MY472 – Week 1: Introduction

Thomas Robinson

Course website: lse-my472.github.io

What is this course about?

The 80/20 rule of data science:
80% data manipulation, 20% data analysis

It is about the 80%

Course outline

1. Introduction to data
2. The shape of data
3. Data visualisation
4. Textual data
5. HTML, CSS, and scraping static pages
6. (*Reading week*)
7. XML, RSS, and scraping non-static pages
8. Working with APIs
9. Creating and managing databases
10. Interacting with online databases
11. Cloud computing

Plan for today

► Administration and logistics
► On the history of data and databases
► R and RStudio
► Git/Github for version control

Course philosophy

How to learn the techniques in this course?

- ▶ Lecture approach: not ideal for learning how to code
- ▶ You can only learn by doing
- → We will cover each concept three times during each week
    1. Introduction to the topic in lecture
    2. Guided coding session in lecture and lab
    3. Course assignments
- ▶ We will move relatively fast

Materials

Course website: `https://lse-my472.github.io/`

▶ Mixed set of readings, very specific to each week
  ▶ Often freely available online, otherwise, available for purchase (often in electronic versions)
  ▶ Some books are (freely) available online and in print, and the online version may be more recent

Course meetings

- ▶ Weekly lectures
- ▶ Ten one-hour classes ("labs") *starting this week*
  - ▶ Group 1: Thursdays 13:00–14:00 (CKK.2.13)
  - ▶ Group 3: Thursdays 14:00–15:00 (CKK.2.18)
  - ▶ Group 2: Thursdays 17:00–18:00 (CKK.1.09)
- ▶ No lecture/class in Week 6
- ▶ Office hours (book via StudentHub)

Assessment

- 1 practise problem set
  - Opportunity to practise format and style of response
  - Due Thursday 12 October, 16:00
- 2 further problem sets will be assessed (50% in total)
  - Submitted via Moodle
  - Only "knitted" R-markdown assignments in HTML accepted
  - Due 2 November and 7 December 2023, 16:00
- Take-home assessment (50% in total)
  - A collaborative project undertaken over winter holidays
  - Deadline: 10 January 2024, 16:00

A note on collaboration

- ▶ All assignments are individual unless we instruct you otherwise
- ▶ You may discuss with others, but your submission (including code) should be your own
- ▶ You can use online resources but always give credit in comments if you borrow code/solutions
- ▶ Any uncited code/solutions/papers/resources, or shared code, are considered plagiarism

ChatGPT (and other generative assistants)

We will **allow** ChatGPT to be used for assignments

- ▶ Ignoring the presence/possibilities of ChatGPT is unwise
- ▶ An opportunity to *learn* how to integrate these tools into your workflow

But beware:

- ▶ We are assessing your ability to deploy these tools in research contexts
- ▶ You need proficiency to recognise "good", and fix broken, code
- ▶ ChatGPT routinely produces incorrect/broken code (and information). Be very careful
- ▶ You are sharing your thoughts, ideas, and work with models that are proprietary

Plan for today

- Administration and logistics
- R and RStudio
- Git/GitHub
- Coding

Why use R

- ▶ It's free and open-source
- ▶ Quite accessible even to novice coders
- ▶ Frequently used in academia and the private sector
- ▶ Flexible and extensible through many *packages*
- ▶ Excellent online documentation and troubleshooting resources
- ▶ A fully-fledged programming language, making it easier to transition to/from other languages
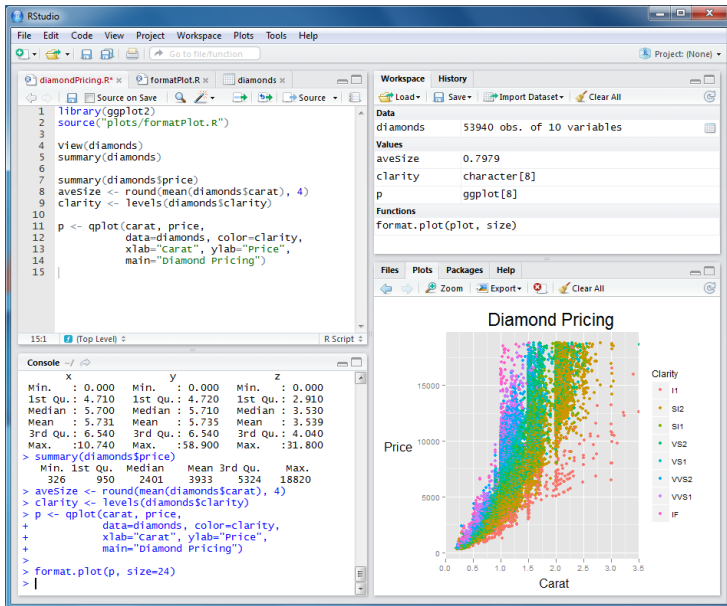
RStudio

