

Вступ до R

Юрій Клебан

2021-04-22

Contents

Загальна інформація	5
1 Вступ до курсу	7
План	7
1.1 Що таке R?	7
1.2 Історія створення R	8
1.3 Основи роботи з R	9
1.4 Основи роботи з пакетами в R	36
2 Базові конструкції мови R	41
План	41
2.1 Введення-виведення даних	41
2.2 Оголошення та ініціалізація змінних	41
2.3 Базові типи даних	43
2.4 Оператори	47
2.5 Корисні математичні функції	52
2.6 Введення-виведення даних	56
3 Основи роботи з даними в R	57
План	57
3.1 Набори даних	57
3.2 Вектори (vectors)	57

4 Приклади задач та їх розв'язки	61
4.1 Задачі	61
4.2 Рішення задач	63
Джерела	69

Загальна інформація

Увага. Курс у процесі розробки. Матеріали додаватимуться по мірі їх написання та рецензування.

ВСТУП ДО R

Базовий курс для тих, хто тільки
розпочинає вивчення аналізу даних
та програмування



Юрій Клебан, 2021



Курс створено у межах проекту “Підготовка, обробка та ефективне використання даних для наукових досліджень (на основі R)”, що підтримує Європейський союз за програмою House of Europe.

Chapter 1

Вступ до курсу

План

- Що таке R?
 - Історія створення R
 - Основи роботи з R
 - R Project
 - * Завантаження та інсталяція R
 - * Перший запуск R GUI
 - * Поняття робочого простору
 - * Поняття робочого каталогу
 - * Допомога (help/?)
 - Робота з R Studio
 - * Завантаження та інсталяція RStudio Desktop
 - * Створення першого проекту в RStudio
 - Робота з Jupyter Notebook
 - Огляд додаткових IDE та сервісів для роботи з R
 - Основи роботи з пакетами в R
 - Команди для роботи з пакетами
 - Робота з пакетами в RStudio
-

1.1 Що таке R?

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 вирішення 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 різноманітних 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* та *mlr*.
- **Автоматизація.** Написаний код та проекти 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*, *shinydashboard*, *flexdashboard*, *rmarkdown*, *knitr* тощо), результати виконаної роботи можна “оживити” або сформувати “на льоту” готові до презентації документи.

1.2 Історія створення R

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 немалою кількістю 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 світу 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") (А.Б. Шипунов, Е.М. Балдин, П.А. Волкова, А.И. Коробейников, С.А. Назарова, С.В. Петров, В.Г. Суфьянов., 2012). За задумом авторів (Robert Gentleman 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'явилися ресурси, в тому числі зручна система 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 IT темою науки про дані, машинним навчанням, аналітикою для бізнесу, охорони здоров'я тощо.

1.3 Основи роботи з R

1.3.1 R Project

R є безкоштовним програмним забезпеченням, що розповсюджується за умовами GNU General Public License. Код, написаний на R компілюється та запускається на різних платформах: UNIX, Windows, MacOS (R Core Team, 2020).

1.3.1.1 Завантаження та інсталяція R

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 воєкрисовуємо Windows, проте на синтаксис мови програмування та процес написання коду це не впливає:

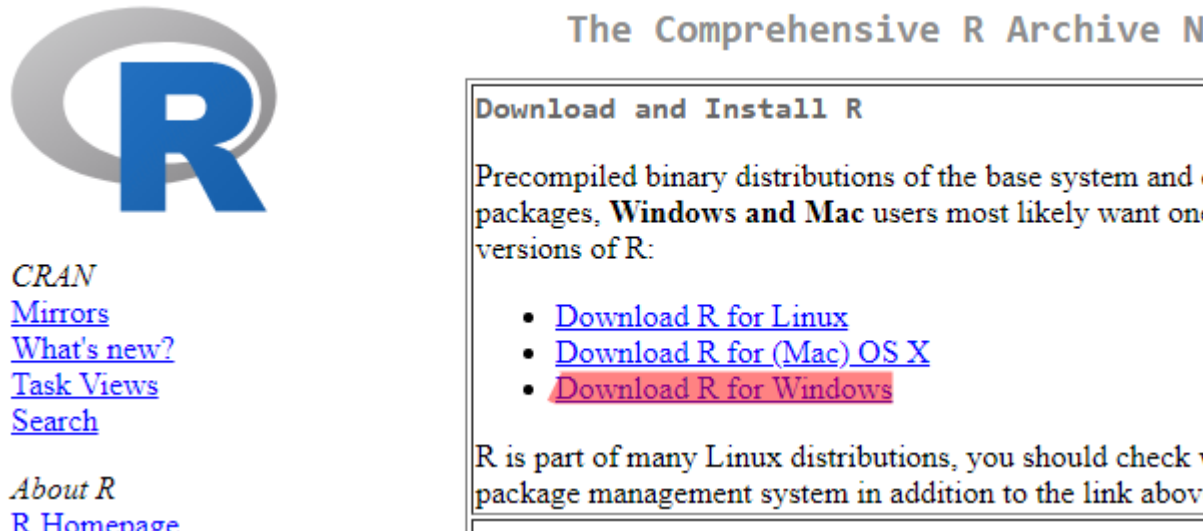


Figure 1.1: Завантаження R. Вибір ОС

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

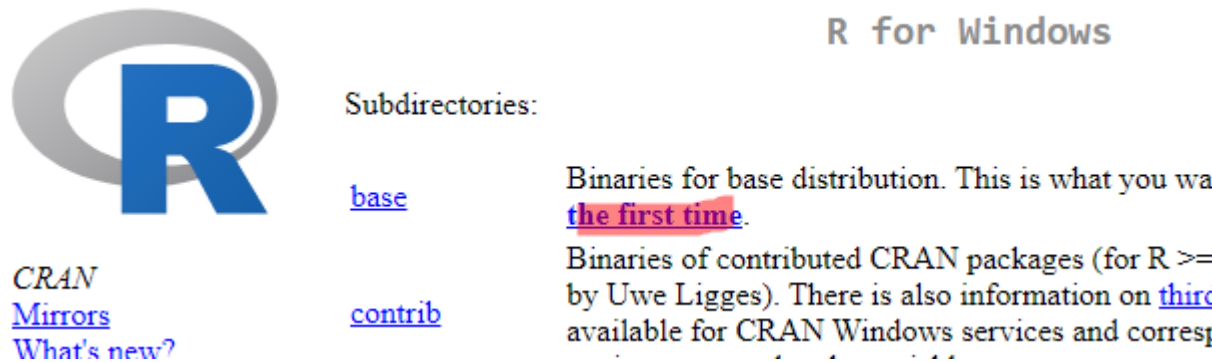


Figure 1.2: Завантаження R. Перша інсталяція

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

Після завантаження файлу інсталяції потрібно його запустити. Зазвичай завантажений файл можна побачити у лівому нижньому кутку браузера або у розділі "Завантаження" Вашого браузера. Наприклад, у браузері Google

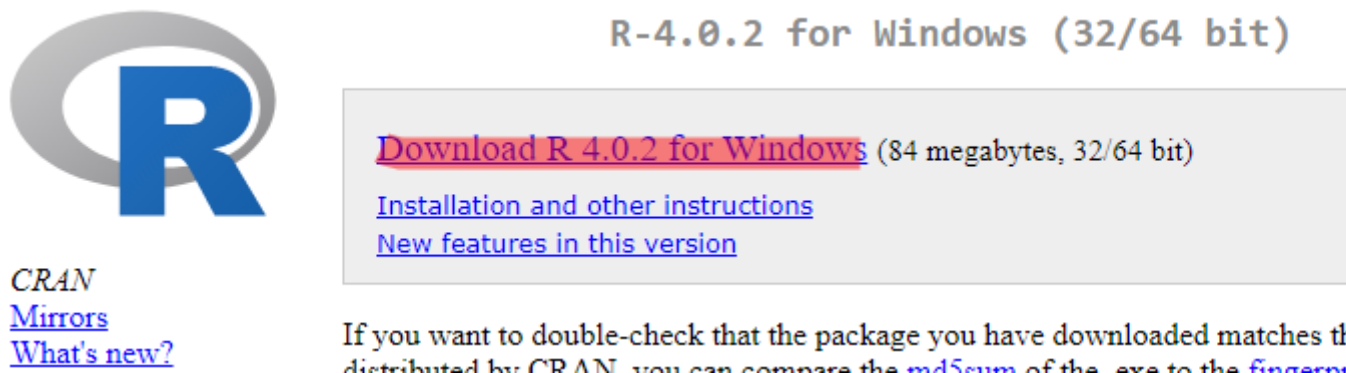


Figure 1.3: Завантаження R. Завантаження версії для ОС

Chrome знайти цей пункт меню так:

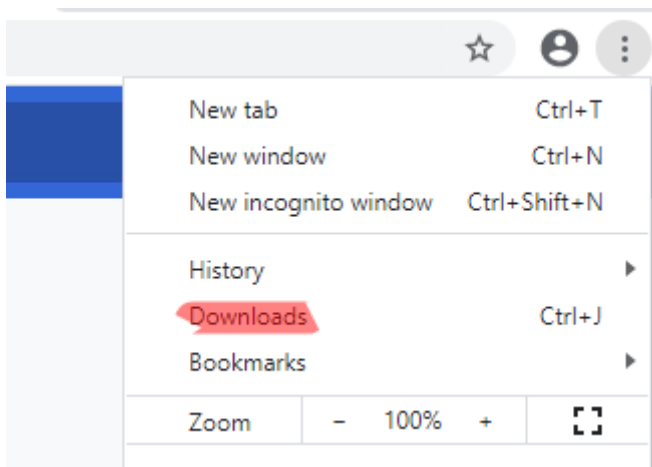


Figure 1.4: Завантаження R. Розділ "Завантаження" у Google Chrome

Процес інсталяції ПЗ не відрізняється від інших програм і детального опису не потребує. Основним тут є запам'ятати шлях встановлення проекту або "відмітити галочками" пункти щодо публікації на *Робочий стіл* чи у меню швидкого доступу ярликів для того, щоб знайти файли запуску.

1.3.1.2 Перший запуск R GUI

За замовчуванням під час інсталяції 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:

