МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Л.М. Олещенко

ТЕХНОЛОГІЇ ОБРОБЛЕННЯ ВЕЛИКИХ ДАНИХ

Комп’ютерний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів, які навчаються
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»)

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти: Чумаченко Сергій Мколайович, д-р техн. наук, проф.
Мошенський Андрій Олександрович, канд. техн. наук, доц.

Відповідальний редактор Легеза Віктор Петрович, д-р техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 2 від 09.12.2021 р.)
за поданням Вченої ради факультету прикладної математики
(протокол № 2 від 27.09.2021 р.)

Електронне мережне навчальне видання

Олещенко Любов Михайлівна, канд. техн. наук, доцент

ТЕХНОЛОГІЇ ОБРОБЛЕННЯ ВЕЛИКИХ ДАНИХ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Технології оброблення великих даних: комп'ютерний практикум з дисципліни «Технології оброблення великих даних» [Електронний ресурс] : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»)/ Л.М. Олещенко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані. – Київ: КПІ ім. Ігоря Сікорського, 2021. – 85 с.

Навчальний посібник розроблено для ознайомлення студентів з практичними прийомами оброблення великих даних та вимогами до виконаних комп'ютерних практикумів, зокрема, правилами їх оформлення. Навчальне видання призначене для студентів, які навчаються за спеціальністю 121 Інженерія програмного забезпечення (освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем») факультету прикладної математики КПІ ім. Ігоря Сікорського.

ЗМІСТ

ВСТУП

Аналітика великих даних сьогодні є однією з найбільш затребуваних у сучасному бізнесі. Знання нових технологій програмування та уміння розробляти програмне забезпечення для управління та аналізу великих об'ємів даних використовуються для забезпечення цифровізації усіх сфер життя.

Дисципліна «Технології оброблення великих даних» входить до циклу професійно-орієнтованих дисциплін плану підготовки бакалаврів, що навчаються за спеціальністю 121 «Інженерія програмного забезпечення» (освітня програма «Програмне забезпечення мультимедійних та інформаційно-пошукових систем»).

Предметом дисципліни є теоретичні та практичні основи оброблення великих даних. Розглядаються загальні методи оброблення великих даних, можливості мов програмування Python та R для аналізу та візуалізації даних.

Мета дисципліни – забезпечити знання теоретичних і практичних основ з оброблення великих даних за допомогою мов програмування Python, R та технологій розподіленого оброблення даних.

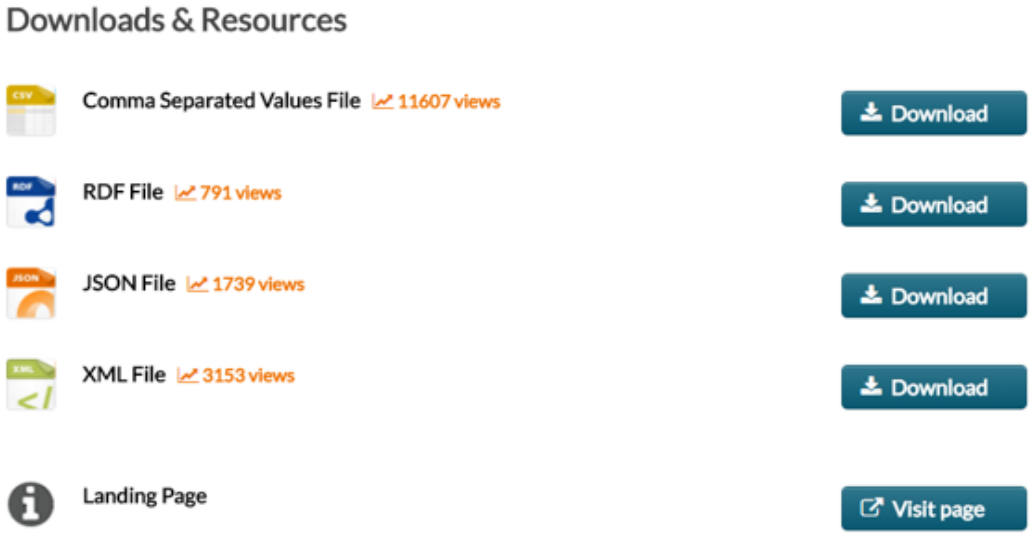
Метою комп'ютерного практикуму є отримання практичних навичок застосування технологій програмування для оброблення великих даних. У комп'ютерному практикумі надаються теоретичні відомості з кожної теми, завдання на комп'ютерний практикум з цієї теми, вказівки щодо виконання завдання, наведено вимоги до оформлення звіту з виконаного комп'ютерного практикуму, контрольні питання для самоперевірки та список рекомендованої літератури. Комп'ютерний практикум з дисципліни «Технології оброблення великих даних» розрахований на 18 академічних годин аудиторних занять. Комп'ютерний практикум складається з 6 розділів, кожен з яких присвячений одному комп'ютерному практикуму з дисципліни «Технології оброблення великих даних».

КОМП'ЮТЕРНИЙ ПРАКТИКУМ 1.
ДОСЛІДЖЕННЯ ДЖЕРЕЛ ВІДКРИТИХ ДАНИХ. ЗАВАНТАЖЕННЯ ДАТАСЕТУ ТА ЗБЕРЕЖЕННЯ ДАНИХ В ФОРМАТІ CSV

Мета роботи*:* дослідити джерела відкритих даних за допомогою Open Government Partnership та вебсайтів, які надають відкриті дані, можливості збереження та візуалізації даних, використовуючи вебсайти www.knoema.com та www.gapminder.org, дослідити право власності на персональні дані, коли ці дані не зберігаються локально та обмеження електронних таблиць при завантаженні даних.

Теоретичні відомості

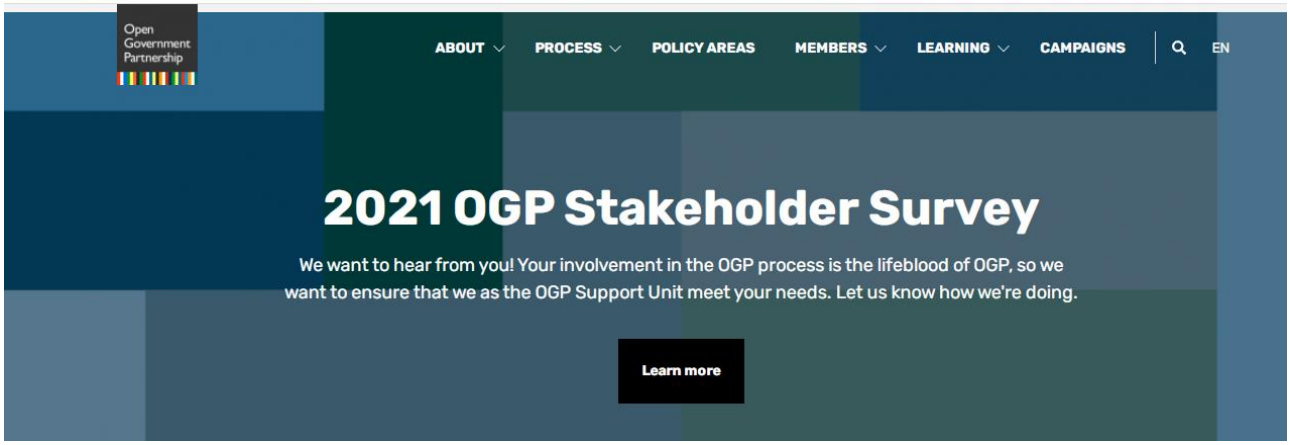
Сьогодні створено та збережено велику кількість даних. Дані корисні лише тоді, коли їх можна використовувати для отримання нових знань. Коли дані надаються у вільний доступ для використання іншим особам без обмежень інтелектуальної власності, ці дані вважаються відкритими даними. Відкриті дані можуть вимагати віднесення до першоджерела. Похідні від оригінального твору також можна позначати як відкриті дані. Партнерство "Відкритий уряд" (Open Government Partnership, OGP) було започатковано у 2011 році. Мета партнерства – створити міжнародну платформу для створення відкритих підзвітних урядів для своїх громадян. Багато організацій, таких як уряди, університети, підприємства та некомерційні організації публікують свої дані про погоду, житло, злочинність тощо у різних форматах (csv, xls, json, pdf, xml).



Завдання до комп'ютерного практикуму

Для виконання комп'ютерного практикуму використовується персональний комп'ютер (ПК) або мобільний пристрій з доступом до мережі Інтернет, програма електронних таблиць, така як Microsoft Excel, Google Sheet, LibreOffice Calc, Apple Numbers або OpenOffice Calc. Необхідно виконати скріншоти виконання роботи на надати відповіді на поставлені питання.

Частина 1. Open Government Partnership



a. Перейдіть на вебсайт **www.opengovpartnership.org**/, щоб дізнатися більше про OGP.

b. Натисніть **Members** (Учасники) у верхній центральній частині екрана.

Перелічіть кілька країн, які беруть участь у OGP.

c. Виберіть країну за допомогою карти або списку внизу сторінки, щоб переглянути інформацію про участь цієї країни у представленні відкритих даних. Наведіть інформацію про участь у OGP України.

d. Угорі сторінки міститься огляд участі цієї країни в OGP. Прокрутіть сторінку вниз, щоб побачити додаткову інформацію, включаючи плани дій та звіти. Опишіть наявну інформацію.

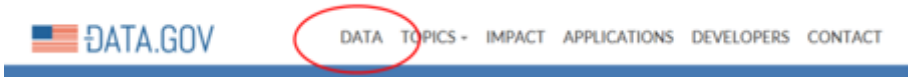
e. Натисніть **Resources** (Ресурси) та перегляньте, які документи є в наявності.

Частина 2. Відкриті вебсайти з даними

a. Перейдіть на сторінку **http://www.data.gov**. Цей вебсайт містить відкриті дані уряду США.

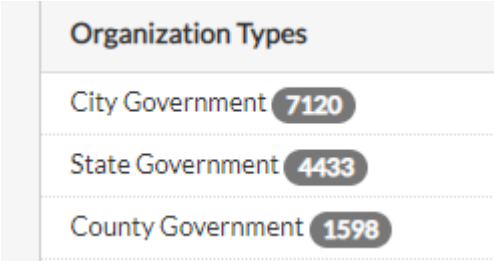
На цьому вебсайті ви можете знайти дані та ресурси для дослідження, розробки програмного забезпечення та інші проєкти, пов'язані з даними. З цього вебсайту ви також можете перейти до оригінальних вебсайтів, які надали опубліковану інформацію.

b. Натисніть **Data** (Дані) у верхній частині екрана.



c. Скільки наявних наборів даних?

d. Дослідіть цікавий набір даних або іншу категорію тем. Наприклад, ви можете фільтрувати дані за різними категоріями. Якщо вас цікавлять лише дані, надані міською владою, натисніть **Data** (Дані) у верхній частині екрана. Натисніть **Local Government** (Місцеве самоврядування). Прокручуйте список вниз, доки не знайдете **Organizational Types** (Типи організацій) на лівій панелі. Натисніть **City Government** (Уряд міста).



e. Знайдіть набір даних про злочини в місті Чикаго, що містить статистику злочинності для міста Чикаго з 2001 року по теперішній час.

f. Ви можете завантажити цей набір даних, прокрутивши вниз до розділу **Downloads and Resources** (Завантаження та ресурси).

Які формати файлів доступні для завантаження?

Натисніть посилання **Landing Page** в розділі **Downloads and Resources**, щоб перейти на Портал даних міста Чикаго про злочини з 2001 р.

g. На цій сторінці ви можете переглядати дані про злочини у місті Чикаго з 2001 року по теперішній час без завантаження даних. Ви також можете переглянути частину цих даних на карті.

Натисніть **City of Chicago** у верхній частині сторінки, щоб перейти до списку наборів даних, доступних на порталі даних **City of Chicago**. У розділі **Типи перегляду** натисніть **Карти**. Перегляньте карту злочинності для міста Чикаго за посиланням https://data.cityofchicago.org/Public-Safety/Crimes-Map/dfnk-7re6. Тут представлена інтерактивна карта району Чикаго з місцями злочинів за останній рік. Ви можете дослідити місця злочину, натиснувши червоні точки або збільшити масштаб за допомогою смуги прокрутки ліворуч. Прогляньте дані інших організацій, повернувшись до http://www.data.gov. Прогляньте подібні вебсайти для інших країн, наприклад:

Австралія: http://data.gov.au/

Великобританія: http://data.gov.uk

Канада: http://data.gc.ca

Україна: https://data.gov.ua/

Які групи наборів даних доступні для дослідження на українському порталі? У яких форматах розміщено ці дані? Які, на ваш погляд, недоліки даного порталу?

Частина 3. Візуалізація даних

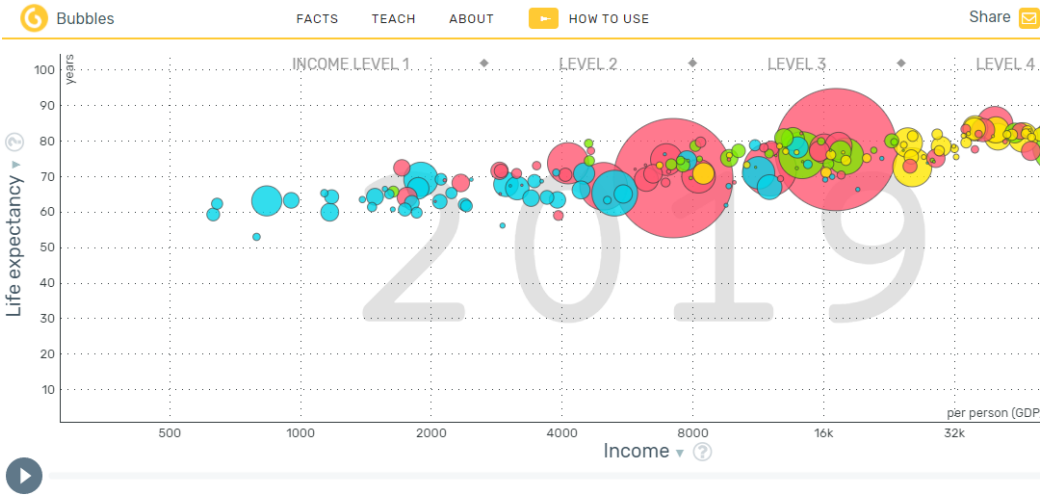
Візуалізація даних може допомогти зрозуміти списки чисел наборів даних і представити їх змістовно. Візуалізація може допомогти «побачити» тенденції, які інакше можна пропустити. Існують вебсайти, які надають візуалізацію даних. На вебсайті https://knoema.com/ є багато способів графічного представлення статистичних даних. https://knoema.com/

a. Прогляньте статистику зростання цін на товари за посиланням https://knoema.com/insights/Commodities. Наведіть приклад візуалізації даних.

b. Угорі головної сторінки натисніть стрілку вниз поруч із даними. Натисніть **World Data Atlas** (Світовий атлас даних). З будь-якої категорії https://knoema.com/atlas ви можете досліджувати різноманітні дані, доступні вам. Із списку країн оберіть **Ukraine** та опишіть, які дані цієї країни розміщені на сайті.

Місія Gapminder Foundation – подолати розрив у знаннях, використовуючи статистичні дані, зібрані з різних джерел та створюючи багату візуалізацію даних та анімовані презентації даних.

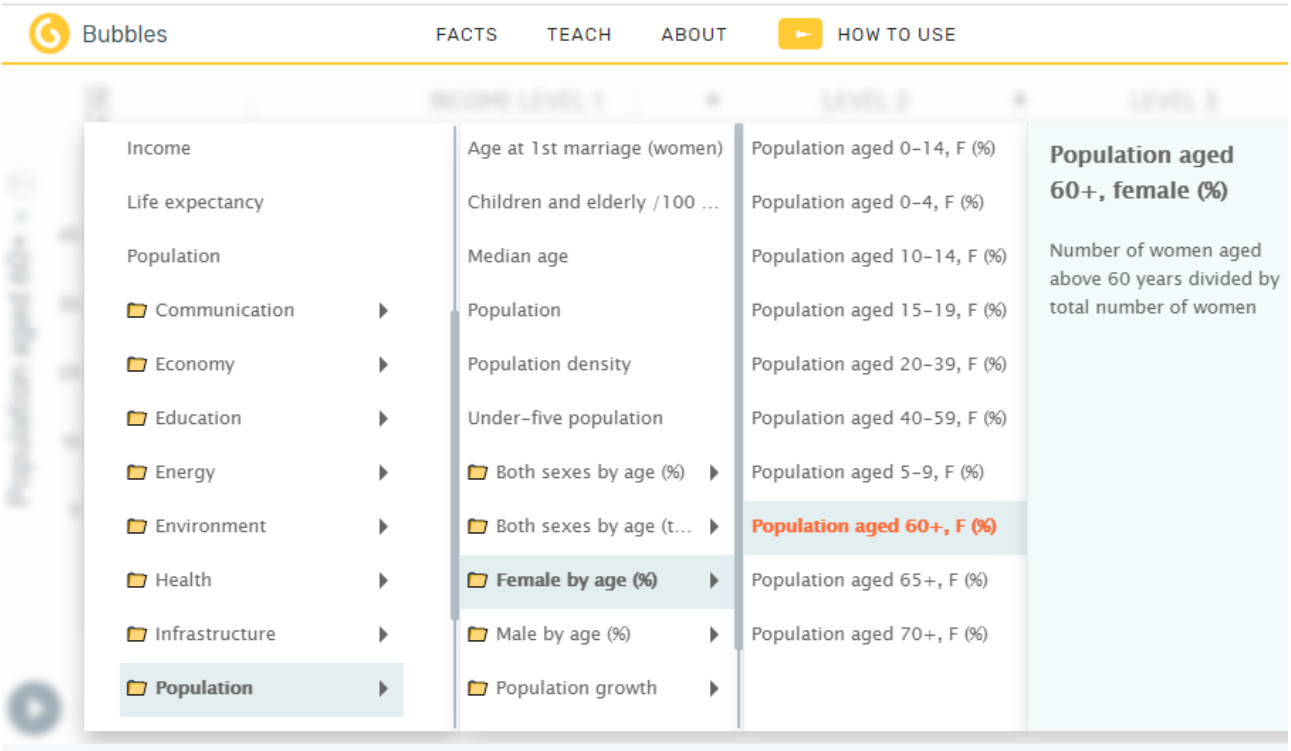
c. Перейдіть на вебсайт https://www.gapminder.org/. Натисніть Відео (https://www.gapminder.org/videos/), щоб переглянути презентації, які оживляють статистичні дані та роблять дані більш актуальними для дослідників. Щоб взаємодіяти з бульбашковими діаграмами, які були представлені у відео, натисніть **Gapminder World** (https://www.gapminder.org/tools/#$chart-type=bubbles&url=v1). Натисніть **Play**.