Installing R and RStudio

- ▶ Please install R and RStudio on your laptop and bring it to lectures and labs
- ▶ Software:
  - ▶ R – Install from `https://www.r-project.org/`
  - ▶ RStudio – Install from `https://posit.co/download/rstudio-desktop/`
- ▶ *Try to install both before the lab this week. If there are any issues with the installation, we can discuss them in the lab*

Plan for today

- Administration and logistics
- R and RStudio
- Git/GitHub
- Coding

Version control

- A version control system (VCS) is key when working on code, particularly when collaborating
- It keeps records of changes in files - *who* made *which* changes *when*
- Possibility of reverting changes and going back to previous states
- When a VCS keeps the entire code and history on each collaborator's machine, it is called distributed

The main idea(s)

Your work

Main

Another person's work
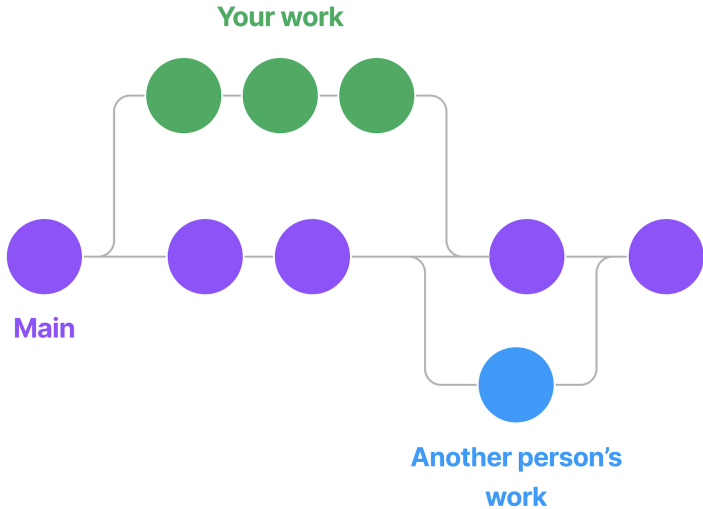
Git/GitHub

- Git: A very popular distributed version control system
- Created by Linus Torvalds in 2005 to facilitate Linux kernel development
- Other options e.g. Mercurial, Subversion
- GitHub: Service to host collections of code online with many extra functionalities (UI, documentation, issues, user profiles...)

Terminology

- *Repository/repo*: A collection of code and other files
- *Clone*: Download a repo to a computer
- *Commit*: Create a snapshot of (code) files and describe how they have changed
- *Push*: Update changes made locally on a computer also in the remote repository
- *Pull*: Obtain changes made by others which are stored in the remote repository

Installing Git

- Mac:
  - Type `git` into your Terminal and hit enter
  - If not installed already,
    `https://git-scm.com/download/mac` (use/first install Homebrew)
- Windows:
  - Downlod from `https://git-scm.com/download/win`
- Register a GitHub account at https://github.com/
  - You can apply for student benefits via
    `https://education.github.com/benefits?type=student`

Creating a repository

- First, log on to https://github.com/ with your account
- Click on the alias in the upper right hand corner $->$ Your repositories $->$ New
- Select a name, e.g. 'firstrepository'
- Select private to make it visible only to you and accounts you can select
- For the .gitignore choose the R pre-set
- Add an empty README
- Click on Create
- The repo now exists on GitHub

