МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет нафти і газу

Кафедра інформаційно-телекомунікаційних технологій та систем

Л. О. Штаєр

СУЧАСНІ ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНИХ СИСТЕМ

ЛАБОРАТОРНИЙ ПРАКТИКУМ
ЧАСТИНА 1

Івано-Франківськ
2017

УДК 004.42
ББК 32.973-01
Ш-87

Рецензент:

Незамай Б. С. кандидат технічних наук, доцент кафедри математичних методів у інженерії Івано-Франківського національного технічного університету нафти і газу

Рекомендовано методичною радою університету
(протокол № 22 від 31.05.2017 р.)

Штаєр Л. О.

Ш-87 Сучасні технології створення програмних систем: лабораторний практикум. – Ч. 1. – Івано-Франківськ: ІФНТУНГ, 2017. – 38 с.

МВ 02070855-10974-2017

Лабораторний практикум містить методичні матеріали для проведення лабораторних занять з дисципліни “Сучасні технології створення програмних систем” у першому семестрі вивчення даної дисципліни. Розроблений відповідно до робочої програми навчальної дисципліни “Сучасні технології створення програмних систем”. Може бути використаний студентами денної та заочної форм навчання.

Призначений для підготовки фахівців за спеціальністю 151 “Автоматизація та комп’ютерно-інтегровані технології” спеціалізації “Комп’ютеризовані системи управління та автоматика”.

УДК 004.42
ББК 32.973-01

МВ 02070855-10974-2017

ЗМІСТ

ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ

Дисципліна «Сучасні технології створення програмних систем» є профілюючою, забезпечує теоретичну та інженерну підготовку фахівців в галузі Автоматизація та приладобудування за освітньо-кваліфікаційним рівнем "магістр" зі спеціальності Автоматизація та комп'ютерно-інтегровані технології.

У курсі запропоновано ознайомлення із технологіями обробки даних та розробкою front-end. Зокрема продемонстровано роботу з сучасними програмними продуктами git, GitHub, Trello, R, RStudio, фреймворками Shiny та AngularJS. Курс дасть змогу студентам стати розробниками клієнтської частини веб-додатків, створювати динамічну та інтерактивну інфраструктуру сучасних веб-порталів, взаємодіяти із наборами даних з метою їх збору, очищення і аналітичного оброблення для одержання нових даних.

При вивченні курсу від студентів вимагається володіння навичками та методами, одержаними при вивченні курсів "Архітектура програмних систем", "Об'єктно-орієнтоване програмування", "Сучасні технології програмування".

Метою викладання є формування цілісних уявлень про обробку даних та розробку клієнтської частини веб-додатків, оволодіння навичками практичного розв'язання задач з розробки web-орієнтованих додатків з використанням сучасних технологій створення програмних систем.

Завдання вивчення дисципліни:

– здобування знань про підходи до збору (з Інтернету, API, баз даних), очистки та обробки статистичних даних з використанням мови програмування R;

– набуття навичок у використанні R для ефективного аналізу даних;

– здобування практичних навичок в узагальненні одержаних даних та здобуванні нових знань з використанням графічних можливостей R;

– засвоєння базових принципів проведення відтворюваних досліджень та ознайомлення з інструментами для поширення наукових знань;

– практична реалізація клієнтської частини web-орієнтованих програмних продуктів;

– ознайомлення з клієнтською частиною фреймворків JavaScript і шаблону проектування MVC;

– вміти реалізовувати додатки на AngularJS;

– вміти використовувати засоби AngularJS, включаючи директиви, фільтри, контролери, маршрутизацію.

Перша частина лабораторного практикуму містить методичні вказівки для проведення лабораторних робіт з дисципліни «Сучасні технології створення програмних систем» в першому семестрі вивчення дисципліни (модуль 1) і за змістом відповідає робочій програмі дисципліни. Модуль 1 «Огляд корисних web-сервісів при розробці ПЗ. Аналіз даних з R» розрахований на 18 годин лекційного матеріалу, 18 годин лабораторних робіт та 54 годин самостійної роботи. Модуль містить 6 лабораторних робіт, для виконання яких застосовуються системи з відкритим програмним кодом, які доступні у вигляді on-line сервісів або через вільно поширювані дистрибутиви та пакети.

Кожна лабораторна робота містить такі елементи як мету, обладнання, теоретичні відомості, порядок виконання роботи, зміст звіту, запитання для самоконтролю, тривалість заняття. Звіт з лабораторної роботи оформляється відповідно до стандартів на оформлення технічної документації, які діють у навчальному закладі, повинен містити такі необхідні елементи: мету роботи, завдання, хід виконання роботи, одержані результати з графічною демонстрацією, текст програм з коментарями, висновки.

ЛАБОРАТОРНА РОБОТА 1.
КЕРУВАННЯ ВЕРСІЯМИ ЗА ДОПОМОГОЮ GIT

Мета роботи: вивчити та закріпити на практиці можливості системи керування версіями, одержати практичний досвід у використанні Git та GitHub.

Обладнання:

– ПК IBM PC x86 CPU з встановленою операційною системою;

– доступ до мережі інтернет.

1.1 Теоретичні відомості

Система керування версіями (англ. *source code management*, SCM) — програмний інструмент для керування версіями одиниці інформації: вихідного коду програми, скрипту, веб-сторінки, веб-сайту, 3D моделі, текстового документу тощо.

Система керування версіями — це потужний інструмент, який дозволяє одночасно, без перешкод один одному, проводити роботу над груповими проектами.

Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у коді додатків, кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.

Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігається автор зробленої зміни та її час.

Інструменти для контролю версій входять до складу багатьох інтегрованих середовищ розробки.

Системи керування версіями існують двох основних типів: з централізованим сховищем та розподіленим.

Централізовані системи контролю версій. Централізована система контролю версій (клієнт-серверна) — система, дані в якій зберігаються в єдиному «серверному» сховищі. Весь обмін файлами відбувається з використанням центрального сервера. Є

можливість створення та роботи з локальними репозиторіями (робочими копіями).

Переваги:

– загальна нумерація версій;
– дані знаходяться на одному сервері;
– можлива реалізація функції блокування файлів;
– можливість керування доступом до файлів.

Недоліки:

– необхідність мережевого з'єднання для оновлення робочої копії чи збереження змін.

До таких систем відносять Subversion, Team Foundation Server.

Розподілені системи контролю версій. Розподілена система контролю версій (англ. Distributed Version Control System, DVCS) — система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів.

Переваги:

– кожний з розробників працює зі своїм власним репозиторієм;
– рішення щодо злиття віток приймається керівником проекту;
– немає потреби в мережевому з'єднанні.

Недоліки:

– немає можливості контролю доступу до файлів;
– відсутня загальна нумерація версії файлу;
– значно більша кількість необхідного дискового простору;
– немає можливості блокування файлів.

До таких систем відносять Git, Mercurial, SVK, Monotone, Codeville, BitKeeper.

Система контролю дозволяє зберігати попередні версії файлів та завантажувати їх за потребою. Вона зберігає повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Місце зберігання даних файлів називають репозиторієм. В середині кожного з репозиторіїв можуть бути створені паралельні лінії розробки — гілки.

Гілки, зазвичай, використовують для зберігання експериментальних, незавершених (alpha, beta) та повністю

робочих версій проекту (final). Більшість систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

Використання системи контролю версії є необхідним для роботи над великими проектами, над якими одночасно працює велика кількість розробників. Системи контролю версії надають ряд додаткових можливостей:

– можливість створення різних варіантів одного документу;

– документування всіх змін (коли і ким було змінено/додано, хто який рядок змінив);

– реалізує функцію контролю доступу користувачів до файлів. Є можливість його обмеження;

– дозволяє створювати документацію проекту з поетапним записом змін в залежності від версії;

– дозволяє давати пояснення до змін та документувати їх.

Словник основних термінів-сленгів:

– транк (trunk) — основна гілка коду;

– бранч (branch) — відгалуження;

– чекін (Check in (submit, commit)) — відправлення коду в репозиторій;

– чекаут (Check out) — одержання зміни з репозиторію;

– конфлікти — виникають, коли кілька людей правлять один і той же код, конфлікти можна вирішувати;

– патч — шматок з записаними змінами, які можна застосувати до сховища з кодом.

1.2 Порядок виконання роботи

Для керування версіями розробки програмного забезпечення обрано сервіс GitHub.

Переваги:

– простий спосіб реєстрації;

– адаптовані під розповсюдженні версії операційних систем клієнти;

– простий спосіб синхронізації між сервісом та локальною версією;

– підтримка багатьох мов програмування;

– можливість відновлення репозиторію після видалення.

Недоліки:

– для Windows необхідний NET Framework 4.0;

– ручна синхронізація;

– для проекту що складається з декількох директорій, потрібно створити відповідно декілька репозиторіїв у системі та відслідковувати кожен з них;

– невірне відображення кирилиці після відновлення з віддаленого репозиторію.

Реєстрація у системі

1 Зайти на сайт https://github.com/.

2 Натиснути на кнопку SING UP FOR GitHub (див. рис. 1.1).