Спостерігайте за змінами діаграми розсіювання щодо тенденції між тривалістю життя **Life expectancy** (роки) та доходом **Income** на людину. Кожна із цих бульбашок уособлює прогрес країни.

Які відносини ви спостерігаєте?

Змініть тривалість життя **Life expectancy** по осі Y (роки) на населення у віці 60+ років, обидві статі (%), наведіть діаграми порівняння відсотку населення старше 60 років з доходом на людину.



Частина 4. Приватність даних

Соціальні мережі та мережеве сховище стали невід'ємною частиною життя багатьох людей. Файли, фотографії та відео передаються друзям та родині. Онлайн-співпраця та зустрічі проводяться на робочому місці з людьми, які знаходяться на великій відстані один від одного. Географічне розташування пристроїв зберігання даних більше не є обмежуючим фактором для зберігання або резервного копіювання даних у віддалених місцях.

Якщо ви використовуєте онлайн-сервіси для зберігання даних або спілкування з друзями чи родиною, ви, ймовірно, уклали угоду з провайдером. Умови надання послуг, також відомі як Умови використання, або Загальні

положення та умови, є юридично обов'язковою угодою, яка регулює правила відносин між вами, вашим провайдером та іншими користувачами, які користуються послугою.

Перейдіть на вебсайт використовуваної онлайн-служби та знайдіть угоду про Умови використання. Нижче наведено список популярних соціальних медіа та послуг зберігання даних в Інтернеті.

Соціальні мережі:

Facebook: https://www.facebook.com/policies

Instagram: http://instagram.com/legal/terms/

Twitter: https://twitter.com/tos

Pinterest: https://about.pinterest.com/en/terms-service

Інтернет – сховища:

icloud: https://www.apple.com/legal/internet-services/icloud/en/terms.html

Dropbox: https://www.dropbox.com/terms2014

OneDrive: http://windows.microsoft.com/en-us/windows/microsoft-services-agreement

Перегляньте умови та дайте відповідь на наступні питання.

a. У вас є обліковий запис у онлайн-провайдері соціальних медіа та/або сховища даних? Якщо так, то чи читали ви угоду про Умови надання послуг?

b. Яка політика використання даних?

c. Які налаштування конфіденційності?

d. Що таке політика безпеки?

e. Які ваші права щодо ваших даних? Чи можете ви попросити копію ваших даних?

f. Що може зробити постачальник із завантаженими вами даними?

g. Що відбувається з вашими даними після закриття облікового запису?

Після того, як ви створили обліковий запис і погодилися з Умовами надання послуг, чи дійсно ви знаєте, на що ви зареєструвалися?

Нижче наведено статтю для початку роботи з **Facebook**.

http://www.telegraph.co.uk/technology/social-media/9780565/Facebook-terms-and-conditions-why-you-dont-own-your-online-life.html

Перегляньте статтю та дайте відповідь на наступні питання.

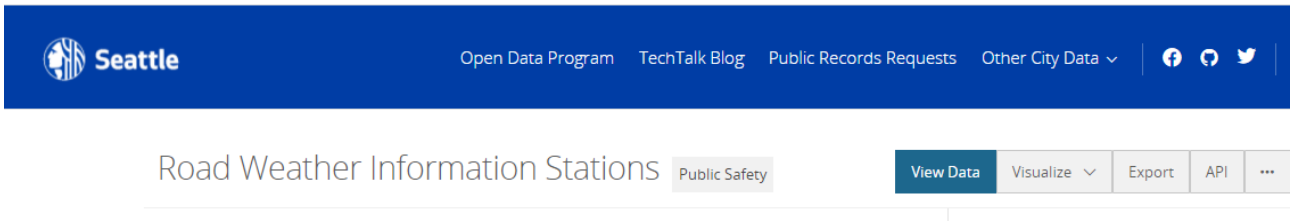
Що ви можете зробити, щоб захистити себе? Що ви можете зробити, щоб захистити свій обліковий запис та захистити свої дані?

Частина 5. Демонстрація обмеження електронних таблиць даних

В Інтернеті існує велика кількість відкритих даних, які можна завантажувати та використовувати для аналізу. Розглянемо обмеження електронних таблиць для аналізу даних, використовуючи відкритий набір даних із міста Сіетл. Завантажте набір даних дорожніх метеорологічних станцій з вебсайту https://data.seattle.gov.

a. Перейдіть на сторінку https://data.seattle.gov/Transportation/Road-Weather-Information-Stations/egc4-d24i?

b. Натисніть **View Data** (Переглянути дані).



У цьому наборі даних перераховуються температура доріг і повітря від датчиків, вбудованих на мостах та наземних вулицях в межах міста Сіетла. Дані оновлюються кожні п’ятнадцять хвилин.

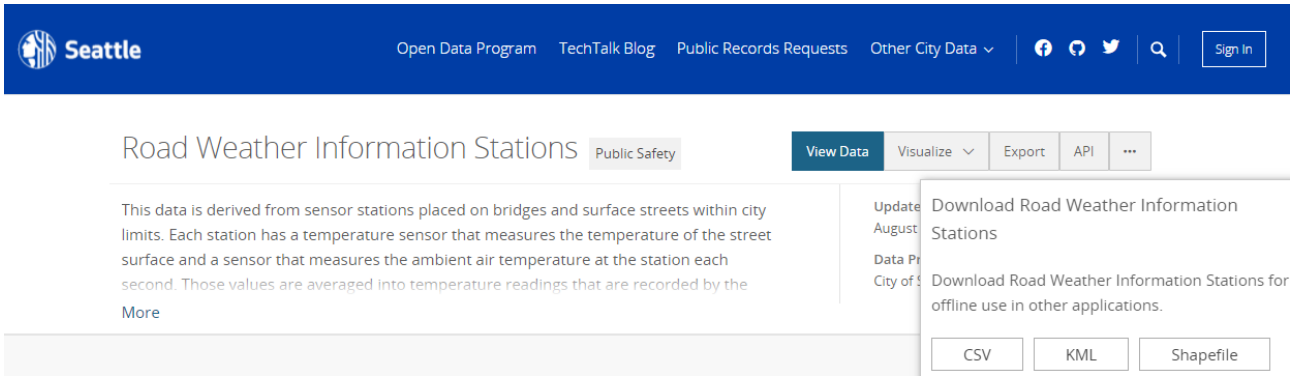
Скільки полів (стовпців) є в наборі даних?

Скільки записів (рядків)? Які назви полів?

c. Натисніть **Export**. Натисніть CSV, щоб завантажити цей набір даних як CSV. Завантаження займе кілька хвилин. Якщо ви не можете завантажити файл, перейдіть до наступної частини.

Якщо вам не вдалося повністю завантажити файл Road_Weather_Information_Stations.csv, знайдіть у своїй файловій системі Road_Weather_Information_Stations.csv.part.

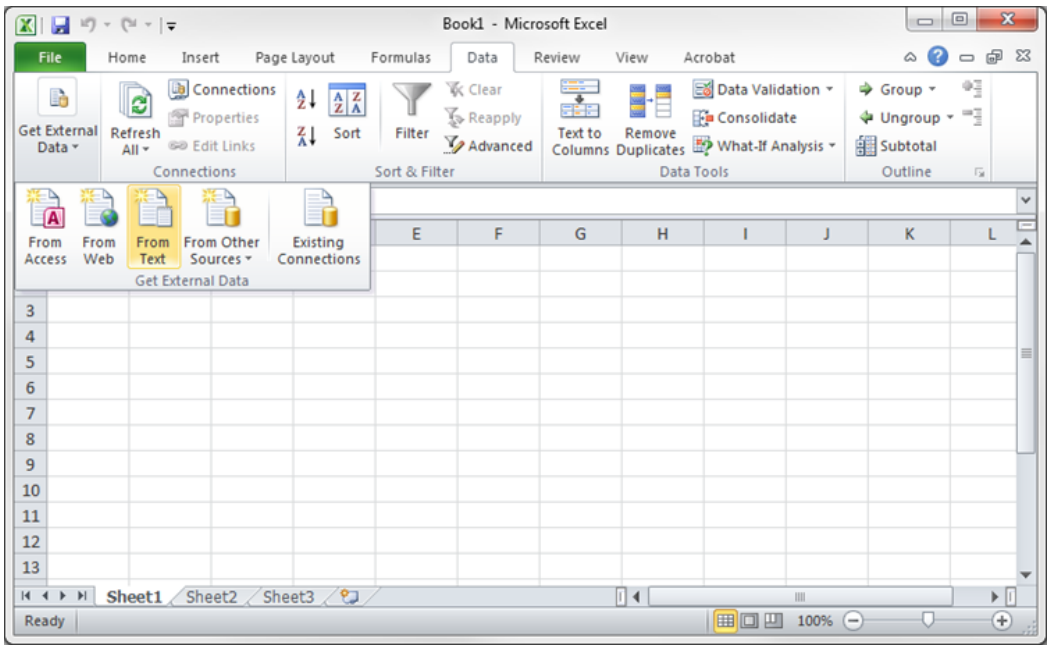
Зазвичай файл знаходиться у папці "Завантажити". Змініть назву файлу на Road_Weather_Information_Stations.csv.



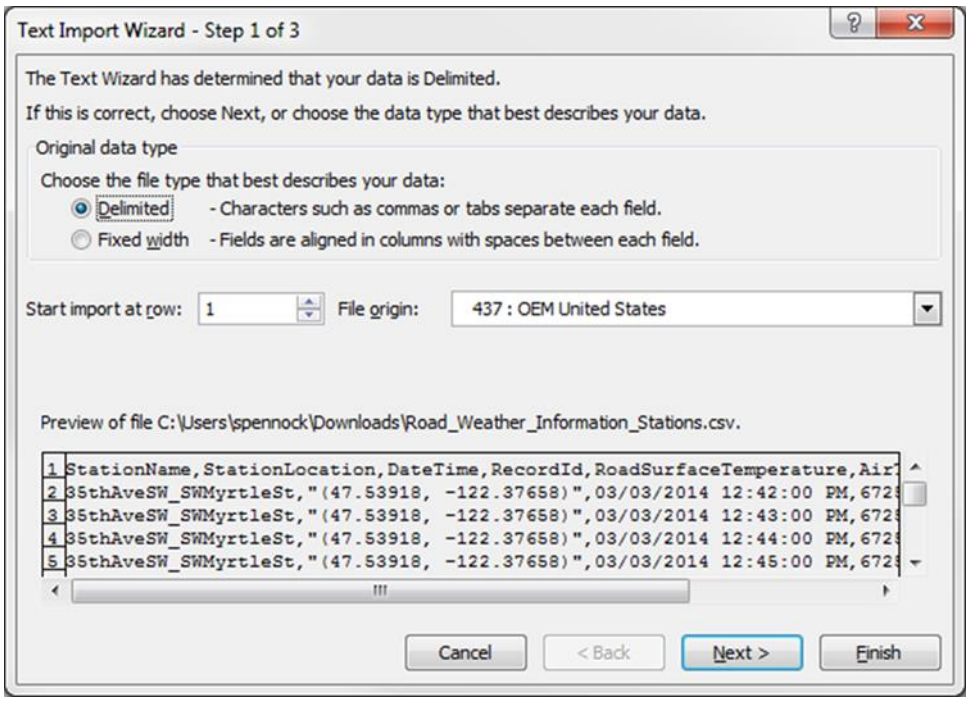
Завантаживши набір даних, відкрийте його у вибраній програмі електронних таблиць. Якщо вам не вдалося завантажити файл, ви можете переглянути кроки та вивчити обмеження електронних таблиць, не відкриваючи файл у програмі електронних таблиць.

Відкрийте програму електронних таблиць, наприклад, Microsoft Excel.

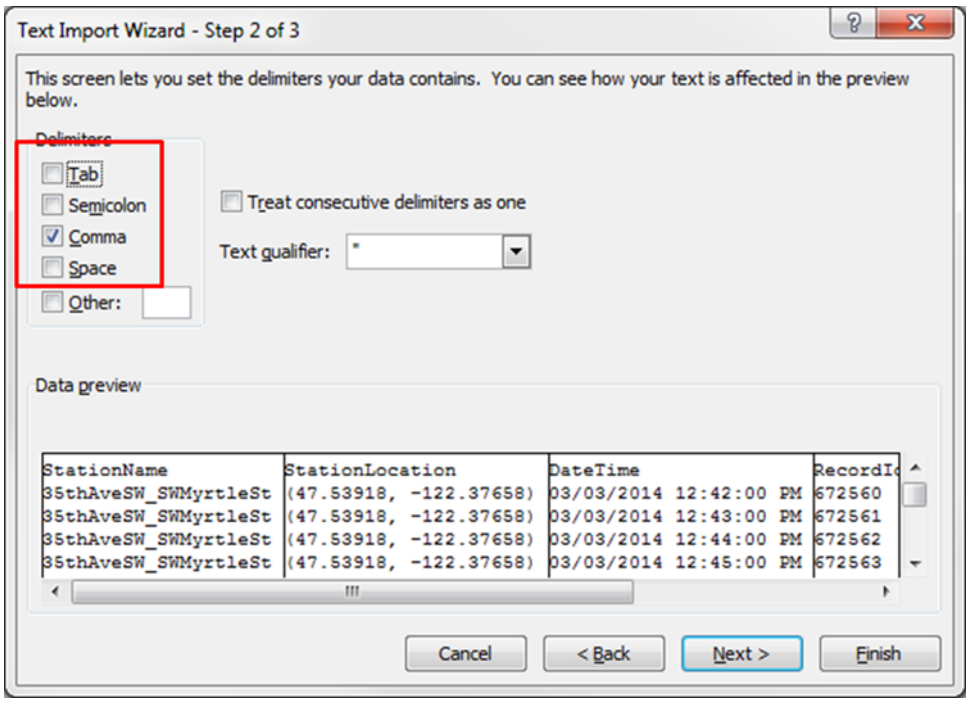
Імпортуйте завантажений файл Road_Weather_Information_Stations.csv. Натисніть **Get External Data** (Отримати зовнішні дані). Натисніть **From Text** та виберіть Road_Weather_Information_Stations.csv.



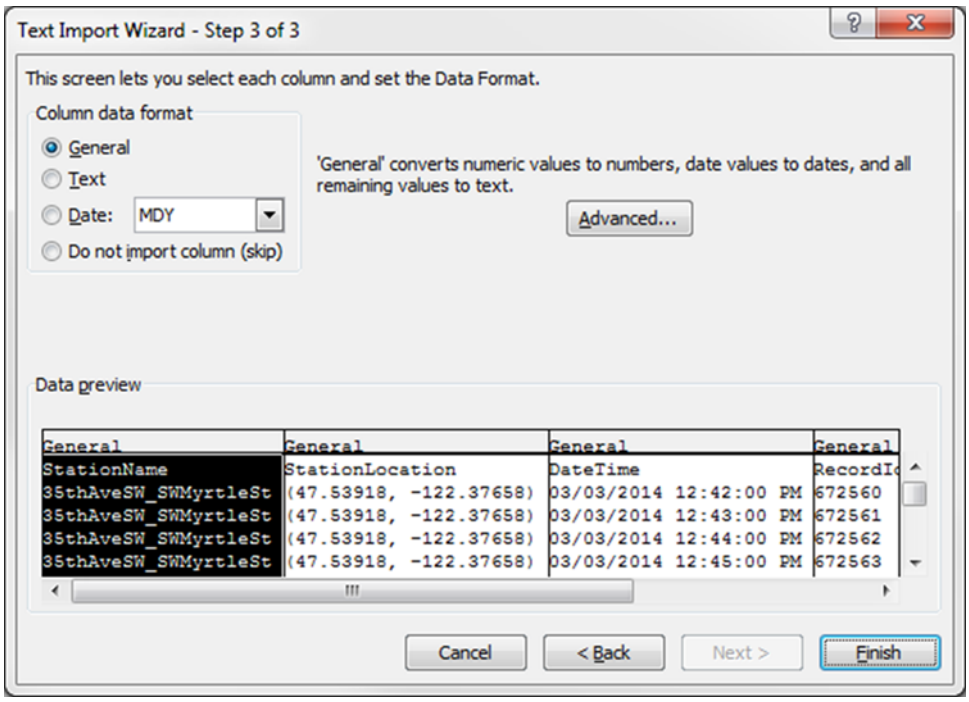
Відкриється майстер імпорту тексту. Натисніть Далі, щоб продовжити.



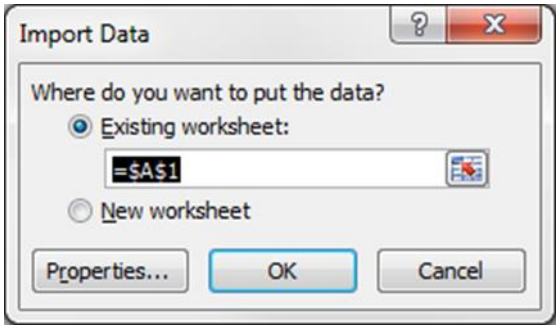
Виберіть **Comma** як роздільник і зніміть прапорець Tab. Натисніть Далі, щоб продовжити.



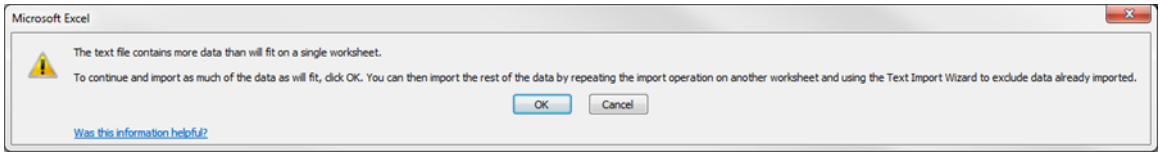
Натисніть **Finish**, щоб продовжити.



Натисніть OK, щоб імпортувати дані.



Яке попереджувальне повідомлення ви отримали?



Як це обмеження перешкоджатиме вашому аналізу даних?

Відкрийте нову електронну таблицю. Спробуйте досягти максимальної кількості рядків у таблиці. Яка максимальна кількість рядків у таблиці на ваш вибір? Порівняйте максимальну кількість рядків у таблиці з кількістю записів у наборі даних дорожніх метеорологічних станцій з https://data.seattle.gov. Використовуючи це спостереження, поясніть обмеження електронної таблиці.

