Вступ до програмування в R

Навчальний посібник з курсу **Вступ до прикладного програмування в R** для студентів спеціальності економічна кібернетика, 1 курс

Юрій Клебан

Invalid Date

Зміст

# Про посібник

Юрій Клебан  
2022-11-12

|  |
| --- |
|  |

Матеріали навчального посібника підготовлені для читання курсу ***“Вступ до прикладного програмування в R”*** [05.250] студентам 1-го року навчання спеціальності економічна кібернетика [Національного університету “Острозька академія”](https://oa.edu.ua).

## Опис навчальної дисципліни

Навчальна дисципліна спрямована на вивчення основ практичного застосування популярної мови R для проведення статистичних досліджень в економіці.

У процесі вивчення курсу розглядаються теми, що стосуються теоретичних основ та практичної реалізації алгоритмів, завантаження, підготовки та обробки економічних даних.

Місце навчальної дисципліни у підготовці здобувачів: програмні результати дисципліни використовуються під час вивчення таких навчальних дисциплін: “Алгоритми та структури даних”, “Аналіз даних в R”, “Прикладне математичне моделювання в R”, “Підготовка аналітичних звітів”. Закріплення на практиці здобутих програмних результатів відбувається під час проходження навчальної практики з курсу “Економіко-математичне моделювання”.

## Мета дисципліни

Мета навчальної дисципліни – формування у студентів теоретичних знань та практичних навичок використання мови програмування R для роботи з даними та базовими структурами мови (типи даних, розгалуження, цикли, функції).

## Рецензенти

##### Зовнішній рецензент

[Sasha Talavera](https://www.otalavera.com/), Ph. D. in Economics, Professor of Financial Economics at the Department of Economics, Birmingham Business School, University of Birmingham.

[Текст рецензії](#Xa39a3ee5e6b4b0d3255bfef95601890afd80709)

##### Внутрішній рецензент

[Анатолій Нікітін](https://qa.oa.edu.ua/front/cv/anatolii_nikitin), доктор фізико-математичних наук, доцент, професор кафедри економіко-математичного моделювання та інформаційних технологій Національного університету “Острозька академія”.

[Текст рецензії](#Xa39a3ee5e6b4b0d3255bfef95601890afd80709)

## Підтримка проєкту

Матеріали навчального посібника створено у межах проєкту “Підготовка, обробка та ефективне використання даних для наукових досліджень (на основі R)”, що підтримується Європейським союзою за програмою [House of Europe](https://houseofeurope.org.ua/).

## Дотримання принципів доброчесності

Викладач та слухач цього курсу, як очікується, повинні дотримуватися Кодексу академічної доброчесності університету:

* будь-яка робота, подана здобувачем протягом курсу, має бути його власною роботою здобувача; не вдаватися до кроків, що можуть нечесно покращити Ваші результати чи погіршити/покращити результати інших здобувачів;
* якщо буде виявлено ознаки плагіату або іншої недобросовісної академічної поведінки, то студент буде позбавлений можливості отримати передбачені бали за завдання;
* не публікувати у відкритому доступі відповіді на запитання, що використовуються в рамках курсу для оцінювання знань здобувачів;
* під час фінальних видів контролю необхідно працювати самостійно; не дозволяється говорити або обговорювати, а також не можна копіювати документи, використовувати електронні засоби отримання інформації.

Порушення академічної доброчесності під час виконання контрольних завдань призведе до втрати балів або вживання заходів, які передбачені Кодексу академічної доброчесності НаУОА.

|  |  |
| --- | --- |
|  | Матеріали курсу створені з використанням ряду технологій та середовищ розробки:   * ☒ [Мова R](https://www.r-project.org) - безкоштована мова програмування для виконання досліджень у сфері статистики, машинного навчання та візуалізацї результатів. * ☒ [Quarto Book](https://quarto.org) - система для публікації наукових та технічних текстів з відкритим кодом (R/Python/Julia/Observable). * ☒ [JupyterLab](https://github.com/jupyterlab/jupyterlab) - середовище розробки на основі [Jupyter Notebook](https://jupyter.org/). JupyterLab є розширеним веб-інтерфейсом для роботи з ноутбуками. * ☒ [Git](https://git-scm.com/)/[Github](https://github.com/) - система контролю версій та, відповідно, сервіс для організації зберігання коду, а також публікації статичних сторінок. * ☒ [RStudio Desktop](https://www.rstudio.com/) - інтегроване середовище розробки (IDE) для мови R з відкритим кодом, що містить в собі редактор коду, консоль, планер, засоби візуалізації та можливості. * ☒ [Visual Studio Code](https://code.visualstudio.com/) - інтегроване середовище розробки (IDE) з відкритим кодом практично для усіх відомих технологій та мов програмування. |

Бібілографічний опис bibtex:

@book{yk-r-intro,  
 author = {Юрій Клебан},  
 title = {Вступ до програмування в R},  
 publisher = {Zenodo},  
 year = 2022,  
 doi = {10.5281/zenodo.7251419},  
 url = {https://doi.org/10.5281/zenodo.7251419}  
}

# Вступ

Юрій Клебан  
2022-11-12

Фахівці спеціальності економічна кібернетика у майбутньому працюватимуть з великими масивами даних, що накопичуються велизними темпами у даний момент і збиралися у попередні дисятиліття. Підготовка, обробка і трансформація даних у зручний формат прийняття рішень забирає все більше часу, а звичні рашіне інструменти аналізу даних, як наприклад, Microsoft Excel не мають достатньо вбудованих можливостей для виконнання задач бізнесу.

На даний час існує велика кількість мов програмування, що інтегруються у суспільні сфери діяльності людини та роботи технічних систем: біоінформатика, а також економіка та бізнес.

Однією з мов програмування, що отримали широке поишення серед економістів-науковців, аналітиків та практиків математичного моделювання (machine learning) є мова програмування (R)(R Core Team 2020). Свою популярність ця мова програмування здобула завдяки простоті у використанні, доступності (безкоштовні як базові компоненти для написання коду, так і середовища розробки), розширюваності (кожен розробник має можливість створювати власні пакети та публікувати їх у відкритому доступі).

Основними задачами курсу “Вступ до прикладного прграмування в R” є ознайомлення студентів з базовми конструкціями мови програмування R, вивчення способів роботи з найпоширенішими типами даних,

# 1. Що таке R?

Юрій Клебан  
2022-11-12

R є поширеною мовою програмування для роботи з даними (DataScience) та машинного навчання (Machine Learning). Але Ви можете скористатися засобами R і для простіших задач: обчислення, візуалізація даних.

Синтаксис мови програмування R є досить простим для вивчення та використання, а широкий набір готових пакетів дозволяє використати готові розробки для виіршення широкого спектру задач від статистичних обчислень до навчання нейронних мереж для розпізнавання/класифікації зображень.

Важливо відмітити, що мова програмування R є безкоштовною (free) і має відкритий код (open source).

R має ряд корисних властивостей, серед яких варто виділити:

* **Візуалізація даних**. Побудова різноманітих видів графіків, робота з мапами, широкий спектр бібліотек та налаштувань до них.
* **Повторне використання коду**. На відміну від електронних таблиць, що мають обмеження на кількість спостережень (наприклад, MS Excel), R дозволяє працювати з великими масивами даних та перезапускати обчислення у потрібний момент не створюючи додаткових копій даних.
* **Машинне навчання**. R дозволяє використати для побудови, навчання та тестування моделей, а також оптимізації гіперпараметрів та відбору факторів дуже велику кількість алгоритмів. Існують також спеціальні пакети, що об’єднують у собі усі описані функції та алгоритми, наприклад, caret(Kuhn 2022) та mlr(Lang et al. 2019).
* **Автоматизація**. Написаний код та проєкти можна перетворити у готові до публікації та впровадження продукти (deployment) або використовувати напрацьовані алгоритми для швидкого вирішення схожих задач (pipeline).

Також можна виділити досить корисні фічі **Розробка веб-застосунків** та **Звітність**, адже, використовуючи спеціальні бібліотеки (shiny(Chang et al. 2022), shinydashboard(Chang and Borges Ribeiro 2021), flexdashboard(Sievert et al. 2022), rmarkdown(Xie, Dervieux, and Riederer 2020), knitr(Xie 2015) тощо) результати виконаної роботи можна “оживити” або сформувати “на льоту” готові до презентації документи.

# 2. Коротка історія мови R

Юрій Клебан  
2022-11-12

Мова програмування R виникла як продовження статистичної мови S. Назва мови S була обрана аналогічно до C. Створена S була у 1976 році компанією Bell Labs. Мова S мала кілька версій і широко використовувалася для комерційного програмування. Найпотужнішою була версія S-Plus, що мала реалізацію за досить немалу кількість функцій під Windows та Unix-платформи, що стримувало її розвиток. Саме в цей момент розпочинається історія R.

Влітку 1993 року двоє молодих новозеландських вчених анонсували свлю нову розробку, яку вони назвали R (є інформація, що буква "R" була обрана тому, що вона стоїть перед "S" у латинському алфавіті, тут є аналогія з мовою "C", якій передувала мова "B"). За задумом авторів (Robert Gentelman та Ross Ihaka) це повинна була бути нова реалізація мови S, що відрізнялася від S-Plus деякими деталями, наприклад, роботою з локальними та глобальними змінними, пам’яттю тощо. Фактично було створено нову мову, що відгалуджується від S.