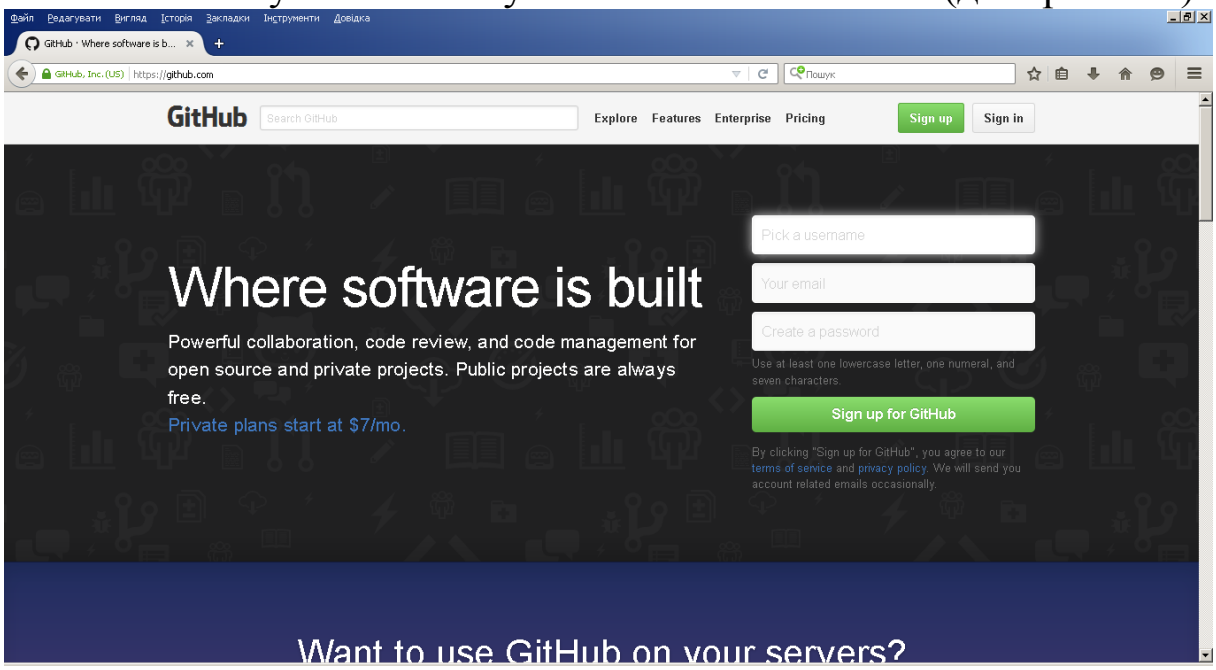

Рисунок 1.1 – Кнопка для реєстрації учасника в системі

3 Заповнити відповідні поля форми реєстрації (див. рис. 1.2).

4 Після реєстрації відкриється головна форма системи, де є можливість створити новий проект та завантажити клієнт відповідно до операційної системи (див. рис. 1.3).

5 Встановлення Git. Перед початком використання Git, необхідно встановити його на локальному комп'ютері. Встановлення Git на Linux за допомогою бінарного пакету – через основний менеджер управління пакетами (для Debian-подібного дистрибутиву, такого як Ubuntu):

```
$ sudo apt-get install git
```

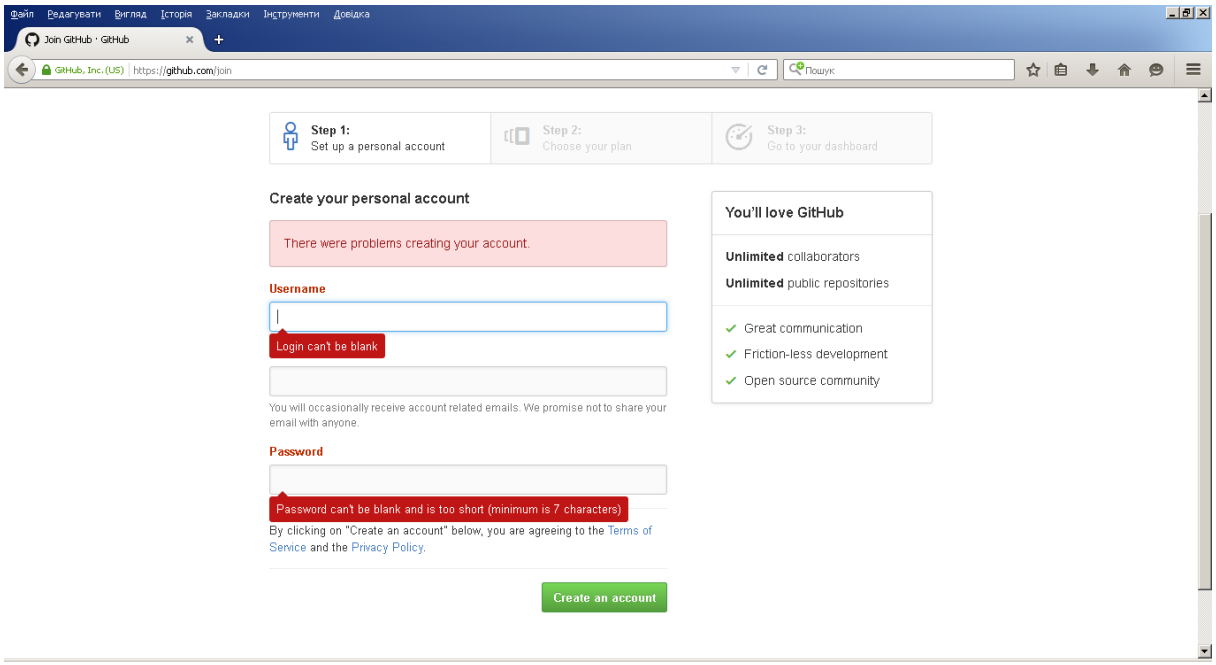

Рисунок 1.2 – Форма реєстрації учасника

6 Інсталяція на Windows: є декілька шляхів встановити Git під Windows. Офіційна збірка доступна для завантаження з сайту Git. Перейдіть до http://git-scm.com/download/win і завантаження почнеться автоматично.

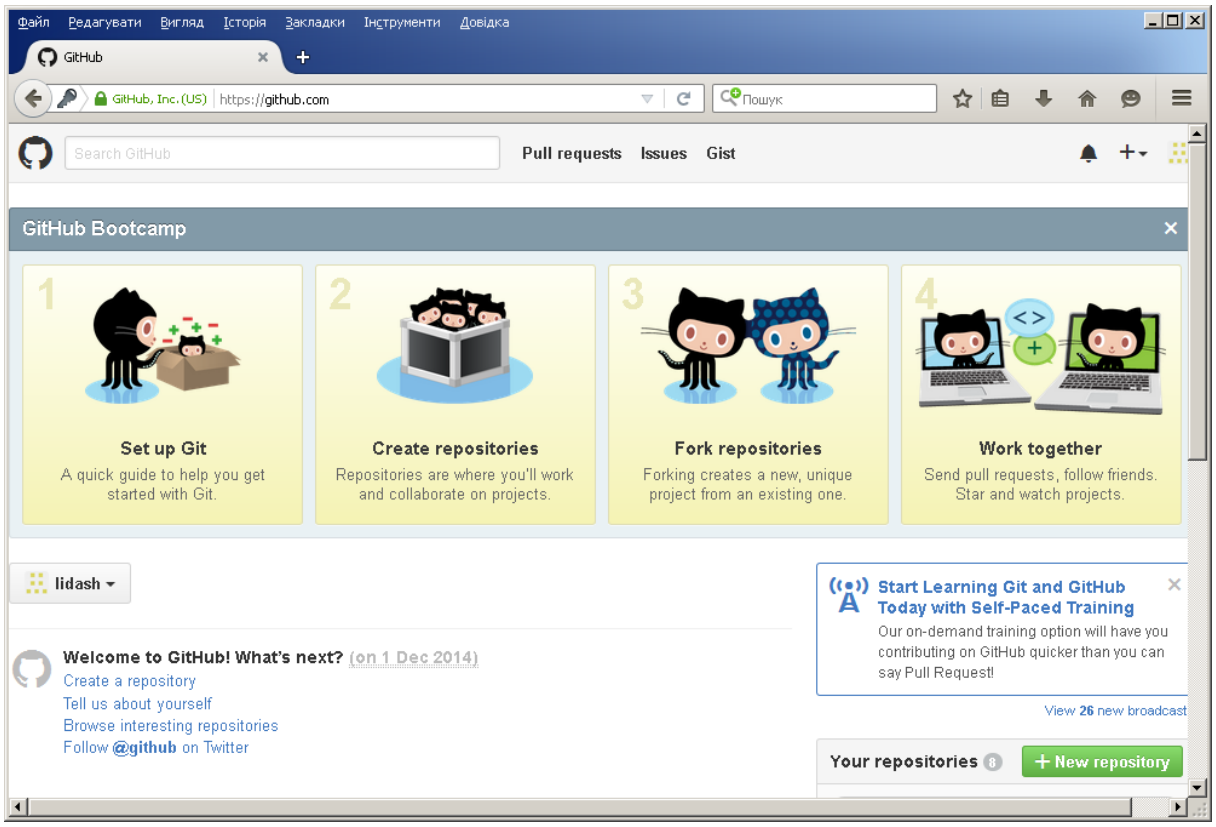

Рисунок 1.3 – Головна форма керування системою

7 Початкове налаштування Git. Налаштування потрібно виконати лише один раз (налаштування зберігаються між оновленнями). Встановлення імені користувача та адреси електронної пошти (кожен коміт в Git використовує цю інформацію і вона включена у коміти):

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Перевірка налаштувань – виконати команду git config --list, щоб переглянути всі налаштування, котрі Git встановив.