Якщо популярні програми електронних таблиць не можуть обробляти більші набори даних, які інструменти аналізу даних доступні? Використайте Інтернет для пошуку можливих інструментів.

Вимоги до оформлення звіту

Звіт має включати:

1. Титульний аркуш.
2. Завдання на комп'ютерний практикум.
3. Хід роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до комп'ютерного практикуму.
4. Висновки.

Питання для самоперевірки

1. Які дані є відкритими?
2. Яка мета партнерства Open Government Partnership?
3. У яких форматах представляються дані з відкритих джерел?
4. Для чого використовується вебсайт knoema.com?
5. Яка основна місія Gapminder Foundation?
6. Які політики конфіденційності персональних даних в соціальних мережах ви знаєте?
7. Які обмеження електронних таблиць для зберігання великих наборів даних?

Рекомендована література

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics
2. Open Government Partnership // Електронний ресурс. Режим доступу: www.opengovpartnership.org/
3. World Data Atlas // Електронний ресурс. Режим доступу: https://knoema.com/atlas
4. Gapminder Foundation // Електронний ресурс. Режим доступу: https://www.gapminder.org/
5. Seattle Open Data // Електронний ресурс. Режим доступу: https://data.seattle.gov/

КОМП'ЮТЕРНИЙ ПРАКТИКУМ 2.
АНАЛІЗ ТА ВІЗУАЛІЗАЦІЯ ДАНИХ У PYTHON

Мета роботи*:* продемонструвати свої знання про життєвий цикл аналізу даних, використовуючи заданий набір даних та вказані інструменти Python.

Теоретичні відомості

NumPy – це основний пакет для наукових обчислень з Python. Він містить потужний N-вимірний об'єкт масиву та складні (трансляційні) функції.

Pandas – це бібліотека з ліцензією BSD з відкритим кодом, що забезпечує високопродуктивні, прості у використанні структури даних та засоби аналізу даних для мови програмування Python.

Matplotlib – це бібліотека для побудови графіків для мови програмування Python та її числового математичного розширення NumPy.

Folim – це бібліотека для створення інтерактивної карти.

Завдання до комп'ютерного практикуму

Частина 1. Імпорт пакетів Python

Потрібно імпортувати пакети Python: pandas, numpy, matplotlib, folium, datetime та csv, необхідні для аналізу та візуалізації набору даних, що містить інформацію про злочини в Сан-Франциско (файл даних Map-Crime_Incidents-Previous_Three_Months.csv).

```
# Code cell 1
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import folium
```

Частина 2. Завантаження даних

Потрібно завантажити набір даних про злочини в Сан-Франциско (SF County). Завантажте дані про злочини Сан-Франциско у дата фрейм.

Імпортуйте дані про злочини в Сан-Франциско з файлу значень, відокремлених комами (comma-separated values, CSV), у фрейм даних.

```
# code cell 2
# This should be a local path
dataset_path = './Data/Map-Crime_Incidents-Previous_Three_Months.csv'
# read the original dataset (in comma separated values format) into a DataFrame
SF = pd.read_csv(dataset_path)
```

Потрібно використати засоби Python та Jupyter, щоб підготувати ці дані до аналізу, проаналізувати їх, побудувати графіки та повідомити про свої результати. Для перегляду перших п'яти рядків файлу csv використовується команда Linux **head**.

```
# code cell 3
!head -n 5 ./Data/Map-Crime_Incidents-Previous_Three_Months.csv
```

Перегляньте імпортовані дані. Набравши в клітинку ім'я змінної дата фрейму, ви можете структуровано візуалізувати верхні та нижні рядки.

```
# Code cell 4
pd.set_option('display.max_rows', 10) #Visualize 10 rows
SF
```

Використовуйте функцію **columns** для перегляду імені змінних у DataFrame.

```
# Code cell 5
SF.columns
```

Скільки змінних міститься у фреймі даних SF?

За допомогою функції **len** визначте кількість рядків у наборі даних.

```
# Code cell 6
len(SF)
```

Частина 3. Підготовка даних

Тепер, коли ви завантажили дані в робоче середовище, потрібно підготувати дані до аналізу.

а. Витягніть місяць і день із поля Дата.

Lambda – це ключове слово Python для визначення анонімних функцій. Lambda дозволяє вказати функцію в одному рядку коду, не використовуючи def та не визначаючи для неї конкретного імені. Синтаксис lambda виразу: **lambda parameters : expression**. Далі функція lambda

використовується для створення вбудованої функції, яка вибирає лише цифри місяця зі змінної **Date**, і **int** для перетворення рядкового подання у ціле число. Потім функція pandas **apply** використовується для застосування цієї функції до цілого стовпця (apply неявно визначає цикл for і передає один за одним рядки до lambda функції). Таку саму процедуру можна зробити для дня.

```
# Code cell 7
SF['Month'] = SF['Date'].apply(lambda row: int(row[0:2]))
SF['Day'] = SF['Date'].apply(lambda row: int(row[3:5]))
```

Щоб перевірити, чи ці дві змінні були додані до дата фрейму SF, використайте функцію **print**, щоб надрукувати деякі значення з цих стовпців і перевірити **type**, чи ці нові стовпці дійсно містять числові значення.

```
# Code cell 8
print(SF['Month'][0:2])
print(SF['Day'][0:2])
# Code cell 9
print(type(SF['Month'][0]))
```

b. Видаліть змінні з дата фрейму SF.

Стовпець IncidntNum містить багато комірок з NaN. У цьому випадку дані відсутні. Стовпець можна вилучити з дата фрейму. Одним із способів видалення небажаних змінних у фреймі даних є використання функції **del**.

```
# Code cell 10
del SF['IncidntNum']
```

Аналогічно, атрибут Location не буде використовуватися в цьому аналізі. Його можна викинути з дата фрейму. Можна використовувати функцію **drop** для дата фрейму, вказавши, що *вісь* дорівнює 1 (0 для рядків), і що команда не вимагає присвоєння іншому значенню для збереження результату (*inplace* = *True*).

```
# Code cell 11
SF.drop('Location', axis=1, inplace=True )
```

Переконайтеся, що стовпці видалено.

```
# Code cell 12
SF.columns
```

Частина 4. Аналіз даних

Тепер, коли дата фрейм підготовлений з даними, настав час проаналізувати дані.

a. Узагальніть змінні для отримання статистичної інформації.

Використовуйте функцію **value_counts** для підсумовування кількості злочинів, скоєних за типом, а потім **print** для відображення вмісту змінної *CountCategory*.

```
# Code cell 13
CountCategory = SF['Category'].value_counts()
print(CountCategory)
```

За замовчуванням підрахунок впорядковується за спаданням. Значення необов'язкового параметра за *зростанням* можна встановити на *True*, щоб змінити цю поведінку.

```
# Code cell 14
SF['Category'].value_counts(ascending=True)
```

Якого виду злочину було скоєно найбільше?

Вклавши дві функції в одну команду, ви можете досягти одного результату за допомогою одного рядка коду.

```
# Code cell 15
print(SF['Category'].value_counts(ascending=True))
```

У якому PdDistrict було найбільше випадків зареєстрованих злочинів?

Надайте команди Python для підтримки вашої відповіді.

```
# code cell 16
# Possible code for the challenge question
print(SF['PdDistrict'].value_counts(ascending=True))
```

b. Підгрупуйте дані у менші дата фрейми (кадри даних).

За допомогою логічного індексування можна вибрати лише ті рядки, для яких виконується дана умова. Витягніть лише ті злочини, скоєні в серпні, і збережіть результат у новому DataFrame.

```
# Code cell 17
AugustCrimes = SF[SF['Month'] == 8]
AugustCrimes
```

Скільки випадків злочинів було за серпень?

Скільки квартирних крадіжок було зареєстровано у серпні?

```
# code cell 18
# Possible code for the question: How many burglaries were reported in the
month  of August?
AugustCrimes = SF[SF['Month'] == 8]
AugustCrimesB = SF[SF['Category'] == 'BURGLARY']
len(AugustCrimesB)
```

Щоб створити підмножину кадру даних SF для певного дня, використайте операнд **query** функції для порівняння місяця та дня одночасно.

```
# Code cell 19
Crime0704 = SF.query('Month == 7 and Day == 4')
Crime0704
# Code cell 20
SF.columns
```

Частина 5. Представлення даних

Візуалізація та подання даних забезпечує миттєвий огляд, який може бути не очевидним, просто переглянувши вихідні дані. Дата фрейм SF містить координати довготи та широти, які можна використовувати для аналізу даних.

а. Побудуйте графік дата фрейму SF, використовуючи змінні X та Y.

Використайте функцію **plot()** для побудови кадру даних SF. Використовуйте необов'язковий параметр, щоб побудувати графік червоним кольором і встановити фігуру маркера в коло за допомогою *ro*.

```
# Code cell 21
plt.plot(SF['X'],SF['Y'], 'ro')
plt.show()
```

Визначте номери відділів поліції, складіть словник *pd_districts*, щоб зв'язати їх рядок із цілим числом.

```
# Code cell 22
pd_districts = np.unique(SF['PdDistrict'])
pd_districts_levels = dict(zip(pd_districts, range(len(pd_districts))))
pd_districts_levels
```

Використайте **apply** та **lambda**, щоб додати ціле число для поліцейського відділу до нового стовпця DataFrame.

```
# Code cell 23
SF['PdDistrictCode'] = SF['PdDistrict'].apply(lambda row:
pd_districts_levels[row])
```

Використайте щойно створений *PdDistrictCode* для автоматичної зміни кольору.

```
# Code cell 24
plt.scatter(SF['X'], SF['Y'], c=SF['PdDistrictCode'])
plt.show()
```

b. Додайте пакети для побудови карт, щоб покращити сюжет.

Ви створили простий сюжет, який показує, де мали місце злочини у окрузі SF. Цей графік корисний, але **folium** надає додаткові функції, які дозволять вам накласти цей графік на карту OpenStreet.

Folium вимагає вказувати колір маркера з використанням шістнадцяткового значення. З цієї причини використайте пакет *colors* і виберіть необхідні кольори.

```
# Code cell 25
from matplotlib import colors
districts = np.unique(SF['PdDistrict'])
print(list(colors.cnames.values())[0:len(districts)])
```

Створіть словник кольорів для кожного окружного відділу поліції.

```
# Code cell 26
color_dict = dict(zip(districts, list(colors.cnames.values())[0:-
1:len(districts)]))
color_dict
```

Створіть карту, використовуючи середні координати даних SF, щоб відцентрувати карту (за допомогою **mean**).

Щоб зменшити час обчислення, використайте *plotEvery* для обмеження кількості побудованих даних. Встановіть це значення в 1 для побудови всіх рядків (візуалізація карти може зайняти багато часу).

```
# Code cell 27
# Create map
map_osm = folium.Map(location=[SF['Y'].mean(), SF['X'].mean()], zoom_start =
12)
plotEvery = 50
obs = list(zip( SF['Y'], SF['X'], SF['PdDistrict']))
```

```
for el in obs[0:-1:plotEvery]:
        folium.CircleMarker(el[0:2], color=color_dict[el[2]],
fill_color=el[2],radius=10).add_to(map_osm)
```

```
# Code cell 28
map_osm
```

Вимоги до оформлення звіту

Звіт має включати:

1. Титульний аркуш.
2. Завдання на комп'ютерний практикум.
3. Хід роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до комп'ютерного практикуму.
4. Висновки.

Питання для самоперевірки

1. Які пакети Python необхідні для аналізу та візуалізації набору даних?
2. Як відбувається завантаження даних у Python?
3. Яка команда використовується для перегляду перших п'яти рядків файлу?
4. Для чого використовуються пакети Python: pandas, numpy, matplotlib, folium, datetime та csv?
5. Які засоби Python використовуються для візуалізації даних?

Рекомендована література

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics
2. Pandas // Електронний ресурс. Режим доступу: https://pandas.pydata.org/
3. NumPy // Електронний ресурс. Режим доступу: https://numpy.org/
4. Matplotlib // Електронний ресурс. Режим доступу: https://matplotlib.org/
5. Guide to Getting Started with Geospatial Analysis using Folium // Електронний ресурс. Режим доступу: https://www.analyticsvidhya.com/blog/2020/06/guide-geospatial-analysis-folium-python/

КОМП’ЮТЕРНИЙ ПРАКТИКУМ 3.
КОРЕЛЯЦІЙНИЙ АНАЛІЗ У PYTHON

Мета роботи*:* продемонструвати практичні навички кореляційного аналізу даних, використовуючи заданий набір даних та вказані інструменти Python.

Теоретичні відомості

Кореляційний аналіз – це статистичне дослідження стохастичної залежності між випадковими величинами (англ. correlation — взаємозв'язок). У найпростішому випадку досліджують дві вибірки (набори даних), у загальному – їх багатовимірні комплекси (групи).

Мета кореляційного аналізу – виявити, чи існує істотна залежність однієї змінної від інших. Кореляція може бути позитивною та негативною.

Від'ємна кореляція – кореляція, при якій збільшення однієї змінної пов'язане зі зменшенням іншої, при цьому коефіцієнт кореляції від'ємний. Додатна кореляція – кореляція, при якій збільшення однієї змінної пов'язане зі збільшенням іншої, при цьому коефіцієнт кореляції додатний.

Найбільш відомою мірою залежності двох величин є **коефіцієнт кореляції Пірсона**, який зазвичай називають спрощено коефіцієнтом кореляції. Він розраховується як відношення коваріації двох випадкових величин на добуток їх стандартних відхилень. Взаємозв’язок між змінними чисельно характеризується за допомогою коефіцієнту кореляції *r*. Коефіцієнт *r* є випадковою величиною, оскільки обчислюється з випадкових величин. Це лінійний коефіцієнт кореляції, який показує лінійний взаємозв’язок між двома змінними і коливається в межах від -1 до 1 (табл. 1.1). За відсутності лінійного зв’язку значення *r* буде близьким до 0.

Таблиця 3.1

Лінійний коефіцієнт кореляції

Значення *r*	Рівень зв'язку між змінними
0,75 – 1.00	дуже високий позитивний
0,50 – 0.74	високий позитивний
0,25 – 0.49	середній позитивний
0,00 – 0.24	слабкий позитивний
0,00 – -0.24	слабкий негативний
-0,25 – -0.49	середній негативний
-0,50 – -0.74	високий негативний
-0,75 – -1.00	дуже високий негативний

Кореляційний аналіз може виконуватися з використанням методу Пірсона або рангового методу Спірмена.

Метод Пірсона використовується для розрахунків, які вимагають точного визначення ступеня зв'язку, що існує між змінними. Досліджувані з його допомогою ознаки повинні виражатися тільки кількісно. Коефіцієнт кореляції обчислюється за формулою:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}. \qquad (3.1)$$

Завдання до комп'ютерного практикуму

У частині 1 ви налаштуєте набір даних. У частині 2 ви дізнаєтеся, як визначити, чи змінні в даному наборі даних є корельованими. У частині 3 ви будете використовувати Python для обчислення кореляції між двома наборами змінних. У частині 4 ви здійсните візуалізацію результатів дослідження.

Для виконання комп'ютерного практикуму необхідні ресурси: 1 ПК з доступом до Інтернету, бібліотеки Python: pandas, numpy, matplotlib, seaborn, файл даних: brainsize.txt.

Частина 1. Завантаження набору даних

Використайте набір даних, який містить зразок 40 студентів з Southwestern university. Для дослідження використовується чотири субтести (словниковий запас, схожість, дизайн блоків та доповнення картинок) переглянутої Wechsler Adult Intelligence Scale (1981). Дослідники використовували магнітно-резонансну томографію (МРТ) для визначення розміру мозку досліджуваних. Також була включена інформація про стать та розмір тіла (зріст та вага). До набору даних застосовано дві прості модифікації:

1. Заміна знаків питання, що використовуються для представлення утриманих точок даних рядком 'NaN'. Заміна була здійснена, оскільки Pandas неправильно обробляє знаки питання.
2. Заміна усіх символів табуляції комами, перетворивши набір даних у набір даних CSV.

Підготовлений набір даних зберігається як brainsize.txt.

a. Завантаження набору даних із файлу.

Перш ніж набір даних можна використовувати, його потрібно завантажити в пам’ять. У наведеному нижче коді перший рядок імпортує модулі pandas та визначається **pd** як дескриптор, який посилається на модуль.

Другий рядок завантажує CSV-файл набору даних у змінну з назвою **brainFile**. Третій рядок **read_csv()** – це метод pandas для перетворення CSV набору даних, збережений в brainFile в dataframe. Потім фрейм даних зберігається у змінній brainFrame.

Запустіть код нижче, щоб виконати описані функції.

```
# Code cell 1
import pandas as pd
brainFile = './Data/brainsize.txt'
brainFrame = pd.read_csv(brainFile)
```

b. Перевірка дата фрейму.

Щоб переконатися, що фрейм даних правильно завантажений і створений, скористайтеся методом **head().** Метод pandas **head()** відображає перші п’ять записів кадру даних.

```
# Code cell 2
brainFrame.head()
```

Частина 2. Діаграми розсіювання та корельовані змінні

a. Метод describe().

Модуль pandas включає метод **describe()**, який виконує однакові загальні обчислення до даного набору даних. На додаток до загальних результатів, включаючи кількість, середнє, стандартне відхилення, мінімальне та максимальне, **describe()** дозволяє швидко тестувати достовірність значень у фреймі даних. Виконайте код нижче, щоб вивести результати виконання методу **describe()**, обчислені для дата фрейму brainFrame.

```
# Code cell 3
brainFrame.describe()
```

b. Графіки діаграм розсіювання.

Графіки діаграм розсіювання потрібні для кореляційного аналізу, оскільки вони дозволяють швидко візуально перевірити природу взаємозв'язку між змінними. У цій роботі використовується коефіцієнт кореляції Пірсона, який чутливий лише до лінійного співвідношення між двома змінними.

Перш ніж побудувати графіки, необхідно імпортувати кілька модулів, а саме **numpy** та **matplotlib**. Запустіть код нижче, щоб завантажити ці модулі.

```
# Code cell 4
import numpy as np
import matplotlib.pyplot as plt
```

Відокремте дані.

Щоб результати не спотворювались через різницю в чоловічому та жіночому тілах, дата фрейм розділений на два кадри даних: один, що містить усі записи чоловіків, а інший – лише жіночі випадки. Запуск коду нижче

створює два нових кадри даних, **menDf** і **womenDf**, кожен з яких містить відповідні записи.

```
# Code cell 5
menDf = brainFrame[(brainFrame.Gender == 'Male')]
womenDf = brainFrame[(brainFrame.Gender == 'Female')]
```

Побудуйте графіки.