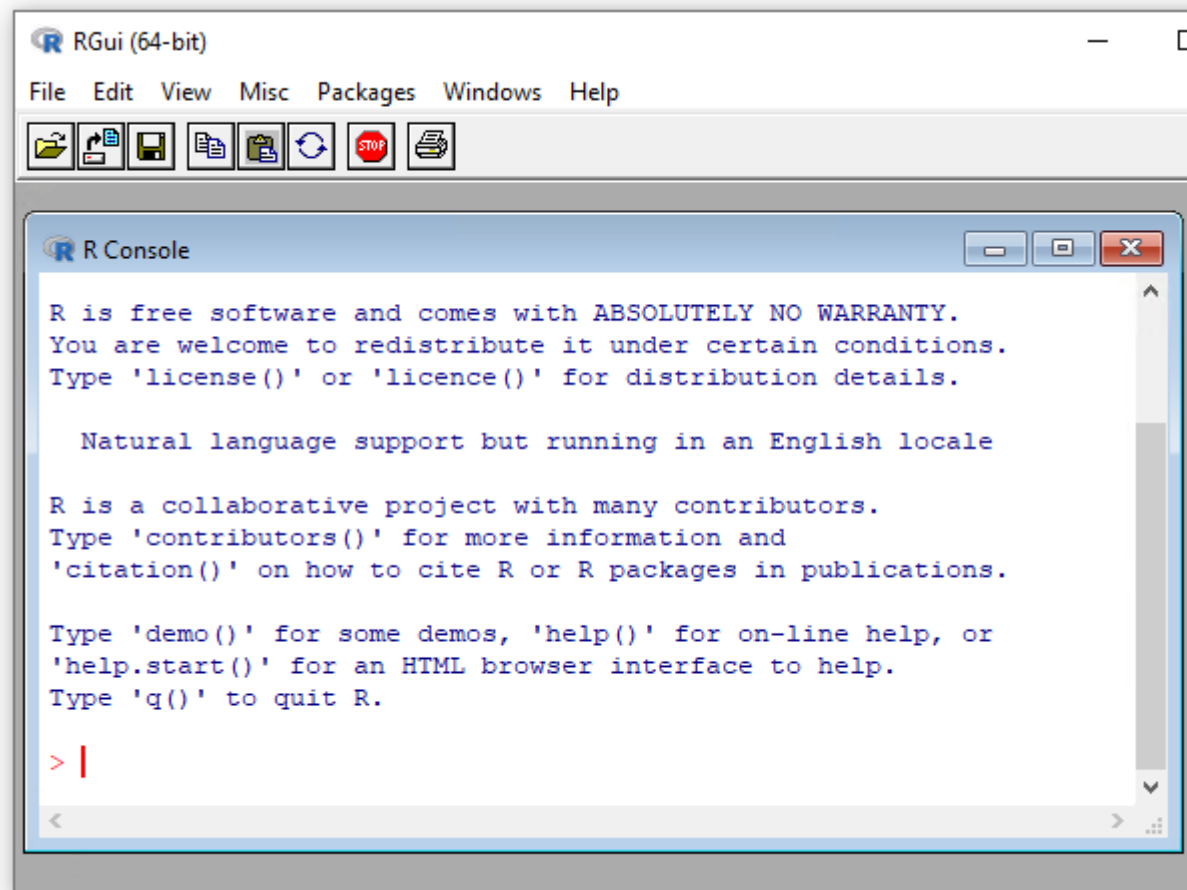


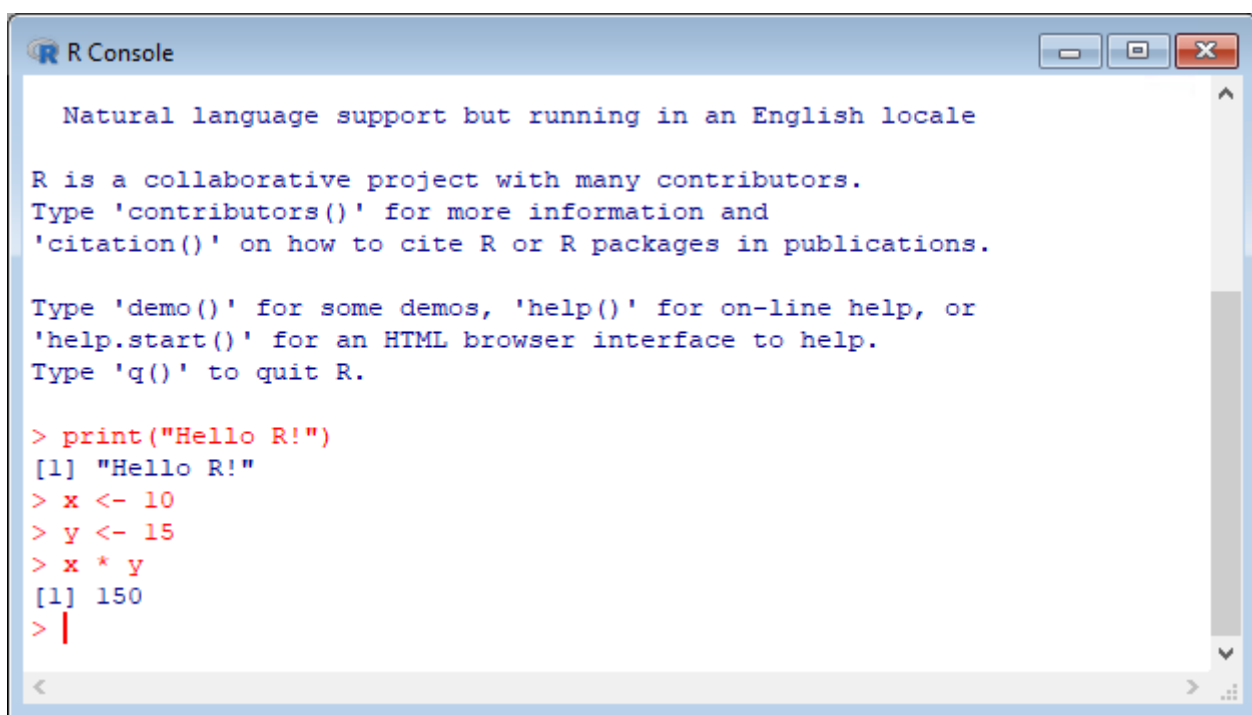
Figure 1.5: Вигляд головного вікна RGui

GUI (Graphical User Interface) - 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 пам'яті програми і можуть бути використані у наступних блоках коду:

Середовище R GUI має широкий спектр функцій і дозволяє написати будь-



The image shows a screenshot of the R Console window. The window has a title bar with the R logo and the text "R Console". Inside the window, there is a text area with the following content:

```
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> print("Hello R!")
[1] "Hello R!"
> x <- 10
> y <- 15
> x * y
[1] 150
> |
```

The text is displayed in a monospaced font. The prompt character ">" is red, and the output is black. The user has entered several commands, and the console has executed them, showing the results. The window also has standard Windows window controls (minimize, maximize, close) in the top right corner.

Figure 1.6: Вигляд консолі для команд RGui

якого рівня складності проекти на R, проте він є лише базовою графічною оболонкою для R. Розглянемо інші зручніші середовища для написання R-коду.

1.3.1.3 Поняття робочого простору

У процесі виконання коду створені об'єкти/змінні та функції зберігаються у поточній сесії. У R є можливість переглянути список збережених елементів, видалити усі або окремі, зберегти стан поточної сесії на диск та завантажити його пізніше, щоб не проходити 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 звільнити 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(" _ .RData")`, а його зчитування за допомогою `load(" _ .RData")`:

```
# Clear workspace
rm(list = ls())

# declare data
a <- 10
b <- a + 15

# Save image to file
save.image("tmp.RData")
```

```
# 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(' ', file=" _ .rds")` та `readRDS(file=" _ .rds")`.

1.3.1.4 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 проект, то цей каталог буде відповідати повному шляху до папки проекту:

```
getwd()
```

```
## [1] "E:/Repos/YuRa/r-intro"
```

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

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] "_bookdown.yml"          "_bookdown_files"       "_output.yml"
## [4] "01-chapter1.Rmd"        "01-chapter1_files"     "01-intro_files"
## [7] "02-chapter2.Rmd"        "02-chapter2_files"     "03-chapter3.Rmd"
## [10] "06-chapter6.Rmd"        "07-references.Rmd"      "book.bib"
## [13] "css"                    "favicon.ico"           "images"
## [16] "inc"                    "index.md"              "index.Rmd"
## [19] "index.utf8.md"          "packages.bib"          "preamble.tex"
## [22] "r-intro.log"            "r-intro.rds"           "README.md"
## [25] "render_commands"        "render581819cb22c1.rds" "RIntro.Rproj"
## [28] "tmp.RData"
```



```
list.files()
```

```
## [1] "_bookdown.yml"          "_bookdown_files"       "_output.yml"
## [4] "01-chapter1.Rmd"        "01-chapter1_files"     "01-intro_files"
## [7] "02-chapter2.Rmd"        "02-chapter2_files"     "03-chapter3.Rmd"
## [10] "06-chapter6.Rmd"        "07-references.Rmd"      "book.bib"
## [13] "css"                    "favicon.ico"            "images"
## [16] "inc"                    "index.md"               "index.Rmd"
## [19] "index.utf8.md"          "packages.bib"           "preamble.tex"
## [22] "r-intro.log"            "r-intro.rds"            "README.md"
## [25] "render_commands"        "render581819cb22c1.rds" "RIntro.Rproj"
## [28] "tmp.RData"
```

1.3.1.5 Допомога (help/?)

Для отримання швидкої довідки в R варто скористатися функції `help(_ ' _ _)` або `? _ ' _ _` :

```
# Get help for intersect() function
help(intersect)
```

Якщо є потреба отримати інформацію про пакет скористайтеся:

```
help(package = "stats")
```

1.3.2 Робота з R Studio

1.3.2.1 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 (R-s, 2021).

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

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), З'єднання з зовнішніми ресурсами даних, наприклад, бази даних (Connections), навчальна інструкція (Tutorial). * Файли каталогу або проекту (Files), Інсталювані пакети (Packages), Допомога (Help), Візуалізація результатів (Plots, Viewer).

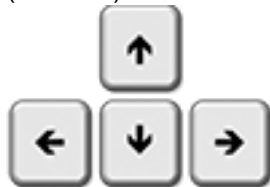
Для першої демонстрації роботи виконаємо у консолі 2 рядки коду:

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** відображаються 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 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**.

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

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



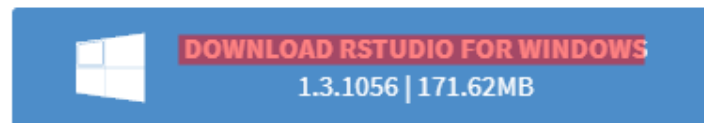


	Open Source Edition	RStudio Desktop Pro
Overview	<ul style="list-style-type: none"> • Access RStudio locally • Syntax highlighting, code completion, and smart indentation • Execute R code directly from the source editor • Quickly jump to function definitions • Easily manage multiple working directories using projects • Integrated R help and documentation • Interactive debugger to diagnose and fix errors quickly • Extensive package development tools 	<p>All of the features of open source</p> <ul style="list-style-type: none"> • A commercial license for organizations not able to use open source software • Access to priority support • RStudio Professional Driver • Connect directly to your R Server Pro instance remotely
Support	Community forums only	<ul style="list-style-type: none"> • Priority Email Support • 8 hour response during business hours (ET)
License	AGPL v3	RStudio License Agreement
Pricing	Free	\$995/year
	DOWNLOAD RSTUDIO DESKTOP	DOWNLOAD FREE RSTUDIO DESKTOP PRO TRIAL

Figure 1.7: Вибір версії RStudio Desktop

RStudio Desktop 1.3.1056 - [Release Notes](#)

1. Install R. RStudio requires [R 3.0.1+](#).
2. Download RStudio Desktop. Recommended for your system:



Requires Windows 10/8/7 (64-bit)

All Installers

Linux users may need to [import RStudio's public code-signing key](#) prior to installation due to security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use

OS	Download
Windows 10/8/7	RStudio-1.3.1056.exe
macOS 10.13+	RStudio-1.3.1056.dmg
Ubuntu 16	rstudio-1.3.1056-amd64.deb