8 Створення Git-репозіторію. Для створення Git-репозиторію використовують два основних підходи. Перший приймає існуючий проект або каталог і імпортує його в Git. Другий клонує існуючий репозіторій Git з іншого сервера.

Ініціалізація репозіторію в існуючому каталозі: для початку використання Git для існуючого проекту необхідно зайти в каталог проекту та виконати:

```
$ git init
```

Це створить новий підкаталог .git, який містить всі необхідні файли репозіторію – скелет Git-репозиторію. На даний момент, у проекті ще нічого не відстежується.

Якщо необхідно додати існуючі файли під версійний контроль, необхідно проіндексувати ці файли і зробити перший коміт. Це можна зробити за допомогою декількох команд git add, що визначають файли, які необхідно відслідковувати, після чого треба виконати git commit:

```
$ git add *.c
$ git add LICENSE
$ git commit -m 'Перша версія проекту'
```

Клонування існуючого репозиторію: для отримання копії існуючого Git-репозиторію (наприклад, проекту, в якому необхідно прийняти участь) потрібна команда git clone. Git отримує повну копію майже всіх даних, що є у сервера. Кожна версія кожного файлу в історії проекту витягується автоматично, коли виконується git clone. Якщо щось станеться з диском серверу, зазвичай можна використати майже будь-який з клонів на будь-якому клієнті щоб повернути сервер до стану на момент клонування.

Щоб клонувати репозиторій треба використати команду git clone [url]. Наприклад, для клонування бібліотеки Git libgit2:

```
$ git clone https://github.com/libgit2/libgit2
```

Це створить директорію під назвою “libgit2”, проведе ініціалізацію директорії .git, забере всі дані для репозиторію, та приведе директорію до стану останньої версії.

Якщо необхідно зробити клон репозиторію в директорію з іншою назвою, можна задати її у наступному параметрі команди:

```
$ git clone https://github.com/libgit2/libgit2 mylibgit
```

Ця команда робить те саме, що й попередня, тільки цільова директорія називається mylibgit.

Git має декілька різних протоколів передачі даних, які можна використовувати: https://, git:// або user@server:шлях/до/репозиторію.git, що використовує SSH протокол.

9 Створити проект з кількома файлами, синхронізувати його з сайтом. Якщо вже є локальний репозиторій Git і треба його викласти в загальний доступ, то спочатку необхідно створити віддалений репозиторій (наприклад на GitHub), а потім виконати команди наведені нижче, змінивши відповідно назву репозиторію.

Зв’язування локального репозиторію з віддаленим:

```
$ git remote add origin https://github.com/n0tb0dy/UpRemote.git
```

Публікування вітки master у віддалений репозиторій:

```
$ git push -u origin master
```

10 Внести зміни до проекту. Відобразити зміни у проекті на сайті. Додавання всіх файлів (в тому числі і папок):

```
$ git add .
$ git commit –m ‘додавання файлів’
```

Встановлення зв'язку з віддаленим репозиторієм, обчислення локальних змін, відсутніх в ньому, і їх передачі у віддалений репозиторій:

```
$ git push
```

11 Внести зміни на сайті і відобразити їх локально в проекті. Якщо гілка налаштована слідкувати за віддаленою гілкою, це можна виконати з допомогою команди git pull для автоматичного отримання змін та злиття віддаленої гілки до поточної гілки (табл. 1.1).

Таблиця 1.1 – Основні команди git

Команда	Призначення
git add	додає вміст робочої директорії в індекс (staging area) для подальшого commit
git status	показує стану файлів в робочій директорії і індексі: які файли змінені, але не додані в індекс; які очікують комітів в індексі
git commit	бере всі дані, додані в індекс за допомогою git add, і зберігає їх зліпок у внутрішній базі даних, а потім переміщує вказівник поточної гілки на цей зліпок. Ключі: -a для додавання всіх змін в індекс без використання git add, -m для передачі повідомлення комітів без запуску редактора
git fetch	зв'язується з віддаленим репозиторієм і забирає з нього всі зміни, яких поки немає і зберігає їх локально
git pull	працює як комбінація команд git fetch і git merge, тобто Git спочатку забирає зміни із зазначеного віддаленого сховища, а потім намагається злити їх з поточної гілкою
git push	використовується для встановлення зв'язку з віддаленим репозиторієм, обчислення локальних змін відсутніх в ньому, і їх передачі в вищезгаданий репозиторій

12 Додати чужий проект.

Основні команди git наведено в табл. 1.1, дія команд в системі git показана на рис. 1.4.

Детальна інформація про можливості роботи з Git – https://git-scm.com/book/uk/v2, основні команди – http://www.fandroid.info/shpargalka-po-komandam-git/.

Отже, в результаті взаємодії з системою, стало можливим відслідковувати зміни файлів проекту. Складність системи не дозволяє повністю автоматизувати внесення змін, але слід розуміти що зміни у проекті відбуваються під контролем розробника і він самостійно може додати до списку змін внесенні ним корективи.

13 Оформити звіт.

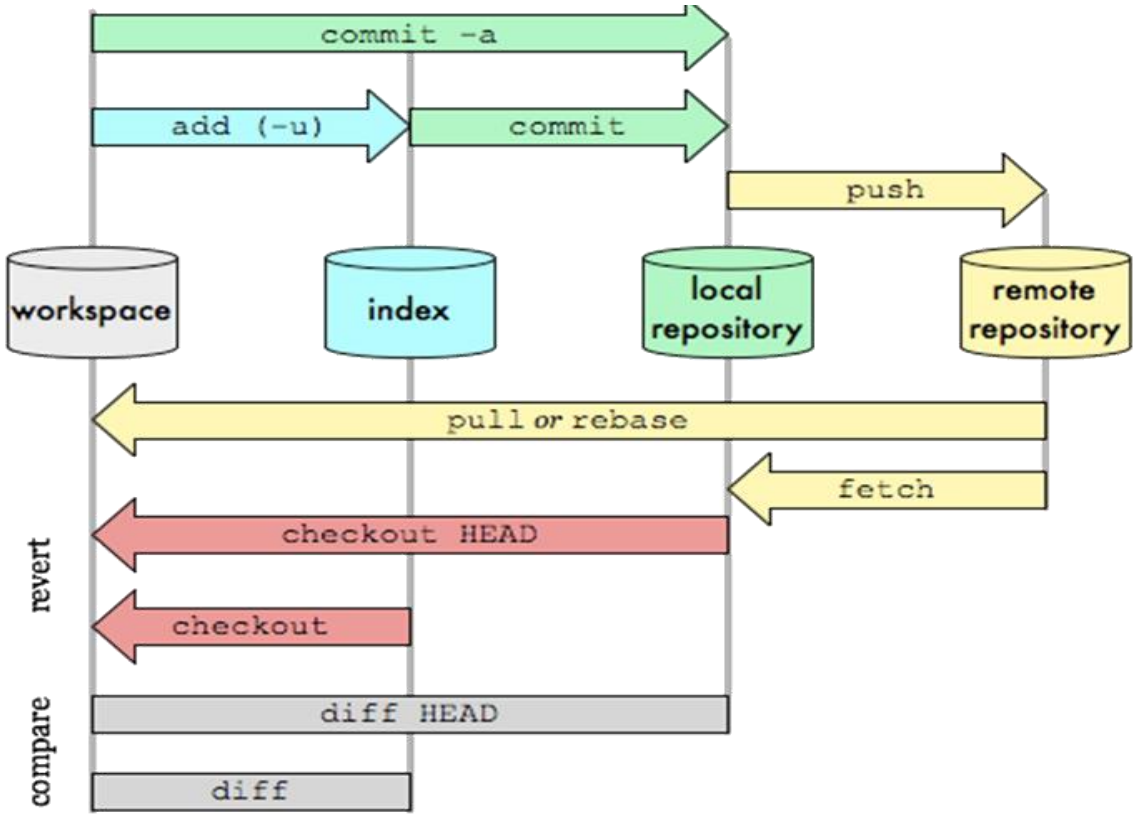

Рисунок 1.4 – Команди переміщення даних Git

1.3 Зміст звіту

Звіт має містити:

– титульний аркуш;

– мету роботи і завдання;

– покроковий опис роботи, копії екранів з результатами виконаної роботи;

– висновки.

Запитання для самоконтролю:

1 Охарактеризуйте можливості систем контролю версій.

2 Як називають основну гілку розробки проекту?

3 За якими критеріями ділять системи контролю версій?

4 Назвіть варіанти створення репозиторію git.

5 Як додати під версійний контроль існуючі файли?

Тривалість заняття: 2 год.

ЛАБОРАТОРНА РОБОТА 2.
ВСТАНОВЛЕННЯ ТА ОЗНАЙОМЛЕННЯ З СЕРЕДОВИЩЕМ ПРОГРАМУВАННЯ НА R

Мета роботи: одержання практичних навичок у встановленні середовища програмування R та його оболонки RStudio; ознайомлення з базовими об'єктами середовища.