Оскільки набір даних включає три різні показники інтелекту (PIQ, FSIQ та VIQ), перший рядок коду нижче використовує метод **mean()** для обчислення середнього значення між трьома значеннями та збереження результату у змінну **menMeanSmarts**. Зверніть увагу, що перший рядок також стосується **menDf**, відфільтрованого кадру даних, що містить лише чоловічі записи.

Другий рядок використовує matplotlib метод **scatter()** для створення діаграми розсіювання між menMeanSmarts змінною та MRI_Count атрибутом. MRI_Count у цьому наборі даних можна вважати мірою фізичного розміру мозку досліджуваних. Третій рядок відображає графік. Четвертий рядок використовується для відображення графіку.

```
# Code cell 6
menMeanSmarts = menDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(menMeanSmarts, menDf["MRI_Count"])
plt.show()
%matplotlib inline
```

Створіть графік розбіжності для відфільтрованого кадру даних лише для жінок.

```
# Code cell 7
# Graph the women-only filtered dataframe
#womenMeanSmarts = ?
#plt.scatter(?, ?)
plt.show()
%matplotlib inline
```

Частина 3. Обчислення кореляції в Python

Метод pandas **corr()** забезпечує простий спосіб обчислення кореляції для кадру даних. Викличте метод для фрейму даних, щоб отримати кореляцію між усіма змінними одночасно:

```
# Code cell 8
brainFrame.corr(method='pearson')
```

Зверніть увагу на діагональ зліва направо у таблиці кореляцій, сформованій вище. Чому діагональ заповнена 1s? Це випадковість? Відповідь поясніть.

Не дивлячись на таблицю кореляцій вище, зауважте, що значення відображаються дзеркально; значення нижче 1 діагоналі мають дзеркальний аналог вище 1 діагоналі. Це випадковість? Відповідь поясніть.

Використовуючи метод **corr()**, розрахуйте кореляцію змінних, що містяться в кадрі даних лише для жінок:

```
# Code cell 9
womenDf.corr(method='pearson')
```

Те саме виконайте для кадру даних, призначеного лише для чоловіків:

```
# Code cell 10
# Use corr() for the male-only dataframe with the pearson method
#?.corr(?)
```

Частина 4. Візуалізація даних

a. Встановіть Seaborn.

Для спрощення візуалізації кореляцій даних можна використовувати графіки кореляційних карт. На основі кольорових квадратів графіки кореляційних карт можуть допомогти швидко виявити кореляційні зв’язки. За допомогою модуля Python **seaborn** можна будувати графіки кореляційних карт. Спочатку запустіть код нижче, щоб завантажити та встановити модуль **seaborn**.

```
# Code cell 11
!pip install seaborn
```

b. Побудуйте графік кореляційної карти.

Тепер, коли кадри даних готові, можна побудувати кореляційні карти.

Рядок 1: Створює таблицю кореляції на основі women NoGenderDf кадру даних та зберігає її в **wcorr**.

Рядок 2: Використовує seaborn метод **heatmap()** для формування та побудови графіку кореляційної (теплової) карти. Зверніть увагу, що **heatmap()** приймає **wcorr** в якості параметра.

Рядок 3: Використовуйте для експорту та збереження сформованої теплової карти як зображення PNG. Поки рядок 3 не активний (перед ним є # – символ коментаря, що змушує інтерпретатор ігнорувати його), він зберігався з інформаційною метою.

```
# Code cell 12
import seaborn as sns
wcorr = womenDf.corr()
sns.heatmap(wcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

Подібним чином створіть теплову карту для кадру даних лише для чоловіків.

```
# Code cell 14
mcorr = menDf.corr()
sns.heatmap(mcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

Багато пар змінних мають кореляцію, близьку до нуля. Що це означає? Навіщо потрібно розділяти стать?

Які змінні мають сильнішу кореляцію з розміром мозку (MRI_Count)? Чи це є очікуваним? Відповідь поясніть.

Вимоги до оформлення звіту

Звіт має включати:

1. Титульний аркуш.
2. Завдання на комп’ютерний практикум.
3. Хід роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до комп’ютерного практикуму.
4. Висновки.

Питання для самоперевірки

1. Для чого використовується кореляційний аналіз даних?
2. Як розраховується коефіцієнт кореляції Пірсона?
3. Для чого використовується метод head()?
4. Для чого використовується метод describe()?
5. Для чого використовується метод scatter()?
6. Для чого використовується метод corr()?

Рекомендована література

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics.
2. Кореляційний аналіз // Електронний ресурс. Режим доступу: https://pidru4niki.com/12461220/statistika/korelyatsiyniy_analiz
3. Wechsler Adult Intelligence Scale // Електронний ресурс. Режим доступу: https://en.wikipedia.org/wiki/Wechsler_Adult_Intelligence_Scale
4. Better Heatmaps and Correlation Matrix Plots in Python // Електронний ресурс. Режим доступу: https://towardsdatascience.com/better-heatmaps-and-correlation-matrix-plots-in-python-41445d0f2bec

КОМП'ЮТЕРНИЙ ПРАКТИКУМ 4.
ПОБУДОВА ЛІНІЙНОЇ РЕГРЕСІЇ В PYTHON

Мета роботи*:* ознайомитись з поняттями лінійної регресії та роботи з даними для прогнозування в Python, проаналізувати запропоновані дані про продажі та побудувати лінійну регресію для прогнозування річного чистого обсягу продажів на основі кількості магазинів у районі.

Теоретичні відомості

У статистиці лінійна регресія – це спосіб моделювання взаємозв'язку між залежною змінною і незалежною змінною. Регресійна модель – це функція незалежної величини та коефіцієнтів з включеними випадковими змінними.

Регресійний аналіз використовується для прогнозу, аналізу часових рядів, тестування гіпотез та виявлення прихованих взаємозв'язків в даних.

Регресійний аналіз тісно пов'язаний з кореляційним аналізом. У кореляційному аналізі досліджується напрям та міцність зв'язку між незалежними змінними. В регресійному аналізі досліджується форма залежності (модель зв'язку, вираженої у функції регресії) між незалежними змінними.

Регресія – залежність математичного очікування (середнього значення) незалежної змінної від однієї або декількох інших змінних. Нехай у точках x_n незалежної змінної x отримані виміри Y_n. Потрібно знайти залежність середнього значення величини $\overline{Y}$ від величини x, тобто $\overline{Y}(x) = f(x|a)$, де a – вектор невідомих параметрів a_i. Функцію $f(x|a)$ називають функцією регресії. Припускають, що $f(x|a)$ є лінійною функцією параметрів a, тобто має вигляд:

$$f(x|a) = \sum_{i=1}^{I} a_i \varphi_i(x), \qquad (4.1)$$

де $f_i(x)$ – задані функції, для визначення параметрів a_i використовують метод найменших квадратів (МНК).

Регресійний аналіз використовується у випадку, якщо відношення між змінними можуть бути виражені кількісно у вигляді деякої комбінації цих змінних. Отримана комбінація використовується для передбачення значення, що може приймати залежна змінна, яка обчислюється на заданому наборі значень незалежних змінних. У найпростішому випадку для цього використовується лінійна регресія.

Необхідно зазначити, що серед регресійних моделей виділяють:

– однопараметричні моделі (залежність від однієї змінної);

– багатопараметричні моделі (залежність від декількох змінних);

– лінійні моделі відносно незалежних змінних;

– моделі нелінійні за змінними та нелінійні за параметрами.

Лінійна регресійна модель має наступний вигляд:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + u \qquad (4.2)$$

де y – залежна змінна; $(x_1, x_2, ..., x_n)$ – незалежні змінні; u – випадкова похибка, розподіл якої в загальному випадку залежить від незалежних змінних, але математичне очікування якої рівне нулю.

Згідно з моделлю (4.2) математичне очікування залежної змінної є лінійною функцією незалежних змінних:

$$E(y) = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + u \qquad (4.3)$$

Вектор параметрів $(\beta_0, \beta_1, ..., \beta_k)$ є невідомим і задача лінійної регресії полягає в оцінці цих параметрів на основі деяких експериментальних значень y і $(x_1, x_2, ..., x_n)$. Тобто для n експериментів є відомі значення $\{y_i, y_{i1}, ..., y_{ip}\}_{i=1}^{n}$ незалежних змінних і відповідне їм значення залежної змінної.

Згідно з визначенням моделі для кожного експериментального випадку залежність між змінними визначається формулою:

$$y_i = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + u_i . \qquad (4.4)$$

На основі цих даних потрібно оцінити значення параметрів $(\beta_0, \beta_1, ..., \beta_k)$, а також розподіл випадкової величини u. Зважаючи на характеристики досліджуваних змінних, можуть додаватися різні додаткові специфікації моделі і застосовуватися різні методи оцінки параметрів.

В залежності від об'єктів, що досліджуються за допомогою лінійної регресії та конкретних цілей дослідження, можуть використовуватися різні методи оцінки невідомих коефіцієнтів. Найпопулярнішим є **метод найменших квадратів (МНК)**. Він приймає за оцінку змінної значення, що мінімізують суму квадратів залишків по всіх спостереженнях:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} \left| y_i - \beta_0 - \sum_{j=1}^{k} X_{ij}\beta_j \right|^2 = \arg\min_{\beta} \left\| y = X\beta \right\|^2 \qquad (4.5)$$

Завдання до комп'ютерного практикуму

Для виконання комп'ютерного практикуму використовується 1 ПК з доступом до Інтернету, бібліотеки Python: pandas, numpy, scipy, matplotlib та файли даних store-dist.csv.

Частина 1. Імпорт бібліотек та даних

У цій частині потрібно імпортувати бібліотеки та дані з файлу stores-dist.csv. Імпортуйте наступні бібліотеки:

```
# Code Cell 1
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Імпортуйте дані.

На цьому кроці ви імпортуєте дані з файлу stores-dist.csv та переконаєтесь, що файл імпортовано правильно.

```
Code Cell 2
# Import the file, stores-dist.csv
salesDist = pd.read_csv('./Data/stores-dist.csv')
# Verify the imported data
salesDist.head()
```

Перейменуйте заголовки стовпців **annual net sales** і **number of stores in district** для полегшення оброблення даних:

- **annual net sales** на **sales**
- **number of stores in district** на **stores**

```
# Code Cell 3
# The district column has no relevance at this time, so it can be dropped.
salesDist = salesDist.rename(columns={'annual net sales':'sales','number of stores in district':'stores'})
```

- `salesDist.head()`

Частина 2. Відображення даних

a. Визначте кореляцію.

На цьому кроці потрібно дослідити кореляцію даних для регресійного аналізу, а також скинути будь-які не пов'язані між собою стовпці за необхідності.

```
# Code Cell 4
# Check correlation of data prior to doing the analysis
# # Hint: check lab 3.1.5.5
```

З коефіцієнта кореляції виявляється, що стовпець **district** має низьку кореляцію до **annual net sales** і **number of stores in the district** (показати це). Отже, колонка **district** не є необхідною як частина регресійного аналізу і може бути виключена з dataframe.

```
# Code Cell 5
# The district column has no relevance at this time, so it can be dropped.
#sales = salesDist.drop(...)
sales.head()
```

За даними коефіцієнта кореляції, який тип кореляції ви спостерігали між річними чистими продажами та кількістю магазинів у районі?

 Введіть тут свою відповідь.

b. Створіть сюжет.

На цьому кроці створіть графік для візуалізації даних та призначте **stores** (магазини) як незалежну змінну ***x***, а **sales** (продажі) як залежну змінну ***p***.

```
# Code Cell 6

# dependent variable for y axis
y = sales['sales']
# independent variable for x axis
x = sales.stores
```

```
# Code Cell 7
# Display the plot inline
%matplotlib inline
# Increase the size of the plot
plt.figure(figsize=(20,10))
# Create a scatter plot: Number of stores in the District vs. Annual Net
Sales
plt.plot(x,y, 'o', markersize = 15)
# Add axis labels and increase the font size
plt.ylabel('Annual Net Sales', fontsize = 30)
plt.xlabel('Number of Stores in the District', fontsize = 30)
# Increase the font size on the ticks on the x and y axis
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
# Display the scatter plot
plt.show()
```

Частина 3. Побудова простої лінійної регресії

У цій частині ви будете використовувати **numpy** для створення лінії регресії для набору даних.

Ви також розрахуєте центроїд для цього набору даних. **Центроїд** – це середнє значення для набору даних. Сформована проста лінійна лінійна регресія також повинна проходити через центроїд.

a. Обчисліть нахил та перетин Y-лінії лінійної регресії.

```
# Code Cell 8
# Use numpy polyfit for linear regression to fit the data
# Generate the slope of the line (m)
# Generate the y-intercept (b)
m, b = np.polyfit(x,y,1)
print ('The slope of line is {:.2f}.'.format(m))
print ('The y-intercept is {:.2f}.'.format(b))
print ('The best fit simple linear regression line is {:.2f}x +
{:.2f}.'.format(m,b))
```

b. Обчисліть центроїд. Центроїд набору даних обчислюється за допомогою функції середнього значення.

```
# Code Cell 9
# y coordinate for centroid
y_mean = y.mean()
# x coordinate for centroid
x_mean = x.mean()
print ('The centroid for this dataset is x = {:.2f} and y =
{:.2f}.'.format(x_mean, y_mean))
```

c. Накладіть лінію регресії та центральну точку на графіку.

```
# Code Cell 10
# Create the plot inline
%matplotlib inline
# Enlarge the plot size
plt.figure(figsize=(20,10))
# Plot the scatter plot of the data set
plt.plot(x,y, 'o', markersize = 14, label = "Annual Net Sales")
# Plot the centroid point
plt.plot(x_mean,y_mean, '*', markersize = 30, color = "r")
# Plot the linear regression line
plt.plot(x, m*x + b, '-', label = 'Simple Linear Regression Line', linewidth
= 4)
# Create the x and y axis labels
plt.ylabel('Annual Net Sales', fontsize = 30)
plt.xlabel('Number of Stores in District', fontsize = 30)
# Enlarge x and y tick marks
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
# Point out the centroid point in the plot
plt.annotate('Centroid', xy=(x_mean-0.1, y_mean-5), xytext=(x_mean-3,
y_mean-20), arrowprops=dict(facecolor='black', shrink=0.05), fontsize = 30)
# Create legend
plt.legend(loc = 'upper right', fontsize = 20)
```

d. Прогнозування.

Використовуючи лінійну лінійну регресію, передбачте річний чистий обсяг продажів на основі кількості магазинів у районі.

```
# Function to predict the net sales from the regression line
def predict(query):
    if query >= 1:      predict = m * query + b
        return predict
```

```
    else:
        print ("You must have at least 1 store in the district to predict
the annual net sales.")
 # Code Cell 12
# Enter the number of stores in the function to generate the net sales
prediction.
```

Який прогнозований чистий продаж, якщо в районі є 4 магазини?

 Введіть тут свою відповідь.

Вимоги до оформлення звіту

Звіт має включати:

1. Титульний аркуш.
2. Завдання на комп'ютерний практикум.
3. Хід роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до комп'ютерного практикуму.
4. Висновки.

Питання для самоперевірки

1. Яке призначення регресійного аналізу?
2. Для чого використовується лінійна регресія?
3. Як оцінюються її параметри лінійної регресії?
4. Як відбувається побудова лінійної регресії в Python?

Рекомендована література

1. Introduction to IoT (Cisco Networking Academy) // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics
2. Інтелектуальний аналіз даних: методичні вказівки до виконання комп'ютерних практикумів з навчальної дисципліни «Інтелектуальний аналіз даних» / Уклад.: д.б.н., с.н.с. Є. А. Настенко, к.т.н. В. С. Якимчук, к.т.н. О. К. Носовець. – К.: НТУУ «КПІ ім. І. Сікорського», 2017. – 51 с.
3. Linear Regression in Python // Електронний ресурс. Режим доступу: https://realpython.com/linear-regression-in-python/

КОМП'ЮТЕРНИЙ ПРАКТИКУМ 5.
АНАЛІЗ ТА ВІЗУАЛІЗАЦІЯ ДАНИХ В R

Мета роботи*:* ознайомитись з можливостями мови програмування R для аналізу та візуалізації даних, використати бібліотеку R dplyr для очищення та трансформації даних та бібліотеку ggplot2 для візуалізації даних.

Теоретичні відомості

R – це мова програмування, яка широко використовується для аналізу даних. У цій роботі використовуються бібліотеки R:

dplyr: для очищення та трансформації даних;

ggplot2: для візуалізації даних.

Встановлення R: перейдіть за посиланням https://cloud.r-project.org/

• оберіть вашу операційну систему;

• завантажте відповідний пакунок;

• інсталюйте його.

Встановлення RStudio: перейдіть за посиланням https://www.rstudio.com/products/rstudio/download/

• оберіть вашу операційну систему;

• завантажте відповідний пакунок;

• інсталюйте його.

Створити новий файл в RStduio:

• в меню обрати пункт меню **File** ;

• далі **New File** ;

• обрати пункт **RScript.**

R має модульну структуру. Ви встановлюєте базову функціональність і розширяєте її потрібними бібліотеками. Для встановлення бібліотеки використовується команда **install.packages**.

Скопіюйте у R файл ці команди:

```
install.packages('dplyr', dependencies = TRUE)
install.packages('ggplot2', dependencies = TRUE)
```

Щоб виконати код, виділіть рядки та натисніть піктограму **Run** з зеленою стрілкою або комбінацію клавіш CTRL + ENTER або COMMAND + ENTER. Також код можна набирати в консолі (нижня ліва панель в RStudio).

Завантажте ці бібліотеки до робочого середовища. Це можна зробити за допомогою функції **library**. Ми встановлюємо бібліотеку один раз, але завантажувати її потрібно щоразу при запуску RStudio. При наступному запуску RStudio команди інсталяції не будуть потрібні і їх можна буде закоментувати використовуючи символ #:

R як калькулятор

```
2+3-8 # додавання та віднімання
## [1] -3
7*5/2 # множення та ділення
## [1] 17.5
pi # константа пі
## [1] 3.141593
sqrt(4) # корінь квадратний
## [1] 2
2^3 # піднесення до степеня
## [1] 8
Символ присвоєння: <-
# x присвоїти значення 2
x <- 2
# вивести значення x
x
## [1] 2
x та X - різні змінні
X <- 3
X
## [1] 3
x
## [1] 2
```

Типи даних в R

Основні типи: логічні – TRUE, FALSE; рядкові – character; числові – numeric, integer, double, complex.

На відміну від мов Java чи C, в R не обов'язково декларувати тип змінної. Дізнатися, який тип має змінна можна з допомогою функції **class**:

```
v1 <- TRUE
class(v1)
## [1] "logical"
v1 <- -10.6
class(v1)
## [1] "numeric"
v1 <- 3L # L вказує, що це ціле число
class(v1)
## [1] "integer"
v1 <- 3+2i
class(v1)
## [1] "complex"
v1 <- "stats"
class(v1)
## [1] "character"
```

Типи об’єктів R

Основні типи об’єктів R: вектор, матриця, список(list), фактор, таблиця даних (data frame). Вектор – набір значень одного типу. Утворюється з допомогою функції **c** (скорочення від **concatenate**).