Figure 1.8: Завантаження RStudio Desktop

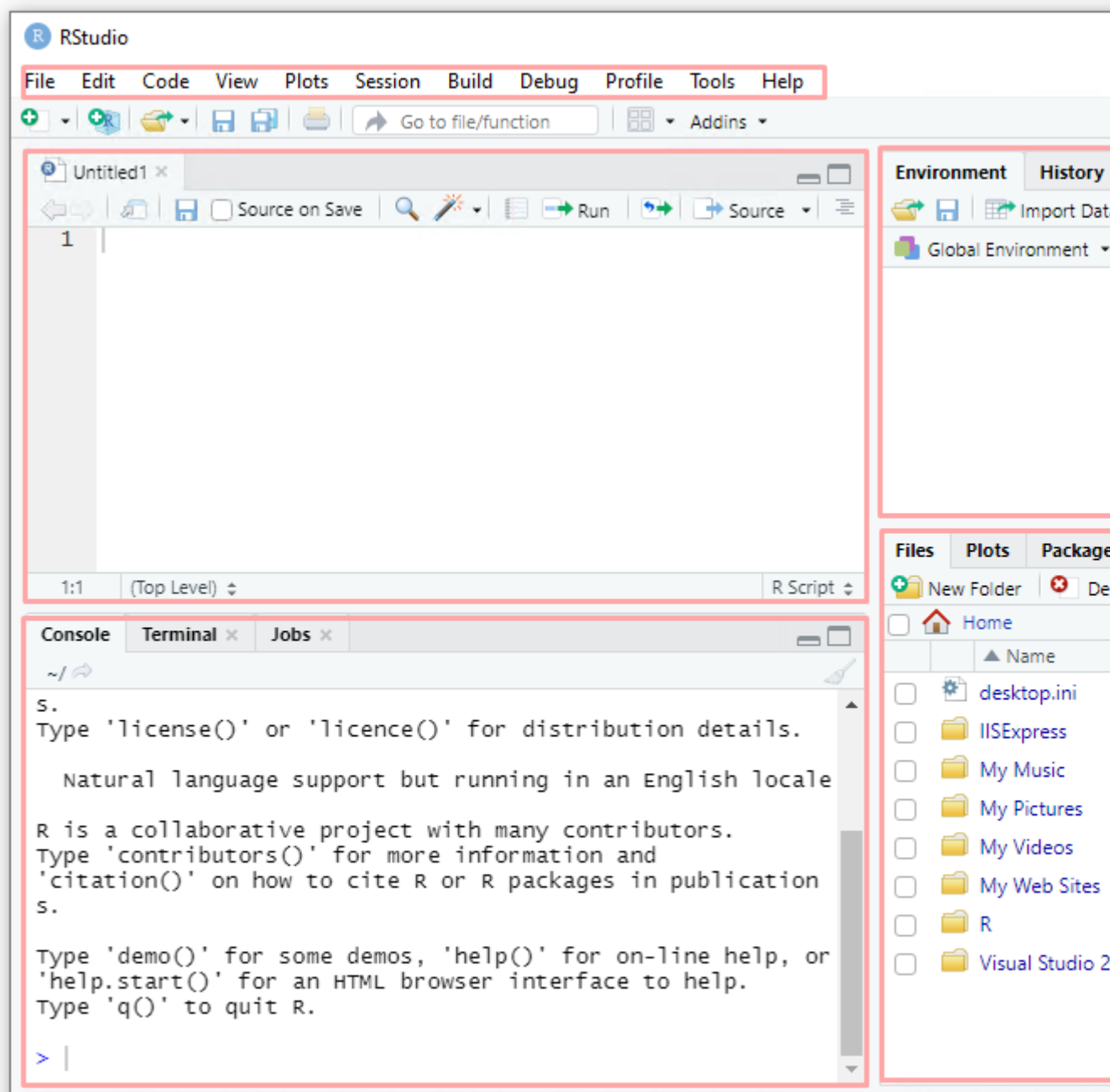


Figure 1.9: Головне вікно RStudio Desktop

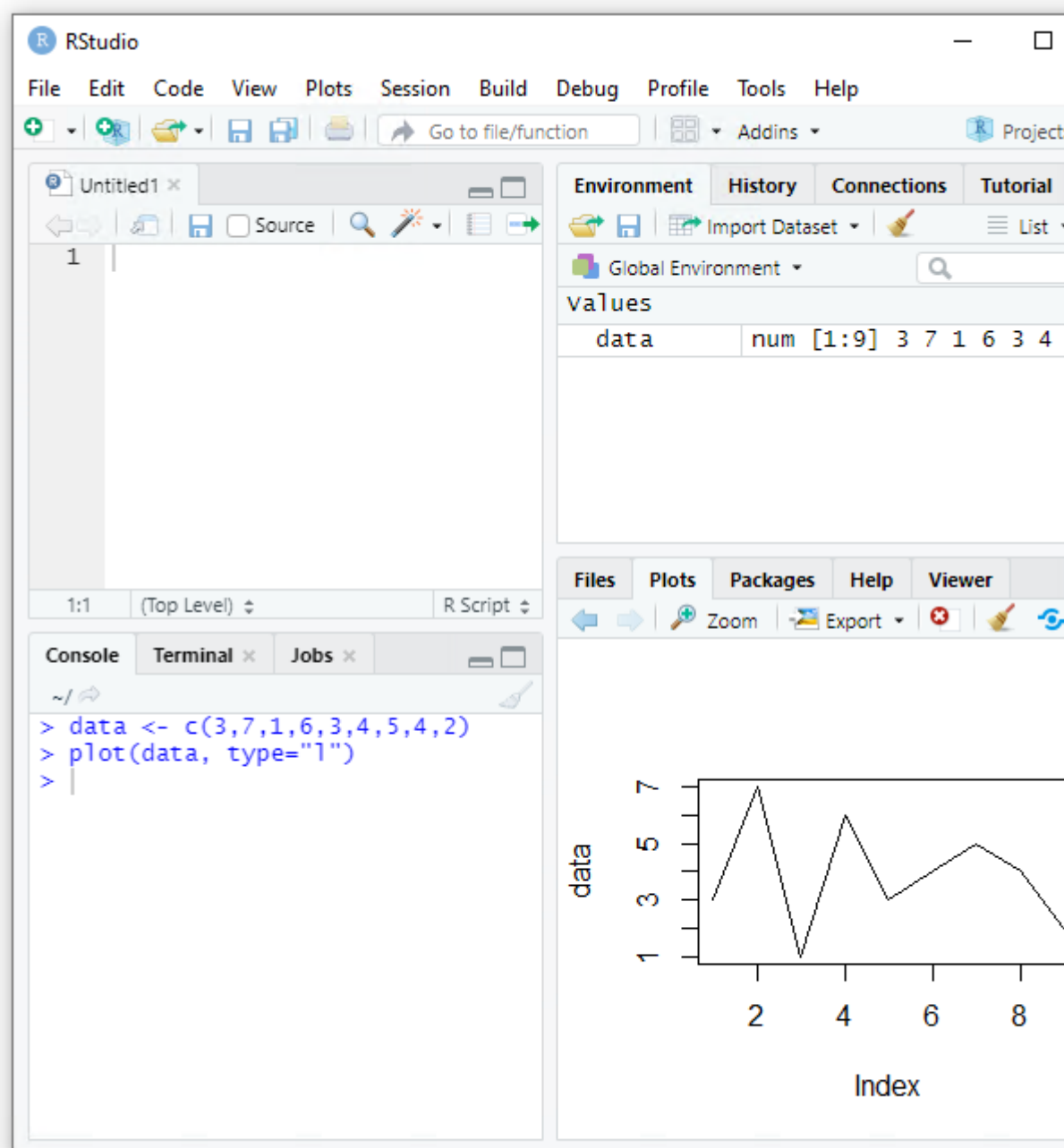


Figure 1.10: Приклад написання коду в RStudio Desktop

1.3.2.2 Створення першого проекту в RStudio {chapter1322}

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.

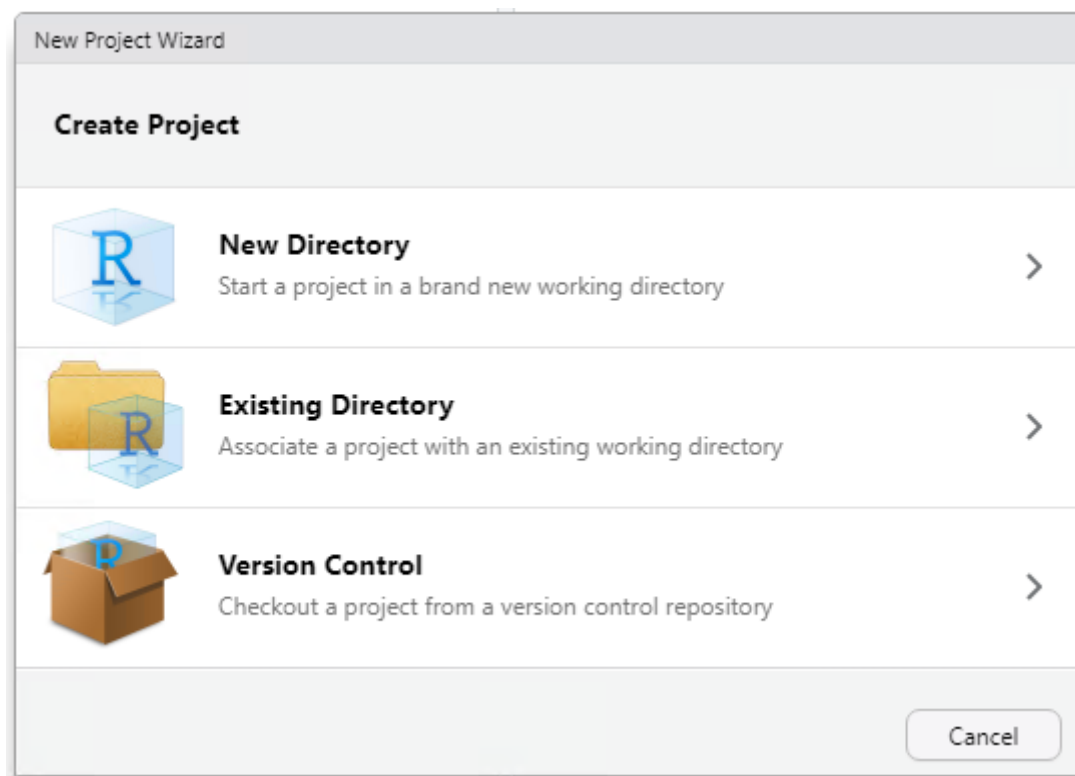


Figure 1.11: RStudio Desktop. Новий проект

На наступному кроці обираємо `New Project`:

Після кліку на `Create Project` буде створено папку за попередньо обраним шляхом. Для запуску проекту 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.