Обладнання:

– ПК IBM PC x86 CPU з встановленою операційною системою;

- доступ до мережі інтернет;
- інсталяційний пакет R та RStudio.

2.1 Теоретичні відомості

R — мова програмування і програмне середовище для статистичних обчислень, аналізу та представлення даних в графічному вигляді. Розробка R відбувалась під істотним впливом двох наявних мов програмування: мови програмування S з семантикою успадкованою від Scheme. R названа за першою літерою імен її засновників Роса Іхаки (Ross Ihaka) та Роберта Джентлмена (Robert Gentleman), працівників Оклендського Університету в Новій Зеландії. Незважаючи на деякі принципові відмінності, більшість програм, написаних мовою програмування S запускаються в середовищі R.

R розповсюджується безкоштовно за ліцензією GNU General Public License у вигляді вільнодоступого вихідного коду або відкомпільованих бінарних версій більшості операційних систем: Linux, Windows, Mac OS X, Solaris. R використовує текстовий інтерфейс, однак існують різні графічні інтерфейси користувача (наприклад, RStudio).

R має значні можливості для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз і багато іншого. R легко розбудовується завдяки використанню додаткових функцій і пакетів доступних на сайті Comprehensive R Archive Network (CRAN – https://cran.r-project.org/). Більша частина стандартних функцій R, написана мовою R, однак існує

можливість підключати код написаний C, C++. Також за допомогою програмного коду на C або Java можна безпосередньо маніпулювати R об'єктами. R підтримує концепцію Об'єктно-орієнтованого програмування (ООП). В мові програмування R всі змінні є об'єктами, кожен об'єкт належить до певного класу.

Пакет програмного забезпечення swirl для R призначений для того, щоб проілюструвати деякі ключові поняття. Пакет перетворює консоль R в інтерактивне середовище навчання. Використання swirl дає можливість працювати в середовищі програмування R для вивчення ключових понять програмування в даному середовищі.

Встановлення R

Swirl встановлюється на версіях R починаючи з 3.0.2. Для визначення версії R в його командному рядку введіть R.version.string. Завантажити останню версію R можна на сайті https://www.r-project.org/.

Встановіть RStudio. Завантажити останню версію RStudio можна на сайті https://www.rstudio.com/products/rstudio/.

Встановлення swirl

Оскільки swirl – це пакет R, для його встановлення необхідно ввести команду з консолі R:

```
install.packages("swirl")
```

Версія swirl повинна бути ≥ 2.2.21. Визначити версію можна так:

```
packageVersion("swirl")
```

Завантаження swirl

Щоразу для використання swirl необхідно завантажити даний пакет. З R консолі:

```
library(swirl)
```

Після завантаження:

```
swirl()
```

прочитайте рекомендації про роботу в даному пакеті. Виберіть для вивчення “R Programming: The basics of programming in R” як це показано на рис. 2.1

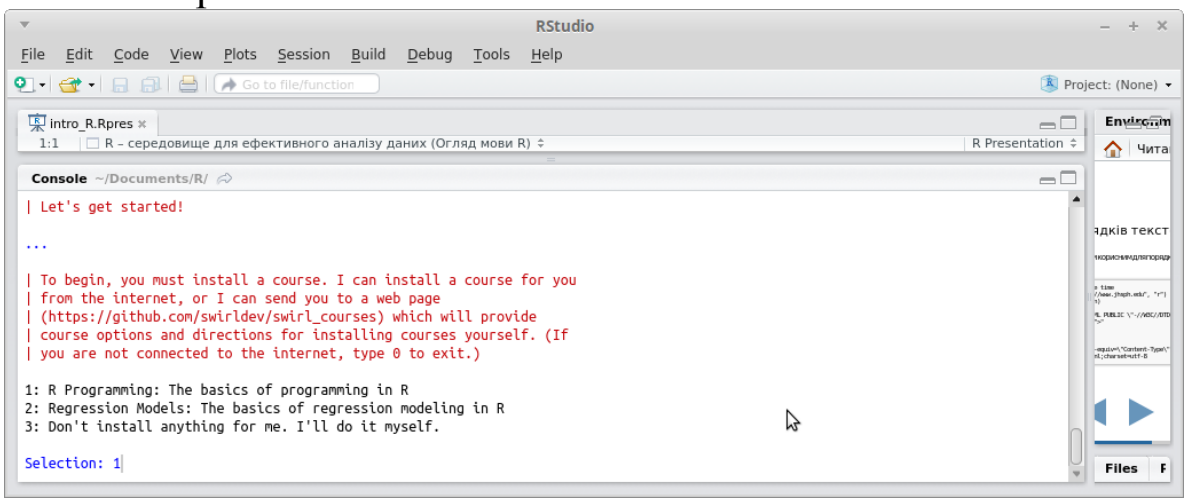

Рисунок 2.1 — Використання пакету swirl

Пакет автоматично встановить курс для використання. У встановленому програмному середовищі навчання поділено на 15 базових уроків. Для виконання лабораторної необхідно пройти 1-7 уроків для ознайомлення з середовищем розробки:

1 Basic Building Blocks
2 Workspace and Files
3 Sequences of Numbers
4 Vectors
5 Missing Values
6 Subsetting Vectors
7 Matrices and Data Frames

2.2 Порядок виконання роботи

1 Скачати і встановити R для Вашої операційної системи. Відповідно до теоретичних відомостей виведіть версію R.

2 Встановіть оболонку для роботи з R – Rstudio.

3 Встановіть пакет для навчання в середовищі R – swirl відповідно до інструкцій, наведених в теоретичних відомостях.

4 Завантажте swirl, виберіть для встановлення “R Programming: The basics of programming in R”, а потім для вивчення “R Programming”.

5 Виконайте завдання з уроків 1-7.

6 Оформити звіт.

2.3 Зміст звіту

Звіт має містити:

- титульний аркуш;
- мету роботи і завдання;
- покроковий опис роботи, команди для встановлення пакетів, копії екранів з результатами виконаної роботи, копії екранів з пройденими уроками (100%) для кожного уроку.
- висновки.

Запитання для самоконтролю:

1 Які бувають класи об’єктів в R?
2 Які об’єкти може містити вектор в R? Якого класу?
3 Чи може list (список містити об’єкти різних класів?
4 Що представляє Inf в R?
5 Які атрибути можуть мати об’єкти R?

Тривалість заняття: 2 год.

ЛАБОРАТОРНА РОБОТА 3.
НАПИСАННЯ ФУНКЦІЙ В R ДЛЯ ОБРОБКИ СТАТИСТИЧНИХ ДАНИХ

Мета роботи: одержати практичні навики у написанні функцій для обробки статистичних даних в R.

Обладнання:

– ПК IBM PC x86 CPU з встановленою операційною системою;

– встановлене програмне забезпечення R з оболонкою RStudio;

– встановлений в R пакет swirl;

– доступ до мережі інтернет.

3.1 Теоретичні відомості

Для одержання практичних навичок в написанні функцій в R виконайте наступні уроки в навчальному середовищі swirl():

8 Logic

9 Functions

14 Dates and Times

3.2 Порядок виконання роботи

1 Виконайте уроки 8, 9 і 14 в навчальному середовищі swirl.

2 Завантажити файл specdata.zip з відповідного каталогу лабораторної роботи. Архівований файл містить 332 файли, значення в яких розділені комами (CSV), що містять дані моніторингу забруднення повітря дрібними твердими частинками (PM) в 332 точках Сполучених Штатів. Кожен файл містить дані з однієї точки моніторингу, ідентифікаційний номер (ID) кожної точки моніторингу міститься в назві файлу. Наприклад, дані з точки моніторингу 200 міститься у файлі "200.csv". Кожен файл містить три змінні:

– дата: дата спостереження в форматі YYYY-MM-DD (рік-місяць-день);

– сульфат: рівень сульфату PM в повітрі в цей день (вимірюється в мікрограмах на кубічний метр);

– нітрат: рівень нітратів PM в повітрі в цей день (вимірюється в мікрограмах на кубічний метр).

3 Розпакуйте файл specdata.zip і створіть каталог 'specdata'. Зверніть увагу, що в кожному файлі є багато днів, коли або сульфат або нітрат (або обидва) не визначені (позначено NA).

4 Напишіть функцію з ім'ям 'pollutantmean', яка обчислює середнє значення для забруднюючої речовини (сульфат або нітрат) для певного переліку точок моніторингу. Функція 'pollutantmean' приймає три аргументи: “каталог”, “забруднювач” і “ідентифікатор”. Враховуючи заданий вектор точок моніторингу (їх ID), функція 'pollutantmean' читає дані про відповідний тип забруднення з каталогу, який відповідає точці забруднення і повертає середнє значення забруднення по всіх точках моніторингу, ігноруючи пропущені значення (NA). Прототип функції повинен виглядати наступним чином:

```
pollutantmean <- function(directory, pollutant, id = 1:332) {
## 'directory' is a character vector of length 1 indicating
## the location of the CSV files

## 'pollutant' is a character vector of length 1 indicating
## the name of the pollutant for which we will calculate the
## mean; either "sulfate" or "nitrate".

## 'id' is an integer vector indicating the monitor ID numbers
## to be used

## Return the mean of the pollutant across all monitors list
## in the 'id' vector (ignoring NA values)
}
```

Результат роботи функції повинен бути наступним:

```
source("pollutantmean.R")
pollutantmean("specdata", "sulfate", 1:10)
## [1] 4.064
pollutantmean("specdata", "nitrate", 70:72)
## [1] 1.706
pollutantmean("specdata", "nitrate", 23)
## [1] 1.281
```

Написана функція повинна давати наведений результат. Код необхідно зберегти у файл з ім'ям pollutantmean.R.

