, інформативне та естетично оформлене графічне зображення, використовуючи виключно функції низького рівня для додавання всіх елементів, окрім початкового полотна.

Завдання 1. Підготовити дані та створити базове полотно.

1. Створити два вектори даних x та y1, а також додатковий вектор y2 для другого набору даних.

```
x <- 1:10
y1 <- x + rnorm(10, mean = 0, sd = 1.5)
y2 <- 10 - x + rnorm(10, mean = 0, sd = 1)
  Створення третьої змінної для функції symbols()
z <- abs(rnorm(10, mean = 5, sd = 2))
```

2. Створити порожнє графічне вікно, використовуючи функцію високого рівня plot() з параметром type='n', щоб визначити межі осей, але не малювати дані.

```
plot(x, y1, type = 'n',
   main = "", xlab = "", ylab = "",
   xlim = c(0, 11), ylim = c(0, 12))
```

Завдання 2. Додавання основних елементів та даних

1. Використати функцію title() для додавання основного заголовку графіка та підписів осей.

```
title(main = "Аналіз двох наборів даних",
   xlab = "Незалежна змінна (X)",
   ylab = "Залежна змінна (Y)")
```

2. Додати рамку навколо графіка за допомогою функції box().

```
box()
```

3. Додати прямокутну сітку на графік за допомогою функції grid().

```
grid(lty = 2, col = "gray")
```

4. Додати перший набір даних (x, y1) як точки, використовуючи функцію points().

```
points(x, y1, pch = 19, col = "blue")
```

5. Додати другий набір даних (x, y2) як лінію, використовуючи функцію lines().

```
lines(x, y2, lty = 1, col = "red")
```

Завдання 3. Виведенняна екран анотації та додаткових графічних об'єктів.

1. Додати горизонтальну лінію, що позначає середнє значення y1, використовуючи функцію abline().

```
abline(h = mean(y1), lty = 2, col = "darkblue")
```

2. Використайте функцію segments() для з'єднання першої та останньої точки набору y1.

```
segments(x0 = x[1], y0 = y1[1], x1 = x[10], y1 = y1[10],
    col = "blue", lty = 3)
```

3. Використайте функцію arrows() для позначення аномалії (наприклад, найбільшого значення y1).

```
max_y1_index <- which.max(y1)
arrows(x0 = x[max_y1_index], y0 = y1[max_y1_index] + 1,
   x1 = x[max_y1_index], y1 = y1[max_y1_index],
   length = 0.1, angle = 30, col = "orange", lwd = 2)
```

4. Додати текстову анотацію до стрілки, використовуючи функцію text().

```
text(x = x[max_y1_index], y = y1[max_y1_index] + 1.2,
  labels = "Аномалія", col = "orange")
```

5. Використайте функцію mtext() для додавання інформації про автора або дату у зовнішнє поле графіка.

mtext("Лабораторна робота: Функції низького рівня", side = 1, line = 4, adj = 1, cex = 0.8)

Завдання 4. Побудова складних фігур та легенд.

1. Використайте функцію rect() для виділення області інтересу (наприклад, діапазону X від 3 до 7).

```
rect(xleft = 3, ybottom = 0, xright = 7, ytop = 12,
   col = rgb(0.5, 0.5, 0.5, 0.2), border = NA)
```

2. Використайте функцію polygon() для затінення області між лінією y2 та віссю X.

```
polygon(c(x, rev(x)), c(y2, rep(0, length(y2))),
    col = rgb(1, 0, 0, 0.1), border = NA)
```

3. Використайте функцію symbols() для відображення змінної z як розміру кіл навколо точок y1.

```
symbols(x, y1, circles = z/15, inches = FALSE, add = TRUE,
    bg = rgb(0, 0, 1, 0.3), fg = NA)
```

4. Додайте легенду до графіка за допомогою функції legend().

```
legend("topright", legend = c("Набір даних 1 (точки)", "Набір даних 2 (лінія)", "Середнє значення Y1"), col = c("blue", "red", "darkblue"),
    pch = c(19, NA, NA),
    lty = c(NA, 1, 2),
    lwd = c(NA, 1, 1),
    bg = "white")
```

5. Побудуйте згладжену лінію (сплайн) для набору y2 за допомогою xspline().

```
xspline(x, y2, shape = 0.4, border = "purple", lwd = 2)
```

6. Оформіть текстовий звіт.

Вимоги до звіту:

Пояснити різницю між функціями високого та низького рівня у візуалізації R.

Описати, які саме функції низького рівня були використані для досягнення кінцевого результату.

Описати, як візуалізація допомагає виявити закономірності (наприклад, спадний тренд у y2 та зростаючий у y1) та аномалії (виділену стрілкою точку).

Лабораторна робота 4. Оператори, індексація та візуалізація виразів у середовищі RStudio

Мета роботи: закріпити практичні навички формування виразів у мові R; відпрацювати механізми індексації векторів і матриць; навчитися будувати ефектні математичні поверхні (параболоїд, сідлова поверхня); опанувати збереження результатів обчислень та графіків у файли.

Теоретичні відомості

У мові R будь-яке обчислення оформлюється у вигляді виразу, який складається з операторів, змінних та значень. Операції в R є векторизованими, що дозволяє працювати з масивами даних без використання циклів.

Індексація дозволяє отримувати доступ до окремих елементів або підмножин даних. Вона широко застосовується при роботі з матрицями та табличними структурами.

Мова R також має потужні засоби візуалізації, які дають змогу наочно представляти результати обчислень, зокрема математичні поверхні.

Хід виконання роботи

Завдання 1. Створити послідовності значень для змінних x та y у діапазоні від -5 до 5 з кроком 0.25.

```
x <- seq(-5, 5, by = 0.25)
y <- seq(-5, 5, by = 0.25)
```

Створити двовимірну координатну сітку:

```
grid <- expand.grid(x = x, y = y)
```

Завдання 2. Задано математичний вираз $z^2 = x^2 + y^2$, що описує поверхню параболоїда.

Обчислити значення z:

```
grid$z <- grid$x^2 + grid$y^2
```

Проаналізувати перші рядки отриманих даних:

```
head(grid)
```

Завдання 3. Індексація та аналіз даних. Вибрати всі точки, для яких z < 10.

```
subset_low <- grid[grid$z < 10, ]
```

Порахувати кількість таких точок.

```
nrow(subset_low)
```

Отримати координати x та y лише для вибраних точок.

```
subset_low[, c("x", "y")]
```

Завдання 4. Перетворення у матричну форму. Для побудови поверхні необхідно перетворити вектор z у матрицю:

```
z_matrix <- matrix(grid$z, nrow = length(x), ncol = length(y))
```

Завдання 5. Побудова 3D-графіка (параболоїд). Використати базову функцію persp():

```
persp(x, y, z_matrix,
  theta = 30,
  phi = 30,
  col = "lightblue",
  main = "Параболоїд: z = x^2 + y^2",
  xlab = "X",
  ylab = "Y",
  zlab = "Z")
```

Проаналізувати форму поверхні та вплив параметрів theta і phi.

Завдання 6. Альтернативний приклад (сідлова поверхня)

Використайте математичний вираз: [z = x^2 - y^2]

```
grid$z2 <- grid$x^2 - grid$y^2
z_matrix2 <- matrix(grid$z2, nrow = length(x))
persp(x, y, z_matrix2,
  theta = 40,
```

```
  phi = 20,
  col = "lightgreen",
  main = "Сідлова поверхня: z = x^2 - y^2")
```

Завдання 7. Збереження результатів:

```
Збереження таблиці даних
write.csv(grid, "surface_data.csv", row.names = FALSE)
Збереження графіка у файл
png("paraboloid.png", width = 800, height = 600)
persp(x, y, z_matrix,
  theta = 30,
  phi = 30,
  col = "lightblue",
  main = "Параболоїд")
dev.off()
```

Завдання 8. Аналіз отриманих результатів:

- пояснити форму побудованих поверхонь;
- описати роль індексації у відборі даних;
- вказати, у яких галузях можуть використовуватись подібні моделі.

Інтегровані завдання до розділу 9

Завдання 1.

1. Побудувати гістограму, використавши змінну набору iris.
2. Створити boxplot для двох числових змінних набору iris.
3. Побудувати кругову діаграму для власного набору даних.

Завдання 2.

1. Побудувати 3D-графік для будь-якої математичної функції.
2. Створити власну матрицю діаграм розсіювання.
3. Змінити параметри theta і phi та проаналізувати результат.

Завдання 3.

1. Побудувати поверхню виду z = sin(x) * cos(y) для заданих X та У.

2. Змінити діапазон та крок координатної сітки і проаналізувати результат.

3. Зберегти графік у форматі PDF.

4. Підготувати короткий звіт із поясненнями отриманих результатів.

Контрольні питання до розділу 9

5. Що таке візуалізація даних та яку роль вона відіграє в аналізі даних?

6. Які основні переваги використання графіків під час статистичного аналізу?

7. Які базові графічні можливості надає мова R?

8. У чому полягає відмінність між базовою графікою R та пакетом ggplot2?

9. Для чого використовується функція plot() і які типи графіків вона підтримує?

10. Яке призначення функцій points() та lines() і як їх комбінувати з plot()?

11. Які основні параметри керують виглядом графіків у базовій графіці R?

12. Які типи графіків найчастіше застосовуються для аналізу числових даних?

13. Для чого використовуються гістограми та які параметри функції hist()?

14. Що таке ящикова діаграма (boxplot) і яку статистичну інформацію вона відображає?

15. У яких випадках доцільно застосовувати стовпчикові діаграми?

16. Що означає поняття багатовимірної візуалізації даних?

17. Яку роль відіграють естетики (aes) у пакеті ggplot2?

18. Які геометричні об’єкти (geom) використовуються для побудови діаграм розсіювання та лінійних графіків?

19. У чому полягає принцип нашарування (layering) графіків у ggplot2?

20. Як реалізується групування та автоматичне створення легенд у ggplot2?

21. Що таке фасетування (facet_wrap(), facet_grid()) і для чого воно використовується?

22. Які способи збереження графіків існують у мові R?

23. Які формати файлів найчастіше використовуються для експорту графіків і чим вони відрізняються?

РОЗДІЛ 10. СУЧАСНІ ПІДХОДИ ДО КЛАСИФІКАЦІЇ ТА ОПРАЦЮВАННЯ ДАНИХ У RSTUDIO

10.1. Класифікація даних у середовищі RStudio

Класифікація даних є одним і етапів аналізу даних, що передбачає віднесення об’єктів до заздалегідь визначених класів або категорій на основі їхніх ознак. У середовищі RStudio класифікація використовується як для початкового впорядкування та опису даних, так і для побудови моделей машинного навчання, які дозволяють автоматизувати процес прийняття рішень.

10.1.1. Поняття класифікації даних та її значення в аналізі

Класифікація – це один із найпоширеніших методів керованого машинного навчання (Supervised Machine Learning). Її основне призначення полягає у прогнозуванні категоріальної (дискретної) цільової змінної на основі одного або кількох вхідних ознак (предикторів). У RStudio (та мові R загалом) класифікація даних виконується для того, щоб автоматично віднести об’єкти або спостереження до певних класів (категорій) на основі їхніх ознак.

На відміну від регресії, яка прогнозує неперервні числові значення (наприклад, ціну будинку), класифікація прогнозує мітки класів (наприклад, "спам" або "не спам", "доброякісна" або "злоякісна" пухлина, "сорт A", "сорт B" або "сорт C" квітки).