Проект з самого початку розвивався досить повільно, але коли у команди розробників R з’явлися ресурси, в тому числі зручна системи створення розширень (пакетів), все більше аналітиків, статистиків, вчених, програмістів почало переходити з S-Plus на R. Коли були усунуті проблеми роботи з пам’яттю перших версій R, на цю мову почали переходити користувачі інших статистичних пакетів (SAS, Stata, SYSSTAT).

Кількість книг та публікацій у мережі Інтернет по роботі з R постійно зростає разом із зацікавленням молодих і вже досвідчених спеціалістів зі сфери ІТ темою науки про дані, машинним навчанням, аналітикою для бізнесу, охорони здоров’я тощо.

# 3. Вcтановлення та налаштування R

Юрій Клебан  
2022-11-12

R є безкоштовним програмним забезпеченням, що розповсюджується за умовами [GNU General Public License](https://www.r-project.org/COPYING). Код, написаний на R компілюється та запускається на різних платформах: UNIX, Windows, MacOS (R Core Team 2020).

Для завантаження актуальної версії R варто перейти на сайт проекту <https://cran.r-project.org/>.

На сайті обираємо завантаження R для потрібної операційної системи. У межах курсу ми вокристовуємо ОС Windows, проте на синтаксис мови програмування та процес написання коду це не впливає:

|  |
| --- |
| Рис. 3.1. Завантаження R. Вибір ОС |

У наступному вікні клікаємо на **install R for the first time**:

|  |
| --- |
| Рис. 3.2. Завантаження R. Перша інсталяція |

Далі обираємо **Download R 4.X.X for Windows**, де 4.X.X версія R, яка може бути відмінною на момент вивчення курсу:

|  |
| --- |
| Рис. 3.3. Завантаження R. Завантаження версії для ОС |

Після завантаження файлу інсталяції потрібно його запустити. Зазвичай завантажений файл можна побачити у лівому нижному кутку браузера або у розділі “Завантаження” Вашого браузера. Наприклад, у браузері Google Chrome знайти цей пункт меню так:

|  |
| --- |
| Рис. 3.4. Завантаження R. Розділ "Завантаження" у Google Chrome |

## 3.1 Перший запуск R GUI

За замовчуванням під час інсталяції R у ОС Windows пропонується шлях C:\Program Files\R\R-4.X.X.

Для запуску R GUI (стандартного графічного інтерфейсу для роботи з R) потрібно зайти у папку bin\x64 (або i386, якщо у Вас 32-х розрядна ОС) та запустити файл Rgui.exe.

Вигляд вікна R GUI зображено нижче:

|  |
| --- |
| Рис. 3.5. Вигляд головного вікна RGui |

|  |
| --- |
| Note |
| **GUI** (**G**raphical **U**ser **I**nterface) - набір візуальних компонентів для інтерактивної взаємодії користувача з програмним забезпеченням. |

У вікні R Console можна вводити команди/інструкції R, що будуть виконуватися:

Результати виконання команд зберігаються у памяті програми і можуть бути використані у наступних блоках коду:

|  |
| --- |
| Рис. 3.6. Вигляд консолі для команд RGui |

Середовище R GUI має широкий спектр функцій і дозволяє написати будь-якого рівня складності проекти на R, проте він є лише базовою графічною обгорткою для R. Розглянемо інші зручніші середовища для написання R-коду.

## 3.2 Поняття робочого простору

У процесі виконання коду створені об’єкти/змінні та функції зберігаються у поточній сесії. У R є можливість переглянути список збережних елментів, видалити усі або окремі, зберегти стан поточної сесії диск та завантажити його пізніше, щоб не проходти усі етапи виконання коду повторно *(інколи дуже складний код може виконувати досить довго і збереження проміжних результатів може бути хорошим рішенням)*.

Для прикладу створимо дві змінні var1, var2 та виведемо на консоль їх значення:

var1 <- 10  
var2 <- sqrt(15)  
var1

[1] 10

var2

[1] 3.872983

Для того аби переглянути список змінних у поточній сесії варто скористатися **ls()**:

ls()

[1] "var1" "var2"

Якщо виникає потреба очистити робочий простір і звільними пам’ять використовується команда rm(). Так, щоб очистити усі змінні можна скористатися rm(list = ls()), якщо ж Ви хочете видалити якусь одну/дві змінних, то просто вкажіть імена:

rm(list = c("var1"))  
ls()

[1] "var2"

Таким чином, після виконання коду вище, залишиться лише змінна var2.

Зберігання образу (image) робочого простору на диск здійснюється за допомогою команди save.image("шлях до файлу.RData"), а його зчитування за допомогою load("шлях до файлу.RData").

Код збереження даних:

# Clear workspace  
rm(list = ls())  
  
# Declare variables  
a <- 10  
b <- a + 15  
  
# Save image to file  
save.image("tmp.RData")

Код читання та виведення даних:

# Clear workspace  
rm(list = ls())  
  
# load image to file  
load("tmp.RData")  
  
print(a)

[1] 10

print(b)

[1] 25

У прикладі 2 не створюєть жодного параметра, проте вони збережні у файлі сесії.

Для того аби зберегти та зчитати окремий об’єкт, а не всі елементи сесії у R є спеціальний формат .RDS, який реалізовується методами saveRDS(об'єкт, file="шлях\_файлу.rds") та readRDS(file="шлях\_файлу.rds").

## 3.3 Поняття робочого каталогу

Робота в будь-якому середовищі передбачає зв’язок із поточним каталогом, відносно якого будуються шляхи до файлів. Звичайно можна писати завжди повний шлях до файла, проте такий підхід є досить негнучким і під час перенесення коду між ПК створює чимало проблем розробникам.

Для визначення базового каталогу R в поточній сесії використовують команду getwd(). Якщо Ви користуєтеся RStudio та створили проєкт, то цей каталог буде відповідати повному шляху до папки проєкту:

getwd()

[1] "E:/Repos/OA\_2023/intro-to-r-source"

Для того аби змінити поточний робочий каталог використовують команду setwd(шлях). Після запуску цієї команди функцій getwd() буде вказутивати уже на нову адресу/шлях.

|  |
| --- |
| Important |
| Варто знати та вміти будувати **абсолютні** та **відносні** шляхи до каталогів та файлів, ці знання корисні для роботи з усіма мовами програмування та більшістю ПЗ для роботи з даними. |

Для запису шляху у ОС Windows можна скористатися 2-ма способами:

* / - ***слеш***, записується як один знак;
* \\ - ***бекслеш***, записується як два знаки.

У прикладі нижче обидва шляхи ведуть до тієї ж папки (drive - буква диска):

setwd("drive:/folder1/folder2/")  
setwd("drive:\\folder1\\folder2\\")

Для перегляду інформації про наявні каталоги та файли у поточній робочій папці можна скористатися командою dir() або list.files():

dir()

[1] "\_language-ua.yml" "\_quarto.yml"   
 [3] "00-intro.qmd" "01-what-is-r.qmd"   
 [5] "02-r-history.qmd" "03-r-environment.qmd"   
 [7] "03-r-environment.rmarkdown" "03-r-environment\_files"   
 [9] "04-r-help.qmd" "05-r-packages.qmd"   
[11] "06-r-input-output.qmd" "07-r-variables.qmd"   
[13] "08-r-basic-types.qmd" "09-r-operators.qmd"   
[15] "10-r-conditions-if-else.qmd" "11-r-loops.qmd"   
[17] "12-userful-math.qmd" "12-userful-math\_files"   
[19] "13-r-functions.qmd" "14-r-vectors.qmd"   
[21] "15-r-data.frames.qmd" "16-r-lists.qmd"   
[23] "commands" "css"   
[25] "demo.qmd" "docs"   
[27] "identity" "img"   
[29] "index.log" "index.qmd"   
[31] "index.tex" "index\_files"   
[33] "intro-to-r-source.Rproj" "references.bib"   
[35] "references.qmd" "site\_libs"   
[37] "solved-tasks.qmd" "summary.qmd"   
[39] "tasks.qmd" "tmp.RData"

list.files()

[1] "\_language-ua.yml" "\_quarto.yml"   
 [3] "00-intro.qmd" "01-what-is-r.qmd"   
 [5] "02-r-history.qmd" "03-r-environment.qmd"   
 [7] "03-r-environment.rmarkdown" "03-r-environment\_files"   
 [9] "04-r-help.qmd" "05-r-packages.qmd"   
[11] "06-r-input-output.qmd" "07-r-variables.qmd"   
[13] "08-r-basic-types.qmd" "09-r-operators.qmd"   
[15] "10-r-conditions-if-else.qmd" "11-r-loops.qmd"   
[17] "12-userful-math.qmd" "12-userful-math\_files"   
[19] "13-r-functions.qmd" "14-r-vectors.qmd"   
[21] "15-r-data.frames.qmd" "16-r-lists.qmd"   
[23] "commands" "css"   
[25] "demo.qmd" "docs"   
[27] "identity" "img"   
[29] "index.log" "index.qmd"   
[31] "index.tex" "index\_files"   
[33] "intro-to-r-source.Rproj" "references.bib"   
[35] "references.qmd" "site\_libs"   
[37] "solved-tasks.qmd" "summary.qmd"   
[39] "tasks.qmd" "tmp.RData"

## 3.4 Робота з RStudio

### 3.4.1 Завантаження та інсталяція RStudio Desktop

**RStudio** - це інтегроване середовище розробки для R. Воно включає у себе консоль, підсвічування синтаксису (підказки), прямий запуск коду, інструменти для візуалізації графіків, html-коду, історію виконаних команд, відлагоджування коду, управління робочими просторами, підтримка різних видів розмітки та багато іншого. RStudio має версію з відкритим кодом та комерційну версію для Windows, Linux та Mac, а також веб-версію для серверів на Linux RStudio Server та RStudio Server Pro (*RStudio Official Website* 2021).

|  |
| --- |
| Note |
| IDE (integrated development environment) - комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду. Wikipedia |

Завантажити продукти можна з сайту https://rstudio.com. Щоб знайти середовище, яке ми будемо використовувати під час вивчення курсу варто виконати наступні кроки:

1. У головному меню сайту обрати Products > RStudio.
2. Знаходимо на сторінці кнопку для завантаження програми RStudio Desktop версії Open Source та натискаємо **DOWNLOAD RSTUDIO DESKTOP**:

|  |
| --- |
| Рис. 3.7. Вибір версії RStudio Desktop |

1. Далі обираємо завантаження безкоштовної версії RStudio Desktop з наданого переліку:

|  |
| --- |
| Рис. 3.8. Завантаження RStudio Desktop |

Після завантаження запускаємо інсталятор RStudio. Особливих кроків у цьому процесі немає.