5 Написати функцію, яка зчитує каталог з файлами і повідомляє про кількість повністю спостережуваних випадків в кожному файлі даних. Функція повинна повертати фрейм даних, де перший стовпець – це ім'я файлу, а другий стовпець – число повних випадків. Прототип цієї функції виглядає так:

```
complete <- function(directory, id = 1:332) {

## 'directory' is a character vector of length 1 indicating
## the location of the CSV files

## 'id' is an integer vector indicating the monitor ID numbers
## to be used

## Return a data frame of the form:
## id nobs
## 1 117
## 2 1041
## ...
## where 'id' is the monitor ID number and 'nobs' is the
## number of complete cases
}
```

Результат роботи функції повинен бути наступним:

```
source("complete.R")
complete("specdata", 1)
##   id nobs
## 1  1  117
complete("specdata", c(2, 4, 8, 10, 12))
##   id nobs
## 1  2 1041
## 2  4  474
## 3  8  192
## 4 10  148
## 5 12   96
complete("specdata", 30:25)
##   id nobs
## 1 30  932
## 2 29  711
## 3 28  475
## 4 27  338
## 5 26  586
## 6 25  463
complete("specdata", 3)
##   id nobs
## 1  3  243
```

Код необхідно зберегти у файл з ім'ям complete.R.

6 Написати функцію, яка в якості аргументів бере каталог файлів даних і поріг для випадків повного спостереження і обчислює кореляцію між сульфатом і нітратом для точки моніторингу, де кількість повністю спостережуваних випадків (по всіх змінних) більша, ніж поріг. Функція повинна повертати вектор кореляцій для точок моніторингу, які відповідають вимогам порогу. Якщо немає точок моніторингу, які відповідають вимогам порогу, то функція повинна повертати числовий вектор довжиною 0. Прототип цієї функції:

```
corr <- function(directory, threshold = 0) {
## 'directory' is a character vector of length 1 indicating
## the location of the CSV files
```

```
## 'threshold' is a numeric vector of length 1 indicating the
## number of completely observed observations (on all
## variables) required to compute the correlation between
## nitrate and sulfate; the default is 0

## Return a numeric vector of correlations
}
```

Для цієї функції застосуйте функцію 'cor' в R, яка обчислює кореляцію між двома векторами.

Результат роботи функції повинен бути наступним:

```
source("corr.R")
source("complete.R")
cr <- corr("specdata", 150)
head(cr)
## [1] -0.01896 -0.14051 -0.04390 -0.06816 -0.12351 -0.07589
summary(cr)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2110 -0.0500  0.0946  0.1250  0.2680  0.7630
cr <- corr("specdata", 400)
head(cr)
## [1] -0.01896 -0.04390 -0.06816 -0.07589  0.76313 -0.15783
summary(cr)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1760 -0.0311  0.1000  0.1400  0.2680  0.7630
cr <- corr("specdata", 5000)
summary(cr)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##
length(cr)
## [1] 0
cr <- corr("specdata")
summary(cr)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.0000 -0.0528  0.1070  0.1370  0.2780  1.0000
length(cr)
## [1] 323
```

Код необхідно зберегти у файл з ім'ям corr.R.

7 Розмістити написані файли у власному обліковому записі на GitHub у репозиторії lab_STSPS.

8 Оформити звіт.

3.3 Зміст звіту

Звіт має містити:

– титульний аркуш;

– мету роботи і завдання;

– покроковий опис роботи, копії екранів з пройденими уроками (100 % для кожного уроку); код написаних функцій; коментарі до реалізації написаних функцій; резальтати тестових

запусків функцій; копії екранів з відображенням вмісту створеного репозиторію на GitHub.

- висновки.

Запитання для самоконтролю:

1 Як можна вийти з циклу типу repeat?

2 Чи можуть в R функції бути передані в якості аргументів в інші функції?

3 Як визначити значення, яке повертає функція?

4 Що таке формальні аргументи функції?

5 Як співставляються аргументи R функцій?

Тривалість заняття: 4 год.

ЛАБОРАТОРНА РОБОТА 4.
НАПИСАННЯ R-СКРИПТУ ДЛЯ ЗБОРУ, ОЧИСТКИ ТА ОБРОБКИ ДАНИХ

Мета роботи: одержання практичних навичок у зборі, очистці та обробці даних.

Обладнання:

– ПК IBM PC x86 CPU з встановленою операційною системою;

– встановлене програмне забезпечення R з оболонкою RStudio;

– доступ до мережі інтернет.

4.1 Теоретичні відомості

Актуальними розробками на даний час є розробка алгоритмів для залучення нових клієнтів як покупців продукції компаній. Дані, які використано в роботі – це дані, отримані від акселерометрів смартфону Samsung Galaxy S. Повний опис є на сайті, де були отримані дані:

http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

Дані для роботи знаходяться в папці лабораторної.

4.2 Порядок виконання роботи

Необхідно створити R скрипт, який називається run_analysis.R, який буде виконувати натупне:

1 Об'єднувати дані про тренування і тестування для утворення одного набору даних.

2 Витягувати тільки вимірювання математичного очікування та дисперсії для кожного вимірювання.

3 Використовувати назви активностей для назв діяльності в наборі даних.

4 Надавати назви даним іменами, які описують суть змінних.

5 З даних пункту 4 створіть інший незалежний набір даних з середнім значенням по кожній змінній за кожним видом діяльності і по кожному предмету.

6 Розмістити написаний скрипт у власному акаунті на GitHub у репозиторії lab_STSPS.

7 Оформити звіт.

4.3 Зміст звіту

Звіт має містити:

- титульний аркуш;
- мету роботи і завдання;
- покроковий опис роботи, код написаного скрипту з коментарями; копії екранів з відображенням результату роботи скрипту та вмісту власного репозиторію на GitHub;
- висновки.

Запитання для самоконтролю:

1 Що робить команда getwd()?

2 Який результат виконання команди file.exists("directoryName")?

3 Охарактеризуйте функцію read.table()?

4 Який символ в якості розділювача встановлює read.csv за замовчуванням?

5 Що означає атрибут na.strings при читанні даних з файлу?

Тривалість заняття: 4 год.

ЛАБОРАТОРНА РОБОТА 5.
ГРАФІЧНІ МОЖЛИВОСТІ R

Мета роботи: одержати практичні навики у побудові графічних залежностей в середовищі R.

Обладнання:

– ПК IBM PC x86 CPU з встановленою операційною системою;

– встановлене програмне забезпечення R з оболонкою RStudio;

– встановлений в R пакет swirl;

– доступ до мережі інтернет.

5.1 Теоретичні відомості

Завантаження swirl

Для використання swirl необхідно завантажити даний пакет. З консолі R:

```
library(swirl)
```

Встановіть пакет swirl “Exploratory Data Analysis” для вивчення можливостей графіки. Встановлення навчального пакету:

```
install_from_swirl("Exploratory Data Analysis")
```

Пакет автоматично встановить курс для використання. У встановленому програмному середовищі навчання поділено на 15 базових уроків.

Після завантаження:

```
swirl()
```

виберіть для вивчення “Exploratory Data Analysis” як це показано на рис. 5.1.

Для виконання лабораторної необхідно пройти 1-5 уроків для ознайомлення з механізмом побудови графіки:

1 Principles of Analytic Graphs

2 Exploratory Graphs

3 Graphics Devices in R

4 Plotting Systems

5 Base Plotting System

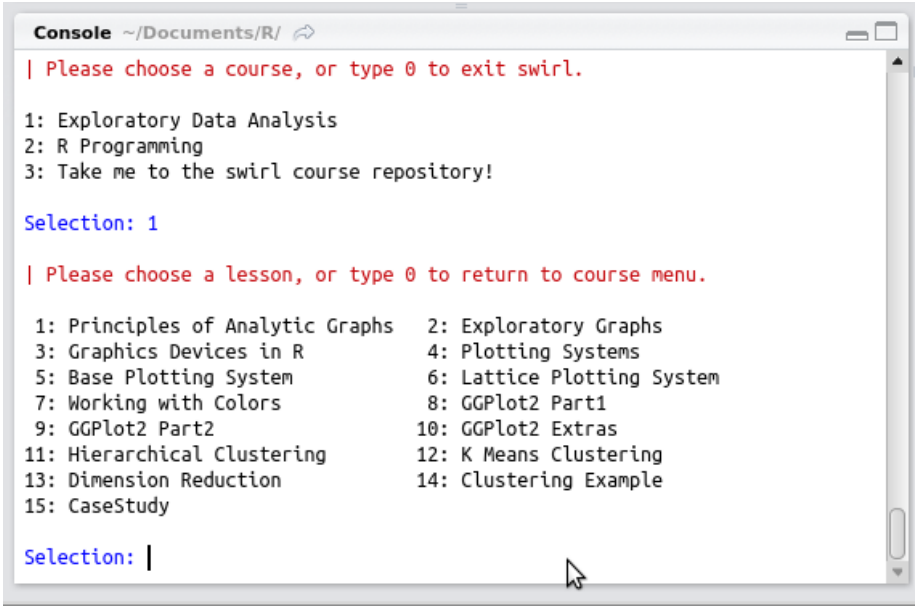

Рисунок 5.1 — Використання пакету swirl

Для виконання завдання лабораторної роботи використовуються дані з популярного репозиторію UC Irvine Machine Learning, який містить набори даних, що часто використовуються для навчання.