```
numeric_vector <- c(1, 10, 49)
boolean_vector <- c(TRUE, FALSE, TRUE)
character_vector<-c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
```

Операції з векторами

```
x <- c(10, 2, 3, 7, 4)
y <- c(2, -1, 3, 2, 6)
# додавання/віднімання
```

```
# додаються/віднімаються поелементно
x + y
## [1] 12 1 6 9 10
x - y
## [1] 8 3 0 5 -2
# множення на скаляр
2*x
## [1] 20 4 6 14 8
# застосування функції до кожного елемента
sqrt(x)
## [1] 3.162278 1.414214 1.732051 2.645751 2.000000
# сума елементів
sum(x)
## [1] 26
# довжина вектора
length(x)
## [1] 5
# об'єданання векторів
z <- c(x, y)
z
## [1] 10 2 3 7 4 2 -1 3 2 6
```

Доступ до елементів вектора

```
x <- 1:20 # інший спосіб задання вектора, вказуємо послідовність від 1 до 20
x
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
# п'ятий елемент (відлік починається з одиниці)
x[5]
## [1] 5
```

```
# елементи з 6 по 12
x[6:12]
## [1] 6 7 8 9 10 11 12
# елементи 6, 10, 13
x[c(6, 10, 13)]
## [1] 6 10 13
# елементи за винятком 6 та 13
x[-c(6, 13)]
## [1] 1 2 3 4 5 7 8 9 10 11 12 14 15 16 17 18 19 20
# елементи, які більші 5
x[x > 5]
## [1] 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
# елементи, які більше 5 і менші 15
x[x > 5 & x < 15]
## [1] 6 7 8 9 10 11 12 13 14
# елементи, які менші 5 або більші 15
x[x < 5 | x > 15]
## [1] 1 2 3 4 16 17 18 19 20
```

Відсутні значення (аналог null) позначаються як **NA** (**Not avaliable**). Впливають на результат обчислень.

```
x <- c(10, 20, NA, 4, NA, 2)
sum(x)
## [1] NA
sum(x, na.rm = TRUE)
## [1] 36
```

Матриця – двовимірний вектор.

```
mat <- matrix(data=c(9,2,3,4,5,6),ncol=3)
mat
## [,1] [,2] [,3]
## [1,] 9 3 5
```

```
## [2,] 2 4 6
```

Список (list). Якщо елементи вектора мають бути одного типу, то елементи списку можуть мати різні типи.

```
a <- list(p_name="Joe", 4, foo=c(3,8,9))
print(a)
## $p_name
## [1] "Joe"
##
## [[2]]
## [1] 4
##
## $foo
## [1] 3 8 9
```

Доступ до елементів з використанням символа [[]] або $ та імені(якщо елемент має ім'я)

```
a[[3]]
## [1] 3 8 9
a[[1]]
## [1] "Joe"
a$p_name
## [1] "Joe"
```

Фактор – вектор для збереження категоріальних даних. Може містити як категоріальні впорядковані, так і категоріальні невпорядковані дані.

Нехай маємо звичайний вектор:

```
mons <-
c("March","April","January","November","January","September","October
","September","November","August",
"January","November","November","February"
,"May","August","July","December","August","August","September",
"November","February","April")
```

```
class(mons)
## [1] "character"
```

Створимо впорядкований фактор, в параметрі level задамо та задамо порядок:

```
mons <-
factor(mons,levels=c("January","February","March","April","May","June
","July","August","September",
"October","November","December"),ordered=TRUE)
class(mons)
## [1] "ordered" "factor"
```

Тепер можемо визначити, чи March < April

```
mons[1]
## [1] March
## 12 Levels: January < February < March < April < May < June < ... <
December
mons[2]
## [1] April
## 12 Levels: January < February < March < April < May < June < ... <
December
mons[1] < mons[2]
## [1] TRUE
```

Дата фрейм

Дата фрейм (data frame) використовується для роботи з таблицями. Є три способи створити data frame.

1. Об'єднати вектори однакової довжини, використовуючи команду data.frame:

```
cause <- c('pilot error', 'mechanical', 'weather', 'sabotage', 'other')
amount <- c(640, 195, 63, 95, 111)
```

```
plane_crash <- data.frame(cause, amount)
```

2. Використовувати вбудовані набори даних:

```
head(airquality)
## Ozone Solar.R Wind Temp Month Day
## 1 41 190 7.4 67 5 1
## 2 36 118 8.0 72 5 2
## 3 12 149 12.6 74 5 3
## 4 18 313 11.5 62 5 4
## 5 NA NA 14.3 56 5 5
## 6 28 NA 14.9 66 5 6
```

Зчитування даних з файла

```
# кількість стовпців
ncol(airquality)
## [1] 6
# кількість рядків
nrow(airquality)
## [1] 153
# назви колонок
colnames(airquality)
## [1] "Ozone" "Solar.R" "Wind" "Temp" "Month" "Day"
# всі дані для 5 місяця
airquality2 <- airquality[airquality$Month == 5, ]
head(airquality2)
```