Після запуску IDE RStudio зазвичай складається з 3-х або 4-х блоків: \* Файл, з яким працювали останнім (зліва зверху). \* Консоль для введення коду та виведення результатів (зліва знизу). \* Змінні середовища (Environment) (справа зверху) + Історія команд (History), Зєднання з зовнішніми ресурсами даних, наприклад, бази даних (Connections), навчальна інструкція (Tutorial). \* Файли каталогу або проекту (Files), Інстальовані пакети (Packages), Допомога (Help), Візуалізація результатів (Plots, Viewer).

|  |
| --- |
| Рис. 3.9. Головне вікно RStudio Desktop |

Для першої демонстрації роботи виконаємо у консолі 2 рядки коду:

|  |
| --- |
| Рис. 3.10. Приклад написання коду в RStudio Desktop |

Перший рядок з кодом data <- c(3,7,1,6,3,4,5,4,2) створює у пам’яті колекцію чисел. Зверніть увагу, що у блоці **Environments** відобраюаться усі змінні, що уснують у поточному робочому просторі (про це буде далі).

Другий рядок plot(data, type="l") дозволяє побудувати простий лінійний графік (type="l" - linear, "p" - point, help(plot) для деталей). Графіки, що “промальовуються” як картинки выдображаються 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**.

Якщо перемкнутися на вкладку **History**, то ми побачимо перелік раніше виконаних команд.

|  |
| --- |
| Note |
| Для швидкого “гортання” уже виконаних раніше команд на консолі (*Console*) можна скористатися клавішами Up/Down на клавіатурі: Arrow Keys |

### 3.4.2 Створення першого проекту в RStudio

На відміну від R Gui в RStudio реалізовано концепцію проектів, що дозволяє організувати код та поєднати різні його частини у межах певного рішення.

Створимо наш перший проект.

Для початку оберемо з верхнього меню пункт File > New Project. У вікні вибору способу створення проекту клікаємо New Directory. Такий спосіб передбачає, що жодного файлу проекту поки не існує або ми пізніше туди скопіюємо уже готовий код.

|  |
| --- |
| Рис. 3.11. RStudio Desktop. Новий проєкт |

На наступному кроці обираємо New Project:

|  |
| --- |
| Рис. 3.12. RStudio Desktop. Новий проєкт. Тип проекту |

Після кліку на Create Project буде створено папку за попередньо обраним шляхом. Для запуску проєкту або швидкого перемикання між проектами можна скористатися як пунктами головного меню, так і підменю проектів справа. Також відкрити проект можна запуском файлу \*.Rproj у провіднику Windows.

|  |
| --- |
| Рис. 3.13. RStudio Desktop. Новий проєкт |

Щоб додати новий файл з кодом R потрібно обрати з головного меню File > New file > R Script або скористатися командою Ctrl+Shift+N. Новий файл буде створено з назвою Untitled[X], тому рекомендую одразу його зберегти, наприклад, як TestCode.R

Для першого проекту розвяжемо наступну задачу:

Написати програму, що генерує вектор з 20-ти випадкових чисел у межах [1;5], обчислює середнє та суму чисел, а також виводить гістограму частоти кожного значення (скільки разів дане число повторюється у векторі).

Код для генерації 20-ти випадкових чесел у діапазоні [1;5] матиме наступний вигляд:

vtr <- sample(1:5, 20, replace=TRUE)  
vtr

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

|  |
| --- |
| Important |
| Результати виконання на Вашому ПК будуть іншими, адже **псевдо**генератор випадкових чисел буде брати іншу “точку відліку” для генерування чисел. Рекомендую перегляду функцію set.seed(точка відліку - число). |

Обчислення та виведення на консоль інформації про суму та середнє значення:

vtr\_sum <- sum(vtr)  
vtr\_mean <- mean(vtr)  
  
print(paste0("Sum: ", vtr\_sum))

[1] "Sum: 68"

print(paste0("Mean: ", vtr\_mean))

[1] "Mean: 3.4"

Виведемо гістограму:

hist(vtr, breaks = 5)

|  |
| --- |
| Рис. 3.14. Приклад візуалізації гістрограми в R |

*Примітка. Детальніше про параметри функції hist() можна почитати тут:* [*https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/hist*](https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/hist).

Орієнтовний вигляд вікна RStudio після викоання усіх описаних вище операцій матиме настпуний вигляд:

|  |
| --- |
| Рис. 3.15. RStudio Desktop. Перегляд змінних |

Варто звернути увагу на виділений блок Environment, де можна переглянути усі доступні змінні, що є на даний момент у пам'яті. До цих параметрів можна звертатися у коді чи з консолі у будь, який момент. *Детальну інформацію про робоче середовище розглянуто нижче.*

## 3.5 Робота з Jupyter Notebook

Ноутбуки стали зручним та поширеним інструментом для аналізу даних, а також послідовного викладення матеріалів чи результатів дослідження. Перевагою такого інструменту є перемішування коду, результатів його виконання та іншого текстового наповнення, що дозволяє сформувати “на льоту” готові до читання документи.

Використання ноутбуків у навчальному процесі дозволяє описати не лише теоретичний матеріал, але приклади коду, що будуть виконувати безпосередньо під час ознайомлення з лекцією. Також слухач курсу може відредагувати наявний код та перевірити результати його виконання.

Розгялнемо процес інсталяції та запуску Anaconda (середовище з відкритим кодом для вирішення задач Data Science) та Jupyter Notebook на ПК.

Для встановлення середовища Anaconda потрібно перейти на сайт проекту та завантажити індивідуальну версію продукту: https://www.anaconda.com/products/individual (*Anaconda. The World’s Most Popular Data Science Platform* 2021).

\_\*Примітка. Усі операції у даному курсі виконуються під операційну систему Windows 11 Education Edition\_.

Процес інсталяції середовища Anaconda не відрізняється від стандарного покрокового вставнолення програм у Windows.

Після запуску Anaconda Navigator для початку потрібно створити нове середовище та налаштувати роботу R:

|  |
| --- |
| Рис. 3.16. Anaconda Navigator |

Для початку потрібно перейти на вкладку Environments та натиснути Create:

|  |
| --- |
| Рис. 3.17. Головне вікно Anaconda Navigator |

У вікні, що відкрилося потрібно відмітити [x] вставновлення інструментів для роботи з R:

|  |
| --- |
| Рис. 3.18. Створення нового середовища на основі R у Anaconda Navigator |

Після встановлення R-інструментів оптрібно переключитися на вкладку Home та робочий простір:

|  |
| --- |
| Рис. 3.19. Зміна середовища у Anaconda Navigator |

Після завантаження робочого простору оберіть Launch для запуску Jupyter Notebook з переліку встановлених засобів. Jupyter Notebook буде запущено у браузері за замовчеванням Вашого ПК. Відкрити ноутбук можна обравши потрібний файл, а створити новий у меню справа New > Notebook > R:

|  |
| --- |
| Рис. 3.20. Створення нового Jupyter Notebook |

## 3.6 Огляд додаткових IDE та сервісів для роботи з R

Окрім середовищ описаних вище існує ряд досить цікавих інструментів, що роблять досить зручною роботу з R-кодом. Розглянемо ці інструменти.

**Visual Studio Code** - безкоштовний редактор коду від Microsoft, орієнтовний на велику кількість мов програмування та фреймворків (*Visual Studio Code* 2021). Серед інших іструментів у VS Code доступні також розширення для роботи з R:

|  |
| --- |
| Рис. 3.21. Вікно середовища Visual Studio Code |

**Visual Studio Community Edition** - безкоштовне середовище розробки від компаній Microsoft. VS створено з самого початку для розробки під платформу .NET та мови програмування C#, VB.NET, F# тощо, але з часом отримало багато розширень, що дозволяють у тому числі, працювати і з проектами в R (*Visual Studio Community Edition* 2021).

**Google Collab** - онлайн сервіс для роботи з ноутбуками для Data Science від компанії Google (*Google Colaboratory* 2021):

|  |
| --- |
| Рис. 3.22. Вікно середовища Google Collab |

*Примітка. Код у прикладі вище написаний на Python.*

[**kaggle.com**](https://kaggle.com) - сервіс для змагань з Data Science та Machine Learning. Окрім переліку змагань, наборів даних сервіс має досить зручні ноутбуки.

|  |
| --- |
| Рис. 3.23. kaggle.com |

Загалом сервісів та середовищ для розробки в R існує досить багато і їх кількість зростає, але це не впливає на принципи написання коду та роботу з даними.