Класифікація даних – це процес побудови моделі, яка:

- навчається на вже розмічених даних (навчальна вибірка);
- використовує ознаки (features);
- прогнозує клас (label) для нових об’єктів.

Класифікація належить до задач контрольованого навчання (supervised learning). Класифікаційні моделі дозволяють автоматично приймати рішення без участі людини.

Приклад 1.

- зараховано / не зараховано вступника;

– ризик / без ризику;

– успішний / неуспішний студент.

RStudio дозволяє:

– проаналізувати, які фактори впливають на належність до класу;

– передбачити клас для нових даних.

Це особливо важливо в освіті, медицині, економіці. Підтримка управлінських рішень. Класифікація використовується як інструмент систем підтримки прийняття рішень (СППР).

Приклад 2.

– відбір абітурієнтів;

– сегментація студентів за рівнем підготовки;

– виявлення груп ризику.

Виявлення прихованих закономірностей. Класифікаційні алгоритми дозволяють:

– знайти неочевидні зв’язки між змінними;

– визначити ознаки, що впливають на клас.

Підготовка та перевірка гіпотез. Результати класифікації допомагають:

– формулювати гіпотези;

– перевіряти ефективність освітніх або управлінських рішень.

Регресія: Прогнозує число (наприклад, 15.7).

Класифікація: Прогнозує категорію (наприклад, "Так" або "Ні").

Процес побудови класифікаційної моделі в RStudio зазвичай включає:

6. Підготовка даних: очищення, перетворення та кодування категоріальних змінних, обробку пропусків, масштабування

7. Побудова моделі:

– Розділення на навчальний набір (Training Set) для тренування моделі та тестовий набір (Testing Set) для оцінки її продуктивності.

– Вибір відповідного алгоритму та його застосування до навчального набору.

– Використання навченої моделі для прогнозування класів на тестовому наборі.

– Використання метрик (наприклад, точність, матриця похибок) для оцінки якості моделі.

– Модель оцінює ймовірність належності об’єкта до певного класу.

Оцінювання якості класифікації:

– матриця помилок;

– точність;

– чутливість;

– специфічність.

Ознаки які відрізняють класифікацію від кластеризації.

Ознака	Класифікація	Кластеризація
Тип навчання	контрольоване	неконтрольоване
Класи	відомі заздалегідь	невідомі
Мета	прогноз класу	пошук груп

10.1.2. Класифікація даних за типами та структурами в R

Мова програмування R надає гнучкі засоби для роботи з різними типами та структурами даних, що є основою коректної класифікації. До базових типів даних у R належать числові (numeric), цілочисельні (integer), логічні (logical), символьні (character) та категоріальні (factor). Кожен із цих типів має свої особливості зберігання та обробки, які необхідно враховувати під час аналізу.

Структури даних у R включають вектори, матриці, списки та таблиці даних (data frame). Найпоширенішою структурою для задач класифікації є data frame, оскільки він дозволяє зберігати змінні різних типів у табличному вигляді. Коректне визначення типів змінних і вибір відповідної структури даних у RStudio забезпечують правильну роботу алгоритмів класифікації та підвищують точність отриманих результатів.

10.1.3. Структуровані, напівструктуровані та неструктуровані дані

За способом організації дані поділяють на структуровані, напівструктуровані та неструктуровані. Структуровані дані мають чітко визначену схему з фіксованими полями та типами, прикладом чого є таблиці баз даних або об'єкти data frame в R. Такі дані найзручніші для аналізу та класифікації, оскільки легко піддаються формалізованій обробці.

Напівструктуровані дані не мають жорсткої табличної структури, але містять елементи організації, наприклад у форматах JSON або XML. Для їх обробки в R використовуються спеціалізовані пакети, що дозволяють перетворювати такі дані у структурований вигляд. Неструктуровані дані, до яких належать текстові документи, зображення, аудіо- та відеофайли, потребують попереднього опрацювання та перетворення ознак перед застосуванням методів класифікації.

10.2. Класифікаційні алгоритми в мові R

Мова R містить екосистему пакетів, які реалізують широкий спектр класифікаційних алгоритмів.

Алгоритм	Призначення	Пакет R	Основна функція
Логістична регресія	Бінарна класифікація (прогнозування ймовірності належності до класу).	stats (вбудований)	glm()
Дерева рішень	Нелінійна класифікація, проста інтерпретація правил.	rpart, party	rpart(), ctree()
k-Найближчих сусідів (k-NN)	Непараметричний, класифікація на	class	knn()

	основі відстані до сусідів.		
Опорні вектори (SVM)	Ефективна класифікація, особливо для даних з високою розмірністю.	e1071	svm()
Наївний Баєс (Naive Bayes)	Ймовірнісний класифікатор, швидкий, добре працює з текстом.	e1071	naiveBayes()

1. Логістична регресія (Logistic Regression)

Хоча назва містить слово "регресія", це класифікаційний алгоритм. Він використовується для моделювання ймовірності того, що спостереження належить до певного класу.

Призначення: Бінарна класифікація (0 або 1, Так або Ні).

В R: використовується функція glm() (Generalized Linear Model) з параметром family = binomial.

Приклад використання:

Пакет для роботи з даними

library(tidyverse)

Побудова моделі

model_log <- glm(Is_Spam ~ Feature1 + Feature2,

data = my_data,

family = binomial)

2. Дерева рішень (Decision Trees)

Дерево рішень – це структура, схожа на блок-схему, де кожен внутрішній вузол представляє перевірку ознаки, кожна гілка – результат перевірки, а кожен кінцевий вузол (лист) – мітку класу.

Призначення: Створення простих для інтерпретації правил класифікації.

Переваги: Легко візуалізувати та пояснити.

Недоліки: Схильність до перенавчання (overfitting).

Функція в R: rpart() (пакет rpart) або ctree() (пакет party).

Приклад використання:

library(rpart)

Побудова дерева

tree_model <- rpart(Species ~ ., data = iris, method = "class")

Візуалізація (потрібен пакет rpart.plot)

rpart.plot(tree_model)

3. k-Найближчих сусідів (k-Nearest Neighbors, k-NN)

Це непараметричний алгоритм, який не вимагає попереднього припущення про розподіл даних. Класифікація нового об'єкта відбувається на основі "голосування" його k найближчих сусідів у просторі ознак.

Призначення: Проста та ефективна класифікація, особливо для невеликих наборів даних.

Параметр k – кількість сусідів, які беруть участь у голосуванні.

knn() – функція з пакету class.

Приклад використання:

library(class)

Класифікація тестових даних

knn_pred <- knn(train = train_data,

test = test_data,

cl = train_labels,

k = 5)

4. Метод опорних векторів (Support Vector Machines, SVM).

SVM знаходить оптимальну гіперплощину (лінію в 2D, площину в 3D), яка максимально розділяє класи.

Призначення: Ефективний для складних, нелінійно розділених даних.

Ядро (Kernel Trick) – дозволяє розділяти класи, які неможливо розділити лінійно, проектуючи дані у простір вищої розмірності.

svm() – функція з пакету e1071.

Приклад використання:

library(e1071)

Побудова моделі з радіальним ядром

svm_model <- svm(Species ~ ., data = iris, kernel = "radial")

Після тренування модель необхідно оцінити, щоб зрозуміти, наскільки добре вона прогнозує нові, невидимі дані.

Матриця похибок (Confusion Matrix) – це таблиця, яка підсумовує результати прогнозування моделі на тестовому наборі.

	Прогноз: Позитивний	**Прогноз: Негативний**
Факт: Позитивний	True Positive (TP)	False Negative (FN)
Факт: Негативний	False Positive (FP)	True Negative (TN)

TP (Істинно позитивний): Модель правильно передбачила позитивний клас.

TN (Істинно негативний): Модель правильно передбачила негативний клас.

FP (Хибно позитивний): Модель помилково передбачила позитивний клас (Помилка I роду).

FN (Хибно негативний): Модель помилково передбачила негативний клас (Помилка II роду).

На основі матриці похибок обчислюються метрики:

Метрика	**Формула**	**Призначення**
Точність (Accuracy)	$(TP + TN) / (TP + TN + FP + FN)$	Загальна частка правильних прогнозів.
Чутливість (Sensitivity/Recall)	$TP / (TP + FN)$	Частка правильно виявлених позитивних випадків.

Метрика	Формула	Призначення
Специфічність (Specificity)	$TN / (TN + FP)$	Частка правильно виявлених негативних випадків.
Точність (Precision)	$TP / (TP + FP)$	Частка істинно позитивних серед усіх, які були спрогнозовані як позитивні.

Функція в R: Пакет caret (Classification And REgression Training) має функцію confusionMatrix(), яка автоматично обчислює всі ці метрики.

library(caret)

Створення матриці похибок

confusionMatrix(data = predicted_classes, reference = actual_classes)

10.3. Data frame як основна форма організації табличних даних

Таблиці (фрейми даних, data frames) – тип 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. Фрейм даних має вигляд двовимірної таблиці (рядки × стовпці), де:

– кожен рядок відповідає окремому спостереженню (об’єкту, експерименту, запису);

– кожен стовпець – це змінна (ознака);

– у межах одного стовпця всі значення мають один тип даних;

– різні стовпці можуть мати різні типи (numeric, character, logical, factor).

На відміну від матриць, data frame не обмежується лише числовими даними, що робить його основною структурою для реальних прикладних задач.

Фрейм даних створюється за допомогою функції data.frame():

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

Наведемо приклади типових функцій для роботи із data frame:

створення окремих векторів:

id <- 1:5 # числовий вектор

score <- c(78, 85, 90, 67, 88) # результати вимірювань

group <- c("A", "A", "B", "B", "C") # групи

students <- data.frame(id, score, group) # формування таблиці даних

students # перегляд таблиці

Для створення навчальних або експериментальних наборів даних часто використовують генерацію випадкових чисел.

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

rnorm(n, mean, sd)

Приклад:

set.seed(123) # фіксація генератора випадкових чисел

values <- rnorm(10, mean = 50, sd = 10)

values

Основні функції для первинного аналізу таблиць: Перегляд та дослідження структури Data Frame

– head(students) # перші 6 рядків
– tail(students) # останні рядки
– str(students) # структура та типи даних
– summary(students) # описова статистика
– nrow(students) # кількість рядків
– ncol(students) # кількість стовпців
– dim(students) # розмір таблиці

Пакет dplyr забезпечує зручний, декларативний підхід до обробки табличних даних.

library(dplyr)

Поширені операції опрацювання таблиць:

– select() – вибір стовпців;

– filter() – фільтрація рядків;

– arrange() – сортування;

– mutate() – створення нових змінних;

– summarize() – агрегування;

– group_by() – групування даних.

Вибір стовпців: select()

select(students,id,score)# вибір конкретних стовпців

select(students, -group)# виключення стовпця

select(cran, -(X:size))#Вибір усіх стовпців, крім заданих.