```
## Ozone Solar.R Wind Temp Month Day
## 1 41 190 7.4 67 5 1
## 2 36 118 8.0 72 5 2
## 3 12 149 12.6 74 5 3
## 4 18 313 11.5 62 5 4
## 5 NA NA 14.3 56 5 5
## 6 28 NA 14.9 66 5 6
# температура для 5 місяця
airquality$Temp[airquality$Month == 5]
## [1] 67 72 74 62 56 66 65 59 61 69 74 69 66 68 58 64 66 57 68 62 59 73 61
## [24] 61 57 58 57 67 81 79 76
```

Встановлення і початок роботи в RStudio

R – це безкоштовне програмне середовище для статистичного аналізу та візуалізації даних, яке складається із наступних основних частин:

• засоби керування даними та система їх зберігання;

• система векторних та матричних розрахунків;

• великий набір інструментів для аналізу даних;

• інструменти візуалізації даних та результатів їх аналізу;

• мова програмування високого рівня, яка включає умовні оператори, цикли, функції, у т.ч. рекурсивні, та системи введення та виведення даних;

• освітні ресурси, які надають можливості для вивчення R та подальших консультацій по роботі із середовищем.

R є вільною (GNU) версією мови програмування S, на відміну від більшості статистичних пакетів (того ж S), які оперують заданим набором функцій, мова R дозволяє створювати власні функції та пакети.

Крім того, для операцій які потребують найбільш ефективної та швидкої роботи у R може бути інтегрований код на мовах програмування C та C++ для прямих маніпуляцій з об'єктами R.

За рахунок відкритості коду та можливості додавання пакетів мова R є надзвичайно гнучкою та різноманітною; існує величезна кількість пакетів, які надають величезні можливості з оперування даними та їх аналізу.

На сьогодні створено понад 12 тис. додаткових бібліотек, які значно розширюють можливості мови. Оскільки мова R доступна на усіх сучасних операційних системах, включаючи Windows, MacOS та UNIX-подібні системи (Linux, FreeBSD та ін.), усі останні досягнення статистичного аналізу одразу з'являються у вигляді пакетів R.

Основним дзеркалом мови R є сайт www.r-project.org. На даному сайті можна знайти усі необхідні програмні засоби, а також звернутися по підтримку.

Встановити і налаштувати базову комплектацію статистичного середовища R дуже просто. Отримати останню версію інсталятора базової програми R можна за адресою: https://cran.r-project.org/bin/windows/base/

Інсталятор завантажується у вигляді exe-файлу. Для інсталяції програми досить запустити цей файл і відповідати на його запитання. При першій спробі роботи з R рекомендовано погоджуватись з усіма пропозиціями, які робить інсталятор.

Вікно консолі R

Після інсталяції R його можна запустити і отримати приблизно таке вікно управління програмою R Console (рис.5.1).

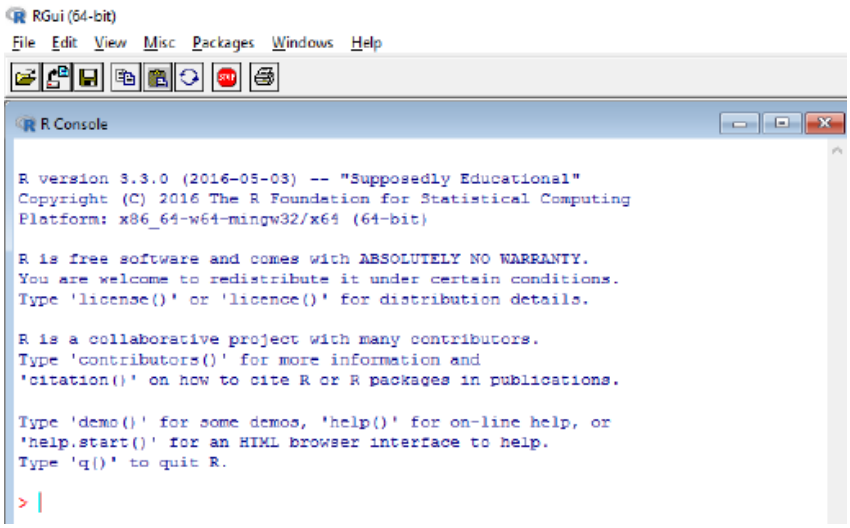


Рис.5.1.

Одразу після запуску з'являється вікно із головним меню, під яким відкрито **«консоль R»,** яка показує хід всіх останніх операцій, і в якій також можна давати інструкції програмі та отримувати її відповіді. Одразу після запуску в даній консолі виведено інформацію про версію програми та підказка щодо отримання допомоги.

Символ > є запрошенням користувачу вводити власні інструкції.

Для перевірки роботи системи можна ввести 2+2 і натиснути Enter. Результат буде виведений у консолі (рис. 5.2).

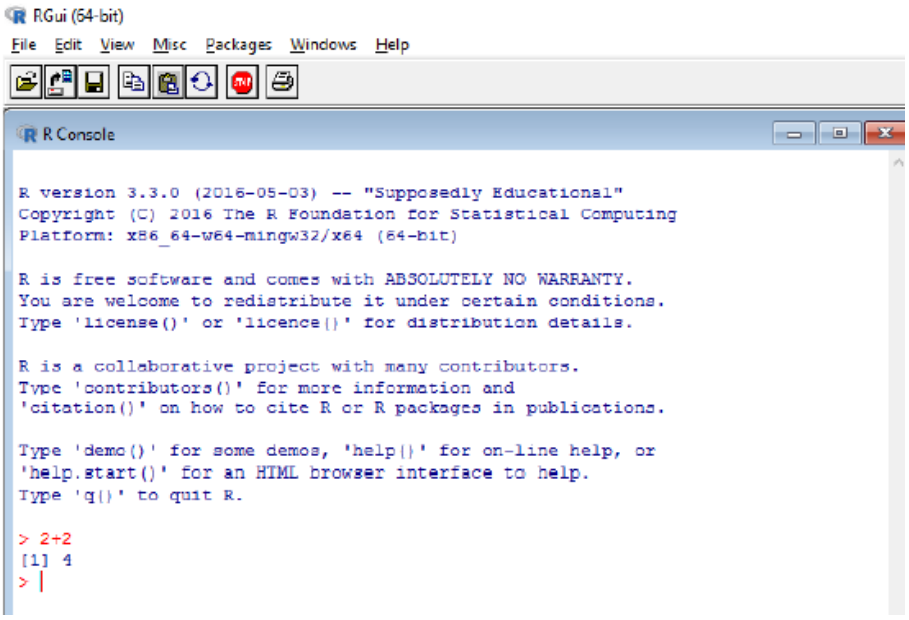


Рис.5.2

R виводить результати виконання безпосередньо після інструкції синім кольором, після чого переходить у режим очікування наступної інструкції, про що повідомляє червоним знаком >.

При роботі з R можна виконувати одразу багато інструкцій, що записані в окремому файлі (script).

Щоб створити скрипт, достатньо натиснути у меню **File ->New Script**. Там за допомогою команди Open Script можна завантажити скрипт, створений раніше (рис.5.3).

Скрипти мають стандартне розширення *.r.

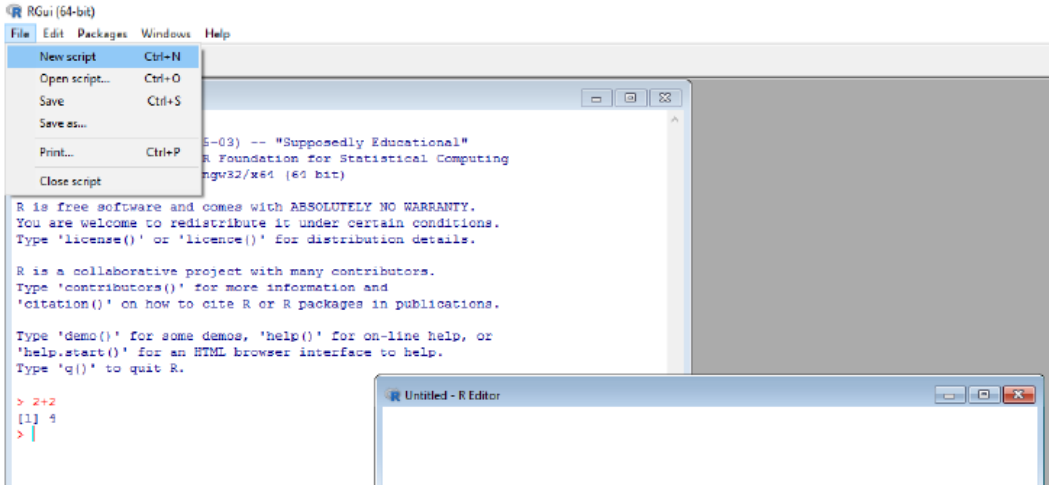


Рис.5.3

При цьому, якщо інструкції у файлі розміщені у окремих рядочках, розділових знаків між ними не потрібно.

Для виконання інструкцій скрипту потрібно натиснути клавішу **F5**. Тоді буде виконана команда, на якій стоїть курсор. Якщо виділити кілька рядків, то після натиснення клавіші **F5** будуть виконані виділені інструкції.

Також виконати завантажений у вікні редактора скрипт повністю можна, використовуючи **Edit->Runall**. Можна також виконати виділену частину скрипту, використовуючи кнопку **Run line or selection**. Закінчивши роботу зі скриптом, його можна зберегти, використовуючи **File->Save**.

Інструкції, вміщені в одному рядочку, розділяють символом «**;**».

Якщо довга команда не вміщується у одному рядку, її можна розбити на кілька рядків, причому, якщо команда не закінчена, при переході до наступного рядка R автоматично виводить символ продовження **+**. **R** здогадується, що команда не закінчена за її синтаксисом. Тому деякі синтаксичні помилки типу забутих дужок можуть сприйматись як незакінчені інструкції. У цьому випадку R виставить **+** на початку наступного рядка і перейде у режим очікування. В такому випадку можна натиснути Escape, щоб перейти у режим введення нової інструкції без продовження аналізу попередньої.

Значок # означає, що все, починаючи з нього до кінця рядка є коментарем.

Приклад.

Створимо числовий вектор **tosses,** компонентами якого є числа 2,1,1,3,6,4:

> **tosses<-c(2,1,1,3,6,4)**

Знак <- означає привласнення. Гострий кінець вказує на об'єкт, якому привласнюється вираження після знаку <-.

Приклад.

>c (2,1,1,3,6,4) -> tosses

Крім того, у R діє стандартний оператор присвоєння С подібних мов =.

Після запуску імені створеного вектору

>tosses

показується зміст вектору tosses:

[1] 2 1 1 3 6 4

R відрізняє великі і маленькі літери, тому **tosses** і **Tosses** – різні змінні.

Робоча пам'ять програми називається **Workspace** – це всі створені під час роботи **R** об'єкти. Інформація про об'єкти, які зберігаються в робочій пам'яті може бути отримана після виконання команди l**s** () або **objects** ().

Приклад.

> ls()

[1] "tosses"

Функція **rm** () видаляє об'єкти з робочої пам'яті.

Приклад.

> rm(tosses)

> ls()

character(0)

Елементарні команди складаються з виразів або присвоєнь. Якщо вираз представлено як команда, він виконується, результат виводиться на екран.

Значення виразу, яке присвоюється змінній, на екран не виводиться, але і не втрачається.

Якщо набрати ім'я змінної, буде виведено її значення.

[1] означає одновимірний об'єкт, тобто воно відповідає, тому, що об'єкт **tosses** є вектором.

Клавіша ↑ повертає попередню виконану команду.

Історія та Workspace

Історія та Workspace зберігаються у робочому каталозі програми (**Working Directory**). Історія має розширення *.RHistory, Workspace – *.RData. Дізнатися робочий каталог програми можна командою **getwd().**

Приклад.

> getwd()

[1] "c:/r"

Робочий каталог може бути змінений командою **setwd()**; при цьому шлях вказується не з символом бекслеш (\), а з символом слеш (/) або подвійний бекслеш (\\).

Приклад.

> setwd("i:/r")

> setwd("i:\\r")

Усі об’єкти, створені або завантажені під час сеансу роботи з **R** і не видалені спеціальною інструкцією, зберігаються у робочому просторі. Наприкінці сеансу **R** запитує, чи зберігати робочий простір на диску:

Save workspace image?

Якщо вибрати збереження, то цей робочий простір буде відновлено на початку наступного сеансу. Такою можливістю варто користуватись дуже обережно, оскільки «старі» об’єкти можуть спотворювати роботу **R** у новому сеансі. Зберігати робочий простір доцільно лише в тому випадку, коли ви збираєтесь наступного разу продовжити свою роботу з того самого місця, на якому зупинились зараз.

Допомога

Існує декілька способів отримати допомогу.

У верхньому меню пройти шляхом: **Help ->R functions (text)**. Після цього відкриється вікно запиту, куди потрібно написати ім'я функції R, по якій потрібна допомога.

Інші можливості отримати допомогу використовують **R Console** вікно управління. Наприклад, інформацію про функцію **mean** можна отримати у такий спосіб.

Приклад.

> ?mean

> help (mean)

Якщо синтаксис функції невідомий, проте вона містить певне словосполучення, можна ввести:

> ??mean

У такому випадку надається список функцій, імена яких включають послідовність символів " mean".

Закінчити роботу з програмою R можна запускаючи команду **q()** або **File -> Exit.**

RStudio

Консольний інтерфейс в **R** є майже повним. Він надає користувачеві історію команд, доповнення по **Tab** (тобто видає при натисненні на цю клавішу список можливих продовжень команд, об'єктів і, навіть, імен файлів), зберігає інформацію і об'єкти між сесіями. Дозволяє працювати на віддаленому комп'ютері. Якщо скористатися програмою **screen**, то можна не боятися

розірваних сесій і випадково закритих терміналів. Мінімальність консольного інтерфейсу спонукає до створення більш зручних сервісів. У роботі з **R** ми будемо використовувати інтегровану оболонку **RStudio**. RStudio (http://www.rstudio.com) — це графічна оболонка для **R**, доступна під вільною ліцензією. Цей графічний інтерфейс (GUI) для **R** працює під основними операційними системами (Windows, Linux, Mac OSX), і надає доступ до **R** через вебінтерфейс. Перед інсталяцією RStudio базовий варіант **R** має бути встановлений. Основне вікно **RStudio** розділене на декілька частин: початковий код редагованого файлу, консоль, вміст робочого простору, історія команд і частина, що надає доступ до списку пакетів, малюнків і файлів в поточному робочому каталозі і довідці (вміст окремих частин вікна можна поміняти через меню **Tools -> Options -> Pane Layout)** (Рис.5.4).

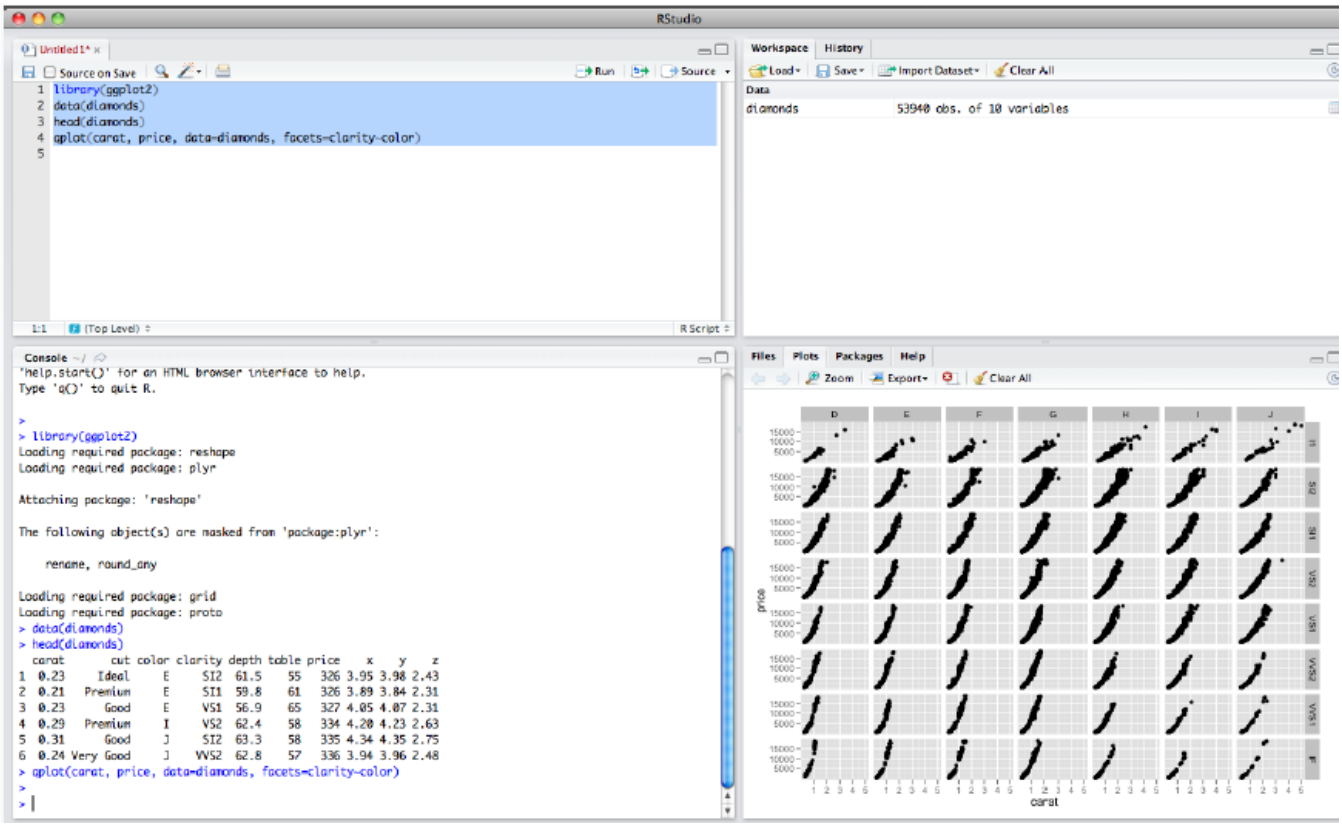


Рис.5.4.

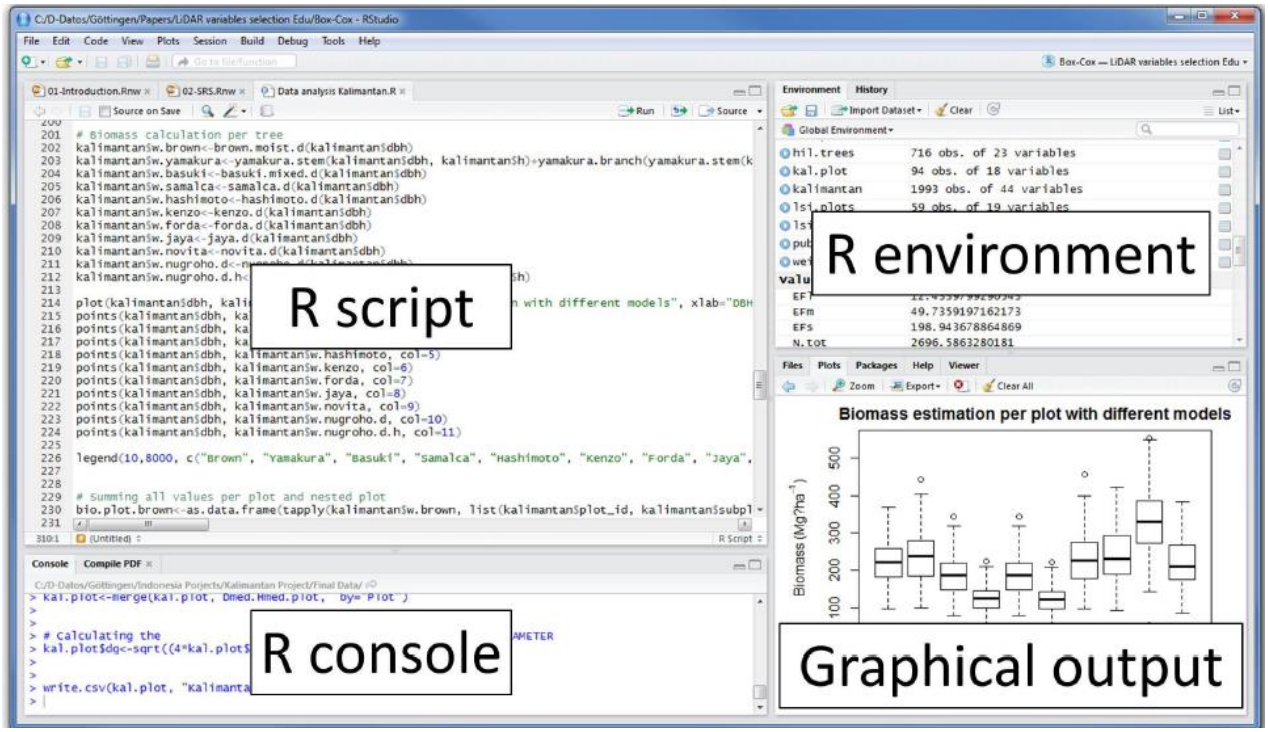
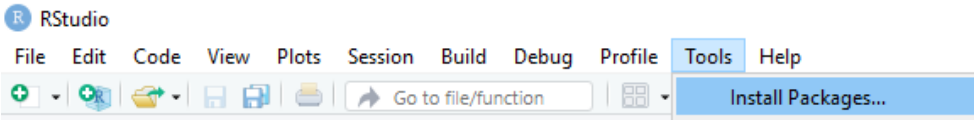


Рис. 5.5.

В якості переваг оболонки слід відмітити редактор коду з автоматичним підсвічуванням, автодоповненням, простими перетворення коду (наприклад, виділення декількох рядків в окрему функцію), підтримку редагування не лише **R** коду, але і документів **Sweave.**

Пакети

Базова програма **R** є лише інтерпретатором команд. Ядро програми **R** не є великим, але воно може розширюватися додатковими пакетами, які можуть скачуватися і встановлюватися на комп'ютері користувачем залежно від його потреб. Статистичні алгоритми, як правило, виконуються у вигляді скриптів на цій мові і зібрані у пакети (**packages**) **R**. Пакети зерігаються у спеціальних депозитаріях. При інсталяції разом з базовою програмою інсталюються і основні пакети, які реалізують найбільш популярні технології статистичного оброблення даних. Приміром, пакет **stats** для елементарних статистичних обчислень є серед них. Якщо потрібний пакет не було інстальовано на комп’ютері, його можна завантажити з інтернет-архіву (депозитарію), використовуючи головне меню (**Packages->Installpackage()**)



Спочатку програма пропонує вибрати інтернет-архів, з якого робиться інсталяція. Варіант **0-cloud**, що пропонується за замовчуванням, як правило, працює цілком задовільно. Після цього треба у списку вибрати потрібний пакет. Якщо цей пакет використовує які-небудь інші, котрих немає на комп'ютері, вони будуть інстальовані автоматично. Після того, як пакети інстальовані, вони зберігаються на вашому комп'ютері, але для роботи з ними під час сеансу їх треба підключати, використовуючи функцію **library()**.

Команда **>library(splines**) підключить пакет **splines**, призначений для роботи зі сплайнами. Ця ж функція, викликана без параметрів, дає перелік усіх завантажених пакетів з коротким поясненням їх функцій (у мінімальному варіанті цього переліку кілька десятків пакетів).

Packages – надається список основних пакетів.

Help → CRAN Home page → Task Views надається перелік галузей знань, далі можна отримати детальніший список завдань з ім'ям пакету для її вирішення.

Help →CRAN Home page → Software, Packages – імена пакетів за абеткою.

Можна за допомогою функцій у вікні **R Console**:

available.packages () #після відповіді на запит mirror-сервера, надається список наявних пакетів

length (available.packages()) # кількість пакетів на сервері

install.packages ("**namel", "name2")** # скачує вказаний пакет

.packages (all.available=TRUE) #список пакетів, встановлених на комп'ютері

library (package.name) # пакет буде прикріплений до шляху пошуку (аналогія: книга береться з книжкової полиці і кладеться на робочий стіл).

search () # список пакетів, прикріплених до шляху пошуку

library (help="stats") # список функцій, що містяться у вказаному статистичному пакеті.

Бібліотека R складається з пакетів.

Пакет складається з функцій і наборів даних.

data (package =.packages (all.available = TRUE)) # імена пакетів даних, встановлених на комп'ютері

data () # список наборів даних, підключених до шляху пошуку.

Завдання до комп'ютерного практикуму

Для виконання комп'ютерного практикуму необхідні ресурси: 1 ПК з доступом до Інтернету, файл даних flats.csv. Потрібно встановити робоче середовище RStudio та завантажити відповідні пакети.

Частина 1.

Дізнатися, який каталог є робочим, створити на диску D (F) папку R_ labs, змінити робочий каталог на створений, для цього використати команду **getwd ()** (get working directory).

getwd ()

Для зміни робочого каталогу використати команду **setwd** ("directory").

setwd("D:/R_labs")

Розглянути пакет даних **Orange:**

data (Orange) # набір даних *Orange* розпаковується і стає доступним для аналізу

help ("Orange") # опис набору даних

Скільки рядків та стовпців містить цей датафрейм?

Orange # друкуються дані

(навести скріншот)

class (Orange) # показується клас об'єкту

(навести скріншот)

str (Orange) # показує змінні набору даних

(навести скріншот)

summary (Orange) # обчислює основні статистики набору даних

(навести скріншот)

boxplot (Orange) # прямокутний графік набору даних у вікні Graph

(навести скріншот)

boxplot(Orange$circumference) # прямокутний графік набору даних circumference у вікні Graph

(навести скріншот)

Зберегти скрипт, робочу пам'ять, історію в робочий каталог: вибрати скрипт; натиснути кнопку "Save script" на панелі інструментів, або пункт меню "File->Save as…" ("File->Save" для файла, який вже зберігали під певною назвою). Збереження робочої пам'яті: в RGui пункт меню "File->Save Workspace". Збереження історії: в RGui пункт меню "File->Save History".

Частина 2.

Одержати загальну інформацію про середовище.

help.start()

(навести скріншот)

Показати інформацію про функцію **diff**. Здійснити пошук інформації щодо цієї функції в системі допоміжної літератури і архіві розсилок.

help(diff)

(навести скріншот)

RSiteSearch("diff")

(навести скріншот)

Показати приклад виконання функції **diff()**.

example(diff)

(навести скріншот)

Здійснити пошук за словом «Cars».

help.search("Cars")

(навести скріншот)

Показати список всіх функцій, що містять "foot".

apropos("foot")

Знайти інформацію по цим функціям:

help(citFooter)

(навести скріншот)

Частина 3.

Виконати команду:

data()

Які статистичні набори даних завантажені та доступні для аналізу?

Виконати команду для відображення повного списку пакетів, з якими інстальовано середовище R, разом з коротким описом:

library()

(навести скріншот)

Завантажити і підключити пакет Matrix.

install.packages("Matrix")

Обчислити значення виразу: $e^{\log_7(\sin 0.5) - arctg\sqrt{1+\cos 0.05}}$

exp(log(sin(1/2),base=7)-atan(sqrt(1+cos(0.05))))

Який результат ви отримали?

Завантажити дані 1, 8, 2, 6, 3, 8, 5, 5, 5, 5 у змінну-вектор:

```
x = c(1, 8, 2, 6, 3, 8, 5, 5, 5, 5) # створення вектора x
x
```

Виконати наступні завдання та скопіювати скрипти та результати їх виконання.

Вибрати компоненти, які знаходяться в інтервалі [5,6]:

y<-x[x>=5 & x<=6] #створення нового вектора, який складається з компонент вектора x, які належать вказаному інтервалу

y

Підрахувати суму, добуток і різницю компонент нового вектора:

```
sum(y) # сума компонент вектора
prod(y) # добуток компонент вектора
diff(y) # різниця між сусідніми компонентами вектора.
```

Створити вектор-послідовність чисел від -15 до 100, які кратні 5:

z<-seq(-15,100,by=5) # генерація вектора z, який складається з чисел від -15 до 100, що кратні 5

У новому векторі замінити кожне число, яке кратне 10 на 0.

x<-replace(z,z%%10==0,c(0)) #заміна компонент вектора, які кратні 10, на 0

x

Виділити компоненти, які більші за середнє значення.

y<-x[x>mean(x)] #виділення тих компонент вектора, які більші за середнє значення вектора x.

y

Знайти суму компонент нового вектора.

sum(y) #сума компонент вектора y

Введіть вектор з парною кількістю елементів. Утворіть з нього матрицю, в якої буде: а) два рядка; б) два стовпчики.

а)

vect<-c(-3,5,6,7,8,9,4,-4,-6,7,8,12,80,-5,0,6,21,-3,7,10) #створення вектора з парною кількістю елементів

M1<-matrix(vect,nrow=2) #створення з елементів вектору vect матриці з двома рядками.

M1

б)

M2<-matrix(vect,ncol=2) #створення з елементів вектору vect матриці з двома стовпчиками

M2

Введіть дані для матриці A. Знайдіть A * A.

vec <- numeric(18) #створення порожнього вектору з 18 елементів.

dim(vec)<-c(3,6) #задання розмірності для майбутньої матриці (3 рядка, 6 стовпчиків).

A <- edit(vec) #введення даних в матрицю з клавіатури

A

A*A #поелементне множення матриць

Введіть дані для вектора довжиною 12. Побудуйте з нього матрицю з трьох рядків і чотирьох стовпців. Транспонуйте матрицю B і позначте її через C. Позначте через D = BC. Чи комутують матриці B і C?

vect1<-c(0,3,2,9,-5,6,0,5,7,3,-3,0) #введення даних у вектор

B<-matrix(vect1,nrow = 3, ncol = 4, byrow = TRUE) #створення матриці з 3 рядками і 4 стовпчиками

B

C<-t(B) #транспонування матриці.

C

D<-B%*%C

D

A<-C%*%B

A

Знайдіть матрицю, обернену до D, за допомогою функції solve.

solve(D) #знаходження оберненої матриці

Введіть квадратну матрицю порядку 4. Елементи головної діагоналі замініть на 0, якщо вони більші за середнє значення всіх елементів матриці.

vec <- numeric(16)

dim(vec)<-c(4,4)

A <- edit(vec)

A

diag(A)<-replace(diag(A), diag(A)> mean(A), c(0)) #заміна діагональних елементів матриці, що більші за середнє значення елементів матриці, на 0

A

a<-c(0,3,-2)

y<-solve(A,a) #розв’язання лінійного рівняння.

y

Запишіть вектор з 20 значень. Знайдіть його квартілі. Розбийте компоненти вектора на 4 інтервали (межі – квартілі). Перетворіть вектор на фактор, де рівнями будуть інтервали, в які попадають значення вектора.

```
vect<-c(-3,5,6,7,8,9,4,-4,-6,7,8,12,80,-5,0,6,21,-3,7,10) #створення вектора
vect
 g<-quantile(vect,c(0,0.25,0.5,0.75,1)) # створення вектора, значеннями якого є квартілі вектора vect
g
u<-cut(vect,breaks=g,labels=c('Перший', 'Другий', 'Третій', 'Четвертий'), include.lowest=TRUE) #поділ на інтервали, кінцями інтервалів є квартілі, останній параметр дозволяє уникнути значення NA для -6
 u
```

Введіть два вектора однакової довжини (наприклад, 15). В першому буде вік пацієнтів, а в другому – значення 1 і 0 (використати функцію sample). 1 означає, що пацієнт одержав ліки, 0 – ні. Створити фактор, використавши ці два вектора. Дізнатися середній вік пацієнтів.