# 4. Документація та допомога

Юрій Клебан  
2022-11-12

Для отримання швидкої довідки в R варто скористатися функціє help(назва\_об'єкта\_або\_функції) або ?назва\_об'єкта\_або\_функції:

help(intersect)

Якщо є потреба отримати інформацію про пакет скористайтеся:

help(package = "stats")

|  |
| --- |
| Note |
| Детальну документацію по роботі з пакетами Ви можете знайти тут:   * <https://cran.r-project.org/web/packages/available_packages_by_name.html> * <https://www.rdocumentation.org/> * <https://rdrr.io/r/> |

Розглянемо для прикладу візуалізацію інформації про функцію mean(), що використовується для знаходження середнього значення:

help(mean)

|  |
| --- |
| Рис. 4.1. Скріншот інформації про функцію mean() з пакету base(R Core Team (2020)) |

Отримана інформація дозволить визначити як правильно викликати функцію, які параметри вона приймає та що вони означають. Наявність прикладів виклику функції такод полегшить роботу з написання коду.

# 5. Робота з пакетами

Юрій Клебан  
2022-11-12

Своєю популярністю R завдячує, у тому числі, і можливості швидко реалізувати досить складні дослідження за допомогою наборів уже готових функції. Такі функції обєднуються у пакети та публікуються вченими, досліджниками та розробниками зі всього світу.

**Пакети в R** - організовані набори методів та класів для виконання вузького набору задач під час програмування на R. Вони містять як функції так і опис способів їх використання, а чтакож дані для відтворення прикладів коду.

Пакети можуть бути завантажені з офіційного сайту проекту [cran.r-project.org](https://cran.r-project.org/web/packages/available_packages_by_name.html) / (R Core Team 2020) або інших джерел (dev-версії є доступні на github).

Завантаження пакетів у R можна здійснювати як з локального диска, так і з серверів у мережі Інтернет.

Для встановлення пакету використовується команда **install.packages()**:

install.packages("fun")

Для підключення пакету та його використання варто скористатися функцією library():

packageDescription("fun")  
help(package = "fun")

|  |
| --- |
| Note |
| Дуже рекомендую почитати детальніше про пакети у статті на DataCamp: R Packages: A Beginner’s Guide. |

## 5.1 Робота з пакетами в RStudio

Робота з пакетами в RStudio організована досить зручно і дозволяє швидко переглянути інформацію про пакет та функції, які він дозволяє використати.

Для інсталяції та оновлення пакетів можна скористатися меню Tools:

|  |
| --- |
| Рис. 5.1. Меню інсталяції пакетів у RStudio Desktop |

Після вибору Install Packages... відкриється вікно, де можна обрати як джерело інсталяції пакету так і сам пакет з переліку, ввівши кілька перших букв його назви:

|  |
| --- |
| Рис. 5.2. Вибір пакету для інсталяції у RStudio Desktop |

RStudio дозволяє також переглянути інстальовані пакети/бібліотеки, розроблені іншими користувачами та завантажені у пам’ять (“галочка” навпроти назви пакету):

|  |
| --- |
| Рис. 5.3. Перегляд інстальованих пакетів у RStudio Desktop |

Доступ до функцій та інших елементів пакету можна здійснювати використавши запис назва\_пакету::назва\_функції() без підключення бібліотеки за допомогою library():

|  |
| --- |
| Рис. 5.4. Приклад використання пакету fun у RStudio Desktop |

Користувачі можуть не тільки завантажувати існуючі пакети, але і створювати власні та роботи їх доступними для дослідників зі всього світу.

## Завдання для студентів

Встановіть пакет fun, використовуючи функцію install.packages() та перегляньте його документацію за допомогою help().

Для прикладу, спробуйте пройти якийсь з тестів.

Нижче наведено приклад роботи з функцією alzheimer\_test() з пакету fun:

library(fun)  
alzheimer\_test(  
 char1 = c("9", "O", "M", "I", "F", "D"),  
 char2 = c("6", "C", "N", "T", "E", "O"),  
 nr = 10,  
 nc = 30))

# 6. Введення-виведення даних

Юрій Клебан  
2022-11-12

Читання даних з консолі R можна здійснювати за допомогою функцій scan() та readline().

readline() - читає рядок з консолі та записує його в змінну. Розглянемо приклад:

text <- readline(prompt = "Text to ask user input: ") # prompt a text to view

Функція scan() зчитує дані з консолі та зберігає їх у векторі. Приклад використання у RStudio Desktop:

data <- scan() 1: 12 5 3: 2 4:  
> z [1] 12 5 2

## 6.1 Виведення даних

Для виведення даних у RStudio можна просто скористатися назвою змінної або використати функцію print().

x <- 5  
y <- 10  
print(x + y)

[1] 15

Ще однією корисною функцією є cat(). Вона використовується для виведення об’єктів, конкантенації тексту та значень змінних. Приклад:

cat("Some text", 1, "more text")

Some text 1 more text

У цьому посібнику часто використовуються також функції paste() та paste0() для конкантенації тексту та значень змінних перед виведенням. Синтаксис функції paste() настпуний paste(..., sep = " ", collapse = NULL). Параметр sep визначає роздільник між елементами, а collapse - роздільник між рядками. Приклад:

paste("file", "number", "32") # default separator is space " "

[1] "file number 32"

paste("file", "number", "32", sep = "\_") # separator underline "\_"

[1] "file\_number\_32"

Усі параметри перед сепаратором (sep) повинні бути розділені комами.

Функція paste0(…, collapse = NULL) працює так само, але не використовує сепаратор між елементами. Приклад:

print(paste0("file", "number", "32"))

[1] "filenumber32"

# 7. Оголошення та ініціалізація змінних

Юрій Клебан  
2022-11-12

Базовим поняттям практично усіх мов програмування є **змінна**. Змінна дозволяє записати значення або об’єкт та назвати його для подальшого доступу, зміни, видалення по імені.

Наприклад, присвоєння змінній my\_variable значення 10 записується так: my\_variable <- 5 або my\_variable = 5.

Операція надання змінній певного значення у програмуванні називається **присвоєнням**.

|  |
| --- |
| Important |
| Важливо! Зверніть увагу, що присвоєння (<-, =) та рівність (==) це різні поняття. Оператор == здіснює перевірку співпадіння значення двох змінних/об’єктів та повертає результат у вигляді логічного значення TRUE (якщо значення рівні) або FALSE (якщо значення не рівні). |

Знак <- не є часто використовуваним у різних мовах програмування, зазвичай для присвоєння користуються =. Проте в R освновним способом засобом початкової ініціалізації змінних є <-.

*Також у програмуванні на R використовуються оператори присвоєння <<-, ->, ->>. Про них можна прочитати за лыками нижче.*

|  |
| --- |
| Tip |
| Рекомендую почитати про різницю між операторами присвоєння у R <- та = тут:   1. [Why do we use arrow as an assignment operator? (Colin FAY)](https://colinfay.me/r-assignment/). 2. [Difference between assignment operators in R (Ren Kun)](https://renkun.me/2014/01/28/difference-between-assignment-operators-in-r/). 3. [Assignment Operators](https://stat.ethz.ch/R-manual/R-devel/library/base/html/assignOps.html). |

Приклад:

x <- 45  
y <- 10  
z <- x + y # z = 45 + 10  
z

[1] 55

Розберемо приклад, описаний вище:

* У першому рядку оголошується змінна x і їй присвоюється значення 45.
* У другому рядку оголошується змінна y і їй присвоюється значення 10.
* У третьому рядку оголошується змінна z і їй присвоюється значення суми x + y. # у R використовується як коментар коду, текст написаний після нього ігнорується.
* У четвертому рядку відбувається виведення на консоль змінної z.

## 7.1 Правила іменування змінних

Є кілька основних правил іменування змінних у R: 1. Ім’я змінної може складатися з **букв** [a-z, A-z], **цифр** [0-9], **крапки** . та нижнього **підкреслювання** \_. 2. Ім’я змінної повинно починатися з **букви або крапки**. Якщо воно починається з крапки, то наступним символом повинна бути буква. 3. Не можна використовувати зарезервовані ключові слова мови програмування для іменування змінних, наприклад, TRUE/FALSE.

Ім’я змінної не може містити пробіл (space). Якщо є потреба назвати об’єкт кількома словами, то їх зазвичай розділяють підкресленням \_ або крапкою .. Наприклад, змінну можна назвати my\_variable\_name або my.variable.name. Назва myVariableName ([camel case](https://en.wikipedia.org/wiki/Camel_case)) теж буде коректно сприйнята мовою програмування R, проте такий запис тут вживається не часто.

Приклад коректного іменування змінних: total, zminna, Sum, .length\_of\_something, Number123, x\_1.

Приклад неправильного іменування змінних: tot@l, 5x\_1, \_variable, FALSE, .0ne.

# 8. Базові типи даних R

Юрій Клебан  
2022-11-12

Усі мови програмування мають власну типізацію даних з якими працюють. Тип даних - це набір властивостей певних об’єктів та операцій, що можна з ними виконувати. Так, наприклад, з **цілими числами** можна виконувати арифметичні операції додавання, віднімання та інші. Набори символів (простими словами **текст**) зазвичай можуть використовуватися для пошуку у них елементів, редагування (шляхом видалення частини існуючого або додавання нового тексту), склеювання та розділення на частини.

У R, на відміну від строго типізованих мов програмування, тип даних визначається на основі поточного значення елемента і може змінюватися у процесі виконання.