Фільтрація рядків: filter()

filter(students,score> 80) # спостереження з балом більше 80

filter(students, group == "B")# об'єкти з групи B

filter(cran, package == "swirl")#вибрати об'єкти, в яких ознака package = "swirl"

filter(cran, r_version == "3.1.1", country == "US")#вибрати об'єкти, в яких r_version = "3.1.1" та country = "US"

filter(cran, country == "US" | country == "IN")#вибрати об'єкти, в яких country ="US" або country = "IN"

filter(cran,size > 100500,r_os=="linux-gnu")#вибрати об'єкти, в яких , size>100500 та r_os = "linux-gnu"

filter(cran, !is.na(r_version)) #вибрати об'єкти, в яких значення ознаки r_version заповнено

Сортування даних: arrange()

arrange(students,score) # сортування за зростанням

arrange(students,desc(score))# сортування за спаданням

arrange(cran2,desc(ip_id))

#відсортувати об'єкти за зростанням значенням ознаки package, а потім за зростанням ip_id

arrange(cran2,package,ip_id)

arrange(cran2,country,desc(r_version),ip_id)

Створення нових змінних: mutate()

```
students2 <- mutate(students,
  score_norm = score / max(score),
passed = score >= 75) # нормалізація значень
students2
```

Групування та агрегування є базовим для аналітики, статистичних досліджень і задач кібернетики. Функція group_by() використовується разом із summarize().

```
students %>%
  group_by(group) %>%
  summarize(
    avg_score = mean(score),
    min_score = min(score),
    max_score = max(score)
  )
```

Пакет dplyr для роботи з табличними даними є стандартним інструментом для маніпуляцій з таблицями даних.

```
library(dplyr)
```

Створення нових змінних: mutate()

```
mutate(cran3, size_mb = size / 2^20, size_gb = size_mb / 2^10)
mutate(cran3, correct_size = size + 1000)
mutate(cran3, size_mb = size / 2^20, size_gb = size_mb/2^10)
> mutate(cran3, correct_size = size + 1000)
```

Узагальнення даних: summarize()

```
summarize(cran, avg_bytes = mean(size))
```

Групування даних: group_by()

```
by_package <- group_by(cran, package)
```

10.4. Основи опрацювання великих даних у середовищі R

У мові R технологія Big Data – це інструмент аналітика, який дозволяє працювати з великими масивами даних завдяки оптимізованим пакетам і інтеграції з розподіленими системами. Базовий R працює в оперативній пам'яті, тому для великих обсягів даних застосовують: оптимізовані структури даних; обробку частинами; інтеграцію з розподіленими системами (Apache Spark, Hadoop).

Підхід	Коли використовувати
base R	Невеликі дані
data.table	Дані до кількох ГБ
chunking	Дані не вміщуються в RAM
sparklyr	Розподілені дані
arrow/parquet	Швидкий I/O

Ефективним засобом опрацювання великих даних у мові R є бібліотека sparklyr. Бібліотека sparklyr – дозволяє працювати з Apache Spark безпосередньо з RStudio, використовуючи знайомий синтаксис R та dplyr.

Вона дозволяє:обробляти великі обсяги даних (Big Data); працювати з даними, що не поміщаються в оперативну пам'ять; виконувати розподілені обчислення; застосовувати машинне навчання Spark MLlib з R.

Дані при цьому фізично зберігаються у Spark, а не в R. Це принципово відрізняє sparklyr від звичайного аналізу в R.

Перед роботою в RStudio на комп'ютері має бути Java

Встановлення пакета:

install.packages("sparklyr")

library(sparklyr)

library(dplyr)

Встановлення Spark

spark_install(version = "3.5.0") (версію можна змінювати)

Підключення до Spark. Локальний режим (для навчання)

```
sc <- spark_connect(
  master = "local",
  app_name = "Spark_RStudio_Demo")
```

Spark працює на вашому комп’ютері що ідеально для лабораторних робіт і навчання.

Передача даних у Spark

```
df_spark <- copy_to(sc, df, overwrite = TRUE)
```

Обчислення

```
df_spark %>%
  summarise(
    avg_score = mean(score),
    max_score = max(score) )
```

Завершення сесії:

```
spark_disconnect(sc)
```

Особливості використання даних у sparklyr

1. У Spark використовується Spark DataFrame, а не data.frame.

```
class(sc)
tbl(sc, "mtcars")
```

2. Передача даних з R у Spark

```
spark_df <- copy_to(sc, mtcars, "cars", overwrite = TRUE)
```

Дані копіюються у пам’ять Spark.

Робота з даними (dplyr-синтаксис)

Для роботи в sparklyr необхідно завантажити бібліотеку dplyr.

```
library(dplyr)
spark_df %>%
  filter(mpg > 20) %>%
  group_by(cyl) %>%
  summarise(avg_mpg = mean(mpg))
```

Обчислення виконуються у Spark, а не в R.

Ліниві обчислення (Lazy Evaluation)

```
result <- spark_df %>%
  filter(mpg > 20)
```

Дані ще не обчислені. Виконання починається лише при:

```
collect(result)
```

collect() – повертає дані в R (обережно з великими обсягами!).

Робота з файлами Big Data. Читання CSV

```
data <- spark_read_csv(sc,
  name = "bigdata",
  path = "data.csv",
  header = TRUE,
  infer_schema = TRUE)
```

Parquet (рекомендовано)

```
spark_read_parquet(sc, "data_parq", "data.parquet")
```

Машинне навчання в sparklyr

sparklyr надає доступ до Spark MLlib.

1. Лінійна регресія

```
model <- ml_linear_regression(
  spark_df,
  mpg ~ wt + cyl
)
summary(model)
```

2. Класифікація

```
model <- ml_logistic_regression(
  spark_df,
  am ~ mpg + wt
)
```

3. Кластеризація

```
model <- ml_kmeans(
  spark_df,
  k = 3,
```

features = c("mpg", "wt"))

4. Пайплайн (ML Pipelines) – це спосіб організувати обробку даних як послідовність логічних кроків, де результат одного кроку автоматично передається в наступний. У R пайплайни найчастіше реалізуються оператором %>% з пакета magrittr (входить до tidyverse). Це промисловий підхід до ML.

```
pipeline <- ml_pipeline(sc) %>%
  ft_vector_assembler(input_cols = c("mpg", "wt"),
    output_col = "features"
  ) %>%
  ml_kmeans(k = 3)
model <- ml_fit(pipeline, spark_df)
```

5. Візуалізація. Spark не візуалізує дані напряму.

Потрібно:

```
small_df <- collect(spark_df)
plot(small_df$wt, small_df$mpg)
```

Коли доцільно використовувати sparklyr.

Критерій	**R**	**sparklyr**
Обсяг даних	до RAM	Big Data
Швидкість	висока	дуже висока
Розподілення	X	✓
ML	caret	Spark ML
Навчання	просте	складніше

Порівняння: R vs sparklyr

- великі CSV / Parquet
- лог-файли
- освітні дані (НМТ, LMS, журнали)
- фінансові транзакції

– ML на великих вибірках

sparklyr – це міст між R і Apache Spark, який дозволяє виконувати розподілений аналіз даних і машинне навчання у знайомому середовищі RStudio.

Лабораторні роботи до розділу 10

Лабораторна робота 1. Класифікація за допомогою k-NN (Набір даних Iris)

Мета роботи: ознайомитися з принципами класифікації в задачах машинного навчання на прикладі алгоритму k-найближчих сусідів (k-NN); навчитися підготовці та нормалізації даних, побудові й налаштуванню класифікаційної моделі на основі набору даних Iris у середовищі RStudio, а також оцінюванню якості класифікації за допомогою основних метрик та матриці похибок.

Теоретичні відомості

k-найближчих сусідів (k-NN) – це простий і наочний алгоритм машинного навчання, який використовується для класифікації та регресії. Він визначає клас або значення нового об’єкта на основі k найближчих до нього точок у просторі ознак, зазвичай використовуючи відстань (наприклад, евклідову). Алгоритм не потребує етапу навчання, але є чутливим до вибору k та масштабування даних.

Набір даних iris є класичним прикладом для демонстрації класифікації, оскільки він містить чотири числові ознаки та цільову змінну Species (три класи: setosa, versicolor, virginica).

Хід виконання роботи

Завдання 1. Завантажити необхідні пакети:

```
#install.packages(c("class", "caret"))
library(class)
```

library(caret)

Завдання 2. Створити об’єкт даних використавши вбудований набір даних iris.

data(iris)

Завдання 3. Врахувати, що k-NN чутливий до масштабу ознак, тому їх потрібно нормалізувати. Виконати підготовку до процесу нормалізації даних.

```
normalize <- function(x)
{
  return ((x - min(x)) / (max(x) - min(x)))
}
```

Провести нормалізацію числових стовпців (1-4)

iris_norm <- as.data.frame(lapply(iris[, 1:4], normalize))

Завдання 4. Розділити дані на навчальний та тестовий набори. Встановити початкове значення для відтворюваності вектора.

set.seed(123)

Створення індексів для 70% навчального набору:

index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)

Створити навчальний набір (70% даних):

iris_train <- iris_norm[index,]

Створити тестовий набір (30% даних)

iris_test <- iris_norm[-index,]

Мітки класів для навчального та тестового наборів

iris_train_labels <- iris[index, 5]

iris_test_labels <- iris[-index, 5]

Завдання 5. Провести тренування моделі k-NN та спрогнозувати результати. Для обчислень використовуємо k = 5 (корінь з кількості спостережень, або непарне число).

```
iris_pred <- knn(train = iris_train,
                 test = iris_test,
                 cl = iris_train_labels,
```

k = 5)

Вивести результати прогнозування та оцінити їх.

print(head(iris_pred))

Завдання 6. Провести оцінку продуктивності моделі за допомогою Матриці похибок. Порівняння прогнозованих міток (iris_pred) з фактичними (iris_test_labels)

confusionMatrix(data = iris_pred, reference = iris_test_labels)

Очікуваний результат (приклад):

```
> confusionMatrix(data = iris_pred, reference = iris_test_labels)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         14         2
  virginica       0          1        13

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.8173, 0.986)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9
```

Пояснення результату:

Accuracy (Точність): Показує загальну частку правильно класифікованих об'єктів (у цьому прикладі 93.33%).

Матриця: Показує, що 15 квіток setosa були правильно класифіковані, 14 versicolor правильно, але 1 versicolor був помилково класифікований як virginica, і 2 virginica був помилково класифікований як versicolor.

Класифікація в RStudio виконується для того, щоб:

- автоматизувати аналіз і прийняття рішень;
- прогнозувати категоріальні результати;
- знаходити закономірності в даних;
- підтримувати управлінські та освітні рішення.

Лабораторна робота 2. Класифікація пасажирів Titanic методами SVM та Дерев рішень у мові R

Мета роботи: набути практичних навичок застосування класифікаційних алгоритмів машинного навчання (Методу опорних векторів (SVM) та Дерев рішень) для аналізу реального набору даних Titanic у середовищі RStudio, а також порівняти ефективність цих алгоритмів.

Теоретичні відомості

Метод опорних векторів (SVM) – алгоритм класифікації, який будує оптимальну гіперплощину, що максимізує відстань між класами. SVM добре працює з багатовимірними даними та може використовувати різні ядра (лінійне, радіальне).

Дерева рішень – інтерпретований алгоритм машинного навчання, який послідовно розбиває дані за умовами ознак, формуючи ієрархічну структуру рішень. Перевагою є наочність і простота пояснення результатів.

Вхідні дані: Використовується класичний набір даних Titanic, який містить інформацію про пасажирів корабля.