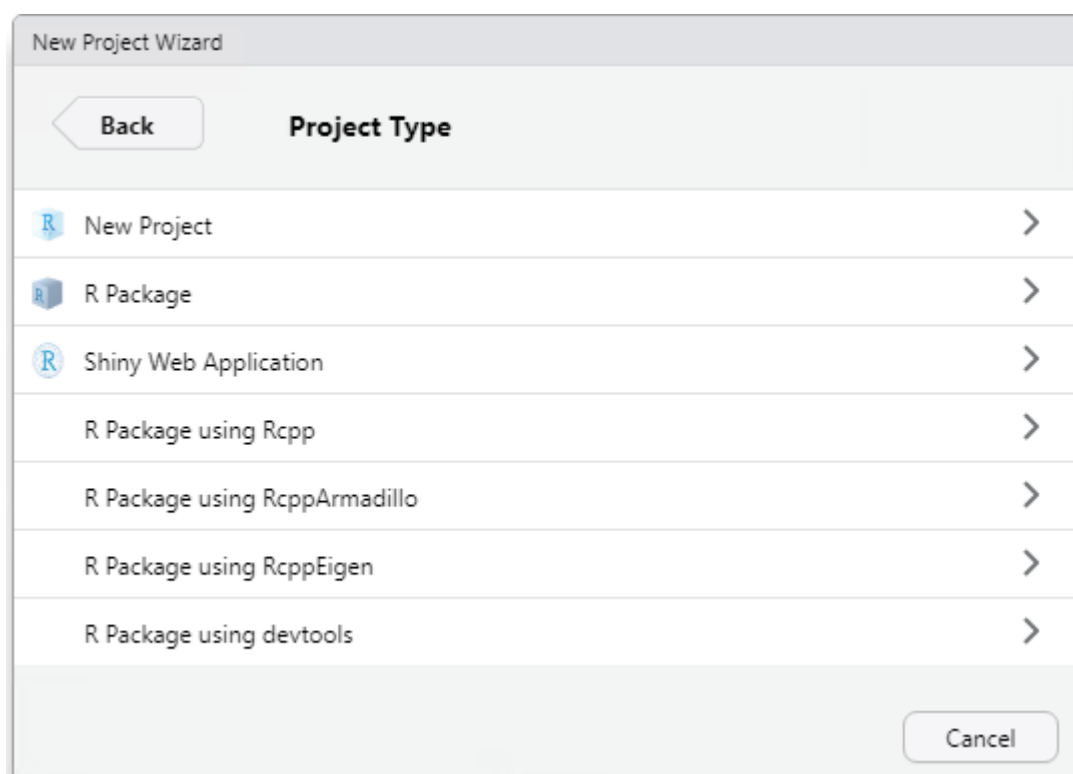


Figure 1.12: RStudio Desktop. Новий проект. Тип проекту

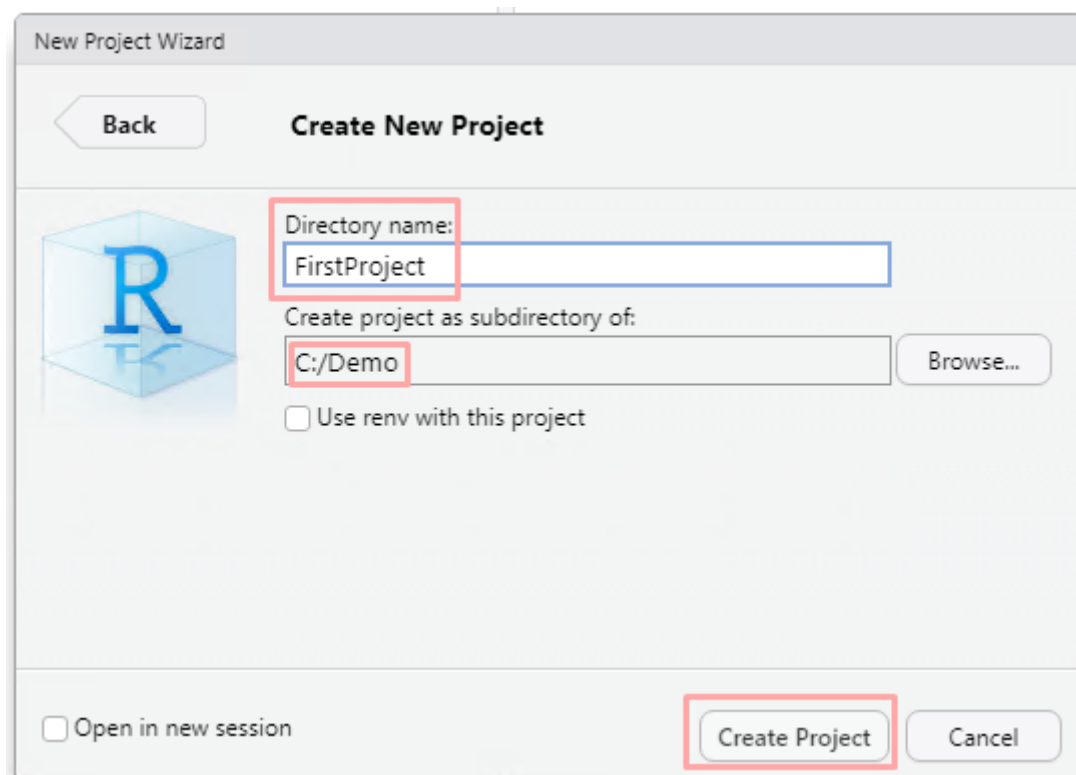


Figure 1.13: RStudio Desktop. Новий проект

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 розв'яжемо 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 чисел у діапазоні [1;5] матиме наступний вигляд:

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

```
## [1] 5 2 4 2 5 1 1 5 1 4 1 3 3 4 4 1 2 4 3 4
```

Результати виконання на Вашому ПК будуть іншими, адже **псевдогенератор** випадкових чисел буде брати іншу “точку відліку” для генерування чисел. Рекомендую перегляду функцію `set.seed(-)`.

P.S. Також зустрічає у мережі інформацію, що робота `set.seed()` для R у 4+ версії може відрізнятися від 3+. Після перевірки інформації оновлю даний текст.

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: 59"
```

```
print(paste0("Mean: ", vtr_mean))
```

```
## [1] "Mean: 2.95"
```

Виведемо гістограму:

```
hist(vtr, breaks = 5)
```

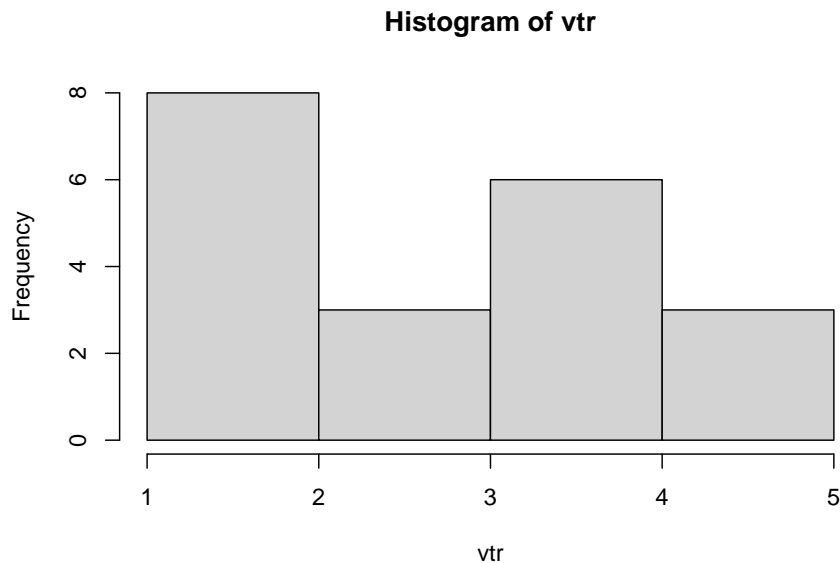


Figure 1.14: Приклад візуалізації гістограми в R

Примітка. Детальніше про параметри функції `hist()` можна почитати тут: <https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/hist>.

Орієнтовний вигляд вікна RStudio після виконання усіх описаних вище операцій матиме наступний вигляд:

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

1.3.3 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.

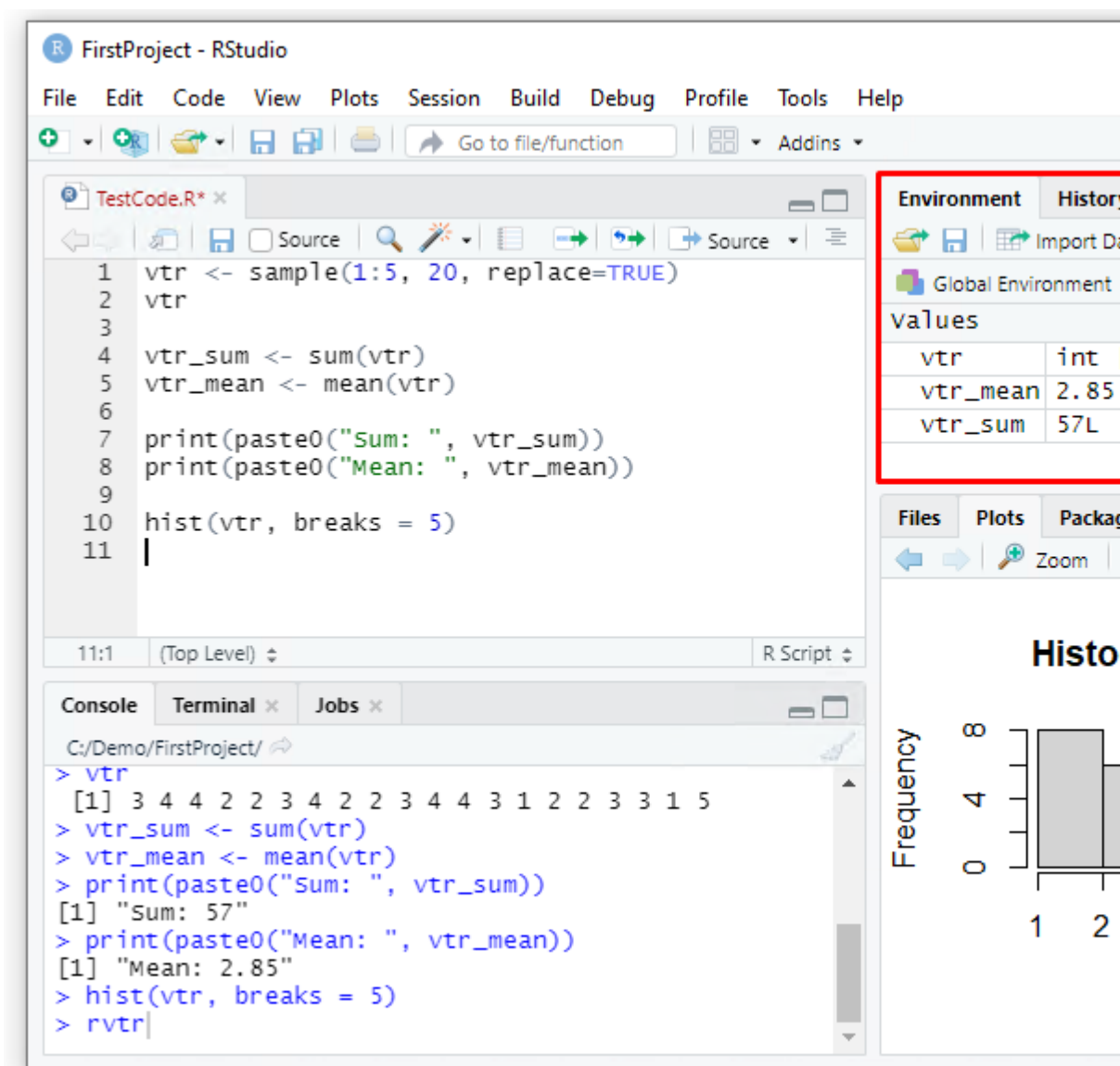


Figure 1.15: RStudio Desktop. Перегляд змінних

*Примітка. Лекційні матеріали даного курсу виконані саме у ноутбуках.

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.

Розглянемо 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> (Ana, 2021).

*Примітка. Усі операції у даному курсі виконуються під операційну систему Windows 10 Education.

Процес інсталяції середовища Anaconda не відрізняється від стандартного покрокового встановлення програм у Windows.

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

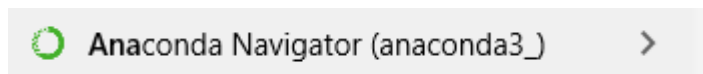


Figure 1.16: Anaconda

Для початку потрібно перейти на вкладку Environments та натиснути Create:

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

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

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:

1.3.4 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.

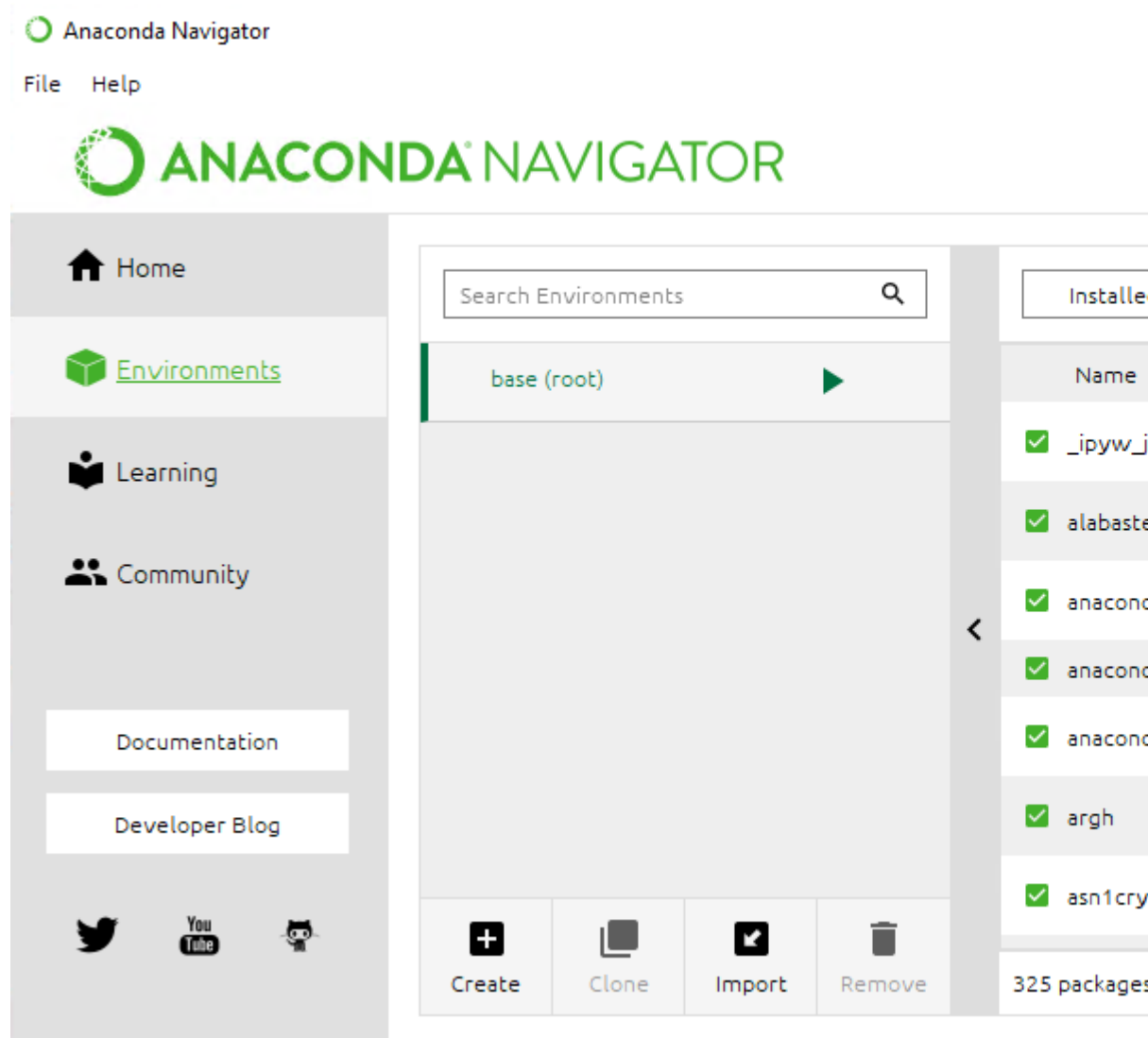


Figure 1.17: Anaconda. Головне вікно

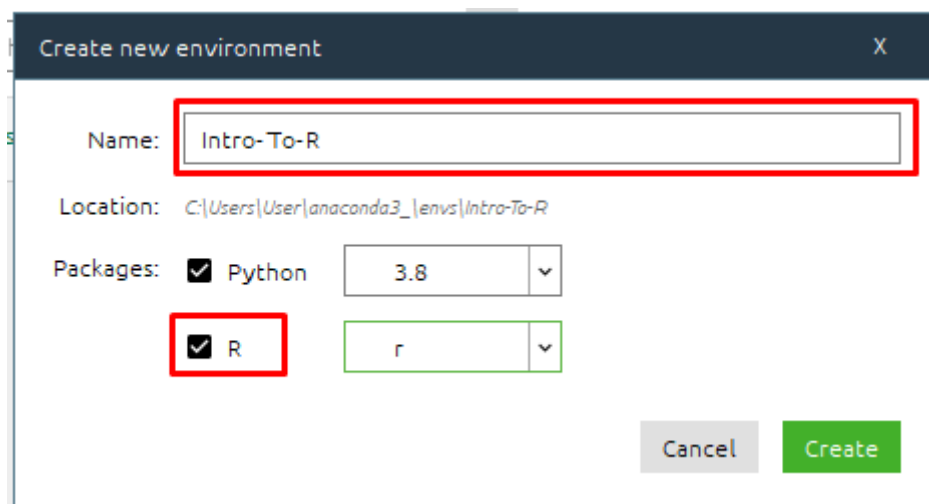


Figure 1.18: Anaconda. Створення нового середовища на основі R

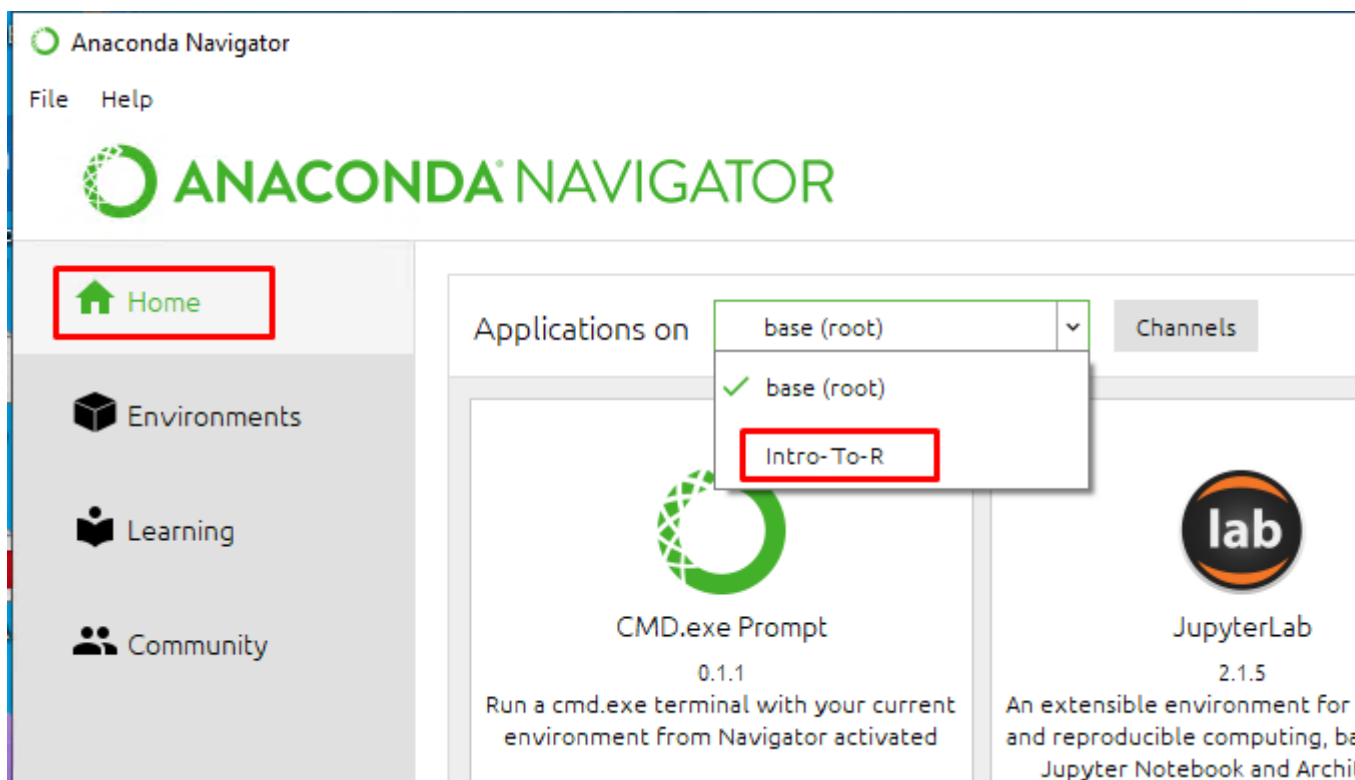


Figure 1.19: Anaconda. Зміна середовища

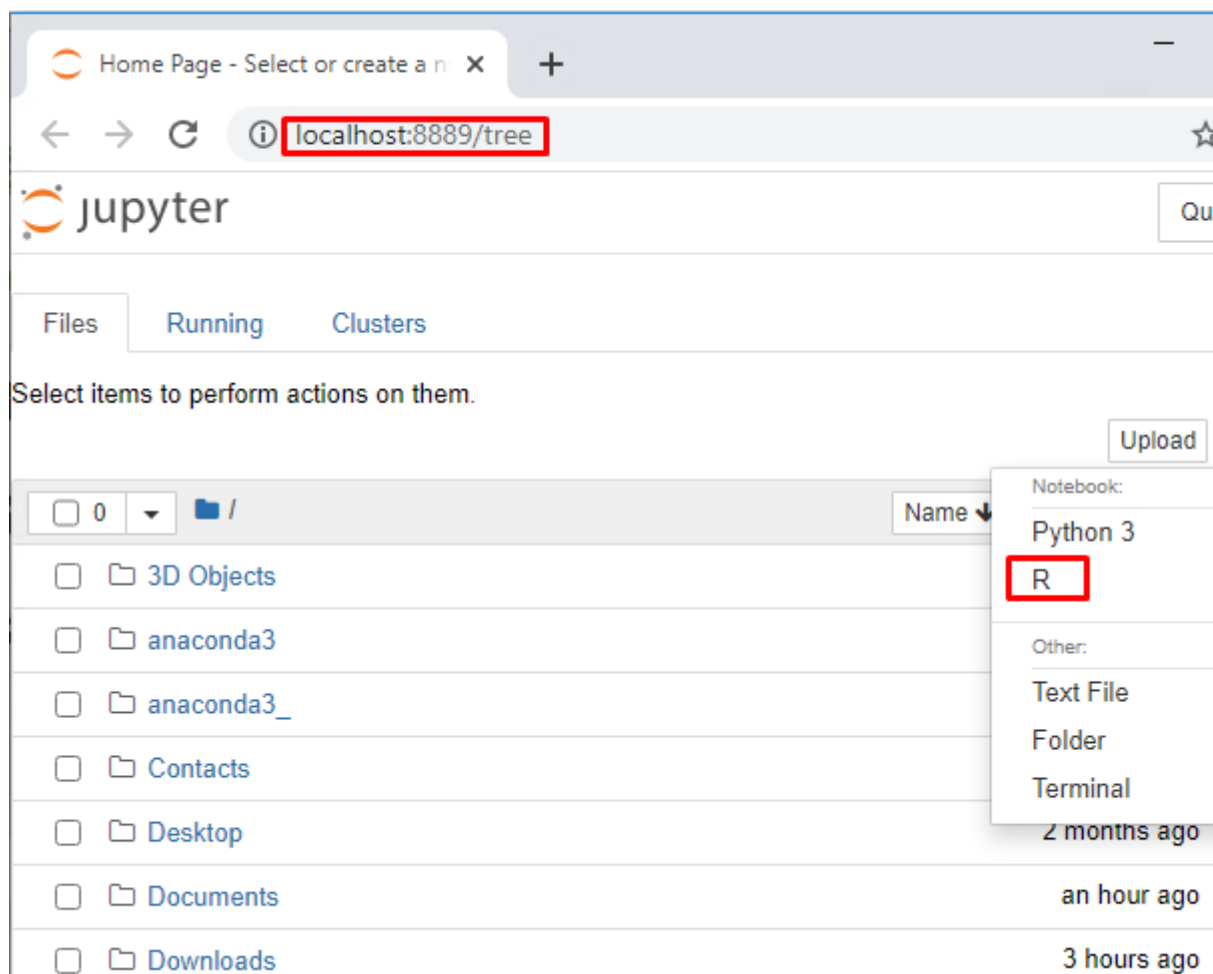


Figure 1.20: Jupyter Notebook. Створення нового ноутбука

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

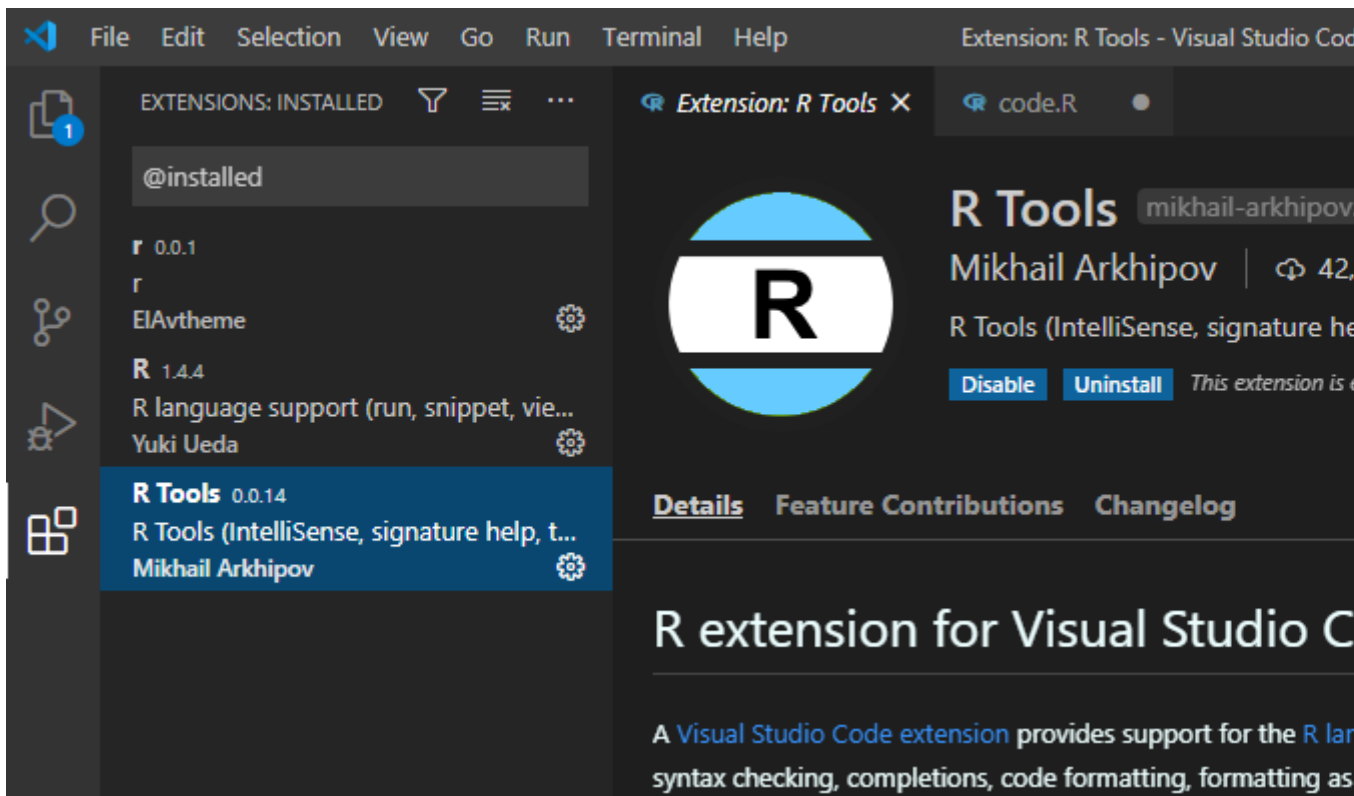


Figure 1.21: Visual Studio Code. Інсталяція RTools

Visual Studio Community Edition - безкоштовне середовище розробки від компанії 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 (vis, 2021).

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

Примітка. Код у прикладі вище написаний на Python.

kaggle.com - сервіс для змагань з Data Science та Machine Learning. Окрім переліку змагань, наборів даних сервіс має досить зручні ноутбуки.

Загалом сервісів та середовищ для розробки в R існує досить багато і їх

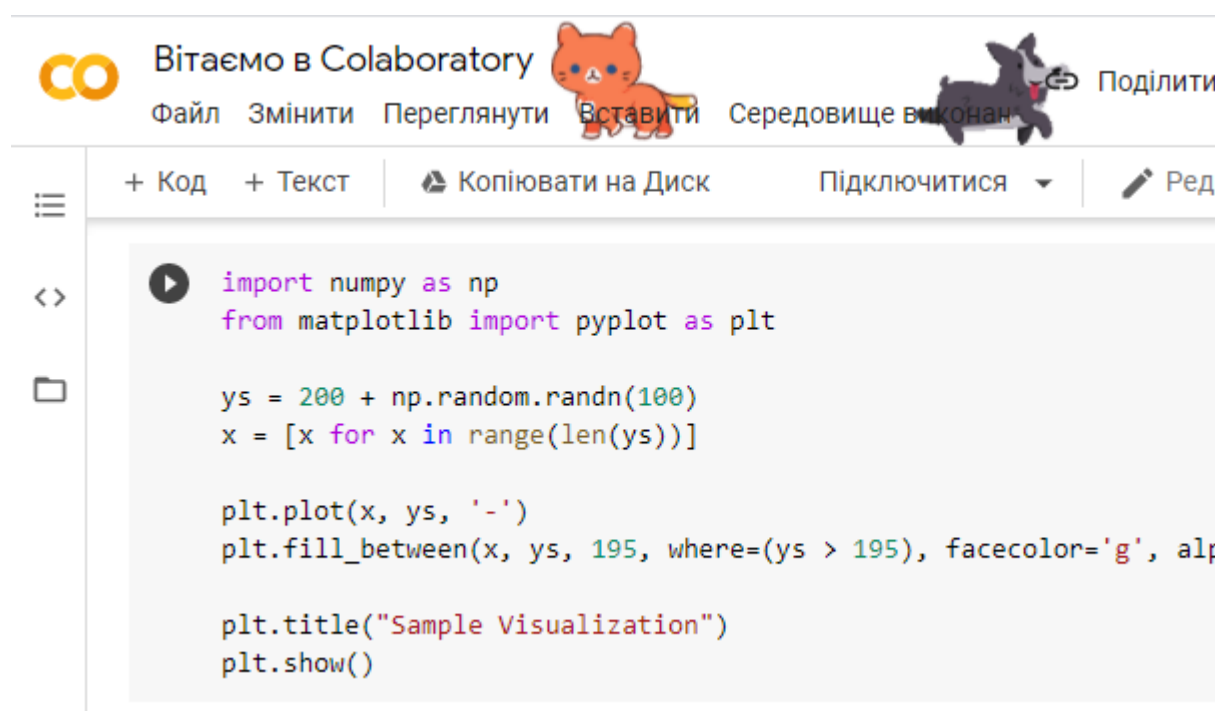


Figure 1.22: Google Collab

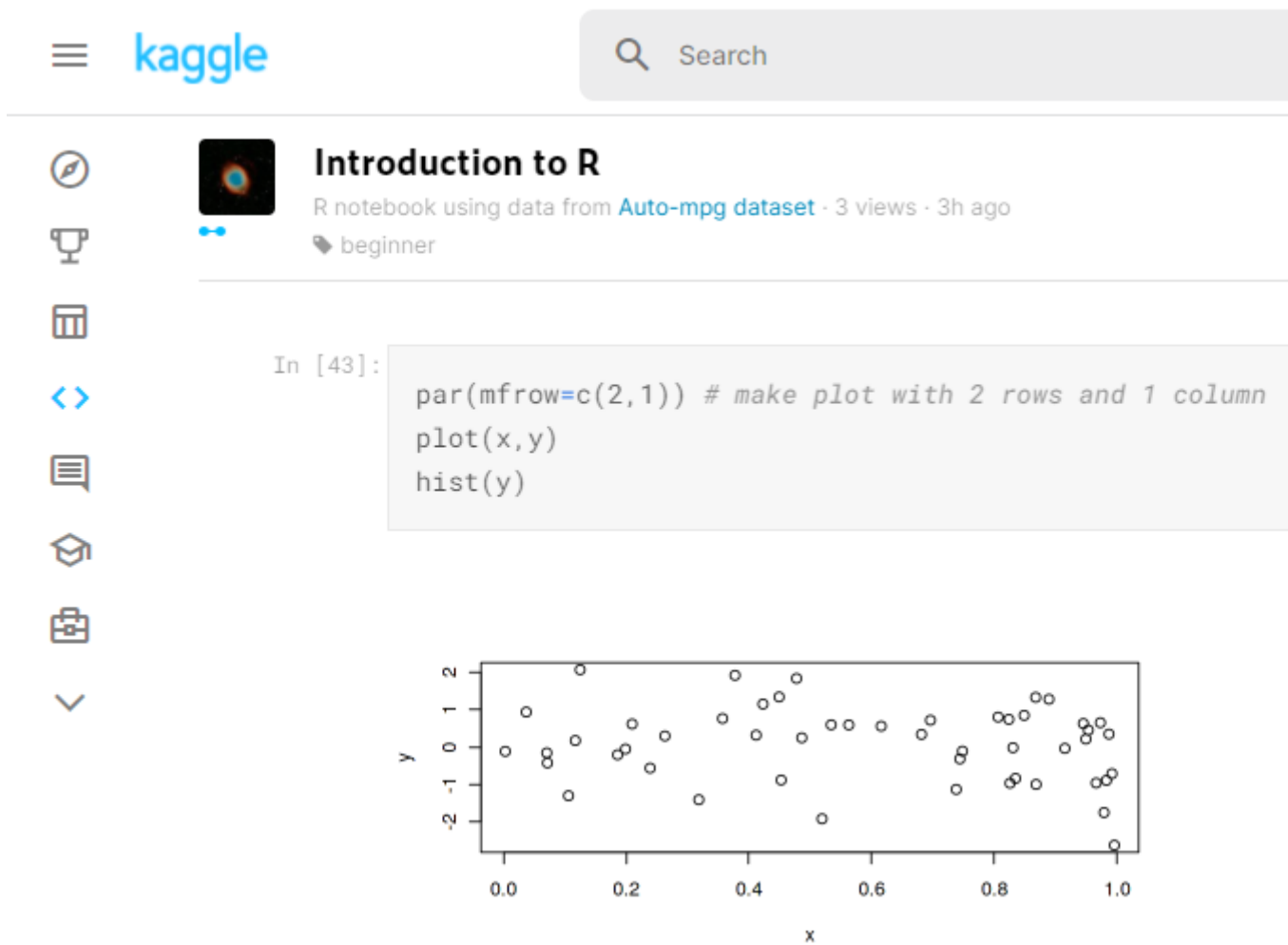


Figure 1.23: Kaggle.com

кількість зростає, але це не впливає на принципи написання коду та роботи з даними.

1.4 Основи роботи з пакетами в R

1.4.1 Команди для роботи з пакетами

Своєю популярністю R завдячує, у тому числі, і можливості швидко реалізувати досить складні дослідження за допомогою наборів уже готових функцій. Такі функції об'єднуються у пакети та публікуються вченими, дослідниками 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 також дані для відтворення прикладів коду.