Розгялнемо приклад коду з мови програмування C# (мова родом із C/Java):

int a = 10;  
a = "some text";

Подібний код у C# передбачає створення нової змінної a типу int (integer - ціле число), а потім відбувається присвоєння для a текстового фрагмента (тип string у С#). Такий код не буде запущено і виникне **помилка компіляції**.

Розглянемо приклад коду з R:

a <- 10  
a <- "some text"  
a

[1] "some text"

Такий код виконається і на консоль буде виведено some text, адже у 1 першому рядку було присвоєно ціле число, у другому - текст. Таким чином R має **динамічну типізацію**, що дозволяє у ту ж саму змінну записати значення різних типів. Проте варто пам’ятати, що попереднє значення буде втрачено.

До базових типів даних у R варто віднести:

* Числа з дробовою частиною (decimal numbers), як наприклад, 4.0, 15.214, що називаються **numeric(s)**.
* Натуральні числа (natural numbers), як наприклад, 4, 15, що називаються **integer(s)**.
* Логічні значення (boolean values), тобто TRUE та FALSE (які також можна скорочено записувати T та F), що називаються **logical**.
* Текст або рядки (string values), як наприклад, "Hello", "12 is number", що називаються **character(s)**.

Оголосимо для прикладу три змінні: my\_numeric - число, my\_character - текст, my\_logical - логічне значення.

my\_numeric <- 5  
my\_character <- "universe"  
my\_logical <- FALSE

Замінимо значення my\_character <- "5" та спробуємо знайти суму значень:

my\_character <- "5"  
my\_sum <- my\_numeric + my\_character

У результаті виклання даного коду ми отримаємо помилку, адже значення 5 та "5" є елементами різних типів даних, перевіримо типи за допомогю функції class():

class(5)

[1] "numeric"

class("5")

[1] "character"

Виконання коду class(5) показує нам, що 5є значенням числового типу даних numeric, а class("5") відповідає тексту character, тому арифметична операція додавання між цими значеннями неможлива.

## 8.1 Перевірка та привдення типів даних

У випадку коли тип даних потрібно визначити у процесі виконання програми/коду та перетворити значення використовується приведння типів даних.

*Приведення типів даних* - операція перетворення значення з одного типу даних в інший. Важливо памятати, що не завжди приведення типів даних може бути здійснено. Так, наприклад, значення "5" (character) можна досить просто привести до 5 (numeric), проте "five" не буде зрозумілим для інтерпритатора.

Для перевірки належності елемента до певного типу даних використовують спеціальну функцію is.назва\_типу(значення). Ця функція повертає TRUE, якщо елемент належить даному типу і FALSE, якщо не належить.

Розглянемо приклад:

my\_numeric <- 5  
my\_character <- "five"  
my\_logical <- FALSE  
  
is.numeric(my\_numeric)

[1] TRUE

is.character(my\_numeric)

[1] FALSE

Для перетворення типу даних можна скористатися функцією as.назва\_типу(значення). У результаті виконання функції буде повернуто значення потрібного типу або пусте значення NA, якщо таке приведення не є можливим:

a <- 5  
b <- "10"  
c <- "10, 20"  
as.numeric(b)

[1] 10

as.numeric(c)

Warning: NAs introduced by coercion

[1] NA

Результат виконання функцій можна записувати у змінні і використовувати у наступних обчисленнях:

a <- 5  
b <- "10"  
b <- as.numeric(b)  
a + b

[1] 15

number <- as.integer(54)  
typeof(number)

[1] "integer"

class(number)

[1] "integer"

Повний перелік типів та методів перевірки і приведення їх типів ображений нижче:

Базові типи даних R, методи перевірки та приведення

| Назва типу | Метод перевірки типу | Метод приведення типу |
| --- | --- | --- |
| Array | is.array() | as.array() |
| **Character** | is.character() | as.character() |
| Complex | is.complex() | as.complex() |
| Dataframe | is.data.frame() | as.data.frame() |
| **Double** | is.double() | as.double() |
| Factor | is.factor() | as.factor() |
| List | is.list() | as.list() |
| **Logical** | is.logical() | as.logical() |
| Matrix | is.matrix() | as.matrix() |
| **Numeric** | is.numeric() | as.numeric() |
| Raw | is.raw() | as.raw() |
| Time series (ts) | is.ts() | as.ts() |
| Vector | is.vector() | as.vector() |

## 8.2 Приклади розв’язання завдань

### Завдання 1. Сума введених чисел

Знайти суму введених з клавіатури чисел. Для введення інформації скористатися функцією readline(prompt = "Text").

Розв’язання:

# works on console and notebook / run this code line by line  
a <- readline(prompt = "Enter A:") # read a  
b <- readline(prompt = "Enter B:") # read b  
  
a <- as.numeric(a) # convert character to numeric  
b <- as.numeric(b) # convert character to numeric  
  
sum <- a + b  
sum

Результат виконання:

Enter A: 10  
Enter B: 45  
55

# 9. Арифметичні та логічні оператори

Юрій Клебан  
2022-11-12

R можна використовувати як звичайни калькулятор.

Розглянемо набір звичних арифметичних операторів, що відомі з початкової школи:

* Додавання: +.
* Віднімання: -.
* Ділення: /.
* Множення: \*.

А також більш складні оператори:

* Піднесення до степеня: ^ (вводиться з клавіатури як Shift+6 на ENG-розкладці клавіатури).
* Остача від ділення (ще може називатися “ділення по модулю”): %% (вводиться з клавіатури як Shift+5).
* Ділення націло: %/%.

Розглянемо приклад **додавання** чисел:

5 + 10

[1] 15

5 + 4 + 15

[1] 24

5 + 53 + 343

[1] 401

(5 + 8) + (4 + 9)

[1] 26

|  |
| --- |
| Note |
| *Примітка*. Використання “круглих” дужок у прогрмуванні виразах має пріоритет аналогічний до загальноприйнятих у математиці. |

Розглянемо приклад **віднімання** чисел:

47 - 21

[1] 26

15 - (10 - 25)

[1] 30

|  |
| --- |
| Note |
| *Примітка*. Заміна знаків до/в “дужках” тут працює так само як працювала у школі :) |

Приклади **множення** чисел:

5 \* 3

[1] 15

5 \* (2 + 5)

[1] 35

Приклади **ділення** чисел:

12 / 2

[1] 6

(4 + 7) / 3

[1] 3.666667

**Піднесення до степеня** за допомогю оператора ^ є досить простим. Так, наприклад, 3^2 (у математиці записується як ) дорівнює 9, а 2^3 - це 2\*2\*2 і дорівнює 8.

5^2

[1] 25

(1+3)^3 + 100

[1] 164

**Остача від ділення** дозволяє знайти залишок одного числа від ділення на інше число.

Наприклад, остача від ділення націло 5 на 2 дорівнює 1, бо 2 \* 2 (=4) + 1 = 5

28 %% 7

[1] 0

17 %% 5

[1] 2

*Примітка.* Пробіли між цифрами та операторами можна не лишати, це робиться для зручності візуального сприйняття коду.

**Ділння націло** залишає лише цілу частину від ідленнядвох чисел:

28 %/% 7

[1] 4

## 9.1 Оператори відношення

**Оператори відношення** відповідають за порівнняння двох об’єктів між собою та повертають значення логічного типу TRUE, якщо результат істинний та FALSE, якщо результат хибний.

Перелік операторів відношення:

* Більше або дорівнює >=.
* Менше <.
* Менше або дорівнює <=.
* Дорівнює ==.
* Не дорівнює !=

Для демонстрації принципів роботи операторів відношення оголосимо 3 змінні a, b та c.

a <- 12  
b <- 5  
c <- 7

Розгялнемо кілька прикладів використання описаних вище операторів.

Оператори, що відповідають за первірку на “більше/менше”:

a > b

[1] TRUE

b + c < a

[1] FALSE

b + c <= a

[1] TRUE

Оператори, що відповідають за перевірку на “рівність/нерівність”:

a != b

[1] TRUE

a == b + c

[1] TRUE

b == c

[1] FALSE

## 9.2 Логічні оператори

До логічних операторів у R відносяться:

* **&** - **І** (амперсант, Shift-7) - виконання усіх умов одночасно.
* **|** - **АБО** (вертикальна риска, Shift+\) - виконання однієї із умов.
* **!** - **НЕ** (знак оклику, Shift+1) - заперечення.

|  |
| --- |
| Important |
| Зверніть увагу, що у R існують також і оператори && та ||. Це також логічні оператори **І** та **АБО**. Проте їх відмінність полягає у тому, що “розширені” оператори порівнюють лише перші елементи векторів, у той час як “скорочені” усі елементи:  x <- c(5, 8, 10) y <- c(3, 2, 5)  print(x)  [1] 5 8 10  print(y)  [1] 3 2 5  x > 5 & y < 7  [1] FALSE TRUE TRUE  x > 5 && y < 7  [1] FALSE |

Важливо розуміти відмінності між цими операторами вміти використовувати результи їх роботи. Для початку варто розглянути таблицю істинності:

Таблиця істинності логічних операторів

| A | B | Оператор **І** | Оператор **АБО** | Заперечення A (**не A**) |
| --- | --- | --- | --- | --- |
| FALSE | FALSE | FALSE | FALSE | TRUE |
| FALSE | TRUE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE | FALSE |
| TRUE | TRUE | TRUE | TRUE | FALSE |

Приклади використання логічних операторів зображені нижче.