Зокрема, ми будемо використовувати дані про споживання електроенергії власниками будинків, які доступні в папці лабораторної роботи:

1 Набір даних: Споживання електроенергії

2 Опис: Вимірювання споживання електроенергії в одному домогосподарстві з частотою дискретизації одна хвилина впродовж періоду 4 роки.

Опис даних 9-ти змінних набору даних:

1 **Date**: дата в форматі dd/mm/yyyy

2 **Time**: час в форматі hh:mm:ss

3 **Global_active_power**: загальна активна потужність на хвилину в господарсті (в кВт)

4 **Global_reactive_power**: загальна реактивна потужність на хвилину в господарсті (в кВт)

5 **Voltage**: напруга на хвилину (у вольтах)

6 **Global_intensity**: загальна сила струму на хвилину в господарсті (в амперах)

7 **Sub_metering_1**: вимірювання енергії No. 1 (у Вт/год активної енергії). Відповідає кухні, яка містить переважно

посудомийну машину, плиту і мікрохвильову піч (плита не електрична, а газова).

8 **Sub_metering_2**: вимірювання енергії No. 2 (у Вт/год активної енергії). Відповідає кімнаті (пральня), що містить пральну машину, електросушку, холодильник і світло.

9 **Sub_metering_3**: вимірювання енергії No. 3 (у Вт/год активної енергії). Відповідає електричному бойлеру і кондиціонеру.

5.2 Порядок виконання роботи

1 Виконайте уроки 1-5 в навчальному середовищі swirl з пакету “Exploratory Data Analysis”.

2 Завантажити дані з файлу завдання лабораторної роботи. При завантаженні набору даних в R, врахуйте наступне:

- Набір даних має 2075259 рядків і 9 стовпців. Оцініть скільки пам'яті необхідно для завантаження набору даних. Переконайтеся, що Ваш комп'ютер має достатній об'єм пам'яті.
- Дані будуть використовуватись відповідно до дат 2007-02-01 і 2007-02-02. Одним з варіантів є зчитування даних тільки з цими датами, а не читати в цілому набір даних і формувати підмножини по цих датах.
- Може бути корисним перетворити змінні Date і Time в класи Date/Time в R з використанням функцій strptime() і as.Date()
- В наборі даних пропущені значення кодуються ?.

3 Побудова графіків. Мета графічної побудови — дослідження зміни використання енергії в господарстві впродовж 2 днів в лютому 2007 року. Завдання – побудувати графічні полотна з використанням базової графічної системи.

Для кожного графіку необхідно:

- Побудувати графік і зберегти його в файл PNG з шириною 480 пікселів і висотою 480 пікселів.
- Назва кожного графіка – plot1.png, plot2.png, і т. д.
- Створити окремий файл з кодом R (plot1.R, plot2.R і т.д.), який створює відповідний графік, тобто код в plot1.R будує графік plot1.png. Код файлу повинен включати в себе код для читання даних, так що графік може бути повністю відтвореним. Необхідно також включити код, який створює файл PNG.
- Помістіть файл PNG і файл коду R до папки верхнього рівня вашого git репозиторію.

4 Після виконання завдання, передайте дані з вашого git репозиторію на GitHub. Повинно бути чотири PNG файли і чотири файли коду R, загалом вісім файлів у папці верхнього рівня репозиторію.

Чотири графіки, які необхідно побудувати, наведені на рис. 5.2-5.5.