– Цільова змінна: Survived (0 – не вижив, 1 – вижив)

– Предиктори: Pclass, Sex, Age, Fare

Хід виконання роботи

Завдання 1. Завантаження та огляд даних

1. Встановіть і підключіть необхідні пакети:

```
#install.packages(c("titanic", "tidyverse", "caret", "e1071", "rpart", "rpart.plot"))
library(titanic)
library(tidyverse)
library(caret)
library(e1071)
library(rpart)
```

library(rpart.plot)

2. Завантажте навчальний набір Titanic:

data <- titanic_train

3. Перегляньте структуру даних:

str(data)

summary(data)

Завдання 2. Підготовка даних

1. Виберіть потрібні змінні:

data_ml <- data %>%

select(Survived, Pclass, Sex, Age, Fare)

2. Видаліть рядки з пропущеними значеннями:

data_ml <- na.omit(data_ml)

3. Перетворіть змінні до потрібних типів:

data_ml$Survived <- factor(data_ml$Survived)

data_ml$Sex <- factor(data_ml$Sex)

4. Масштабуйте числові ознаки (для SVM):

preproc <- preProcess(data_ml[, c("Age", "Fare")], method = c("center", "scale"))

data_ml[, c("Age", "Fare")] <- predict(preproc, data_ml[, c("Age", "Fare")])

Завдання 3. Розбиття на навчальну та тестову вибірки

set.seed(123)

train_index <- createDataPartition(data_ml$Survived, p = 0.7, list = FALSE)

train_data <- data_ml[train_index,]

test_data <- data_ml[-train_index,]

Частина І. Класифікація методом опорних векторів (SVM)

Завдання 4. Побудова SVM-моделі

svm_model <- svm(

Survived ~ .,

data = train_data,

kernel = "radial"

)

Завдання 5. Прогнозування та оцінювання

svm_pred <- predict(svm_model, test_data)

confusionMatrix(svm_pred, test_data$Survived)

Зробіть висновок щодо якості SVM-класифікації.

Частина ІІ. Класифікація за допомогою Дерев рішень

Завдання 6. Побудова дерева рішень

dtree_model <- rpart(

Survived ~ .,

data = train_data,

method = "class"

)

Завдання 7. Візуалізація дерева

rpart.plot(dtree_model)

Завдання 8. Прогнозування та оцінювання

dtree_pred <- predict(dtree_model, test_data, type = "class")

confusionMatrix(dtree_pred, test_data$Survived)

Завдання 9.

1. Додайте ознаку FamilySize.

2. Змініть ядро SVM (linear).

3. Обмежте глибину дерева рішень та порівняйте результати.

4. Побудуйте ROC-криву для обох моделей.

Аналіз результатів та висновки

Проаналізувати отримані результати та підготувати письмовий звіт, який включає:

1. Матриці похибок для обох моделей.

2. Порівняльну таблицю основних метрик (Accuracy, Sensitivity, Specificity) для SVM та Дерева рішень.

Метрика	Дерево рішень	SVM
Accuracy		
Sensitivity		
Specificity		
P-Value		
Kappa		

Аналізуючи дані врахуйте:

- Accuracy – вказує яку частину спостережень модель правильно класифікує, чи є це хорошим результатом для задачі з реальними соціальними даними.
- P-Value [Acc > NIR] – вказує чи є моделі суттєво ефективніша, ніж у наївна модель, яка завжди прогнозує найпоширеніший клас.
- Kappa – вказує на рівень узгодженості між прогнозами моделі та реальними значеннями з урахуванням випадкових збігів.

Аналіз чутливості та специфічності:

Sensitivity – вказує як модель добре розпізнає клас 0 (негативний клас): більшість пасажирів, які не вижили, визначені правильно.

Specificity – оцінює здатність моделі коректно ідентифікувати клас 1 (вижили).

Balanced Accuracy = показує, що з урахуванням дисбалансу класів модель має стабільну, або ідеальну якість.

Висновки:

1. Яка модель показала кращу загальну точність (`Accuracy`)?
2. Яка модель краще прогнозує клас "High" (`Sensitivity`)?
3. Яка модель краще прогнозує клас "Low" (`Specificity`)?
4. Які переваги та недоліки кожного алгоритму ви помітили в процесі роботи?
5. Яку модель ви б обрали для цієї задачі і чому?
6. Яка модель краще працює з нелінійними залежностями?

Лабораторна робота 3. Фрейми даних (Data Frames) у R

Мета роботи: навчитися створювати, переглядати, модифікувати та аналізувати фрейми даних у R.

Теоретичні відомості

Фрейм даних (Data Frame) – це структура даних у R, що дозволяє зберігати табличні дані:

Рядки → спостереження (observations)

Стовпці → змінні (variables)

Кожен стовпець може мати свій тип: числовий, текстовий, фактори, логічний. Основні властивості:

Функція	**Призначення**
data.frame()	створення фрейму
nrow(), ncol()	кількість рядків та стовпців
dim()	розміри фрейму
names() або colnames()	назви стовпців
str()	структура фрейму
head(), tail()	перші/останні рядки
$	доступ до стовпця
[]	індексація (рядки, стовпці)

Хід виконання роботи

Завдання 1. Створіть фрейм даних про студентів за поданим зразком.

```
students <- data.frame(
  ID = 1:5,
  Name = c("Andriy", "Olena", "Ivan", "Kateryna", "Oleg"),
  Age = c(20, 19, 21, 20, 22),
  Major = c("CS", "CS", "CS", "CS", "CS"),
  Score = c(95, 88, 76, 90, 85)
)
students
```

- Виведіть на екран перший стовпець даних.

students$Name

- Виведіть перший рядок даних

students[1,]

- Виведіть перші 3 рядки

students[1:3,]

- Вивести рядки 2-4, стовпці Name та Score

students[2:4, c("Name", "Score")]

Самостійно вивести всі імена студентів, чий Score > 85.

Завдання 2. Виконайте команду додавання стовпця.

students$Passed <- students$Score >= 80

students

Додати стовпець Grade за правилами: Score >= 90 → "A", 80–89 → "B", <80 → "C". Поясніть які зміни відбулись в таблиці.

- Виконайте команду видалення стовпця

students$Major <- NULL

students

Завдання 3. Виконайте наступні операції над даними:

- Визначте кількість студентів

nrow(students)

- Визначте середній бал

mean(students$Score)

- Визначте мінімальний та максимальний бал

min(students$Score)

max(students$Score)

Вивести імена студентів з максимальним балом.

Завдання 4. Ознайомтесь із командами фільтрації та сортування. Перший крок library(dplyr).

Виконайте команду фільтрації та поясніть умову вимоги сформованої у поданій команді.

high_scores <- students %>% filter(Score > 85)

high_scores

Виконайте сортування за Score та поясніть умову вимоги сформованої у поданій команді.

students_sorted <- students %>% arrange(desc(Score))

students_sorted

Відсортувати студентів за віком та вивести тільки стовпці Name і Age.

Завдання 5. Ознайомтесь із командами збереження та завантаження фрейму. Вкажіть шлях завантаження.

- Виконайте збереження у файлі CSV

write.csv(students, "students.csv", row.names = FALSE)

- Виконайте читання з файлу CSV

students_new <- read.csv("students.csv")

students_new

Зберегти свій модифікований фрейм у файл.

Завдання 6. Підсумкове завдання. Створити фрейм даних 10–12 студентів. Додати стовпець Passed та Grade. Відфільтрувати студентів, які отримали A. Зберегти фрейм у CSV і знову прочитати його у нову змінну.

Підготувати короткий опис статистики (середній, мінімальний, максимальний бал)

Додаткові підказки.

– Використовуйте str(students) щоб перевірити структуру.

– Функції head() та tail() допоможуть швидко вивести дані.

– Для великих фреймів використовуємо dplyr для швидких фільтрацій і агрегацій.

Лабораторна робота 4. Таблиці даних (Data Frames) в середовищі RStudio

Мета роботи: набути практичних навичок роботи з табличними даними

(data frames) у середовищі RStudio, зокрема: створення та перегляду фреймів даних; генерації даних та формування таблиць; вибірки, фільтрації, сортування даних; створення нових змінних; групування та узагальнення даних за допомогою пакета dplyr.

Теоретичні відомості

Фрейм даних (data frame) – це двовимірна таблиця, у якій кожен стовпець може містити дані різного типу (числові, логічні, символьні, фактори), але всі елементи одного стовпця мають однаковий тип. Фрейми даних є основною структурою для зберігання та аналізу експериментальних, статистичних та аналітичних даних в R.

Фрейм даних створюється за допомогою функції:

data.frame(data1, data2, ...)

Приклад:

```
x <- 1:5
y <- c("A", "B", "C", "D", "E")
z <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
frm <- data.frame(x, y, z)
frm
```

Функція rnorm() використовується для генерації випадкових чисел з нормальним розподілом:

rnorm(n, mean, sd)

де:

n – кількість значень;

mean – математичне очікування;

sd – середньоквадратичне відхилення.

Пакет dplyr є стандартним інструментом для маніпуляцій з таблицями даних. Підключення пакета:

```
library(dplyr)
```

Вибір стовпців: select()

select(cran, -(X:size))

Вибір усіх стовпців, крім заданих. Фільтрація рядків: filter()

filter(cran, package == "swirl")

filter(cran, r_version == "3.1.1", country == "UA")

filter(cran, country == "UA" | country == "IN")

filter(cran, size > 100500, r_os == "linux-gnu")

filter(cran, !is.na(r_version))

Сортування: arrange()

arrange(cran2, desc(ip_id))

arrange(cran2, package, ip_id)

arrange(cran2, country, desc(r_version), ip_id)

Створення нових змінних: mutate()

mutate(cran3, size_mb = size / 2^2, size_gb = size_mb / 2^3)

mutate(cran3, correct_size = size + 1000)

Узагальнення даних: summarize()

summarize(cran, avg_bytes = mean(size))

Групування даних: group_by()

by_package <- group_by(cran, package)

Хід виконання роботи

Завдання 1. Створення таблиці даних. Згенерувати вектор value із10 випадкових значень з нормальним розподілом (mean = 50, sd = 10).

Створити вектор group категорій з трьома рівнями ("A", "B", "C").

Сформувати фрейм даних із трьома стовпцями: id, value, group.

Завдання 2. Аналіз структури таблиці. Вивести перші 6 рядків таблиці. Переглянути структуру та типи даних. Визначити розмір таблиці.

Завдання 3. Фільтрація та вибірка

Відібрати спостереження з value > 55.

Відібрати лише об’єкти групи "B".

Вибрати лише два стовпці на власний вибір.

Завдання 4. Відсортувати таблицю за зростанням value. Відсортувати за спаданням value всередині кожної групи.

Завдання 5. Створити новий стовпець value_norm, поділивши value на максимальне значення. Додати стовпець з логічною умовою (value > mean(value)).

Завдання 6. Згрупувати дані за group. Для кожної групи обчислити середнє, мінімальне та максимальне значення value.

Лабораторна робота 5. Об'єднання, нормалізація та аналіз фреймів даних у мові R

Мета роботи: навчитися працювати з кількома фреймами даних у R: створювати таблиці, об'єднувати їх, нормалізувати числові дані, виконувати фільтрацію за категоріальними ознаками та будувати графіки.

Теоретичні відомості

У практичному аналізі даних інформація часто зберігається у кількох таблицях, які потрібно:

- об'єднати за спільним ідентифікатором merge(),
- очистити та підготувати до аналізу,
- нормалізувати числові змінні (привести до спільного масштабу),
- відфільтрувати за певними ознаками,
- візуалізувати результати.