```
vik<-c(21,18,76,35,72,29,45,67,43,23,25,87,45,24,25) #створення вектора.
x<-sample(c(0,1),15,replace=TRUE)#створення вектора, який складається з 1 і 0 (всього 15).
x
mean(vik) #середній вік пацієнтів
```

З набору даних Loblolly вибрати стовпчики, які відповідають висоті і віку сосен. Утворити числовий вектор, який складається з середньої висоти дерева для таких категорій віку сосни: до 10 років, 10-15, 16-20, більше 20.

```
x<-
c(mean(Loblolly$height[Loblolly$age<10]),mean(Loblolly$height[Loblolly$age>10&Loblolly$age<16]),mean(Loblolly$height[Loblolly$age>15&Loblolly$age<21]),mean(Loblolly$height[Loblolly$age>20]))
# числовий вектор, який складається з середньої висоти дерева для таких категорій віку сосни: до 10 років, 10-15, 16-20, більше 20
```

x

З набору даних Loblolly вибрати тільки ті дані, які відповідають насінню типу 301 і 315. Зеленою пунктирною лінією зобразити графік залежності висоти дерева для насіння 301 від віку сосни, а неперервною фіолетовою лінією – для насіння 315. Зобразити в одному вікні. Підписати рисунок.

h301<-Loblolly$height[Loblolly$Seed==301] #формування вектору даних, які відповідають висоті древ з насіння типу 301. > a301<-Loblolly$age[Loblolly$Seed==301] #запис у вектор даних, які відповідають віку древ насіння типу 301

a315<-Loblolly$age[Loblolly$Seed==315] #запис у вектор даних, які відповідають віку древ насіння типу 315. > h315<-Loblolly$height[Loblolly$Seed==315] #запис у вектор даних, які відповідають висоті древ з насіння типу 315

plot(a301,h301,type='l',lty=5,col='green',ylab='Висота дерева', xlab="Вік дерева", main="Графік залежності висоти дерева від віку") #побудова графіку залежності висоти дерев типу 301 від віку

lines(a315,h315, add=T,col = "violet") #побудова графіку залежності висоти дерев типу 315 від віку

(навести скріншот отриманого графіка)

Частина 4. Імпорт, аналіз та візуалізація даних в R

Використаємо бібліотеки **dplyr** для очищення та трансформації даних та **ggplot2** для візуалізації даних.

Ці бібліотеки завантажуються з допомогою команди **install.packages**.

Створіть новий файл (File -> New File -> RScript) та скопіюйте наступні рядки:

install.packages("dplyr")

install.packages("ggplot2")

Щоб виконати код, виділіть рядки та натисніть піктограму **Run** з зеленою стрілкою або комбінацію клавіш CTRL + ENTER або COMMAND + ENTER.

Далі завантажте ці бібліотеки до робочого середовища. Це можна зробити за допомогою функції **library**. Ми встановлюємо бібліотеку один раз, але завантажувати її потрібно щоразу при перезапуску RStudio. Тобто при наступному запуску RStudio команди інсталяції не будуть потрібні і їх можна буде закоментувати використовуючи символ #:

#install.packages("dplyr")

#install.packages("ggplot2")

Додайте рядки:

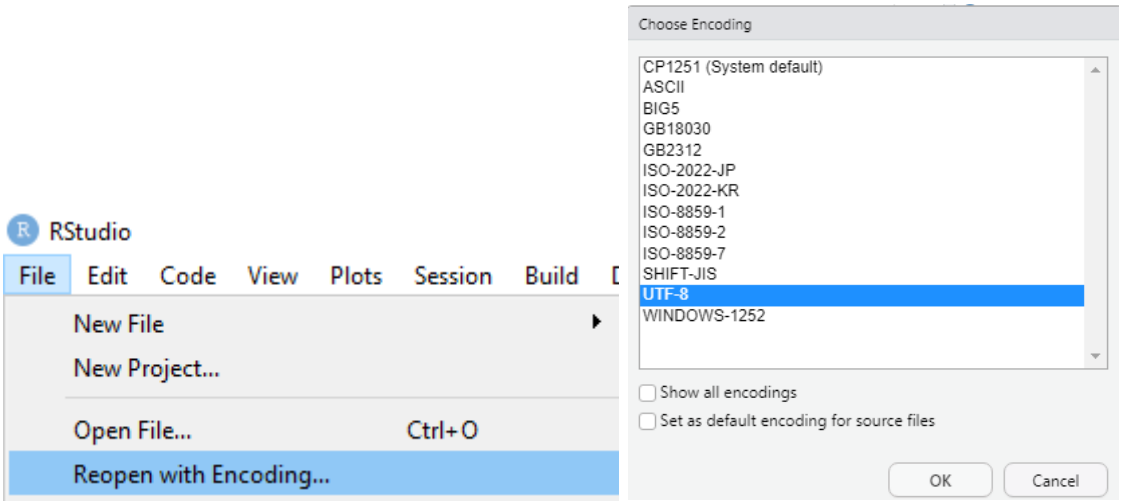
library(dplyr)

library(ggplot2)

Імпорт даних

Завантажте файл "flats.csv" та помістіть у ту ж папку, де знаходиться створений вами RScript. Завантажте дані з файла "flats.csv" у змінну flats, використовуючи функцію **read.csv**.

flats <- read.csv("flats.csv", stringsAsFactors=FALSE, encoding="UTF-8")



Параметр encoding="UTF-8" використовується для коректного відображення кирилиці у OS Windows. Параметр stringsAsFactors=FALSE вказує, що змінні, які мають тип character не будуть перетворюватись у тип даних **factor**. Цей тип використовується для роботи з категоріальними змінними. Справа у верхньому куті в розділі **Environment** натисніть на назву файлу **flats** та опишіть даний датасет.

Скільки спостережень наведено в датасеті? Які назви стовпців відображено?

Якщо отримали помилку Error in file(file, "rt"): cannot open the connection In addition: Warning message: In file(file, "rt"): cannot open file 'flats.csv': No such file or directory, вкажіть шлях до цієї директорії використовуючи команду setwd (скорочення від set working directory). Виконання цієї команди дозволяє не вказувати повний шлях до цієї директорії:

setwd("шлях до файла"), наприклад setwd("D:/R")

Визначимо клас обє'кта flats з допомогою команди class():

class(flats)

[1] "data.frame"

Клас об'єкта flats data.frame або ж таблиця даних. Кожен рядок цієї таблиці репрезентує спостереження, а кожна колонка відображає змінну, тобто частину інформації про це спостереження. В R можна використовувати функцію **str** (скорочення від structure), щоб швидко оцінити, чи правильно зчиталися дані.

str(flats)

(навести результат виконання функції)

Переглянемо документацію по функції read.csv використовуючи функцію **?**

?read.csv

В якості десяткового розділювача за замовчуванням використовується крапка dec ='.'. А в наших даних десятковим розділювачем є кома.

Заново зчитайте дані, вказавши параметр десяткового розділювача:

flats <- read.csv("flats.csv", stringsAsFactors=FALSE, dec= ",")

Перевірте ще раз їх структуру:

str(flats)

(навести результат виконання функції)

Дослідження даних

1. Перевірте розмірність датасету:

dim(flats)

(навести результат виконання функції)

2. Відобразіть першу частину об'єкта, першим параметром є об'єкт (тут таблиця даних flats, другим параметром можна вказати кількість рядків):

head(flats)

(навести результат виконання функції)

3. Відобразіть останню частину об'єкта, можна вказати кількість рядків:

tail(flats)

(навести результат виконання функції)

4. Відобразіть імена, пов'язані з об'єктом:

names(flats)

(навести результат виконання функції)

Трансформація даних

В R можна використовувати функції **str()** та **summary()**, щоб отримати перші знання про таблицю. Бібліотека **dplyr** має функцію **glimpse()** для швидкого узагальнення даних таблиці.

str(flats)

(відобразити результат)

summary(flats)

(відобразити результат)

Перегляд узагальнених даних:

glimpse(flats)

(відобразити результат)

Дізнайтеся, яка кількість квартир продається у кожному місті, для цього використайте в бібліотеці **dplyr** функцію **count**:

count(flats, Місто)

(відобразити результат)

Для виконання послідовно кілька операцій в **dplyr** можна використати оператор %>%, який дозволяє застосувати наступну команду до результатів виконання поточної.

Відсортуйте дані по кількості квартир у кожному місті у зростаючому порядку:

```
flats %>%
count(Місто) %>%
arrange(n)
(відобразити результат)
```

Вилучіть дані по Києво-Святошинському району з відображення, використовуючи команду filter. Умова дорівнює позначається як ==, а не дорівнює як !=. Відсортуйте результати в спадаючому порядку за допомогою **arrange(desc(n))**.

```
flats %>%
filter(Місто != "Києво-Святошинський") %>%
filter(Кімнат == 3) %>%
count(Місто) %>%
arrange(desc(n)) # arrange – сортування, desc – спадаючий порядок
(відобразити результат)
```

Виберіть лише квартири з кількістю кімнат 2:

```
flats %>%
filter(Кімнат == 2) %>%
filter(Місто != "Києво-Святошинський") %>%
count(Місто) %>%
arrange(desc(n))
(відобразити результат)
```

Візуалізація даних

Для візуалізації даних використайте бібліотеку **ggplot2**. Для побудови графіків використовується функція **ggplot()**. Після виконання коду ви побачите графік у вкладці **Plots** у нижній правій панелі в RStudio.

Першим аргументом цієї функції є набір даних (dataset).

Далі ми вказуємо змінні з набору даних як параметр aesthetic, які будуть відображатись, наприклад, по осях x та y.

Наступним кроком потрібно додати ще один рівень (об'єднавши їх знаком +) щоб задати geometric об'єкт. Наприклад, для графіка розсіювання це

geom_point, для лінійного графіка geom_line, для стовпчикової діаграми geom_bar.

Побудуйте стовпчикову діаграму для кількості кімнат:

```
ggplot(flats, aes(x=Кімнат)) +geom_bar(fill="lightblue",col="grey") +
ylab('Кількість')
```

(відобразити результат)

Побудуйте стовпчикову діаграму для змінної Загальна_площа:

```
p <- ggplot(flats, aes(x=Загальна_площа)) +
geom_bar(fill="lightblue",
col="grey") +
ylab('Кількість')
p
```

(відобразити результат)

Графік розсіювання

Побудуйте графік залежності ціни від загальної площі.

```
library(ggplot2)
ggplot(flats, aes(x=Загальна_площа, y=Ціна)) +
geom_point()
```

(відобразити результат)

Вимоги до оформлення звіту

Звіт має включати:

1. Титульний аркуш.
2. Завдання на комп’ютерний практикум.
3. Хід роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до комп’ютерного практикуму.
4. Висновки.

Питання для самоперевірки

1. Назвіть сайт, який є основним дзеркалом середовища R.
2. З якого сайту можна скачати останню версію R?
3. За допомогою якої клавіші виконується скрипт R?
4. Яке розширення мають файли, у яких зберігають скрипти R?
5. Якою командою з меню можна виконати редагування скрипта R?
6. Як дізнатися назву робочого каталогу?
7. Як встановити робочий каталог?
8. З якого сайту можна скачати Rstudio?
9. Як дізнатися про склад пакетів R, встановлених на вашому комп'ютері?
10. Що є пакетом в R?
11. Як завантажити дані в середовище Rstudio, переглянути розмірність та кілька перших рядків датасету?
12. Які бібліотеки використовуються для аналізу та візуалізації даних в R?
13. Які є типи даних в R?
14. Які є типи R об'єктів в R?
15. Які операції з векторами в R ви знаєте?
16. Як відбувається доступ до елементів вектора в R?
17. Як відбувається зчитування даних з файла в R?

Рекомендована література

1. Microsoft R Application Network // Електронний ресурс. Режим доступу: https://mran.microsoft.com/documents/what-is-r

2. R програмування // Електронний ресурс. Режим доступу: https://coderlessons.com/tutorials/mashinnoe-obuchenie/r-programmirovanie/r-programmirovanie

3. R Introduction // Електронний ресурс. Режим доступу: https://www.w3schools.com/r/r_intro.asp

КОМП'ЮТЕРНИЙ ПРАКТИКУМ 6.
РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ ДАНИХ З ВИКОРИСТАННЯМ SPARK-КЛАСТЕРА ТА МОВИ R

Мета роботи*:* встановити Spark на локальній машині, виконати розподілені обчислення для набору даних з використанням Spark-кластера у середовищі R.

Теоретичні відомості

В екосистемі R є багато пакетів, що дозволяють користувачам працювати з віддаленими базами даних і виконувати масштабовані розподілені обчислення за допомогою призначених для цього програмних платформ, або фреймворків. Однією з таких платформ є Apache Spark. Хоча платформа Hadoop дозволила багатьом компаніям успішно застосовувати парадигму MapReduce для розподілених обчислень над величезними обсягами даних, кожен раз при виникненні нового завдання потрібне написання нового коду для операцій map і reduce, що є незручним. Для вирішення цієї проблеми в 2008 р. інженери з Facebook створили Hive – систему управління базами даних на основі Hadoop. Головною особливістю Hive стала підтримка SQL-подібних запитів до даних, що зберігаються в HDFS. У 2009 р. в Каліфорнійському університеті в Берклі був запущений дослідницький проект Spark з метою підвищити ефективність розподілених обчислень методом MapReduce і створити універсальну платформу для таких обчислень. У 2010 р. Spark був опублікований як проєкт з відкритим кодом, а в 2013 р. переданий фонду Apache Software Foundation.

Spark є однією з найбільш широко використовуваних платформ для роботи з великими даними і характеризується такими особливостями:

- швидкодія, що досягається за рахунок виконання обчислень в оперативній пам'яті великої кількості комп'ютерів, об'єднаних в один кластер, а також завдяки ефективним протоколам передачі даних по мережі;

- універсальність: Spark підтримує багато технологій кластерних обчислень і має кілька бібліотек-надбудов для вирішення поширених аналітичних задач, включаючи Spark SQL (SQL-подібні запити до даних), MLlib (алгоритми машинного навчання), GraphX (аналіз графів) і Spark Streaming (обробка потокових даних).

Усі обчислення в системі R виконуються в оперативній пам'яті комп'ютера. Персональні комп'ютери сьогодні мають достатньо оперативної пам'яті для обробки даних, і цього цілком може виявитися достатньо для вирішення широкого кола практичних задач. Однак обсяг багатьох сучасних наборів даних (наприклад, зібраних високонавантажених веб-додатками і промисловими системами) набагато перевищує розмір оперативної пам'яті, доступний на окремому комп'ютері. Такі дані зазвичай зберігаються в віддаленій базі даних або в сховищі іншого типу. Залежно від завдання для обробки таких даних за допомогою R можна скористатися однією з перерахованих нижче стратегій.

1. Створення репрезентативної вибірки обмеженого розміру.

Практично усі класичні методи статистики припускають, що дослідник працює з репрезентативною вибіркою з деякої генеральної сукупності. Тому для застосування таких методів можна випадковим (або іншим відповідним ситуації) чином вибрати кілька тисяч або навіть сотень тисяч записів з бази даних, а потім виконати необхідний аналіз локально на комп'ютері користувача. Такий підхід підійде на початкових стадіях проєкту, коли потрібно швидко ознайомитися з властивостями даних або створити прототип прогнозної моделі. При цьому користувачеві буде доступне усе розмаїття існуючих для R додаткових пакетів. Недолік цього підходу в тому, що іноді створити репрезентативну вибірку буває складніше, ніж здається. Працюючи ж з зміщеною вибіркою, дослідник ризикує зробити невірні висновки щодо властивостей генеральної сукупності.

2. Розбиття даних на частини.

Рішення ряду завдань передбачає обробку логічно розділених наборів даних, наприклад, відповідних певним часовим періодам, окремим географічним областям, компаніям, користувачам тощо. Такі набори можна легко витягти з бази даних і далі послідовно (а іноді паралельно) обробити за допомогою R на локальному комп'ютері дослідника. Ця стратегія зручна у тому, що аналізу піддаються усі наявні дані. Однак ці дані повинні бути нероздільні на окремі частини, що не завжди можливо або не завжди має сенс. Більш того, в залежності від обсягу даних, такий підхід може зайняти занадто багато часу і обчислювальних ресурсів.

3. Виконання ресурсномістких обчислень на стороні бази даних.

Часто перед виконанням статистичного аналізу або побудовою прогнозної моделі до даних необхідно застосувати такі операції, як фільтрація, групування, агрегування тощо. Більшість баз даних чудово справляються з такими операціями і тому має сенс спочатку зробити подібні обчислення саме на стороні бази даних, а потім завантажити отриманий набір даних меншого розміру на комп'ютер користувача і провести його подальшу обробку за допомогою R. До цієї стратегії варто віднести можливість побудови деяких прогнозних моделей з використанням обчислювальних ресурсів бази даних. Наприклад, пакет **modeldb** дозволяє таким чином створювати моделі лінійної регресії. Тут же варто згадати і спроби деяких компаній реалізувати можливість виконання R-скриптів повністю на стороні бази даних. Зокрема, така можливість є в Microsoft SQL Server. Залежно від використовуваної бази даних, необхідна користувачу функціональність іноді може бути недоступна. Більш того, не всі бази даних однаково ефективні: виконання ресурсоємних запитів може привести до істотного уповільнення деяких з них, що, в свою чергу, може викликати інші небажані наслідки (наприклад, уповільнення веб-сайту, який обслуговується подібною базою даних).

4. Використання спеціалізованих програмних платформ для роботи з великими даними.

Якщо описані вище стратегії з тих чи інших причин не підходять, варто звернутися до спеціалізованих програмних платформ, призначених для організації і виконання розподілених обчислень над великими даними. Найбільш відомими і широко використовуваними серед таких платформ є Hadoop і Spark (обидві входять до складу проектів фонду Apache Software Foundation і тому їх також часто називають Apache Hadoop і Apache Spark). В R є кілька пакетів (зокрема, **sparklyr**), що надають зручний інтерфейс для роботи з цими платформами.

Пакет **sparklyr** надає зручний інтерфейс для роботи зі Spark-кластерами з середовища R. Зокрема, з його допомогою можна: встановлювати з'єднання з кластером; виконувати звичайні операції перетворення, фільтрації і агрегування даних з використанням синтаксису **dplyr**; будувати прогнозні моделі з використанням алгоритмів машинного навчання, реалізованих в бібліотеці MLlib для Spark; працювати з іншими R-пакетами, які використовують Spark для виконання розподілених обчислень.

Можливість працювати з платформою Spark локально – це велика її перевага, що дозволяє виконувати розробку і налагодження коду перед його виконанням на віддаленому кластері, який може складатися з декількох комп'ютерів.

Завдання до комп’ютерного практикуму

Частина 1. Встановлення та підключення до локального Spark-кластеру

Програмна платформа Spark написана на Scala і Java і вимагає Java Virtual Machine (JVM). Тому перед інсталяцією Spark необхідно переконатися, що на комп'ютері встановлено мову Java версії, не нижче 1.8.

Оскільки Java потрібна для виконання багатьох програм, швидше за все вона на вашому комп'ютері вже є. З середовища R це можна перевірити с допомогою команди:

system ("java -version")

Якщо Java немає, то скористайтеся офіційною інструкцією з інсталяції (https://www.java.com/ru/download/help/windows_manual_download.html).

Якщо на вашому комп'ютері є кілька версій Java, то, можливо, вам доведеться змінити значення змінної середовища JAVA_HOME на потрібну версію. З R це можна зробити за допомогою команди Sys.setenv () , наприклад:

Sys.Setenv(JAVA_HOME = "<шлях до Java потрібної версії>")

Примітка. Не використовуйте пробіли в назвах файлів і папок, працюючи на комп'ютерах під управлінням Windows – це може привести до різного роду проблем. Проблеми можуть виникнути також при вживанні кирилиці в назвах файлів і папок.

Далі встановіть пакет **sparklyr**:

install.packages("sparklyr")

Програмну платформу Spark можна встановити як вручну, так і за допомогою призначеної для цього команди **spark_install ()** з пакету sparklyr. Другий спосіб більш зручний і саме їм ми і скористаємося. При цьому, щоб уникнути "сюрпризів" у ході виконання задач, встановіть стабільну версію Spark 2.3 (незважаючи на те, що вже доступні новіші версії):

require(sparklyr)

spark_install("2.3")

Виконання останньої команди призведе до інсталяції як запитаної версії Spark, так і необхідної для її коректної роботи платформи Hadoop.

За допомогою команди **spark_installed_versions ()** перевірте, які версії Spark і Hadoop є на вашому комп'ютері:

spark_installed_versions()

За замовчуванням платформа Spark буде встановлена в "... \ AppData \ Local \ spark" на Windows-машинах і в "~ / spark" на Mac і Linux-машинах. Щоб

змінити ці місця установки, можна скористатися командою options (spark.install.dir = "<бажаний шлях>") перед виконанням **spark_install ()**. Місце інсталяції Spark називається "домашньою директорією" і задається за допомогою змінної середовища SPARK_HOME.

При інсталяції Spark за допомогою команди **spark_install ()** значення цієї змінної буде присвоєно автоматично і R буде "знати", де шукати Spark для виконання локальних обчислень. Однак, у хмарі Amazon цю змінну потрібно буде налаштувати самостійно.

Підключення до локального Spark-кластеру

Для підключення до Spark-кластеру використайте функцію **spark_connect ()** з пакету sparklyr. У цій функції є кілька аргументів, але нам будуть потрібні лише два з них – **master** і **version**:

```
sc <- spark_connect(master = "local", version = "2.3")
```

Параметр **master** використовується для вказівки IP- адреси або URL "головного комп'ютера" в кластері, відомого також як "керуючий вузол" (driver node). Ця термінологія пов'язана з тим, що в "справжніх" кластерах є один керуючий вузол, який розподіляє обчислювальні завдання по "робочим вузлам" (workers, або worker nodes). Оскільки зараз нас цікавить робота з локальним кластером, то ми просто присвоюємо параметру значення "local" .

Як випливає з його назви, параметр version служить для вказівки необхідної версії Spark. Цей параметр використовується тільки для роботи з локальним кластером і дозволяє автоматично виявити домашню директорію відповідної версії Spark і запустити кластер.

Об'єкт, що повертається функцією **spark_connect ()**, містить різного роду службову інформацію про кластер. Цьому об'єкту прийнято давати ім'я **sc**.

Для цієї роботи використайте дані одного з багатьох пакетів даних, – пакета nycflights13, який містить таблиці з описом 336776 авіарейсів, які стартували з аеропортів Нью-Йорка в 2013 р. Встановіть пакет nycflights13:

```
install.packages("nycflights13")
require(nycflights13)
```

require(dplyr)

glimpse(flights)

(відобразити результат)

Для початку потрібно завантажити таблицю flights в локальний Spark-кластер. Для цього використайте функцію **copy_to ()**, на яку ми подаємо створений раніше об'єкт **sc** і копійований в таблицю flights :

fl <- copy_to(sc, flights)

fl

(відобразити результат)

Важливо підкреслити відмінність об'єкта fl від вихідної таблиці flights. Об'єкт flights – це звичайна таблиця класу tibble. В об'єкті ж fl зберігається тільки службова інформація по завантаженій нами в Spark таблиці з даними – самих даних в цьому об'єкті немає (виконайте str (fl) , щоб переконатися в цьому). Проте, ввівши в консолі R ім'я цього об'єкта і натиснувши клавішу Enter, ми отримаємо перші кілька спостережень з скопійованої таблиці, як якщо б це була звичайна таблиця з даними. Така поведінка об'єкта fl обумовлена тим, що при його виклику з консолі R Spark автоматично збирає (collect) перші кілька спостережень з скопійованої таблиці і виводить їх на екран. Можливість такого роду інтерактивного аналізу – це ще одна велика перевага Spark.

Для виконання запитів до даних в Spark-кластері з середовища R можна використовувати або SQL, або функції з пакета dplyr.

Припустимо, ми хочемо з'ясувати загальну кількість рейсів з кожного аеропорту Нью-Йорка (змінна origin).

Для виконання SQL-запиту потрібен пакет DBI і функція dbGetQuery (), що входить до його складу. На функцію dbGetQuery () потрібно подати об'єкт sc з інформацією для підключення до Spark-кластеру, а також сам запит:

require(DBI)

q <- "SELECT `origin`, count(*) AS `N`

FROM `flights`

GROUP BY `origin`"

```
dbGetQuery(sc, q)
```

(відобразити результат)

Використання SQL потрібне при виконанні складних запитів, які повинні бути оптимізовані для забезпечення швидкодії. Однак в більшості випадків для виконання стандартних операцій над даними використання функцій з пакета dplyr буде більш зручним. Виконайте наступні команди:

```
result <- fl %>%
group_by(origin) %>%
summarise(N = n())
result
```

(відобразити результат)

Слід пам'ятати про те, що об'єкт result (подібно створеному нами раніше об'єкту fl) не є таблицею даних: він лише зберігає інформацію про те, як такі дані витягти, або "зібрати", з Spark-кластера.

Для отримання даних з Spark в середовище R використайте функцію **collect ()**:

```
true_result <- result %>% collect()
true_result
```

(відобразити результат)

```
class(result)
## [1] "tbl_spark" "tbl_sql"   "tbl_lazy"  "tbl"
```

(відобразити результат)

```
class(true_result)
## [1] "tbl_df"     "tbl"    "data.frame"
```

(відобразити результат)

Після завершення роботи необхідно відключитися від кластера. Для цього служить команда spark_disconnect () :

```
spark_disconnect(sc)
```

Наведена команда також видалить кластер і всі дані, що зберігаються у ньому.

Частина 2. Аналіз даних в Spark-кластері за допомогою пакета dplyr в R

При роботі з великими даними, безпосередній аналіз яких в системі R неможливий, ми змушені звертатися до спеціалізованих платформ для виконання розподілених обчислень, зокрема Spark. Завдяки таким пакетам, як **sparklyr**, ми можемо використовувати R в якості клієнта, який відправляє обчислювальні завдання ("Push compute") на Spark-кластер, а потім збирає результати обчислень ("Collect results") для подальшого аналізу в самій системі R. Є два способи формального представлення подібних обчислювальних задач перед їх відправкою на кластер: або з використанням SQL-команд, або за допомогою функцій з пакета **dplyr**. Хоча SQL (точніше, Spark SQL, який, в свою чергу, заснований на діалекті HiveQL) дозволяє формулювати як завгодно складні операції над даними, в більшості стандартних випадків dplyr більш зручний, оскільки він добре знайомий більшості сучасних користувачів R і володіє набагато більш лаконічним синтаксисом.

Використайте дані з пакета nycflights13, який містить таблиці з описом 336776 авіарейсів, що вилетіли з аеропортів Нью-Йорка в 2013 р. Запустіть локальний Spark-кластер з необхідниими бібліотеками і завантажте в нього необхідні таблиці:

```
require(sparklyr)
require(nycflights13)
# підключення до кластеру:
sc <- spark_connect(master = "local", version = "2.3")
# завантаження даних в кластер:
flights_tbl <- copy_to(sc, flights, "flights") # дані по рейсам
airlines_tbl <- copy_to(sc, airlines, "airlines") # дані по авіакомпаніям
```

Потрібно побудувати модель, яка передбачає ймовірність прибуття затриманого рейсу без запізнення (за умови затримки вильоту на 15-30 хвилин). Розглянемо це завдання як випадок бінарної класифікації: цікавий для нас

відгук приймає значення 1, якщо затриманий рейс прибув без запізнення, і 0 у протилежному випадку.

Процес побудови прогнозних моделей включає кілька кроків, першим з яких є підготовка та розвідувальний аналіз даних.

Розвідувальний аналіз може складатися з декількох кроків, таких як виявлення і усунення проблем з якістю аналізованих даних (пропущені спостереження, викиди тощо), розрахунок описових статистик, виявлення найбільш перспективних предикторів для подальшого включення в модель тощо.

За допомогою пакета **dplyr** можна виконувати такі стандартні види обчислень на Spark-кластері:

- вибір, фільтрація і агрегування змінних;
- використання віконних функцій;
- об'єднання декількох таблиць за допомогою join-операторів;
- імпорт результатів обчислень з Spark в середовище R.

До найбільш важливих команд dplyr відносяться select (), filter (), mutate (), summarise () і arrange (), для виконання групових операцій використовується команда group_by (). Ці та інші команди dplyr можна об'єднувати в "ланцюжки" за допомогою оператора %>% з пакета **magrittr** (підвантажується одночасно з dplyr і тому окремо його викликати не потрібно).

Підрахуйте загальну кількість польотів, виконаних кожною авіакомпанією за 2013 р., а потім виберіть 5 авіакомпаній з найбільшим числом польотів:

```
require(dplyr)
flights_tbl %>%
group_by(carrier) %>%
summarise(N = n()) %>%
arrange(desc(N)) %>%
head(5)
```

(відобразити результат)

Оскільки Spark "розуміє" тільки SQL, то усі обчислення, задані в R за допомогою команд dplyr, перед відправкою на Spark-кластер автоматично переводяться на SQL наступним чином:

select () -> SELECT

filter () -> WHERE

mutate () -> оператори + , - , / , * , log та ін.

summarise () -> функції агрегування SUM, MIN, MAX та ін.

arrange () -> ORDER

group_by () -> GROUP BY

Виконайте функцію **show_query ()** з пакету dplyr для перегляду SQL-запиту, який формується з відповідного коду R:

```
flights_tbl %>%
group_by(carrier) %>%
summarise(N = n()) %>%
arrange(desc(N)) %>%
head(5) %>%
show_query()
```

(відобразити результат)

Об'єднання таблиць

У пакеті dplyr є кілька функцій для виконання стандартних JOIN – операцій над двома таблицями: inner_join () , left_join () , right_join () , full_join () , semi_join () , nested_join () і anti_join ().

Виконайте LEFT JOIN таблиці flights_tbl з таблицею airlines_tbl за полем carrier:

```
flights_tbl %>%
left_join(airlines_tbl, by = "carrier") %>%
glimpse()
```

(відобразити результат)

Функції Hive Query Language

Хоча стандартні команди dplyr і деякі базові функції R автоматично переводяться на SQL, їх буде недостатньо для більш складних обчислень над даними за допомогою Spark.

Spark SQL заснований на мові запитів Hive Query Language (HiveQL) і всі функції цієї мови можна використовувати в поєднанні з командами dplyr.

Повний перелік функцій HiveQL можна знайти в офіційній документації (https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF).

Наприклад, для обчислення медіанного значення змінної dep_delay (затримка рейсу, хв.) з таблиці flights_tbl ми не можемо просто скористатися базовими функціями R median () або quantile () – це призведе до помилки.

Застосуйте Hive-функцію percentile ():

```
flights_tbl %>%
summarise(median = percentile(dep_delay, 0.5))
```

(відобразити результат)

Коли при перекладі коду R на SQL dplyr зустрічає незнайому функцію, то він просто включає її в SQL-запит "як є":

```
  flights_tbl %>%
  summarise(median = percentile(dep_delay, 0.5)) %>%
  show_query()
```

(відобразити результат)

Hive-функція percentile () дозволяє одночасно обчислити кілька процентилей. Для цього на неї потрібно подати масив (array ()) з необхідними значеннями процентилей:

```
flights_tbl %>%
  summarise(perc = percentile(dep_delay, array(0.25, 0.5, 0.75)))
# Source: spark<?> [?? x 1]
  perc
```

(відобразити результат)

Як бачимо, результатом виконання наведеної команди є список з трьома значеннями. Щоб автоматично отримати ці значення зі списку, використайте Hive-функцію **explode ()**:

```
flights_tbl %>%
summarise(perc = percentile(dep_delay, array(0.25, 0.5, 0.75))) %>%
mutate(perc = explode(perc))
```

(відобразити результат)

Отже, ми розібралися з тим, як використовувати пакет dplyr для формування обчислювальних задач для Spark. Застосуємо тепер отримані знання в ході розвідувального аналізу даних з таблиці flights_tbl.

Для початку з'ясуємо, чи є в наших даних пропущені значення. Це можна зробити декількома способами.

Виконайте підрахунок пропущених значень для всіх стовпців таблиці flights_tbl за допомогою команди summarise_each () з пакету dplyr в поєднанні з анонімною функцією, яка задає логіку обчислень:

```
flights_tbl %>%
summarise_each(list(~sum(as.integer(is.na(.))))) %>%
glimpse
```

(відобразити результат)

Яка максимальна кількість пропущених значень? Скільки це становить від загального числа спостережень в таблиці?

Розмірність таблиці можна з'ясувати за допомогою команди sdf_dim () з пакету sparklyr, яка є аналогом базової R-функції dim ().

Якщо частка пропущених значень невелика, ми можемо видалити відповідні рядки з таблиці без особливого ризику вплинути на якість подальшого аналізу. Для цього використайте базову функцію **na.omit ()**:

```
flights_full <- flights_tbl %>% na.omit()
```

(відобразити результат)

```
flights_full %>% sdf_dim()
```

(відобразити результат)

Оскільки нас цікавлять рейси, затримка яких склала від 15 до 30 хв. (включно), далі нам потрібно відфільтрувати дані відповідним чином. Паралельно додайте новий стовпець target зі значеннями залежної змінної:

```
flights <- flights_full %>%
  filter(dep_delay >= 15, dep_delay <= 30) %>%
 mutate(target = as.integer(arr_delay <= 0))
# розмірність таблиці:
flights %>% sdf_dim()
```

(відобразити результат)

Таблицю якої розмірності ви отримали?

Це зовсім невелика таблиця за мірками будь-якого сучасного комп'ютера і вже зараз її можна було б імпортувати з Spark в R (за допомогою команди collect ()) для подальшого аналізу. З метою демонстрації можливостей роботи Spark з R ми залишимо цю таблицю в пам'яті кластера.

Спробуємо з'ясувати, які з наявних змінних корелюють із залежною змінною target. Логічно очікувати, що ймовірність прибуття затриманого рейсу за розкладом в значній мірі визначається відстанню між аеропортом вильоту і аеропортом прибуття (стовпець distance, виражається в милях).

Розрахуємо медіанне значення цієї відстані для обох класів залежною змінною:

```
  flights %>%
  group_by(target) %>%
  summarise(median_dist = percentile(distance, 0.5))
```

(відобразити результат)

Чим більше відстань між аеропортами, тим більше шанс у затриманого рейсу надолужити згаяний час і прибути без запізнення. Можливість надолужити згаяний час може визначатися різними факторами, пов'язаними з авіакомпанією, яка виконує той чи інший рейс (досвід пілотів, характеристики літака тощо). Тому категоріальна змінна carrier, що містить скорочені назви авіакомпаній, також може виявитися корисним предиктором.

Щоб зрозуміти, чи це так, згрупуйте дані за рівнями змінної carrier і підрахуйте кількість затриманих рейсів, які прибули із запізненням і без. Далі візуалізуйте дані з отриманої таблиці спряженості за допомогою мозаїчної діаграми (для цього буде потрібно пакет ggmosaic, який потрібно встановити зі сховища CRAN):

```
install.packages("ggmosaic")
require(ggmosaic)
flights %>%
group_by(target, carrier) %>%
tally() %>% # тут закінчуються обчислення на стороні Spark-кластера
collect %>% # імпорт результатів обчислень в R
ggplot() +  # візуалізація даних в R
geom_mosaic(aes(product(target, carrier), weight = n)) +
labs(x = "Авіакомпанія", y = "Відгук")
```

Авіакомпанії можуть відрізнятися за часткою рейсів, які прибули без запізнення, в зв'язку з чим змінну carrier можна розглядати як потенційно корисний предиктор. Спочатку ми відправляємо з R інструкції для виконання певних обчислень на Spark-кластері, а потім імпортуємо отриманий результат в середовище R і вже продовжуємо аналіз звичайними для R засобами (в даному випадку – зображуємо дані графічно за допомогою ggplot2 і ggmosaic).

Вимоги до оформлення звіту

Звіт має включати:

1. Титульний аркуш.
2. Завдання на комп’ютерний практикум.
3. Хід роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до комп’ютерного практикуму.
4. Висновки.

Питання для самоперевірки

1. Які особливості і переваги Spark для розподілених обчислень?

2. Як відбувається встановлення та підключення до локального Spark-кластеру?

3. Як відбувається аналіз даних в Spark-кластері за допомогою пакета dplyr в R?

Рекомендована література

1. Spark і sparklyr для роботи з великими даними в R // Електронний ресурс. Режим доступу: https://r-analytics.blogspot.com/2020/02/spark-intro.html
2. Локальний Spark-кластер // Електронний ресурс. Режим доступу: https://r-analytics.blogspot.com/2020/02/spark-r-connect.html
3. Аналіз даних в Spark-кластері за допомогою пакета dplyr // Електронний ресурс. Режим доступу: https://r-analytics.blogspot.com/2020/03/spark-dplyr.html