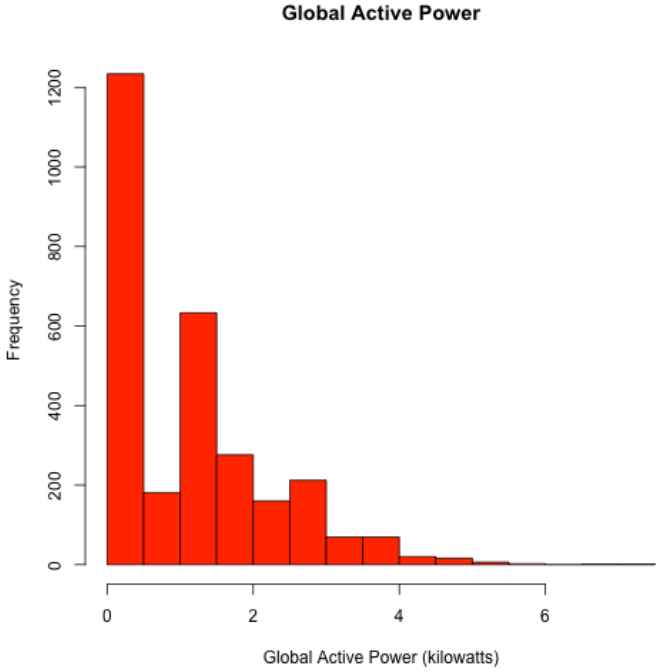

Рисунок 5.2 — Графік 1

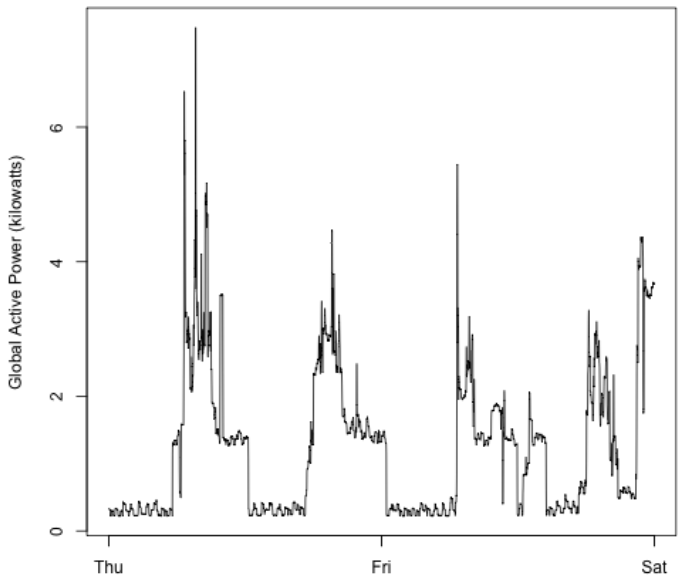

Рисунок 5.3 — Графік 2

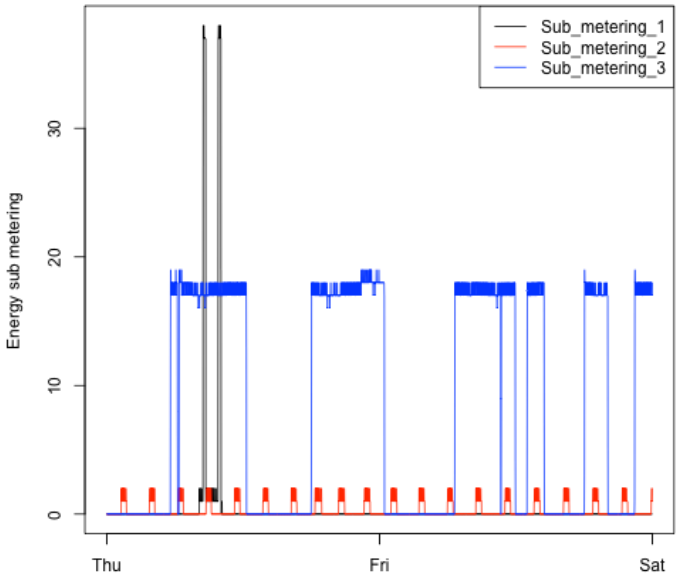

Рисунок 5.4 — Графік 3

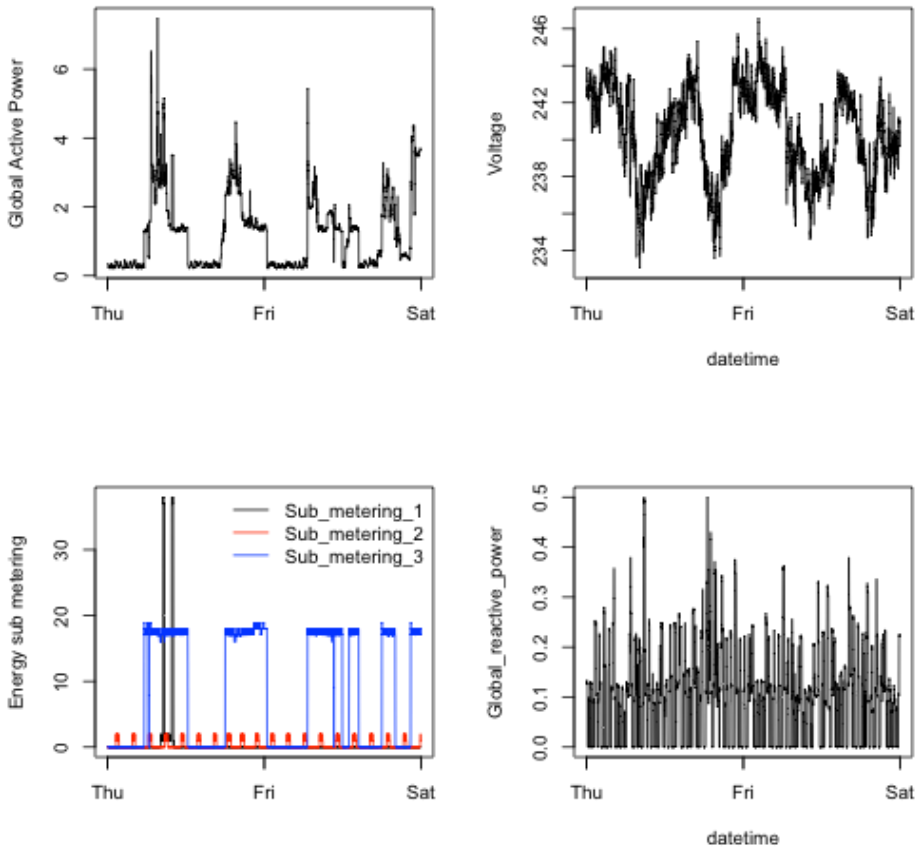

Рисунок 5.5 — Графік 4

5 Оформити звіт.

5.3 Зміст звіту

Звіт має містити:

- титульний аркуш;
- мету роботи і завдання;
- покроковий опис роботи, копії екранів з пройденими уроками (100 % для кожного уроку); код написаних файлів для побудови графіків з коментарями; копії екранів з відображенням результату роботи кодів та вмісту власного репозиторію на GitHub;
- висновки.

Запитання для самоконтролю:

1 Які Ви знаєте графічні системи побудови R?

2 Назвіть команди, які відповідають базовій графіці в R.

3 В якій графічній системі R вся побудова визначається однією функцією?

4 Що таке графічний пристрій?

5 Яка функція відкриває графічний пристрій екрану у Windows?

Тривалість заняття: 2 год.

ЛАБОРАТОРНА РОБОТА 6.
РОЗРОБКА WEB-ДОДАТКА З ВИКОРИСТАННЯМ SHINY ТА ПАКЕТІВ R

Мета роботи: одержати практичні навики у розробці Shiny-додатків та їх розгортання на серверах RStudio.

Обладнання:

– ПК IBM PC x86 CPU з встановленою операційною системою;

– програмне забезпечення R (RStudio IDE);

– доступ до мережі інтернет.

6.1 Теоретичні відомості

Shiny – платформа для створення інтерактивних програм R, які вбудовані у веб-сторінки. З Shiny можна легко створити форму вводу, яка буде викликати R; реалізовувати алгоритм прогнозування; відображати результати. Використовуючи Shiny, час для створення простих веб-продуктів на основі інтерактивних даних в R мінімальний. Shiny реалізовується в R Studio.

Встановлення:

– встановити пакет Rtools (за необхідності);

– встановити пакет Shiny:

```
install.packages(“shiny”)
libray(shiny)
```

– детальна інформація про пакет: http://rstudio.github.io/shiny/tutorial/

Проект Shiny – це папка з мінімум двома файлами:

– ui.R – інтерфейс користувача;

– server.R – контролює функціонування додатку.

Наприклад, код файлу ui.R:

```
library(shiny)
shinyUI(pageWithSidebar(
        headerPanel("Data science FTW!"),
        sidebarPanel(
                h3('Sidebar text')
        ),
        mainPanel(
                h3('Main Panel text')
        )
```

```
))
```

Код файлу server.R:

```
library(shiny)
shinyServer(
        function(input, output) {
        }
)
```

Опублікування shiny-додатку як веб-сторінки

Shiny дає можливість поділитися розробленим додатком з людьми, які не мають R (і не володіють знаннями про нього). Перевага опублікування власного додатоку в інтернеті – можливість апробації розробки широким колом користувачів, використання реалізованих можливостей додатку.

RStudio пропонує три способи для розміщення Shiny-додатків як веб-сторінок:

- Shinyapps.io;
- Shiny Server;
- Shiny Server Pro.

Shinyapps.io

Найпростіший спосіб перетворити Shiny додаток у веб-сторінку є використання shinyapps.io, хостинг RStudio для додатків Shiny. Shinyapps.io дозволяє завантажувати додаток з R сесії на сервер через RStudio. Ви повністю контролюєте ваш додаток, включаючи інструменти адміністрування.

Shiny Server

Shiny сервер є супутником Shiny, який будує веб-сервер, призначений для розміщення Shiny додатків. Він безкоштовний, з відкритим вихідним кодом і доступний з Github.

Shiny сервер – це програма-сервер, з якою сервери Linux можуть запускати Shiny додатки як веб-сторінки. Щоб використовувати Shiny сервер, необхідний сервер Linux, який має підтримку для Ubuntu 12.04 або більше (64 біт) і CentOS / RHEL 5 (64 біт).

Ви можете розмістити кілька Shiny додатків з кількох веб-сторінок на одному Shiny сервері.

Shiny Server Pro

Shiny сервер буде отримувати свій додаток в Інтернеті і піклуватися про всіх ваших потреб Shiny публікації. Тим не менше,

якщо Ви використовуєте Shiny в комерційних умовах, Ви можете додати серверні інструменти, які поставляються з більшістю платних програм серверів, таких як

- аутентифікація паролю
- підтримка SSL
- Інструменти адміністратора
- пріоритетна підтримка
- і т.д.

Все це доступно в платній професійній версії Rstudio – Shiny Server Pro.

Створення облікового запису shinyapps.io і розгортання додатку в хмарі

Shinyapps.io – це платформа як послуга (PaaS) для розміщення веб-додатків Shiny (вказівки для початку роботи з платформою розміщені за адресою http://shiny.rstudio.com/articles/shinyapps.html).

Встановлення devtools

Для використання Shinyapps.io необхідно оновити devtools до версії 1.4 або вище. Для встановлення devtools з CRAN запустіть код:

```
install.packages('devtools')
```

Встановлення rsconnect

Пакет rsconnect розгортає додатки на сервісі shinyapps.io. Встановити пакет rsconnect необхідно з сторінки розробки на Github. Це можна зробити, виконавши команду R:

```
devtools::install_github('rstudio/rsconnect')
```

Після цього завантажте пакет в сесію R:

```
library(rsconnect)
```

Створення аккаунту shinyapps.io

Необхідно перейти до shinyapps.io і натиснути кнопку "Log In." На сайті буде можливість увійти в свій обліковий запис, використовуючи Google.

Перший раз при вході в систему shinyapps.io запропонує налаштувати аккаунт. Shinyapps.io використовує ім'я облікового запису в якості доменного імені для всіх ваших додатків. Імена користувачів повинні бути від чотирьох до 63 символів і може

містити тільки букви, цифри та дефіс (-). Імена користувачів не можуть починатися з цифри або тире, і вони не можуть закінчуватися тире (див RFC 952). Деякі імена облікових записів можуть бути зарезервовані.

Налаштування rsconnect

Після того як Ви створили свій аккаунт в shinyapps.io, необхідно налаштувати пакет rsconnect для використання Вашого облікового запису. Shinyapps.io автоматично генерує token і secret, які пакет rsconnect використовує для доступу до облікового запису (рис. 6.1).