Оператор **І** (and):

a <- 12  
b <- 7  
  
a > 10 & b < 10

[1] TRUE

a > 10 & b > 10

[1] FALSE

a < 10 & b < 10

[1] FALSE

a < 10 & b > 10

[1] FALSE

Оператор **АБО** (or):

a > 10 & b < 10

[1] TRUE

a > 10 | b > 10

[1] TRUE

a < 10 | b < 10

[1] TRUE

## Приклади виконання завдань

### Завдання 1. Ділення по модулю

Яким буде результат виконання наступних команд?

a <- 10  
b <- 4  
  
a %% b > 0 # what is result

[1] TRUE

Результат:

TRUE

Пояснення: %% , бо остача від ділення на рівна , а .

### Завдання 2. Ділення по модулю

Яким буде результат виконання наступних команд?

Підзавдання 1:

a <- 7  
b <- 12  
c <- a - b %% 2  
  
c

Результат:

7

Підзавдання 2:

c > 0 & b > a

Результат:

TRUE

Підзавдання 3:

c < 0 | b > a

Результат:

TRUE

# 10. Розгалуження

Юрій Клебан  
2022-11-12

Якщо є потреба перевірити у коді програми виконання якоїсь умови та за її результатами перейти до одного із блоків виконання зазвичай використовують оператор if-else. Загальний вигляд формату використання оператора if-else:

if (умова) {  
 #інструкція  
 }  
 else {  
 #інструкція  
 }

Вираз умова повинен повертати результат у вигляді елемента логічного типу. Тобто, умова може бути або true, або false.

Якщо умова рівна true, то виконається інструкція (або код) записана у дужках після if. Якщо ж умова рівна false, то виконається інструкція записана у дужках після else.

|  |
| --- |
| Note |
| Символ **#** перед кодом означає коментар, тобто код, який не бачитиме інтепретатор мови програмування. |

Якщо після if або else тільки одна інструкція (можна ототожнити з один рядком коду), то «фігурні» дужки {} не є обов’язковими.

Конструкції if-else можуть вкладеними одна в одну без обмежень по глибині. Проте дуже глибокої вкладеності варто уникати, адже це ускладнює читання та розуміння коду програми.

if (умова1)  
{  
 if (умова2)  
 {  
 #інструкція  
 }  
}

|  |
| --- |
| Note |
| if є обов’язковим і повинно існувати у всіх подібних конструкціях, у той же час else – необов’язковий. |

Також конструкція if-else може мати структуру дерева із багатьма розгалуженнями. Така структура зазвичай записується як if-else-if:

if (умова1)  
 { #код }  
 else if (умова2)  
 { #код }  
 else if (умова3)  
 { #код }  
 else { #код }

Вирази обчислюються зверху-вниз. Якщо один із виразів (умова) отримає істинний результат (true), то програмний код, пов’язаний із цією гілкою розгалуження виконається, а всі інші гілки будуть пропущені. Якщо ж не виконається жодна з умов, то буде запущено код всередині останньої інструкції else (у цьому випадку else теж не обов’язковий).

Розгялнемо приклади використання конструкції if-else. Код нижче дозволяє перевірити, яке з двох цілих чисел є більшим:

a <- 10  
b <- 5  
  
if(a > b) {  
 print("A > B")  
} else if (b > a){  
 print("B > A")  
} else {  
 print("A equals B")  
}

[1] "A > B"

## 10.1 Тернарний оператор

Тернарний оператор – це спосіб запису умовного виразу в один рядок. Він використовується, коли потрібно присвоїти значення змінній в залежності від умови. Синтаксис тернарного оператора у R відрізняється від інших мов програмування (класично це умова ? вираз\_якщо\_true : вираз\_якщо\_false).

Оператор розгалуження if-else у R може повертати та записувати значення у змінну:

value <- 10  
x <- if(value %% 2 == 0) "even" else "odd"  
x

[1] "even"

Проте сильною стороною R є саме робота з векторами. Тому, якщо потрібно виконати умовну перевірку для кожного елемента вектора, то краще використати функцію ifelse(). Синтаксис виклику:

ifelse(condition, a, b)

де condition – вектор умов, a – вектор значень, які будуть повернуті, якщо умова condition істинна, b – вектор значень, які будуть повернуті, якщо умова condition не істинна.

Розглянемо приклад. Потрібно створити вектор, який містить значення 1 для значень >= 0 та 0 для значень < 0 на основі наявного вектора x:

x <- c(-6, 2, -3, 4, 5)  
 binary\_vector <- ifelse(x > 0, 1, 0)  
 print(binary\_vector)

[1] 0 1 0 1 1

Також оператор ifelse можна використовувати і для заміни значень вектора. Наприклад, потрібно замінити всі значення Yes/No на 1/0:

x <- c("Yes", "No", "Yes", "Yes", "No")  
print(x)

[1] "Yes" "No" "Yes" "Yes" "No"

x <- ifelse(x == "Yes", 1, 0)  
print(x)

[1] 1 0 1 1 0

## Приклади розв’язання завдань

### Завдання 1. Парні/непарні числа

Напишіть програму, що виводить консоль інформацію про те число парне чи ні. Значення потрібно потрібно також перевірити і вивести у разі його введення "ZERO".

Розв’язання

a <- readline(prompt = "Enter A:") # read a  
a <- as.numeric(a)   
remainder <- a %% 2  
if(a == 0) {  
 print("ZERO")   
} else if(remainder == 0) {  
 print("EVEN")  
} else if (remainder == 1) {  
 print("ODD")  
}

Результат виконання коду:

Enter A: 2  
[1] "EVEN"

# 11. Цикли у R

Юрій Клебан  
2022-11-12

Цикли у програмуванні використовуються для виконання повторюваних операцій. В R є три типи циклів: for(“R for Loop,” n.d.), while(“R for Loop,” n.d.) та repeat(sabir, n.d.).

Цикл for у R використовується для повторюваних операцій над елементами вектора. Ми раніше уже використовували вектори для фнших прикладів:

x <- c(2, 4, 5, 7, 9, 11, 14)  
x

[1] 2 4 5 7 9 11 14

Синтаксис циклу for виглядає наступним чином:

for (value in sequence) {   
 code  
}

Де value - це змінна, яка буде використовуватись для доступу до елементів вектора sequence. code - це код, який буде виконуватись для кожного елемента вектора sequence.

Для прикладу, давайте знайдемо кількість непарних елементів вектора x:

# створюємо змінну count та присвоюємо їй значення 0  
even\_count <- 0   
# створюємо вектор x  
x <- c(2, 4, 5, 7, 9, 11, 14)   
  
for (i in x) {  
 # якщо елемент i не парний  
 if (i %% 2 != 0) {   
 # збільшуємо значення even\_count на 1   
 even\_count <- even\_count + 1   
 # виводимо значення i та even\_count  
 print(paste("i: ", i, " | even\_count: ", even\_count))   
 }  
}

[1] "i: 5 | even\_count: 1"  
[1] "i: 7 | even\_count: 2"  
[1] "i: 9 | even\_count: 3"  
[1] "i: 11 | even\_count: 4"

print(even\_count)

[1] 4

У прикладі вище, цикл for виконався 7 разів, бо саме стльіки значень у векторі x. На кожній ітерації циклу змінні i отримує нове значення з вектора x.

Також цикл for можна використовувати для роботи з індексами елементів вектора (або іншої колекції). Розвязати попередній приклад можна також так:

# створюємо змінну count та присвоюємо їй значення 0  
even\_count <- 0   
# створюємо вектор x  
x <- c(2, 4, 5, 7, 9, 11, 14)   
  
for (i in 1:length(x)) {  
 # якщо елемент \*\*x[i]\*\* не парний  
 if (x[i] %% 2 != 0) {   
 # збільшуємо значення even\_count на 1   
 even\_count <- even\_count + 1   
 # виводимо значення i та even\_count  
 print(paste("i: ", i, " | even\_count: ", even\_count))   
 }  
}

[1] "i: 3 | even\_count: 1"  
[1] "i: 4 | even\_count: 2"  
[1] "i: 5 | even\_count: 3"  
[1] "i: 6 | even\_count: 4"

print(even\_count)

[1] 4

|  |
| --- |
| Tip |
| Функція length() повертає довжину вектора/послідовності. |

## 11.1 Цикл while

Цикл while виконується поки вказана умова є істинною, а синтаксис виглядає наступним чином:

while (condition) {  
 code  
}

де condition - це умова, яка повинна бути істинною, а code - це код, який буде виконуватись поки умова condition є істинною.

Наприклад, виведемо на консоль усі парні числа від 1 до 10:

i <- 1  
  
while(i <= 10) {  
 # збільшуємо значення i на 1 на кожній ітерації циклу  
 i <- i + 1   
 # перевіряємо i на парність  
 if (i %% 2 == 0) {  
 print(i)  
 }  
}

[1] 2  
[1] 4  
[1] 6  
[1] 8  
[1] 10

Використовуючи цикл while Ви моежете виконувати

## 11.2 Команди break та next

Команди break та next дозволяють виконувати певні дії в циклі.

Так, break зупиняє виконання поточного циклу, а next переходить на виконання наступної ітерації циклу, на виконуючи код після неї.