Нормалізація даних – це перетворення числових значень до стандартного масштабу, наприклад:

- від 0 до 1 (min–max),
- або зі середнім 0 і стандартним відхиленням 1 (scale()).

У ході лабораторної роботи буде виконано роботу з кількома фреймами даних, їх об'єднання, нормалізацію, фільтрацію за категоріальними ознаками та базову візуалізацію результатів у середовищі R

Хід виконання роботи

Завдання 1. Створення двох фреймів даних

Створіть таблицю з особистими даними студентів:

```
students <- data.frame(
  ID = 1:8,
  Name = c("Andriy", "Olena", "Ivan", "Kateryna", "Oleg", "Maria", "Taras", "Iryna"),
  Gender = c("M", "F", "M", "F", "M", "F", "M", "F"),
  Age = c(20, 19, 21, 20, 22, 18, 23, 21)
)
```

Створіть окрему таблицю з результатами навчання:

```
scores <- data.frame(
  ID = 1:8,
  Math = c(88, 92, 75, 90, 85, 95, 70, 89),
  Programming = c(90, 94, 78, 92, 88, 96, 74, 91)
)
```

Перегляньте структуру обох таблиць str().

Завдання 2. Об’єднання таблиць

Об’єднайте таблиці за ідентифікатором студента:

```
library(dplyr)
students_full <- left_join(students, scores, by = "ID")
students_full
```

Поясніть, чому саме поле ID використовується для об’єднання.

Завдання 3. Нормалізація числових даних

Виконайте нормалізацію оцінок з математики та програмування:

```
students_full$Math_norm <- scale(students_full$Math)
students_full$Programming_norm <- scale(students_full$Programming)
```

Перегляньте оновлений фрейм та поясніть, що означають отримані значення.

Завдання 4. Фільтрація даних за статтю

1. Відіберіть лише студентів жіночої статі:

```
female_students <- students_full %>% filter(Gender == "F")
female_students
```

2. Відіберіть лише студентів чоловічої статі:

```
male_students <- students_full %>% filter(Gender == "M")
male_students
```

Порівняйте середні нормалізовані результати для кожної групи.

Завдання 5. Побудова графіка за статтю

Обчисліть середній бал з програмування для кожної статі:

```
avg_by_gender <- students_full %>%
  group_by(Gender) %>%
  summarise(AvgProgramming = mean(Programming))
```

Побудуйте стовпчикову діаграму:

```
barplot(
  avg_by_gender$AvgProgramming,
  names.arg = avg_by_gender$Gender,
  main = "Середній бал з програмування за статтю",
  xlab = "Стать",
  ylab = "Середній бал"
)
```

Зробіть висновок на основі побудованого графіка.

Лабораторна робота 6. Концепція розподілених обчислень та Apache Spark

Мета роботи: ознайомитися з концепцією розподілених обчислень та Apache Spark; навчитися підключатися до Spark із середовища RStudio за допомогою бібліотеки sparklyr; набути практичних навичок роботи зі Spark DataFrame; виконувати фільтрацію, групування та агрегування великих наборів даних; ознайомитися з базовими

можливостями машинного навчання в Spark; навчитися коректно повертати результати обчислень у середовище R.

Теоретичні відомості

Apache Spark – це платформа для швидкої обробки великих даних у пам’яті з використанням розподілених обчислень. Бібліотека sparklyr забезпечує інтеграцію Spark з мовою R та дозволяє використовувати звичний синтаксис dplyr для роботи з великими даними.

Особливістю роботи зі Spark є ліниве виконання обчислень (lazy evaluation): більшість операцій не виконуються одразу, а лише під час запиту результату.

Обладнання та програмне забезпечення:

– встановлений пакет sparklyr;

– Apache Spark (локальний режим).

Завдання 1. Виконайте стандартну процедуру підключення бібліотек.

```
library(sparklyr)
library(dplyr)
```

Завдання 2. Процес встановлення (інсталяції) spark проводиться один раз. Виконайте запуск запуск Spark.

```
#spark_install(version = "3.5.0")
sc <- spark_connect(master = "local")
```

Перевірити активне з’єднання зі Spark.

```
sc
```

Завдання 3. Передавання даних у Spark. При роботі із великими даними необхідно провести копіювання даних у середовище Spark.

```
cars_tbl <- copy_to(sc, mtcars, "cars", overwrite = TRUE)
```

Перевірити структуру таблиці:

```
cars_tbl
```

Завдання 4. Виконуємо аналіз даних у Spark. Типовими є фільтрація та вибірка. Відібрати автомобілі з витратою пального більше 20 mpg.

```
filtered_tbl <- cars_tbl %>%
```

filter(mpg > 20)

Завдання 5. Відпрацюйте групування та агрегування даних. Обчислити середню витрату пального для кожної кількості циліндрів.

summary_tbl <- cars_tbl %>%

group_by(cyl) %>%

summarise(avg_mpg = mean(mpg))

Звернути увагу, що обчислення ще не виконані фізично.

Завдання 6. Для отримання результатів у R необхідно провести команду: виконати обчислення та повернути дані.

result_df <- collect(summary_tbl)

result_df

Пояснити, чому функцію collect() слід застосовувати обережно.

Завдання 7. До базових методик машинного навчання відносять побудову лінійної регресії. Побудувати у Spark модель залежності витрати пального від ваги та кількості циліндрів.

model <- ml_linear_regression(

cars_tbl,

mpg ~ wt + cyl

)

Ознайомитися з результатами моделі:

summary(model)

Завдання 8. Spark не підтримує візуалізацію безпосередньо, тому дані необхідно спочатку повернути у R. Побудова графіка:

cars_r <- collect(cars_tbl)

plot(cars_r$wt, cars_r$mpg,

main = "Залежність витрати пального від ваги",

xlab = "Вага",

ylab = "MPG")

Завдання 9. Завершення роботи зі Spark

spark_disconnect(sc)

За результатами роботи підготуйте звіт.

У звіті необхідно:

- пояснити принцип роботи sparklyr;
- описати виконані операції фільтрації та групування;
- проаналізувати результати регресійної моделі;
- зробити висновки щодо переваг Spark для обробки великих даних.

Лабораторна робота 7. Аналіз реальних освітніх даних із використанням Apache Spark та бібліотеки sparklyr

Мета роботи: сформувати розуміння особливостей аналізу великих освітніх даних; навчити працювати з реальними наборами освітніх даних у середовищі RStudio + Apache Spark; набути практичних навичок використання бібліотеки sparklyr; виконувати групування, агрегування та фільтрацію освітніх показників; ознайомитися з базовими аналітичними та прогностичними моделями в освіті; навчитися інтерпретувати результати аналізу освітніх даних.

Теоретичні відомості

Освітні дані (Educational Data Mining, Learning Analytics) включають інформацію про:

- результати навчання студентів;
- оцінки та рейтинги;
- відвідуваність;
- демографічні характеристики;
- успішність за дисциплінами.

Такі дані часто є багатовимірними та об’ємними, що робить доцільним використання Apache Spark для їх обробки. Бібліотека sparklyr дозволяє виконувати розподілений аналіз безпосередньо в R.

Структура даних students_performance.csv

Змінна	Опис	Тип
student_id	Унікальний ідентифікатор студента	integer
gender	Стать студента	factor
faculty	Факультет	factor
course	Курс навчання	integer
subject	Навчальна дисципліна	factor
grade	Підсумкова оцінка (0–100)	numeric
passed	Чи складено дисципліну	logical
attendance	Відвідуваність (%)	numeric

Дані не містять персональної інформації.

Підготовка середовища.

Завдання 1. Підключіть необхідні бібліотеки.

```
library(sparklyr)
library(dplyr)
```

Завдання 2. Виконайте підключення до Spark.

```
sc <- spark_connect(master = "local")
```

Завдання 3. Завантажте освітні дані у Spark зчитавши їх із CSV-файлу.

```
students_tbl <- spark_read_csv( sc,
  name = "students",
  path = "students_performance.csv",
  header = TRUE,
  infer_schema = TRUE
)
```

Ознайомтися зі структурою таблиці:

```
students_tbl
```

Первинний аналіз освітніх даних Загальна статистика

Завдання 4. Обчисліть вказані параметри вибірки: середню оцінку, мінімальну та максимальну оцінки, середню відвідуваність.

```
students_tbl %>%
  summarise(
    avg_grade = mean(grade),
    min_grade = min(grade),
    max_grade = max(grade),
    avg_attendance = mean(attendance)
  )
```

Завдання 5. Проведіть аналіз успішності за групами та факультетами.

```
faculty_perf <- students_tbl %>%
  group_by(faculty) %>%
  summarise(
    avg_grade = mean(grade),
    pass_rate = mean(passed)
  )
```

Завдання 6. Проведіть аналіз даних за статтю та курсом.

```
gender_course_perf <- students_tbl %>%
  group_by(gender, course) %>%
  summarise(avg_grade = mean(grade))
```

Завдання 7. Проведіть вибірку з метою виявлення проблемних студентів групи ризику. Відібрати студентів, у яких:

оцінка < 60;

відвідуваність < 70%.

```
risk_students <- students_tbl %>%
  filter(grade < 60 & attendance < 70)
```

Завдання 8. Обчислення проводились на Spark. Для повернення результатів у R. необхідно використати collect().

Існує небезпека застосування collect() для великих наборів даних. Чому?

```
risk_df <- collect(risk_students)
head(risk_df)
```

Прогнозування успішності. Логістична регресія.

Завдання 9. Побудувати модель прогнозування складання дисципліни (passed).

```
model <- ml_logistic_regression(
  students_tbl,
  passed ~ attendance + grade
)
```

Переглянути результати моделі:

```
summary(model)
```

Завдання 10. Візуалізація результатів Графічний аналіз

```
students_r <- collect(students_tbl)
boxplot(grade ~ faculty, data = students_r,
   main = "Розподіл оцінок за факультетами",
   ylab = "Оцінка")
```

За результатами проведеного аналізу даних, підготуйте звіт.

Необхідно:

- описати структуру освітніх даних;
- пояснити результати агрегування;
- інтерпретувати модель прогнозування;
- зробити висновки щодо якості навчального процесу.

Лабораторна робота 8. Опрацювання великих даних у RStudio (на прикладі результатів НМТ 2019, 2025 років)

Мета роботи:

- продемонструвати можливості використання Apache Spark та бібліотеки sparklyr для аналізу реальних освітніх даних;
- виконати порівняльний аналіз рівнів досягнутих результатів з математики за матеріалами ЗНО/НМТ 2019, 2025 років;
- використовуючи результати отриманих обчислень візуалізувати середні показники за регіонами;
- зробити аналітичні висновки.

Вхідні дані: За вхідні дані використати відкриті ресурси порталу Українського центру оцінювання якості освіти. Використайте підсумкові CSV файли (Odata2019File.csv та Odata2025File.csv) із https://testportal.gov.ua/zvity-dani-7/.

Первинний огляд даних проводимо з метою перевірка структури, пошук аномальних значень, оцінка масштабів набору даних. (glimpse(nmt), summary(nmt)). Частину полів на час виконання завдання можна видалити. Приблизна структура набору даних НМТ після видалення:

- регіон;
- предмет;
- тестовий бал;
- рейтинговий бал;
- тип закладу;
- рік складання.

Використавши подану нижче програму, ми отримали фрейм даних до якого включили (RegName mean_ukr mean_math mean_hist) та обчислили середні значень по регіонах за 2024 рік. За результатами побудували лінійний графік.