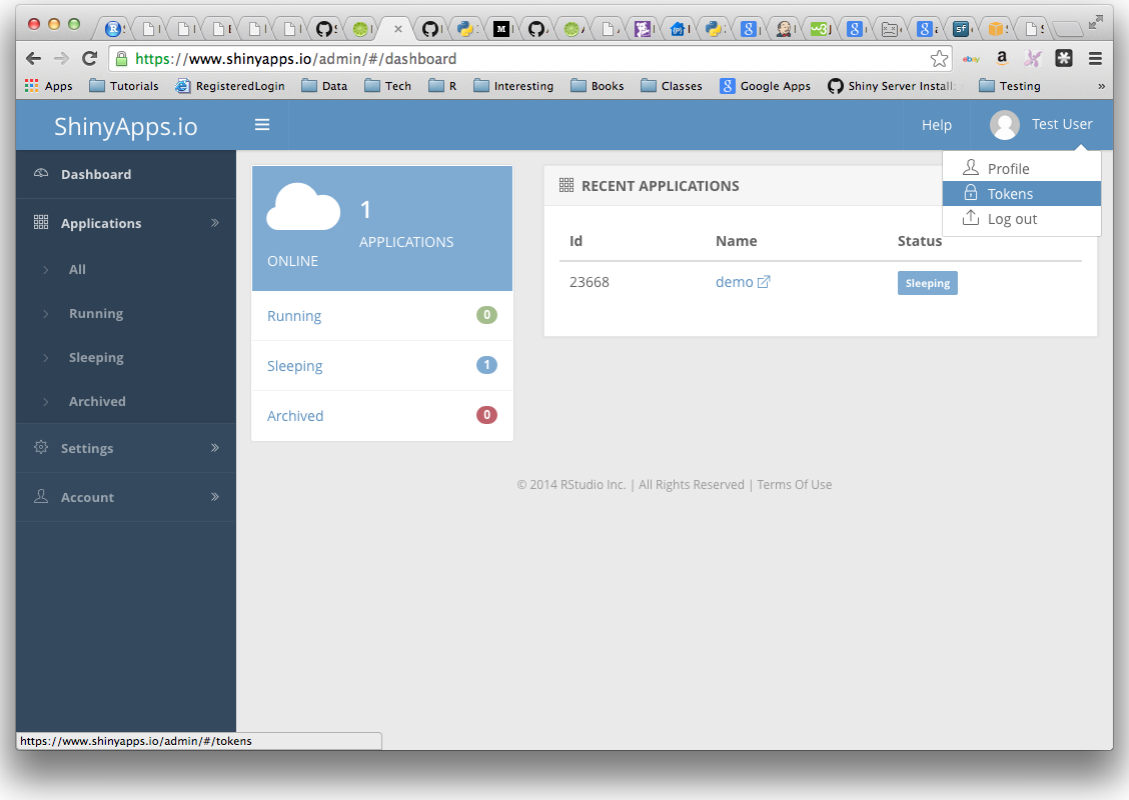

Рисунок 6.1 – Налаштування сервісу Shinyapps.io

Для використання власного облікового запису пакет rsconnect може бути налаштований двома методами:

метод 1: натисненням кнопки «Show», копіюванням вмісту параметрів функції rsconnect::setAccountInfo та вставлянням його в командний рядок в RStudio (рис. 6.2);

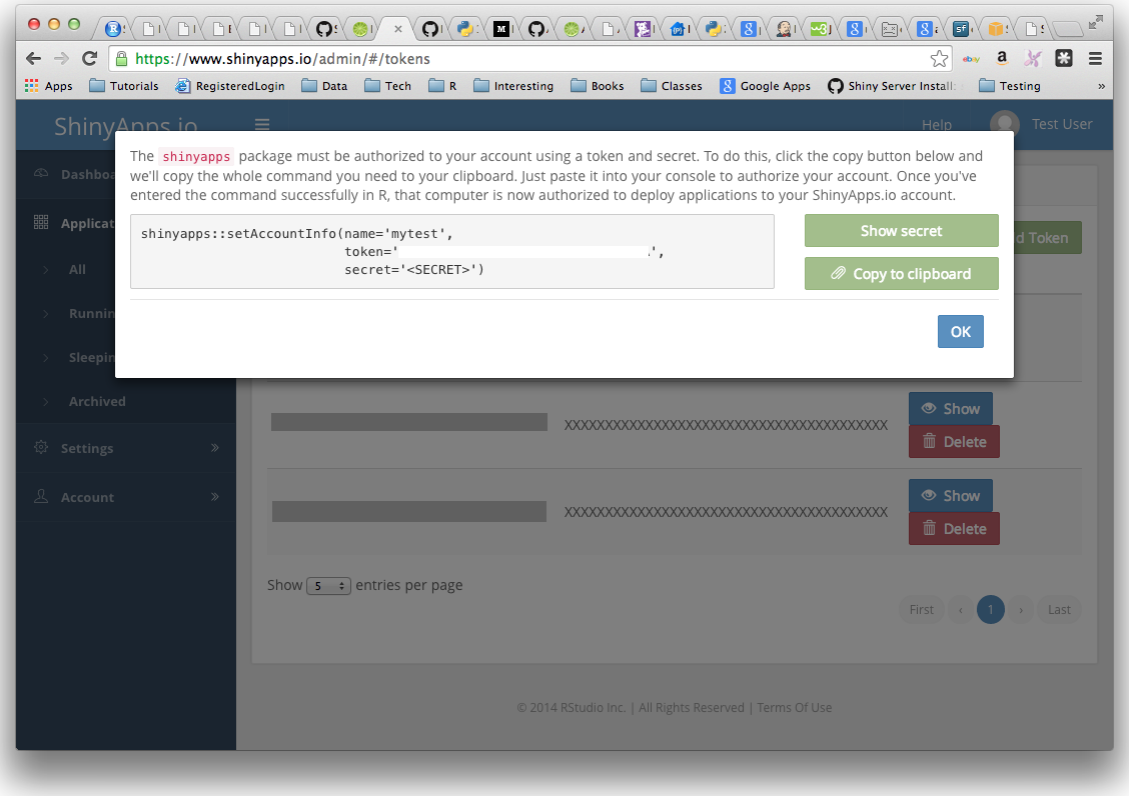

Рисунок 6.2 – Налаштування пакету rsconnect

метод 2: запуском функції setAccountInfo з пакету rsconnect з введенням даних token і secret:

```
rsconnect::setAccountInfo(name="<ACCOUNT>",        token="<TOKEN>",
secret="<SECRET>")
```

Після налаштування rsconnect, його можна використовувати для завантаження додатків на shinyapps.io.

При використанні RStudio IDE, можна керувати shinyapps.io аккаунтами через Tools -> Global Options -> Publishing.

6.2 Порядок виконання роботи

1 Встановити пакет shiny.

2 Написати shiny-додаток з документацією. Документація повинна бути подана як інструкція для початку використання додатку.

Додаток повинен включати в себе наступне:

- форми вводу (віджетів: текстовому, радіо-кнопки, прапорця, ...);
- деякі операції вводу UI в sever.R;

− деякі вихідні відображення відповіді в результаті розрахунків сервера;

− надати достатньо документації для того, щоб користувач міг почати використовувати додаток.

Тема, якій буде присвячено додаток — довільна. Можна використати обробку даних, які вбудовані в пакети R. Наприклад, простий алгоритм прогнозування на основі даних одного з пакетів R. Завантаження пакету datasets та одержання інформації про набори даних, які можна використовувати:

```
library(datasets)
library(help='datasets')
```

3 Розгорнути додаток на shiny-сервері Rstudio.

4 Опублікувати код server.R і ui.R на GitHub.

5 Оформити звіт.

6.3 Зміст звіту

Звіт має містити:

− титульний аркуш;

− мету роботи і завдання;

− покроковий опис роботи, код розроблених файлів додатку, копії екранів з виконуваним додатком в RStudio та після опублікування додатку на сервері; інтернет-адреса для доступу до додатку; опис функціонування додатку.

− висновки.

Запитання для самоконтролю:

1 Що таке фреймворк?

2 Мета застосування фреймворку Shiny.

3 З чого складається проект Shiny?

4 Охарактеризуйте модель надання хмарних обчислень PaaS (платформа як послуга). Наведіть приклад.

5 Як розгорнути Shiny-додаток в хмарі?

Тривалість заняття: 4 год.

ПЕРЕЛІК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1 Chacon S. Pro Git [Електронний ресурс] / Scott Chacon, Ben Straub. – Apress, 2014. – 2nd Edition. – Режим доступу: https://git-scm.com/book/en/v2

2 Venables W. N. An Introduction to R: A Programming Environment for Data Analysis and Graphics [Електронний ресурс] / William N Venables, David M Smith. – Network Theory Ltd, 2009. – 2nd edition. – 144 p. – Режим доступу: https://cran.r-project.org/doc/manuals/R-intro.pdf

3 Knell R. J. Introductory R [Електронний ресурс] / Robert J Knell. – 2014. – 107 p.– Режим доступу: http://www. introductoryr.co.uk/Introductory%20R%20example%20chapters.pdf

4 Matloff N. The Art of R Programming [Електронний ресурс] / Norman Matloff. – 2011. – 400 p. – Режим доступу: http://it-ebooks.org/book/no_starch/the_art_of_r_programming

5 Мастицкий С. Э. Статистический анализ и визуализация данных с помощью R [Електронний ресурс] / С. Э. Мастицкий, В. К. Шитиков. – 2015. – 400 с. – Режим доступу: https://github.com/ranalytics/r-tutorials/blob/master/Edition_2015/Book/Mastitsky_and_Shitikov_2015.pdf

6 Шипунов А. Б. Наглядная статистика. Используем R! [Електронний ресурс] / А. Б. Шипунов, Е. М. Балдин, П. А. Волкова, А. И. Коробейников, С. А.Назарова, С. В. Петров, В. Г. Суфиянов. – М.: ДМК Пресс, 2012. – 298 с. – Режим доступу: http://ashipunov.info/shipunov/school/books/rbook.pdf

7 Зорин А. В. Введение в прикладной статистический анализ в пакете R. Учебно-методическое пособие. [Електронний ресурс] / А. В. Зорин, М. А. Федоткин. – Нижний Новгород: Нижегородский гос. ун-т им. Н. И. Лобачевского, 2010. – 50 с. – Режим доступу: http://www.unn.ru/pages/e-library/methodmaterial /2010/3.pdf

8 Pace L. Beginning R : An Introduction to Statistical Programming [Електронний ресурс] / L. Pace. – 2012. – 310 p. – Режим доступу: http://www.pdfiles.com/pdf/files/English/ Desktop_Apps_Programming/Beginning_R.pdf