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

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

Дуже рекомендую почитати детальніше про пакети у статті на DataCamp: [R Packages: A Beginner's Guide](#).

1.4.2 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`:

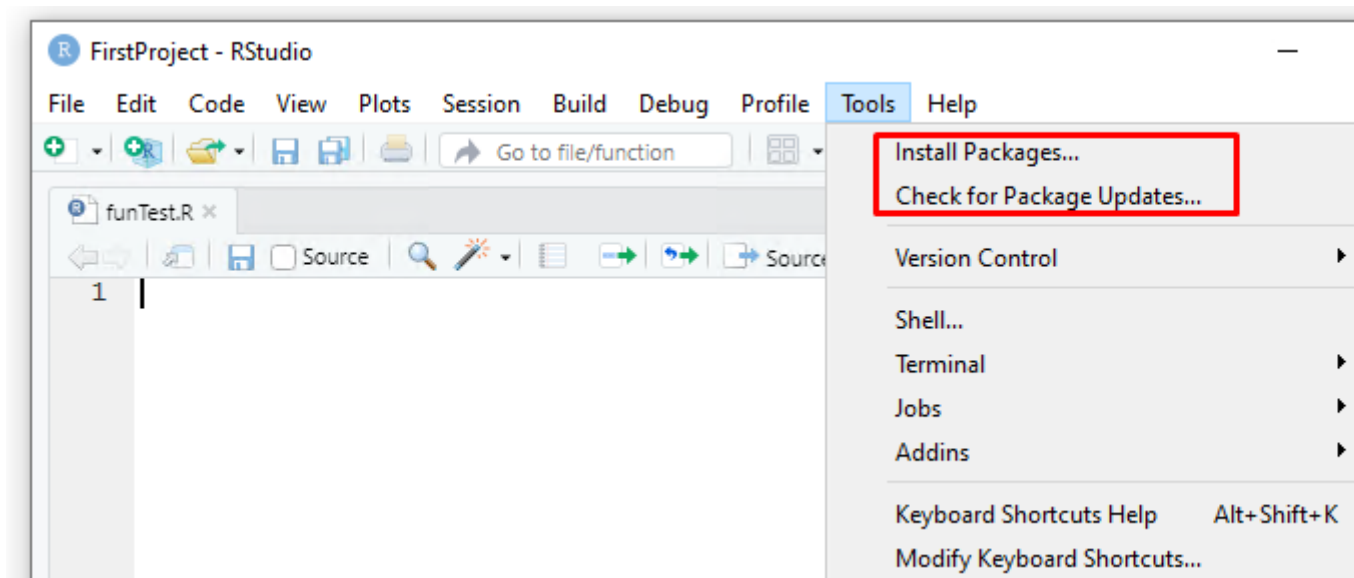


Figure 1.24: RStudio Desktop. Меню інсталяції пакетів

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:

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

Доступ до функцій та інших елементів пакету можна здійснювати використавши запис `_ :: _ ()` без підключення бібліотеки за допомогою `library()`:

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

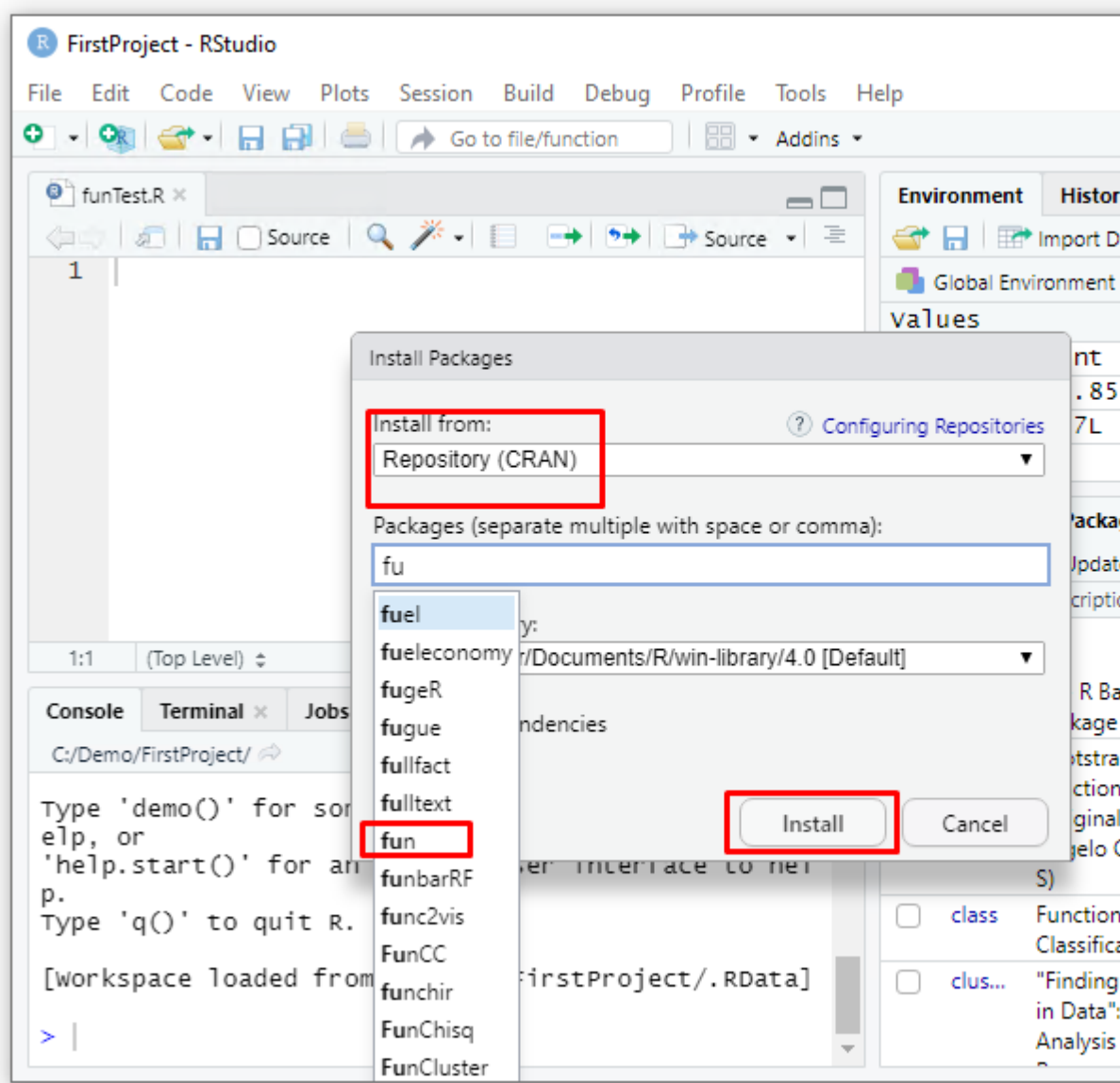


Figure 1.25: RStudio Desktop. Вибір пакету для інсталяції

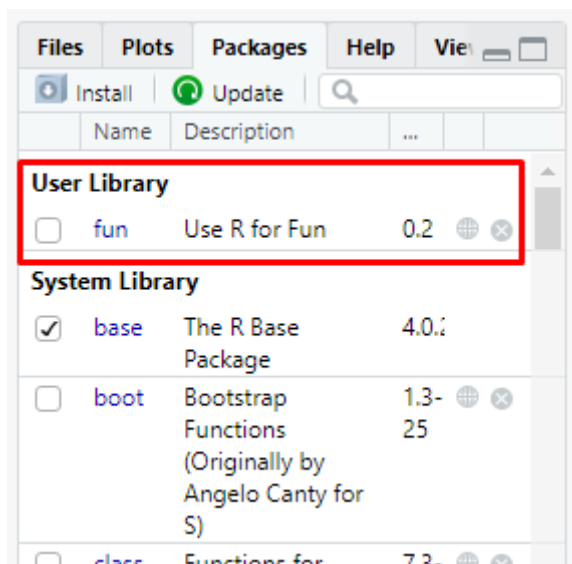


Figure 1.26: RStudio Desktop. Перегляд інстальованих пакетів

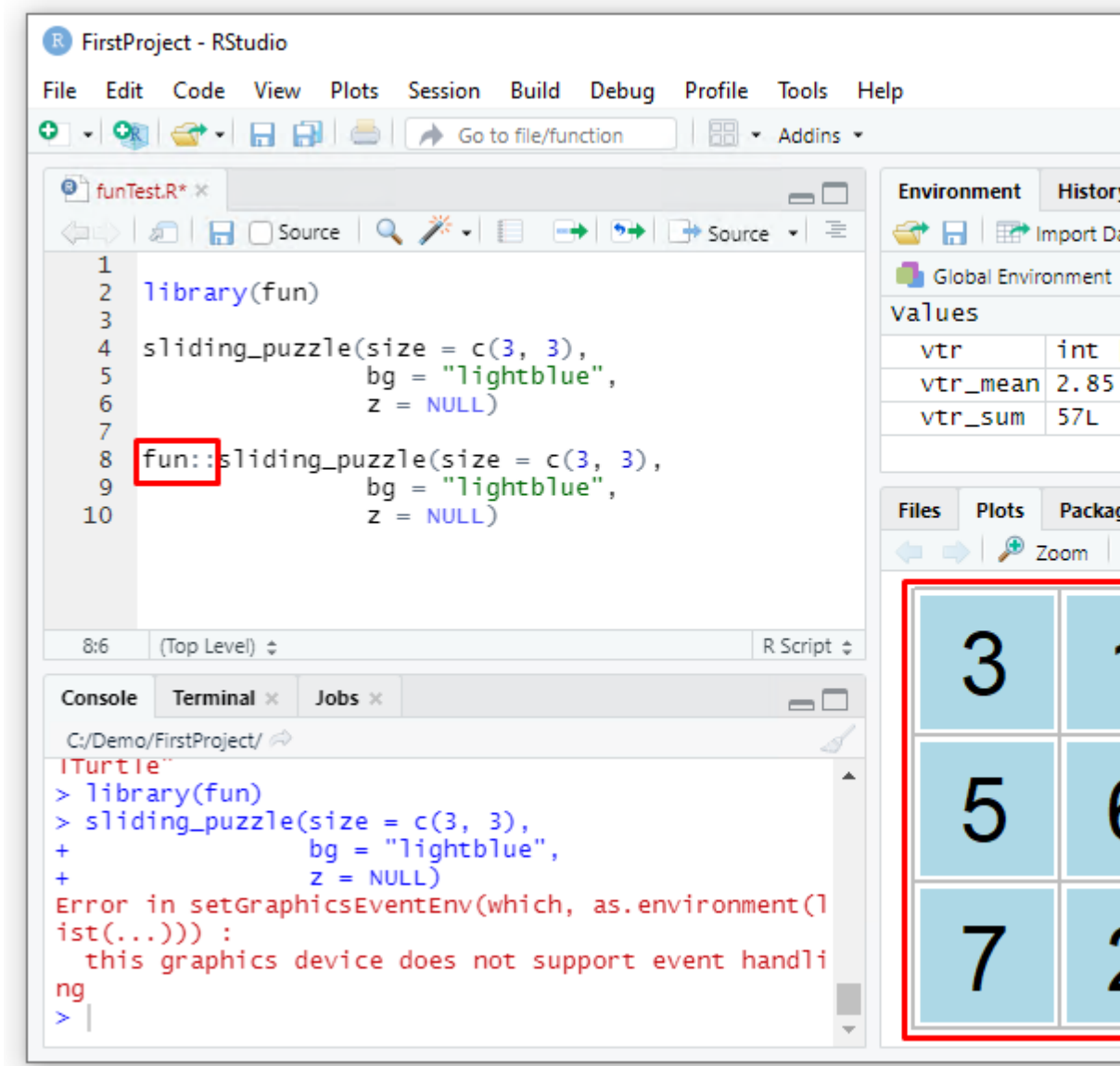


Figure 1.27: RStudio Desktop. Приклад використання пакету 'fun'

Chapter 2

Базові конструкції мови R

План

- Оголошення та ініціалізація змінних
 - Поняття змінних та оператор присвоєння
 - Правила іменування змінних
- Базові типи даних
 - Типізація в R
 - Перевірка та приведення типів даних
- Оператори
 - Арифметичні оператори
 - Оператори відношення
 - Логічні оператори
- Корисні математичні функції
 - Заокруглення чисел(round, ceiling, floor, trunc, signif)
 - Послідовності чисел (seq, rep)
 - Генерація псевдовипадкових чисел
 - Інші математичні функції та константи R
 -

2.1 Введення-виведення даних

2.2 Оголошення та ініціалізація змінних

2.2.1 Поняття змінних та оператор присвоєння

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**.

Важливо! Зверніть увагу, що присвоєння (`<-`, `=`) та рівність (`==`) це різні поняття. Оператор `==` здійснює 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 основним способом засобом початкової ініціалізації змінних є `<-`.

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

Рекомендую почитати про різницю між операторами присвоєння у R `<-` та `=` тут:

1. Why do we use arrow as an assignment operator? (Colin FAY).
2. Difference between assignment operators in R (Ren Kun).
3. Assignment Operators.

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`.

2.2.2 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) 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@1, 5x_1, _variable, FALSE, .One.

2.3 Базові типи даних

2.3.1 Типізація в R

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.

Розглянемо 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 а типу int (integer - ціле число), а потім відбувається присвоєння для а текстового фрагмента

(тип `string` у `#`). 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
```