Хід виконання роботи

Завдання 1. Підготувати Apache Spark та бібліотеки sparklyr для роботи.

Завдання 2. Використовуючи підготовлене середовище завантажте дані із Odata2019File.csv та Odata2025File.csv на Spark.

Завдання 3. Виконайте обчис-лення середніх значень по регіонах для чотирьох навчальних дисциплін.

Завдання 4. Порівняйте отримані значення, оцініть ефективність роботи Spark.

Оформіть висновок:

- вкажіть переваги використання RStudio та Spark як інструментів опрацювання великих даних;
- інтерпретуйте отримані результати опрацьованнння освітніх даних;

– сформулюйте рекомендації за результатами регіональних і предметний аналізів;

– обґрунтуйте застосовану вами візуалізацію розподілів результатів;

– інтерпретуйте результати статистичних узагальнень.

Завдання для самостійної роботи

1. Порівняти результати НМТ з математики та української мови.
2. Побудувати boxplot результатів по регіонах.
3. Визначити 5 регіонів з найвищими середніми балами.
4. Проаналізувати залежність між тестовим і рейтинговим балом.

Лабораторна робота 9. Дослідження ефективності моделей лінійних регресій

Мета роботи: Вивчити методики аналізу даних що запропоновані у теоретичній частині роботи. Ознайомитись із результатами роботи запропонованих моделей при дослідженні ефективність прогнозування вартості діамантів від їх характеристик. Використати запропоновані моделі лінійних регресій для дослідження даних контрольного файлу.

Теоретичні відомості

Нижче подано приклади використання моделей лінійних регресій, що використовуються для аналізу даних, які містяться у файлі diamonds.csv. Вивчіть деталі запропонованих методик.

1. Модель «Лінійна регресія» (найпростіша модель)

```
library(dplyr)
# 1. Зчитування даних
diamonds <- read.csv("diamonds.csv", stringsAsFactors = TRUE)
# Перетворення факторних змінних у числові
diamonds_num <- diamonds %>%
```

```
 mutate(cut = as.numeric(as.factor(cut)),
  color = as.numeric(as.factor(color)),
  clarity = as.numeric(as.factor(clarity)))
model_lm <- lm(price ~ ., data = diamonds_num)
summary(model_lm)
pred_lm <- predict(model_lm, newdata = diamonds_num)
mse_lm <- mean((pred_lm - diamonds_num$price)^2)
cat("MSE лінійної моделі:", mse_lm, "\n")
```

2. Random Forest (рекомендована модель – хороша точність)

```
library(randomForest)
library(dplyr)
diamonds <- read.csv("diamonds.csv", stringsAsFactors = TRUE)
# Перевести фактори в числові
diamonds_num <- diamonds %>%
 mutate(cut = as.numeric(as.factor(cut)),
  color = as.numeric(as.factor(color)),
  clarity = as.numeric(as.factor(clarity)) )
set.seed(123)
rf_model <- randomForest(
 price ~ ., data = diamonds_num,
 ntree = 500,  mtry = 4, importance = TRUE)
rf_model
pred_rf <- predict(rf_model, diamonds_num)
# MSE
mse_rf <- mean((pred_rf - diamonds_num$price)^2)
cat("MSE random forest:", mse_rf, "\n")
importance(rf_model)
varImpPlot(rf_model)
```

3. Модель Gradient Boosting (xgboost – найточніша)

```
library(xgboost)
library(dplyr)
diamonds <- read.csv("diamonds.csv", stringsAsFactors = TRUE)
diamonds_num <- diamonds %>%
 mutate(cut = as.numeric(as.factor(cut)),
  color = as.numeric(as.factor(color)),
  clarity = as.numeric(as.factor(clarity)))
# Матриця ознак
X <- as.matrix(diamonds_num %>% select(-price))
y <- diamonds_num$price
# Модель XGBoost
xgb_model <- xgboost(
 data = X,
 label = y,
 objective = "reg:squarederror",
 nrounds = 300, max_depth = 6, eta = 0.1, verbose = 0)
pred_xgb <- predict(xgb_model, X)
mse_xgb <- mean((pred_xgb - y)^2)
cat("MSE XGBoost:", mse_xgb, "\n")
```

4 Модель із використанням neuralnet

```
library(neuralnet)
library(dplyr)
diamonds <- read.csv("diamonds.csv", stringsAsFactors = TRUE)
diamonds_num <- diamonds %>%
 mutate(cut = as.numeric(as.factor(cut)),
  color = as.numeric(as.factor(color)),
  clarity = as.numeric(as.factor(clarity)))
normalize <- function(x) { return((x - min(x)) / (max(x) - min(x)))}
```

```
diamonds_norm <- as.data.frame(lapply(diamonds_num, normalize))
features <- colnames(diamonds_norm)
features <- features[features != "price"]
formula_nn <- as.formula(paste("price ~", paste(features, collapse = " + ")))
set.seed(123)
nn_model <- neuralnet(  formula = formula_nn,
 data = diamonds_norm,
 hidden = c(10),    # один прихований шар на 10 нейронів
 linear.output = TRUE, stepmax = 1e6)
# Граф мережі
plot(nn_model)
pred <- compute(nn_model, diamonds_norm[, features])$net.result
denorm <- function(norm_value, orig_vector) {
 min_val <- min(orig_vector)
 max_val <- max(orig_vector)
 return(norm_value * (max_val - min_val) + min_val) }
pred_price <- denorm(pred, diamonds_num$price)
comparison_table <- data.frame(
 Real_Price = diamonds_num$price,
 Predicted_Price = pred_price)
head(comparison_table, 20)
```

Завдання 1. Побудуйте регресійні моделі та використайте їх для прогнозування одного із параметрів використавши дані контрольного файлу отриманого із сайту «https://www.kaggle.com/datasets/ayeshasiddiqa123/student-perfirmance».

Завдання 2. Проаналізуйте отримані результати та сформулюйте висновок за поданою нижче схемою.

Оформіть висновок:

1. Підготувати та виконати програму, що використовує модель «Лінійна регресія» (найпростіша модель

2. Прокоментувати результати отримані цільовою змінною

3. Інтерпретувати результати summary(model_lm)

4. Інтерпретувати результати змінної mse_lm.

5. Виконайте програму, що реалізує модель Random Forest.

6. Порівняйте результати двох моделей за значеннями mse_rf та mse_lm.

7. Виконайте програму моделі Gradient Boosting.

8. Виконайте програму моделі neuralnet.

9. Порівняйте отримані значення параметра MSE та оцініть ефективність моделей.

10. Порівняйте результати прогнозування використавши отримані моделі.

Лабораторна робота 10. Аналіз залежності між вартістю основних засобів і випуском продукції

Мета роботи:

- набути практичних навичок роботи з числовими даними в RStudio;
- навчитися будувати лінійну регресійну модель;
- обчислювати коефіцієнт кореляції, детермінації та еластичності;
- перевіряти статистичну значущість моделі за допомогою t- та F-критеріїв;
- інтерпретувати отримані результати економічного аналізу.

Теоретичні відомості

1. Лінійна регресійна модель. Модель описує залежність випуску продукції від вартості основних засобів. Лінійна регресія описує залежність між змінними у вигляді: [Y = a + bX + \varepsilon]

$$Y = a + bX + \varepsilon$$

де:

- (Y) — результативна ознака (випуск продукції);

- (X) — факторна ознака (вартість основних засобів);

- (a) — вільний член;

- (b) — коефіцієнт регресії;

- (ε) – varepsilon — випадкова похибка.

2. Коефіцієнт кореляції

Коефіцієнт кореляції (r) показує тісноту та напрям зв’язку між змінними:

(r приблизно -1) — сильний прямий зв’язок;

(r приблизно -1) — сильний обернений зв’язок;

(r приблизно - 0) — слабкий зв’язок.

3. Коефіцієнт детермінації

R^2 – показує, яку частку варіації результативної ознаки пояснює модель.

4. Коефіцієнт еластичності

$$E = b \cdot \frac{\bar{X}}{\bar{Y}}$$

Показує, на скільки відсотків зміниться (Y), якщо (X) зміниться на 1%.

5. Критерії значущості

t-критерій Стьюдента — значущість окремого коефіцієнта;

F-критерій Фішера — загальна значущість регресійної моделі.

Вхідні дані:

Вартість основних засобів (X): 12, 8, 10, 6, 9, 15, 11, 13, 14, 10

Випуск продукції (Y): 5.6, 4.0, 4.0, 2.4, 3.6, 5.0, 4.6, 6.5, 7.0, 4.5

Хід виконання роботи

Завдання 1. Введення даних у R.

1. Сформуйте вектор X – «Вартість основних засобів» та вектор Y – «Випуск продукції».

```
X <- c(12, 8, 10, 6, 9, 15, 11, 13, 14, 10)
Y <- c(5.6, 4.0, 4.0, 2.4, 3.6, 5.0, 4.6, 6.5, 7.0, 4.5)
```

2. Побудуйте лінійну регресійну модель

```
model <- lm(Y ~ X)
summary(model)
```

3. Коефіцієнт еластичності

```
b <- coef(model)[2]
elasticity <- b * mean(X) / mean(Y)
elasticity
```

Показує, на скільки відсотків зміниться випуск продукції при зміні основних засобів на 1%.

4.Дисперсія змінних:

```
var_X <- var(X)
var_Y <- var(Y)
var_X
var_Y
```

Дисперсія залишків моделі:

```
residual_variance <- var(residuals(model))
```

residual_variance – Характеризує розсіювання значень навколо середнього.

5.Коефіцієнт кореляції

```
correlation <- cor(X, Y)
correlation
```

Інтерпретація:

- значення близьке до 1 → сильний прямий зв’язок;
- близьке до 0 → слабкий зв’язок.

6.Коефіцієнт детермінації (R^2)

```
R² <- summary(model)$r.squared
```

R^2 – Показує, яку частку варіації Y пояснює змінна X.

7. F-критерій Фішера

```
F_stat <- summary(model)$fstatistic[1]
F_stat
```

Перевіряє загальну значущість регресійної моделі.

8. t-критерій Стьюдента для коефіцієнта регресії

t_stat <- summary(model)$coefficients[2, "t value"]

t_stat – Перевіряє статистичну значущість впливу X на Y.

9. Повна таблиця результатів

```
results <- data.frame(
 Показник = c(
  "Коефіцієнт еластичності",
  "Дисперсія X",
  "Дисперсія Y",
  "Коефіцієнт кореляції",
  "Коефіцієнт детермінації R²",
  "F-критерій Фішера",
  "t-критерій Стьюдента"
 ),
 Значення = c(
  elasticity,
  var_X,
  var_Y,
  correlation,
  R2,
  F_stat,
  t_stat
 ))
results
 Графік регресії
plot(X, Y,
   main = "Залежність випуску продукції від вартості основних засобів",
   xlab = "Вартість основних засобів",
   ylab = "Випуск продукції",
   pch = 19)
```

abline(model, col = "blue")

10. За результатами проведеного аналізу даних дайте відповідь на наступні питання:

– оцініть тісноту зв’язку між показниками;

– на скільки відсотків зміниться випуск продукції при зміні основних засобів на 1%;

– інтерпретуйте результати перевірки статистичної значущість моделі;

– якими результатами обчислень підтверджено можливість використання лінійної регресії для аналізу;

– яке рівняння отримано при побудові лінійної регресійної модель у RStudio;

– поясніть результати виконаного кореляційно-регресійного аналізу;

– оцініть статистичну значущість моделі;

ЗАВДАННЯ:

Проведіть дослідження аналогічне наведеному у лабораторній роботі використавши для цього дані:

Глобальні продажі та ринкові дані BMW –

https://www.kaggle.com/datasets/payaldhokane/bmw-global-sales-and-market-data

Лабораторна робота 11. Аналітичне дослідження даних

Мета: Провести самостійне дослідження даних. Вивчити структуру даних. Оцінити потребу у проведенні підготовчих операцій із даними. Виконати структиризацію та логічну оцінку. Сформувати ідею моделі дослідження даних. При потребі сформувати робочу гіпотезу. Самостійно провести всі аналітичні операції із використанням середовища RStudio. Сформувати висновок за отриманими результатами.