Розглянемо наступний приклад: потрібно вивести на консоль усі парні числа з поданого вектора x <- c(4, 6, 7, 1, -4, 0, 6, -8), але при цьому пропускати від’ємні числа, а при знаходженні 0 зупинити виконання циклу.

x <- c(4, 6, 7, 1, -4, 0, 6, -8)  
  
for(i in x) {  
 # якщо i < 0, то переходимо на наступну ітерацію циклу  
 if(i < 0)   
 next  
 # якщо i == 0, то зупиняємо виконання циклу та виводимо повідомлення  
 if(i == 0) {  
 print("ZERO!!!")  
 break  
 }  
 # якщо i парне, то виводимо його на консоль  
 if(i %% 2 == 0)   
 print(i)  
}

[1] 4  
[1] 6  
[1] "ZERO!!!"

## 11.3 Цикл repeat

Цикл repeat на відміну від for та while зупиняється лише коли виконано команду break та не має жодних умов виходу з нього. Синтаксис виглядає наступним чином:

repeat {  
 code  
 if(condition) {  
 break  
 }  
}

Приклад використання циклу repeat:

i <- 1  
repeat {  
 print(i)  
 i <- i + 1  
 if(i > 10) {  
 break  
 }  
}

[1] 1  
[1] 2  
[1] 3  
[1] 4  
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10

У прикладі вище ми вивели на консоль усі числа від 1 до 10.

# 12. Корисні математичні функції

Юрій Клебан  
2022-11-12

Функції у програмуванні мають ім'я/назву та () (круглі дужки). Виклик функції зазвичай має вигляд function\_name(params).

У цьому розділі описано роботу функцій: round(), ceiling(), floor(), trunc(), signif().

Як ми знаємо з математики, що заокруглення чисел буває “вверх”, “вниз” або відносно деякого значення, зазвичай пов’язаного із цифрою 5 (3.6 заокруглюємо до цілого як , а як , ввжаючи межею.

|  |
| --- |
| Warning |
| Увага! Заокрулення чисел у програмуванні може призводити до помилок у результатах обчислень. Для задач бізнесу, фінансів або технічних процесів мінімальні відхилення можуть призводити до викривлених результатів або збоїв у системах. |

### 12.0.1 Функція round()

Примітка. Тут і надалі функції будуть позначатися як назва() (назва і “круглі” дужки).

Для заокруглення дійних чисел (з дробовою частиною) за правилом <0.5 & >=0.5 (не знаю як називається науково) використовується функція round(x, y), де x - число, y - точність (кількість знаків після коми/крапки). Наприклад:

round(3.557, 2)

[1] 3.56

round(3.241, 2)

[1] 3.24

round(-3.557, 2)

[1] -3.56

round(-3.241, 2)

[1] -3.24

Також можна використати round(x) з одним параметром, тоді заокруглення відбудеться до цілої частини, наприклад:

round(124.345)

[1] 124

### 12.0.2 Функція floor()

Для заокруглення до найближчого меншого цілого числа слід скористатися функцією floor():

floor(3.557)

[1] 3

floor(3.241)

[1] 3

floor(-3.557)

[1] -4

floor(-3.241)

[1] -4

### 12.0.3 Функція ceiling()

Для заокруглення до найближчого більшого цілого числа слід скористатися функцією ceiling():

ceiling(3.557)

[1] 4

ceiling(3.241)

[1] 4

ceiling(-3.557)

[1] -3

ceiling(-3.241)

[1] -3

### 12.0.4 Функція trunc()

Функція trunc() у R використовується для отримання найбільшого цілого числа, яке більше або рівне x. Простими словами це означає, що для чисел менших 0 (x < 0) trunc() працює як celing(), а для чисел більших нуля x > 0, як floor():

x <- 5.34  
print(paste("trunc:", trunc(x), "celing:", ceiling(x), "floor:", floor(x), sep = " "))

[1] "trunc: 5 celing: 6 floor: 5"

x <- x \* -1  
print(paste("trunc:", trunc(x), "celing:", ceiling(x), "floor:", floor(x), sep = " "))

[1] "trunc: -5 celing: -5 floor: -6"

### 12.0.5 Функція signif()

Інколи виникає потреба заокруглити не десяткову частину числа, а десятки, сотні, тисячі і так далі. Розглядемо варіант, коли у нас є велике число і нам потрібно коротко його записати як . Для таких задач можна використати функцію signif(x,y), де x - число, яке потрібно заокруглити до певного порядку, y - порядок заокруглення (рахувати від початку). Наприклад:

big\_number <- 11547741.3  
rounded\_big\_number <- signif(big\_number,3)  
rounded\_big\_number

[1] 11500000

rounded\_big\_number / 1000000

[1] 11.5

## 12.1 Послідовності чисел

Послідовності у програмуванні на R це вектори, які містять набір чисел. Найпростіший спосіб створити послідовність у R це використання оператора “двокрапка” :, наприклад:

seq <- 1:4  
seq

[1] 1 2 3 4

class(seq)

[1] "integer"

### 12.1.1 Функція seq()

Функція seq() використовується для створення послідовностей чисел. Ця функція приймає наступні аргументи:

* from - початок послідовності
* to - кінець послідовності
* by - крок послідовності
* length.out - довжина послідовності

Розглянемо кілька прикладів використання функції seq().

Виведемо числа від до :

seq(1, 10)

[1] 1 2 3 4 5 6 7 8 9 10

Виведемо числа від до з кроком :

seq(0, 10, by = 2.5)

[1] 0.0 2.5 5.0 7.5 10.0

Виведемо та покажемо на графіку 50 чисел від до (крок у такому випадку обчислюється функцією самостійно):

s <- seq(0, 10, length=50)  
print(s)

[1] 0.0000000 0.2040816 0.4081633 0.6122449 0.8163265 1.0204082  
 [7] 1.2244898 1.4285714 1.6326531 1.8367347 2.0408163 2.2448980  
[13] 2.4489796 2.6530612 2.8571429 3.0612245 3.2653061 3.4693878  
[19] 3.6734694 3.8775510 4.0816327 4.2857143 4.4897959 4.6938776  
[25] 4.8979592 5.1020408 5.3061224 5.5102041 5.7142857 5.9183673  
[31] 6.1224490 6.3265306 6.5306122 6.7346939 6.9387755 7.1428571  
[37] 7.3469388 7.5510204 7.7551020 7.9591837 8.1632653 8.3673469  
[43] 8.5714286 8.7755102 8.9795918 9.1836735 9.3877551 9.5918367  
[49] 9.7959184 10.0000000

plot(seq(0, 10), seq(0, 50, length=11), type="l", col="blue")

|  |
| --- |
| Рис. 12.1. Приклад візуалізації послідовності у R |

### 12.1.2 Функція rep()

Функція rep() також використовується для створення послідовностей чисел, що повторюються. Ця функція приймає наступні аргументи:

* x - послідовність, що повторюється
* times - кількість повторень
* each - кількість повторень кожного елемента

Синтаксис виклику:

rep(x = <вектор>,   
 times = <кількість повторень ветора>,   
 each = <кількість повторень кожного елемента>)

Приклади використання та результаи виконна коду подано нижче.

# 123 повторюється 2 рази  
rep(1:3, times = 2)

[1] 1 2 3 1 2 3

# вектор x повторюється 2 рази  
x <- c(5, 2, 7)  
rep(x, times = 2)

[1] 5 2 7 5 2 7

# 1 повторюється 2 рази  
# 2 повторюється 1 раз  
# 3 повторюється 4 рази  
rep(1:3, times=c(2,1,4))

[1] 1 1 2 3 3 3 3

# кожен елемент повторюється 2 рази  
rep(1:3, each=2)

[1] 1 1 2 2 3 3

x <- c(1,3,5)  
rep(x, each = 3, times = 2)

[1] 1 1 1 3 3 3 5 5 5 1 1 1 3 3 3 5 5 5

## 12.2 Генерація псевдовипадкових чисел

R має вбудовані функції для генерації псевдовипадкових чисел: runif(), sample().

### 12.2.1 Функція runif()

runif() генерує послідовність випадкових чисел з рівномірним розподілом на відрізку . Ця функція приймає наступні аргументи:

* n - кількість чисел;
* min - мінімальне значення (за замовчуванням );
* max - максимальне значення (за замовчуванням ).

Наприклад, згенеруємо випадкових значень на відрізку :

runif(5)

[1] 0.52355325 0.28162764 0.08812311 0.32439173 0.74645481

Виведемо графік з випадкових знань на відрізку :

x <- runif(1000)  
plot(x)

|  |
| --- |
| Рис. 12.2. Приклад візуалізації 1000 випадкових знань на відрізку [0, 1] з використанням функції runif() |

### 12.2.2 Функція sample()

Функція sample() дозволяє вибрати числа випадковим чином із деякого набору (вектора).

Синтаксис виклику:

sample(x = <вектор>,   
 size = <кількість випадкових значень>,   
 replace = <TRUE/FALSE, >,   
 prob = <вектор з ймовірностями>)

Припустимо у нас є деякий вектор x:

x <- 10:20  
print(x)

[1] 10 11 12 13 14 15 16 17 18 19 20

Для того аби виьрати одне із значень вектора x випадковим чином, використаємо функцію sample():

sample(x)

[1] 14 16 18 19 20 15 11 13 12 10 17

Якщо ми хочемо вибрати декілька значень вектора x випадковим чином, то використаємо аргумент size:

sample(x, size = 4)

[1] 12 13 20 15

sample(x, size = 4)

[1] 15 13 19 12

*Зверніть увагу, що у такому випадку усі значення завжди будуть різними у межах одного вибору.*

У випадку, якщо потрібно вибрати значення з повтореннями, то використаємо аргумент replace:

sample(x, size = 15, replace = TRUE)

[1] 10 13 13 13 17 17 16 16 19 20 10 16 20 19 10

Якщо потрібно вибрати значення з деякими ймовірностями, то використаємо аргумент prob:

t <- sample(c(1,2), size = 1000, replace = T, prob = c(80, 20))  
hist(t, breaks = 2, col = c('blue', 'green'))

|  |
| --- |
| Рис. 12.3. Приклад візуалізації гістограми у R |

## 12.3 Інші математичні функції та константи R

Окрім описаного вище набору функцій R містить дуже велику кількість реалізованих функцій з різних сфер науки, бізнесу, техніки тощо. Прочитати про них можна з офіційної документації пакетів, у яких вони реалізовані та знайти за допомогою функції help() або ?name.