Configuring Git user and email

- ▶ Next, you will once need to configure Git on your computer and link it to GitHub
- ▶ Open Mac Terminal or Windows Git Bash
- ▶ Set your username in Git by pasting in Terminal/Git Bash: `git config --global user.name "Your Name"` (replace with your name before hitting enter)
- ▶ Set your commit email in Git: `git config --global user.email your@email.com`
- ▶ Then navigate to the folder where you would like to locate the repository on your computer with cd (change directory)

Cloning a respository

▶ The next step is to copy (clone) the online repository to your computer

▶ On your repository page on GitHub, click on Code and copy the URL (https)

▶ In the command line, enter `git clone ...` and replace ... with the copied url

▶ You will now be asked to enter your user name and password, for this we will have to create an access token as the last step in this setup (**note: some users are instead asked at this point to enter their password via a pop-up window - in this case, no access token has to be created manually and you can skip the next slide**)

GitHub authentication

- On GitHub, click on the alias in the upper right hand corner $->$ Settings $->$ Developer settings $->$ Personal access tokens $->$ Generate new token
- Pick a name, e.g. "command line", choose an expiration, select "repo" (this will allow to access private and public repos from the command line), and click Generate token
- Copy the token (it will only be visible once)
- Now go back to the command line, enter your GitHub user name and as password paste the token
- That's it, the setup of Git & GitHub is done and the repository was copied as well (no need to repeat the authentication until the token expires)

Creating a file

- ▶ We will now create a new file in the repository and log these changes
- ▶ With RStudio or a text editor (e.g. download VS Code at https://code.visualstudio.com/), add a file somecode.R into the repo folder
- ▶ At the command line, change into the repo folder with cd firstrepo (change directory)
- ▶ Now you are ready to commit the changes that were made to the repo

Committing changes

- First check whether anything changed with `git status` (make sure you are in the repo folder on your computer)
- Next add all untracked changes to the so-called staging area with `git add .` (we can also add only specific files)
- Commit/log changes with `git commit -m "added a code sample"`
- That's it!
- To study this again, add another line of code to the file and repeat the above steps
- Run `git log` to see the history of commits

Pushing changes to the remote repository

- To store these changes also in the remote repository, run `git push` afterwards
- It is now possible to review the changes in the browser which is very helpful for large code files
  - First, go to the repository page on GitHub and click on the clock symbol next to 'commits' in the upper right hand corner
  - Click on the key describing a specific commit, which could e.g. look something like '472cb9d', then you will see which lines of code changed
- If someone else has changed the online repository, run `git pull` to obtain the newest files

Review of key commands

- `git clone ...`: Download online repository to local computer
- `git status`: See status of files in repository
- `git add .`: Stage all changes made (alternatively add distinct file names to be staged)
- `git commit -m "some message "`: Commit (i.e. record) staged changes
- `git push`: Upload local changes to remote repository
- `git pull`: If files changed online, update local repository first

Some further concepts

- *Fork*: Own copy of a repository (pushed changes to this copy do not affect the original remote repository - different from git clone)
- *Branch*: A parallel version of the code originating from a duplication at one point
- *Merge*: Combine branches
- *Pull request*: GitHub based request to merge a branch or a fork into other code
- We will discuss these in the lab

Extensions for Git/GitHub

- ▶ People often use a combination of Git via the command line and the user interface of the GitHub website
- ▶ There is also a graphical user interface from GitHub to replace the command line (GitHub Desktop), or Git can be used directly through RStudio as an R-specific alternative to using the more general command line
- ▶ For detailed online manuals and books that discuss Git, see e.g. `https://git-scm.com/book/en/v2`
- ▶ To review GitHub, see e.g. `https://docs.github.com/e`

Useful command line prompts for Mac/Linux

- `pwd` – "Print working directory"
- `cd` – "Change directory" using relative filepaths
  - `cd ..` goes back one folder level
- `ls` – "lists" all folders and files in the current directory
- Other helpful commands can be `mkdir`, `rmdir`, `rm`, and `touch`

Plan for today

- ▶ Administration and logistics
- ▶ R and RStudio
- ▶ Git/GitHub
- ▶ Coding

Coding

Let's review some R code:

- `01-rmarkdown.Rmd`
- `02-vector-lists-dfs.Rmd`