Теоретичні відомості

Для виконання цієї лабораторної роботи необхідно обрати файл із

даними, над якими в подальшому проводиться аналітична робота. Рекомендуємо обрати файл із сайту *kaglle.com*. Контент даних, що зберігаються і файлах CSV різноплановий.

Як приклад наведено повний цикл дослідження даних одного із доступних файлів. Він пропонує аналітикам даних можливість проводити кореляційні дослідження між академічними факторами та дослідницьким аналізом даних (EDA).

Опис набору даних:

Цей набір даних містить інформацію про 200 студентів, зосереджуючись на факторах, що впливають на академічну успішність. Кожен запис представляє одного студента, ідентифікованого унікальним ідентифікатором. Він містить шість ключових характеристик, наприклад: години навчання, тривалість сну, та інші.

Опис колонок:

- student_id – спеціальний номер, який надається кожному студенту.
- hours_studied – скільки годин студент загалом витратив на навчання перед тестом.
- sleep_hours – скільки годин сну студент зазвичай отримує щодня.
- відсоток_відсутності – відсоток відвідуваності студента на заняттях.
- попередні_оцінки (previous_scores) – загальний бал (зі 100) за попередні тести студента.
- exam_score – бал поточного іспиту (зі 100).

Дані дозволяють досліджувати взаємозв'язки між звичками навчання, способом життя та результатами іспитів. За його допомогою можна виконувати аналіз даних, візуалізацію та розробку моделей прогнозування для оцінки успішності учнів.

Аналіз даних будуємо за наступним сценарієм.

Графік Розсіювання. (Години Навчання vs. Бал Іспиту)

Покаже найсильніший зв'язок у даних: залежність балу іспиту

(exam_score) від годин навчання (hours_studied).

Це класичний графік розсіювання, доповнений лінією регресії, що демонструє прогнозну модель.

Виконаний графік наочно демонструє винятково сильний позитивний лінійний зв'язок між кількістю годин, витрачених на навчання (hours_studied), та балом на іспиті (exam_score). Точки даних щільно групуються навколо висхідної лінії регресії, що підтверджує коефіцієнт кореляції Пірсона r ≈ +0.89. Таким чином, ми візуально встановили, що час, присвячений навчанню, є домінуючим фактором, що визначає академічний успіх у цьому наборі даних.

Графік Розсіювання (Попередні Оцінки vs. Бал Іспиту)

Цей графік покаже залежність Балу Іспиту (exam_score) від Попередніх Оцінок (previous_scores).

Цей графік також є діаграмою розсіювання з лінією регресії. Ми бачимо, що попередня академічна успішність є сильним, але не домінуючим фактором. Лінія регресії спрямована вгору, підтверджуючи позитивний зв'язок (кореляція r ≈ +0.73). Однак, на відміну від першого графіку, точки тут більш розкидані навколо лінії. Це свідчить про те, що попередня оцінка хоча й важлива, але не так точно прогнозує фінальний бал, як години навчання.

Графік Слабких Факторів (Фасетний Графік)

Оскільки sleep_hours (сон) і attendance_percent (відвідуваність) мають дуже низьку кореляцію (близько r ≈ +0.05 та r ≈ +0.04 відповідно), найкращий спосіб їх візуалізувати — це порівняти їхню "плоску" залежність на одному графіку.

Цей код створить один графік із двома окремими панелями (*фасетами*) для наочного порівняння впливу годин сну та відвідуваності на бал іспиту.

Цей фасетний графік візуально підтверджує практичну відсутність лінійного зв'язку між балом іспиту та годинами сну або відвідуваністю. Лінії регресії на обох панелях є майже горизонтальними. Це означає, що незалежно від того, спить студент 4 чи 9 годин, або чи має він 50% чи 95% відвідуваності, його прогнозований бал іспиту залишається приблизно однаковим (тобто,

близько середнього значення). Точки на графіку дуже сильно розсіяні без чіткої тенденції, що є графічним підтвердженням низької кореляції (близько нуля).

Графік Прогнозування з Кольоровою Класифікацією

Цей код створює модель, обчислює залишки, а потім додає логіку для класифікації кожного студента як "Перевершив" (Overperformer), "Відповідає Прогнозу" (Meets Expectation) або "Не Досягнув" (Underperformer).

Побудова моделі прогнозування

Модель лінійної регресії, що пояснює exam_score через hours_studied та previous_scores.

Класифікація студентів за залишками

Створюємо нову категоріальну змінну для класифікації Результат.

Побудований графік є Графіком Залишків (*Residual Plot*), який візуалізує точність вашої моделі множинної лінійної регресії.

Залишок (.resid на осі Y) — це помилка прогнозування. Різниця між фактичним балом іспиту, який отримав студент, і прогнозованим балом, який розрахувала модель:

Позитивний залишок (точка над нульовою лінією) означає, що фактичний бал був вищим, ніж прогнозувалося.

Негативний залишок (точка під нульовою лінією) означає, що фактичний бал був нижчим, ніж прогнозувалося.

Залишок, близький до нуля (точка біля лінії) означає, що прогноз був високоточним.

Теплова Карта Кореляційної Матриці (Heatmap)

Це графік, який використовує колірну інтенсивність (теплоту) для відображення сили та напрямку зв'язку (кореляції) між усіма парами числових змінних у вашому наборі даних.

Матриця: Це квадратна таблиця, де кожен рядок і кожен стовпець представляє одну зі змінних (наприклад, exam_score, hours_studied, sleep_hours тощо). На перетині рядка і стовпця знаходиться коефіцієнт кореляції Пірсона (r) між цими двома змінними.

Теплова Карта: Замість того, щоб читати чисті цифри, ми використовуємо колір.

Тепло" (колір) на карті має чітке значення:

Колір	Коефіцієнт (r)	Значення
Інтенсивний Теплий (Червоний/Помаранчевий)	r ≈ +1	Сильний Позитивний Зв'язок: Зі зростанням однієї змінної, зростає й інша. (Наприклад, hours_studied та exam_score).
Нейтральний (Білий/Світлий)	r ≈ 0	Відсутність Зв'язку: Змінні не впливають одна на одну. (Наприклад, sleep_hours та exam_score).
Інтенсивний Холодний (Синій/Фіолетовий)	r ≈ -1	Сильний Негативний Зв'язок: Зі зростанням однієї змінної, зменшується інша.

Інтерпретація результатів.

Домінуючі Фактори

– hours_studied vs. exam_score (r = +0.78): Це найсильніший фактор.

Це все ще сильна позитивна кореляція, що підтверджує, що час навчання є головним рушієм успіху.

– previous_scores vs. exam_score (r = +0.43): Це помірна позитивна кореляція.

Це означає, що попередні оцінки важливі, але набагато менше, ніж години навчання.

Слабкі Фактори

– attendance_percent vs. exam_score (r = +0.23): Це слабка позитивна кореляція.

– sleep_hours vs. exam_score (r = +0.19): Це дуже слабка позитивна кореляція.

В результаті комплексного розвідувального аналізу даних та прогнозного моделювання встановлено, що головним фактором успіху на іспиті є кількість годин навчання, тоді як попередні оцінки мають лише помірний вплив. Побудована модель лінійної регресії є надійною, пояснюючи значну частину результатів, але найціннішим є виявлення студентів, які перевершили або недотягнули до спрогнозованого балу, що вимагає подальшого, глибшого вивчення неврахованих факторів.

Завдання лабораторної роботи:

1. Проаналізуйте опис файлів доступних на вказаному сайті. Оберіть зрозумілу для вас тематику і скачайте файл даних (.CSV).

(https://www.kaggle.com/datasets/sehaj1104/student-lifestyle-and-academic-performance-dataset)

2. Побудуйте модель дослідження даних вашого файлу, передбачте виконання наступних завдань:

- Імпорт даних
- Очищення та трансформація даних
- Статистичний аналіз
- Статистичне моделювання
- Візуалізація даних
- Машинне навчання та прогнозування
- Інтерпретація результатів
- Документування та створення звітів.

Інтегровані завдання до розділу 10

Завдання 1.

1. Створити фрейм даних teachers та заповнити його перші п’ять рядків:

- ID
- Name
- Age
- Subject
- Experience (роки)

2. Підготувати програму яка виведе на екран:
   - структуру фрейму;
   - перші 5 рядків;
   - назви стовпців.
3. Підготувати оптимальну програму яка знайде і виведе:
   - середній вік викладачів;
   - мінімальний та максимальний стаж роботи.
4. Підготувати програмний код, для того щоб додати да фрейму teachers стовпець Category із врахуванням таких умов:
   - Experience ≥ 15 → "Senior"
   - 5–14 → "Middle"
   - <5 → "Junior"
5. Вивести імена викладачів із фрейму teachers зі стажем понад 10 років.
6. Відсортувати фрейм teachers та зберегти результати в різних фреймах**:**
   - за стажем (спадання);
   - за віком (зростання).
7. Вивести на екран тільки стовпці Name та Experience для викладачів категорії "Senior".
8. Творче завдання:
   - Зберегти створений фрейм у файл CSV з власною назвою, завантажити його назад і перевірити: чи збереглись типи даних; чи співпадає структура з оригіналом.
   - Сформулювати 2–3 пункти висновків за отриманими статистичними результатами (у текстовому вигляді).

Завдання 2.

1. Створити фрейм даних students_extended з 5-10 студентів, де будуть ID, Name, Age, Group, Score
2. Підготувати код який додасть стовпці до існуючої таблиці даних:
   - Passed (TRUE/FALSE);
   - Grade (A, B, C).
3. Вивести на екран відфільтровані результати для студентів:
   - які не склали предмет;
   - які отримали оцінку "A".
4. Підготувати програму, яка використовуючи таблицю даних, обчислить:
   - кількість студентів у кожній оцінковій категорії;
   - середній бал тільки серед студентів, які склали предмет.
5. Творче завдання:
   - Зберегти створений фрейм у файл CSV з власною назвою, завантажити його назад і перевірити: чи збереглись типи даних; чи співпадає структура з оригіналом.
   - Сформулювати 2–3 пункти висновків за отриманими статистичними результатами (у текстовому вигляді).

Завдання 3.

1. Виконати аналіз набору даних, завантаженого з CSV-файлу.
2. Обчислити мінімальне та максимальне значення змінної mpg для кожної групи cyl.
3. Побудувати модель логістичної регресії у Spark.
4. Підготувати короткий звіт з поясненням отриманих результатів.

Завдання 4.

1. Проаналізувати успішність за дисциплінами.