Далі розглянемо перелік найпоширеніших функцій, що використовуються для розв’язання навчальних задач під час вивчення основ програмування.

Популярні математичні функції в R

| Функція | Призначення, опис |
| --- | --- |
| log(x) | Логарифм числа x за основою e |
| log(x,n) | Логарифм числа *x* за основою n |
| exp(x) | e у степені x |
| sqrt(x) | Корінь квадратний числа x |
| factorial(x) | Факторіал числа x |
| abs(x) | Модуль числа x |

Також у R доступні ряд тригонометричних функцій, які вивчалися у школі і не тільки, серед них cos(x), sin(x), tan(x), а також acos(x), asin(x), atan(x), acosh(x), asinh(x), atanh(x).

Детальніше про кожну з них можна почитати у документації за допомогою кодманди help(function).

## Приклади розв’язання завдань

### Завдання 1. Генерація випадкових чисел

Згенеруйте випадкових чисел у діапазоні . Заокругліть їх до десятих (“2 знаки після коми/крапки”). Побудуйте графік за допомогою функції plot().

Розв’язання

x <- runif(1000)  
x <- round(x, 1)  
plot(x)

|  |
| --- |
| Рис. 12.4. Графік до Завдання 1. Генерація випадкових чисел |

# 13. Створення функцій

Юрій Клебан  
2022-11-12

Усі мови програмування окрім набору вбудованих функцій також дозволяють створювати власні функції, що інколи жє наобхідним для вирішення задачі, а інколи спрощує написаний код та полегшує його тестування.

Усі функції мають кілька спільних характеристик:

* Функція - це блок коду, що виконується при її виклику по імені.
* Функція може приймати аргументи, які використовуються в коді функції.
* Функція може повертати значення, яке може бути використано в коді, де функція була викликана.

Створення фукнцій у R відбувається з використанням ключового слова function, а синтаксис має вигляд:

my\_func <- function() {  
 print("this is my first function")  
}  
  
my\_func()

[1] "this is my first function"

У рядках **1-3** створюється функція my\_func, яка не приймає аргументів, але виконує дію виводу тексту "this is my first function" на екран. У рядку **5** функція викликається.

*Увага. Виконання функції відбувається у момент виклику, а не у момент створення.*

У прикладі вище my\_func є імям функції, а my\_func() - її виликом.

## 13.1 Передача аргументів та повернення значень

Функції також можуть приймати аргументи. Ви можете додати стільки аргументів, скільки вам потрібно, розділяючи їх комами.

Наприклад, створимо функцію для конвретації валюти з гривні у долари:

uah\_to\_usd <- function(uah) {  
 usd\_rate <- 40  
 usd\_amount <- uah / usd\_rate  
 print(usd\_amount)  
}  
  
uah\_to\_usd(200)

[1] 5

У прикладі вище ми вивели значення прямо з функції, проте частіше виникає потреба повернути значення з функції, щоб використати його далі. Для цього використовується ключове слово return. Перепишемо нашу функцію:

uah\_to\_usd <- function(uah) {  
 usd\_rate <- 40  
 usd\_amount <- uah / usd\_rate  
 return(usd\_amount)  
}  
  
total\_usd <- uah\_to\_usd(200)  
print(paste("Total USD:", total\_usd))

[1] "Total USD: 5"

Насправді у R можна не використовувати return, а просто використовувати останнє значення, яке буде обчислено в функції. Однак, це не є хорошою практикою, оскільки це може призвести до помилок, які важко знайти у коді програми:

uah\_to\_usd <- function(uah) {  
 usd\_rate <- 40  
 usd\_amount <- uah / usd\_rate  
 return(usd\_amount)  
}

## 13.2 Аргументи функції за замовчуванням

Розширимо нашу попереню функцію та будемо також передавати курс USD/UAH як параметр. Якщо курс не передано, то використовуємо курс за замовчуванням:

uah\_to\_usd <- function(uah, usd\_rate = 40) {  
 usd\_amount <- uah / usd\_rate  
 return(usd\_amount)  
}  
  
# Буде використано курс за замовчуванням  
uah\_to\_usd(200)

[1] 5

# Буде використано курс 42.5  
uah\_to\_usd(200, 42.5)

[1] 4.705882

## Приклади розв’язання завдань

### Завдання 1. Середнє значення

Напишіть функцію, що обчислює середнє значення чисел у векторі та повертає його. Точність обчислень - знаки після коми.

*Примітка. Під час розвязання задачі не можна користуватися жодними готовими вбудованими функціямию*

Розв’язання

average <- function(x) {  
 count <- 0  
 sum <- 0  
 for(i in x) {  
 count <- count + 1  
 sum <- sum + i  
 }  
 return(round(sum/count, 2))  
}  
  
vector <- c(5, 2, 7, 9, 2, 1)  
  
average(vector)

[1] 4.33

### Завдання 2. Створення вектору із випадковими числами

Написати функцію, що генерує вектор випадкових чисел у вказаному діапазоні. Якщо діапазон користувачем не вказано, то генеруються значення у діапазоні від до . Також варто врахувати, що випадкові числа можуть мати дробову частину. У такому випадку потрібно передати

Розв’язання

randomNumbers <- function(n, min = 0, max = 100, intOnly = TRUE) {  
 if(intOnly) {  
 return(sample(min:max, n, replace = TRUE))  
 } else {  
 return(runif(n, min, max))  
 }  
}  
  
randomNumbers(10, intOnly = F)

[1] 83.293581 78.291728 78.613134 86.771048 79.975703 8.413933 35.002182  
 [8] 70.950235 80.050888 87.926001

randomNumbers(10, 100, 200)

[1] 129 138 167 149 137 128 184 132 194 147

### Завдання 3. Депозитний калькулятор

Напишіть функцію, що обчислює суму депозиту на кінець терміну вкладу. Функція повинна приймати наступні аргументи:

* P - сума депозиту у гривні.
* n - термін депозиту у місяцях.
* r - щомісячна відсоткова ставка за депозитом
* monthly capitaliztion

During caluculations print every month info in style: “Month: 1 / Total: 1010 / 10 EUR” Suppose I deposit 1000 (P) dollars in bank every month for 60 (n) months and bank pays me 1 (rate) percent per month as interest. Hence I can calculate the interest on each deposit as:

The formula used is A = P(1+r/n) ^ n, where ‘A’ represents final amount procured, ‘P’ represents principal, ‘r’ represents annual interest rate, ‘n’ represents the number of times that interest has been compounded, ‘t’ represents the tenure.

# 14. [-] Вектори

Юрій Клебан  
2022-11-12

# 15. [-] Дата-фрейми

Юрій Клебан  
2022-11-12

# 16. [-] Списки

Юрій Клебан  
2022-11-12

# [-] Задачі для самостійної підготовки

Юрій Клебан  
2022-11-01

## Типи даних та змінні

## Послідовності

## Функції

## Читання/запис даних

# [-] Приклади розв’язання задач

Юрій Клебан  
2022-11-01

# Список використаних джерел

*Anaconda. The World’s Most Popular Data Science Platform*. 2021. 206 379 Broadway Ave., Suite 310 New York, NY 10013, USA: Anaconda Inc. <https://www.anaconda.com/>.

Chang, Winston, and Barbara Borges Ribeiro. 2021. *Shinydashboard: Create Dashboards with ’Shiny’*. <https://CRAN.R-project.org/package=shinydashboard>.

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2022. *Shiny: Web Application Framework for r*. <https://CRAN.R-project.org/package=shiny>.

*Google Colaboratory*. 2021. Google LLC. <https://colab.research.google.com/>.

Kuhn, Max. 2022. *Caret: Classification and Regression Training*. <https://CRAN.R-project.org/package=caret>.

Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. 2019. “mlr3: A Modern Object-Oriented Machine Learning Framework in R.” *Journal of Open Source Software*, December. <https://doi.org/10.21105/joss.01903>.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.

“R for Loop.” n.d. *W3Schools*. <https://www.w3schools.com/r/r_for_loop.asp>.

*RStudio Official Website*. 2021. 250 Northern Ave, Boston, MA 02210: RStudio, PBC. <https://rstudio.com/>.

sabir. n.d. “R Repeat Loop.” *R-Bloggers.com*. <https://www.r-bloggers.com/2022/07/r-repeat-loop/>.

Sievert, Carson, Richard Iannone, JJ Allaire, and Barbara Borges. 2022. *Flexdashboard: R Markdown Format for Flexible Dashboards*. <https://CRAN.R-project.org/package=flexdashboard>.

*Visual Studio Code*. 2021. Microsoft. <https://code.visualstudio.com/>.

*Visual Studio Community Edition*. 2021. Microsoft. <https://visualstudio.microsoft.com/>.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. <https://yihui.org/knitr/>.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. <https://bookdown.org/yihui/rmarkdown-cookbook>.