Замінімо значення `my_character <- "5"` та спробуємо знайти суму значень:

```
my_character <- "5"
my_sum <- my_numeric + my_character
```

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

```
class(5)
```

```
## [1] "numeric"
```

```
class("5")
```

```
## [1] "character"
```

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

2.3.2 Перевірка та приведення типів даних

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

Приведення типів даних - операція перетворення значення з одного типу даних в інший. Важливо пам'ятати, 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. _ ()`. 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. _ ()`. 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 об'єктів нижче:

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

2.4 Оператори

2.4.1 Арифметичні оператори

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
```

Примітка. Використання “круглих” дужок у програмуванні виразах має пріоритет аналогічний до загальноприйнятих у математиці.

Розглянемо приклад **віднімання** чисел:

```
47 - 21
```

```
## [1] 26
```

```
15 - (10 - 25)
```

```
## [1] 30
```

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 допомогою оператора \wedge є досить простим. Так, наприклад, 3^2 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
```

Ділення націло залишає лише цілу частину від ділення двох чисел:

```
28 %/% 7
```

```
## [1] 4
```

```
17 %/% 5
```

```
## [1] 3
```

```
Sys.setlocale("LC_CTYPE", "ukrainian")
```

```
## [1] "Ukrainian_Ukraine.1251"
```

```
# ,
```

В окремих випадках інтерпритатори R можуть некоректно читати або взагалі ввжати за помилку наявність кирилиці у коді. Тоді варто вказати явно локалізацію, яку Ви бажаєте використовувати. Для української локалізації варто на початку коду додати рядок:

```
Sys.setlocale("LC_CTYPE", "ukrainian")
```

2.4.2 Оператори відношення

Оператори відношення відповідають за порівняння 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
```

Розглянемо кілька прикладів використання описаних вище операторів.

Оператори, що відповідають за перевірку 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
```

2.4.3 Логічні оператори

До логічних операторів у R відносяться:

- **I** & (амперсанти, Shift-7) - виконання усіх умов одночасно.
- **AБО I** (вертикальна риска, Shift+\) - виконання однієї із умов.
- **НЕ !** (знак оклику, Shift+1) - заперечення.

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

A	B	Оператор I	Оператор AБО	Заперечення A (не A)
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	FALSE

Матеріали розділу не доповнені прикладами.

2.5 Корисні математичні функції

2.5.1 Заокруглення чисел (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 4, а 3.2 як 3, вважючи 3.5 межею).