2. Побудувати модель лінійної регресії для прогнозу оцінки.
3. Додати фактор "форма навчання" та повторити аналіз.
4. Підготувати рекомендації для підвищення успішності студентів.

Контрольні питання до розділу 10

1. Дайте визначення False Positive (FP) та False Negative (FN).
2. Для чого використовується F-критерій?
3. Для чого використовується пакет dplyr?
4. Для чого використовується функція collect() і чому вона може бути небезпечною?
5. Для чого використовується функція mutate()?
6. До якого типу машинного навчання належить класифікація і чому?
7. У чому економічний зміст коефіцієнта еластичності?
8. У чому полягає ідея методу опорних векторів (SVM)?
9. У чому полягає принципова відмінність між класифікацією та регресією?
10. У чому різниця між summarize() та mutate()?
11. У якому форматі доцільно зберігати табличні дані для подальшої обробки в R?
12. Чим Spark DataFrame відрізняється від data.frame у R?
13. Чому алгоритм SVM потребує масштабування даних?
14. Чому для алгоритму k-NN необхідно виконувати нормалізацію даних?
15. Чому логістична регресія є класифікаційним алгоритмом?
16. Що означає поняття «ліниві обчислення» в Spark?
17. Що показує коефіцієнт кореляції?
18. Що таке Apache Spark і для чого він використовується?
19. Що таке матриця похибок?
20. Що таке фрейм даних (data frame) у мові R і чим він відрізняється від матриці?
21. Як визначити структуру, кількість рядків і стовпців фрейму даних?

22. Яка різниця між доступом до даних за допомогою $ та []?
23. Яке призначення бібліотеки sparklyr?
24. Яке призначення коефіцієнта детермінації?
25. Який механізм дозволяє SVM розділяти нелінійно розділені класи?
26. Які переваги використання RStudio для статистичного аналізу?
27. Які типи даних можуть зберігатися в стовпцях data frame?
28. Яку інформацію дає візуалізація даних у процесі аналізу?
29. Яку роль відіграє group_by() у процесі аналізу даних?
30. Яку роль відіграє параметр *k* в алгоритмі k-NN?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Dzhun, J.V., Yasinsky, A.M., Shportko, A.V. et al. Software Implementation of a Numerical Method for Obtaining Maximum Likelihood Estimates of Parameters of the Jeffreys Modification of the Pearson Type VII Distribution. Cybern Syst Anal 61, 857–863 (2025). https://doi.org/10.1007/s10559-025-00817-4

2. Michael F. Lawrence, John Verzani. Programming Graphical User Interfaces in R. Chapman and Hall/CRC, Boca Raton, FL, 2012. 480p. ISBN 978-1-4398-5682-6. http://www.crcpress.com/product/isbn/9781439856826

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

4. Аналіз результатів національного мультипредметного тесту із використанням інструментів статистичного моделювання в RSTUDIO / авторів Ясінський А.М., Соловей Л.Я., Колупаєв Б.Б., Ільків О.С. //Вісник Міжнародного економіко-гуманітарного університету імені Академіка Степана Дем'янчука. Серія: Педагогіка та психологія. - Рівне : Видав. дім "Гельветика", 2025. - № 4. - С.49-56.

5. Вашків П.Г., Пастер П.І., Сторожук В.П. Ткач Є.І. Теорія статистики: Навчальний посібник–К: Либідь, 2001. -320с.

6. Гнатюк В. Вступ до R на прикладах / Віктор Гнатюк. – Харків : ХНЕУ, 2010. – 101 с

7. Застосування середовища RSTUDIO при дослідженні задач з математики, фізики та економіки / Ясінський Андрій, Соловей Людмила // Цифрова платформа: інформаційні технології в соціокультурній сфері: наук. журн. / Київ. нац. ун-т культури і мистецтв. – Київ: Вид. центр КНУКіМ, Том 7 № 2, 2024.с. 224–231

8. Івахненко В. Математичні методи і моделі ринкової економіки [Електрон-ний ресурс] : методичні рекомендації до лабораторних робіт для

студентів спеціальності 051 "Економіка" другого (магістерського) рівня / уклад. О. В. Івахненко. – Харків : ХНЕУ ім. С. Кузнеця, 2019. –65 с.

9. Комп'ютерна статистика : підручник / Р. Є. Майборода. – К. :ВПЦ "Київський університет", 2019. – 589 с. ISBN 978-966-933-010-9

10. 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.

11. Майборода Р. Є. Аналіз даних за допомогою пакета R 6 : навч. посібник : [Електронний ресурс] / Р. Є. Майборода, О. В. Сугакова. – Київ, 2015. – 65 с. Режим доступу: http://matphys.rpd.univ.kiev.ua/downloads/courses/mmatstat/ Statistics_with_R.pdf.

12. Методичні рекомендації до самостійної роботи з математичних дисциплін з використанням програмного середовища R для студентів усіх спеціальностей першого (бакалаврського) рівня [Електронний ресурс] / укл. Л.М. Малярець, О.Г. Тижненко; Харківський національний економічний університет ім. С. Кузнеця. - Електрон. текстові дан. (2.93 МБ). - Х. : ХНЕУ ім. С. Кузнеця, 2017. - 84 с.

13. Опрацювання статистичних даних педагогічного дослідження засобами мови R / Й. В. Джунь, А. М. Ясінський, Л. Я. Соловей // Вісник Міжнародного економіко-гуманітарного університету імені академіка Степана Дем'янчука.Серія: "Педагогіка та спихологія". Міжнар. економ.-гуманіт. ун-т ім. Степана Дем’янчука. - Рівне : Видав. дім "Гельветика", 2025. - № 1. - С.49-56.

14. Порівняльний аналіз прогнозування за допомогою лінійної регресії та нейронної мережі засобами бібліотеки Neuralnet /Ясінський А.М., Джунь Й.В., Кундос М.Г., Соловей Л.Я // Вчені записки Таврійського національного університету імені В.І. Вернадського.Серія: Технічні науки. Видавничий дім «Гельветика».Том 36 (75) № 1, с.271-276, 2025

15. Програмування мовою R для підвищення ефективності математичної освіти та досліджень: методи, інструменти та практичні застосування / Андрій Миколайович Ясінський, Людмила Ярославівна Соловей, Вячеслав Андрійович Ковальчук// Вісник Міжнародного економіко-гуманітарного університету імені Академіка Степана Дем'янчука. Серія: Педагогіка та психологія Рівне : Видав. дім "Гельветика", 2024. - № 2. - С.59-66.
16. https://cran.r-project.org/
17. http://data.worldbank.org/
18. http://www.rstudio.com/
19. https://www.codecademy.com/learn
20. https://www.coursera.org/
21. https://www.kaggle.com/

ГЛОСАРІЙ

Boxplot – графічний метод відображення розподілу даних на основі квартилів.

CRAN (Comprehensive R Archive Network) – офіційне сховище пакетів мови R.

CSV-файл – текстовий файл для зберігання табличних даних з розділенням комами або іншими символами.

Data frame – таблична структура даних, у якій кожен стовпець може мати власний тип даних.

ggplot2 – пакет мови R для побудови складних і багатовимірних графіків.

R – мова програмування та середовище для статистичних обчислень, аналізу даних і побудови графіків, що широко використовується в наукових дослідженнях та освіті.

RStudio – інтегроване середовище розробки (IDE) для мови програмування R, яке забезпечує зручні інструменти для написання коду, аналізу даних, візуалізації та документування результатів.

Tibble – сучасна версія data frame з покращеним відображенням і додатковими можливостями.

Автоматизація аналізу – використання програмних засобів для повторюваного та ефективного виконання аналітичних операцій.

Вектор – базова структура даних у R, що містить елементи одного типу.

Векторизація – виконання операцій над цілими векторами без використання циклів.

Великі дані (Big Data) – набори даних великого обсягу та складності, що потребують спеціальних методів обробки.

Візуалізація даних – графічне подання даних з метою їх аналізу та інтерпретації.

Гістограма – графік для відображення розподілу числових даних.

Діаграма розсіювання – графік, що відображає взаємозв’язок між двома змінними.

Експорт даних – збереження результатів аналізу у файли зовнішніх форматів.

Імпорт даних – процес завантаження даних із зовнішніх джерел у середовище R.

Індексація – механізм доступу до окремих елементів або підмножин даних.

Класифікація даних – процес віднесення об’єктів до певних класів за заданими ознаками.

Логічна фільтрація – відбір даних на основі логічних умов.

Логічний тип (logical) – тип даних, що приймає значення TRUE або FALSE.

Маніпуляція даними – перетворення, фільтрація, сортування та об’єднання наборів даних.

Масив (array) – багатовимірна структура даних у R.

Матриця – двовимірна структура даних з елементів одного типу, організованих у рядки та стовпці.

Очищення даних – процес виявлення та усунення помилок, пропусків і некоректних значень у даних.

Пакет (package) – набір функцій, даних та документації, що розширює базові можливості мови R.

Проєкт RStudio – організований робочий простір у RStudio, що об’єднує файли коду, дані та результати аналізу в межах одного каталогу.

Пропущені значення (NA) – відсутні або невідомі значення в наборі даних.

Символьний тип (character) – тип даних для зберігання текстової інформації.

Сімейство apply – набір функцій у R (apply, lapply, sapply тощо), призначених для обробки даних без явних циклів.

Список (list) – структура даних, що може містити елементи різних типів.

Статистичні характеристики – числові показники, що описують властивості даних (середнє, медіана, дисперсія тощо).

Структура даних – спосіб організації та зберігання даних у пам’яті комп’ютера.

Тип даних – характеристика змінної, що визначає вид даних та можливі операції над ними (числовий, символьний, логічний тощо).

Фактор (factor) – спеціальний тип даних у R для представлення категоріальних змінних.

Функціональне програмування – парадигма програмування, що базується на використанні функцій як основних елементів обчислень.

Функція – іменований блок коду, що виконує певну задачу та може приймати аргументи.

Числовий тип (numeric) – тип даних у R для зберігання числових значень.

ПЕРЕЛІК СКОРОЧЕНЬ

Big Data – великі за обсягом та складністю набори даних, що потребують спеціальних методів опрацювання.

CRAN – Comprehensive R Archive Network, офіційне сховище пакетів мови R.

CSV – Comma-Separated Values, текстовий формат файлів для зберігання табличних даних.

DBI – Davies–Bouldin Index, кількісний показник оцінювання якості кластеризації, що характеризує співвідношення внутрішньокластерної компактності та міжкластерної відокремленості; менші значення індексу відповідають кращій структурі кластерів і часто використовуються в R для порівняння результатів алгоритмів кластеризації (зокрема k-means)..

ETL – Extract, Transform, Load, процес вилучення, перетворення та завантаження даних.

GUI – Graphical User Interface, графічний інтерфейс користувача.

I/O – Input/Output, введення та виведення даних.

IDE – інтегроване середовище розробки (Integrated Development Environment).

JSON – JavaScript Object Notation, формат обміну структурованими даними.

ML – Machine Learning, машинне навчання.

NA – Not Available, позначення пропущених значень у мові R.

R – мова програмування для статистичних обчислень і аналізу даних.

RStudio – інтегроване середовище розробки для мови програмування R.

SQL – Structured Query Language, мова структурованих запитів для роботи з базами даних.

Tidyverse – колекція пакетів мови R для аналізу та візуалізації даних.

TXT – текстовий формат файлів для зберігання даних.

XML – eXtensible Markup Language, мова розмітки для зберігання та передавання даних.