Увага! Заокруглення чисел у програмуванні може призводити до помилок у результатах обчислень. Для задач бізнесу або технічних процесів мінімальні відхилення можуть призводити до викривлених результатів або збоїв у системах.

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
```

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
```

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
```

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 *ceiling()*, а для чисел більших нуля $x > 0$, як *floor()*:

```
x <- 5.34
```

```
print(paste("trunc:", trunc(x), "ceiling:", ceiling(x), "floor:", floor(x), sep = " "))
```

```
## [1] "trunc: 5 ceiling: 6 floor: 5"
```

```
x <- x * -1
```

```
print(paste("trunc:", trunc(x), "ceiling:", ceiling(x), "floor:", floor(x), sep = " "))
```

```
## [1] "trunc: -5 ceiling: -5 floor: -6"
```

Функція *signif()*

Іноколи виникає потреба заокруглити не десяткову частину числа, а десятки, сотні, тисячі і так далі. Розглядемо варіант, коли у нас є велике число 11 547 741.3 і нам потрібно коротко його записати як 11.5. 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
```

2.5.2 Послідовності чисел (seq, rep)

Матеріали розділу у процесі підготовки.

2.5.3 Генерація псевдовипадкових чисел

Матеріали розділу у процесі підготовки.

```
runif(10)
```

```
## [1] 0.235335378 0.244405587 0.009424044 0.178302423 0.137479838 0.846481698
## [7] 0.439165844 0.394814532 0.695522728 0.032971312
```

```
sample(100)
```

```
## [1] 66 19 2 24 55 32 8 75 89 25 96 22 53 99 85 27 39 52
## [19] 80 16 81 43 86 82 47 30 40 33 71 74 77 64 38 94 60 68
## [37] 97 76 54 10 18 28 93 6 83 3 62 15 51 13 26 36 72 20
## [55] 42 58 91 70 14 69 41 92 4 12 100 46 9 5 7 11 67 21
## [73] 29 78 59 65 35 17 95 45 90 79 23 56 48 1 87 49 98 57
## [91] 88 63 34 44 61 73 37 50 31 84
```

2.5.4 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.

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

Функція	Призначення, опис
<i>sqrt(x)</i>	Корінь квадратний числа x
<i>factorial(x)</i>	Факторіал числа x
<i>abs(x)</i>	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 $\arccos(x)$, $\arcsin(x)$, $\arctan(x)$, $\cosh(x)$, $\sinh(x)$, $\tanh(x)$.

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

2.6 Введення-виведення даних

Матеріали розділу у процесі підготовки.

Chapter 3

Основи роботи з даними в R

План

- Набори даних
 - Вектори (vectors)
 - Поняття та спосіб представлення
 - Оголошення векторів
 - Операції над векторами
-

3.1 Набори даних

Матеріали розділу у процесі підготовки.

3.2 Вектори (vectors)

3.2.1 Поняття та спосіб представлення

Вектори є найпростішим способом представлення колекції даних. З своїм змістом **вектор** - це послідовність однорідних елементів. Якщо ж говорити

про мову програмування, то **вектор** - це послідовність елементів одного типу, що розміщені за деяким порядком (індексом).

Вектор прийнято позначати, як $x = (x_1, x_2, \dots, x_n)$, де x - назва вектора, n - кількість елементів вектора,

3.2.2 Оголошення векторів

Вектор - базовий тип даних у R, що дозволяє записати колекцію елементів одного типу за допомогою `c()` або без нього, якщо це послідовність значень.

Примітка. По суті функція `c()` дозволяє об'єднати кілька векторів.

Розглянемо для прикладу звичайну змінну x :

```
x <- 10
```

По своїй суті x у даному випадку є вектором, що складається з одного значення 10. Ми можемо також записати кілька елементів у змінну x :

```
x <- c(1, 2, 2.5, 3)
x
```

```
## [1] 1.0 2.0 2.5 3.0
```

Елементами вектора можуть бути значення будь якого типу: `numeric`, `character`, `logical` тощо:

```
v1 <- c(1, 3, 4, 6, 7)
v2 <- c(T, F, F, T, F)
v3 <- c("Hello", "my", "friend", "!")
```

Елементами вектора також послідовності, створені на за допомогою функцій `rep()`, `seq()` та оператора `::`:

```
vtr <- 2:7
vtr
```

```
## [1] 2 3 4 5 6 7
```

```
vtr <- 7:2
vtr
```

```
## [1] 7 6 5 4 3 2
```

Якщо є потреба об'єднати кілька векторів, скористайтеся функцією `c()`:

```
x <- 2:3
y <- c(4,6,9)
z <- c(x, y, 10:12, 100)
z
```

```
## [1] 2 3 4 6 9 10 11 12 100
```

Переглянути коротку описову статистику по вектору можна за допомогою функції `summary()`:

```
summary(z)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      2.00   4.00   9.00   17.44  11.00  100.00
```

3.2.3 Операції над векторами

Chapter 4

Приклади задач та їх розв'язки

-
- Задачі
 - Послідовності та вектори
 - Функції
 - Рішення задач
 - Послідовності та вектори
 - Функції
-

4.1 Задачі

4.1.1 Послідовності

4.1.1.1 Задача

Написати програму, що обчислює $y = (x + 2)^2 + \ln(x)$, де x число з послідовності 100, 105, 110, ..., 200. Результат вивести у вигляді `data.frame` з колонками X та Y .

4.1.1.2 Задача

Написати програму, що обчислює $y = \frac{\sqrt{x+2}}{z}$, де x число з послідовності 10, 15, 20, ..., 100, а z - випадкові значення з діапазону $[-10, 10]$, що відповідає кількості елементів у x . Результат вивести у вигляді `data.frame` з колонками X , Z та Y .

4.1.1.3 Задача

Без використання спеціальних функцій написати програму, що сортує вектор за зростанням та спаданням. Елементи вектора є випадковими числами з діапазону $[10, 100)$. Кількість елементів у векторі 10.

4.1.1.4 Задача

Відсортувати вектор таким чином, щоб всі додатні елементи знаходилися на початку, а всі від'ємні – вкінці, і при цьому зберігся початковий порядок елементів в обох групах. Елементи вектора є випадковими числами з діапазону $[-100, 100]$. Кількість елементів у векторі 10.

Наприклад, якщо початковий вектор x складається з елементів:

```
x <- c(1, -5, 10, -8, -2, 5, 4, -9)
```

то після "сортування" він матиме вигляд:

```
## [1] 1 10 5 4 -5 -8 -2 -9
```

4.1.1.5 Задача

Задано натуральне число N (вводиться з клавіатури). Знайти суму його цифр.

4.1.1.6 Задача

Задано одновимірний масив. Знайти два серед його елементів, модуль різниці яких має найменше значення.

4.1.1.7 Задача

Знайти найбільший спільний дільник двох натуральних чисел, використавши алгоритм Евкліда. Алгоритм Евкліда полягає в наступному: від більшого числа віднімається менше до тих пір, поки вони не стануть рівними; отримане в результаті число і буде найбільшим спільним дільником.

4.1.1.8 Задача

Знайти мінімальний елемент серед тих елементів масиву A , які не є елементами масиву B .

4.1.1.9 Задача

Написати програму, що обчислює **середнє** значення серед парних елементів вектора. Елементи вектора генеруються випадковим чином у діапазоні $[1; 10)$. Кількість елементів вектора 10.

4.1.1.10 Задача

Написати програму, що знаходить середнє значення елементів вектора. Елементи вектора генеруються випадковим чином у діапазоні $[100; 200]$. Усі елементи вектора повинні бути кратними 7-ми. Генерацію випадкового числа кратного 7-ми винести в окрему функцію.

4.1.2 Функції

4.1.2.1 Задача

Написати функцію, що обчислює для поданого вектора суму, середнє, медіану, мінімум та максимум. Результат роботи функції повертається у вигляді списку (`list`).

4.2 Рішення задач

4.2.1 Послідовності та вектори

Рішення до задачі 6.1.1.3:

```
x <- sample(10:100, size = 10)

for(j in 1:(length(x)-1)) {

  for(i in 1:(length(x)-1)) {

    if(x[i] > x[i+1]) {
      tmp = x[i]
      x[i] = x[i+1]
      x[i+1] = tmp
    }
  }
}

print(x)
```

```
## [1] 35 42 44 48 49 53 55 86 92 100
```

Рішення до задачі 6.1.1.4:

```
x <- sample(-100:100, size = 10)
print("Vector before sort:")
```

```
## [1] "Vector before sort:"
```

```
print(x)
```

```
## [1] -61 -69 12 -87 94 82 -70 72 -3 -80
```

```
for(j in 1:(length(x)-1)) {

  for(i in 1:(length(x)-1)) {

    if(x[i] < 0 & x[i+1] > 0) {
      tmp = x[i]
      x[i] = x[i+1]
      x[i+1] = tmp
    }
  }
}

print("Vector after sort:")
```



```
## [1] "Vector after sort:"
```

```
print(x)
```

```
## [1] 12 94 82 72 -61 -69 -87 -70 -3 -80
```

Рішення до задачі 6.1.1.5:

```
#number <- as.numeric(readline(prompt = "      :"))
number <- 15783
sum <- 0

while(number > 0) {
  last_digit = number %% 10
  sum = sum + last_digit
  number = (number - last_digit) / 10
  print(paste0("Number: ", number, " | Sum: ", sum, " | Last: ", last_digit))
}
```

```
## [1] "Number: 1578 | Sum: 3 | Last: 3"
## [1] "Number: 157 | Sum: 11 | Last: 8"
## [1] "Number: 15 | Sum: 18 | Last: 7"
## [1] "Number: 1 | Sum: 23 | Last: 5"
## [1] "Number: 0 | Sum: 24 | Last: 1"
```

Рішення до задачі 6.1.1.9 (спосіб 1):

```
x <- sample(1:10, size = 4, replace = T)
x
```

```
## [1] 1 8 9 4
```

```
x_parni <- c()

for(i in 1:length(x)) {

  if(x[i] %% 2 == 0) {
    x_parni <- c(x_parni, x[i])
  }
}
```

```
}  
  
}  
  
mean(x_parni)
```

```
## [1] 6
```

Рішення до задачі 6.1.1.9 (спосіб 2):

```
x <- sample(1:10, size = 10, replace = T)  
x
```

```
## [1] 6 4 3 1 8 7 9 8 3 6
```

```
sum <- 0  
count <- 0  
  
for(i in 1:length(x)) {  
  
  if(x[i] %% 2 == 0) {  
    sum = sum + x[i]  
    count = count + 1  
  }  
  
}  
  
mean_value = sum / count  
mean_value
```

```
## [1] 6.4
```

4.2.2 Функції

Рішення до задачі 6.1.2.1:

```
x <- sample(10:100, size = 10)  
print(x)
```

```
## [1] 82 64 44 22 19 12 75 95 99 85
```

```
vector.info <- function(vector) {  
  x <- list()  
  x$Sum <- sum(vector)  
  x$Mean <- mean(vector)  
  x$Median <- median(vector)  
  x$Min <- min(vector)  
  x$Max <- max(vector)  
  return(x)  
}
```

```
vector.info(x)
```

```
## $Sum  
## [1] 597  
##  
## $Mean  
## [1] 59.7  
##  
## $Median  
## [1] 69.5  
##  
## $Min  
## [1] 12  
##  
## $Max  
## [1] 99
```

```
Sys.setlocale("LC_CTYPE", "ukrainian")
```


Джерела

Bibliography

(2021). *Anaconda. The World's Most Popular Data Science Platform*. Anaconda Inc, 206 379 Broadway Ave., Suite 310 New York, NY 10013, USA.

(2021). *Google Colaboratory*. Google LLC.

(2021). *RStudio official website*. RStudio, PBC, 250 Northern Ave, Boston, MA 02210.

(2021). *Visual Studio Code*. Microsoft.

(2021). *Visual Studio Community Edition*. Microsoft.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

А.Б. Шипунов, Е.М. Балдин, П.А. Волкова, А.И. Коробейников, С.А. Назарова, С.В. Петров, В.Г. Суфиянов. (2012). *Наглядная статистика. Используем R!* ДМК Пресс, Москва, Россия.