

Python Data Collection and Management for Public Policy Research

Day 2: Intro to the Command Line, Sublime Text, Setting Up Git

Blake Miller[†]

2 July, 2024

[†]Assistant Professor, Department of Methodology, London School of Economics and Political Science (E-mail: b.a.miller@lse.ac.uk)

Agenda for Today

- Questions from Lecture 1?
- Sublime Text and Text Editors
- Coding Session 1: Practicing Sublime Text Shortcuts
- Quiz
- The Unix Shell
- Coding Session 2: Practicing and Installing the Unix Shell
- Coding Session 3: Creating a Github Account, SSH Authentication

Questions from Lecture 1?

Sublime Text and Text Editors


What is a Text Editor?

- A **text editor** is a type of software used for editing plain text.
- Essential for writing and editing code, scripts, markup, and simple text.
- Offers various features to help improve the efficiency and accuracy of writing and coding.



Why Use a Text Editor?

- **Simplicity:** Simpler than integrated development environments (IDEs)
- **Flexibility:** Customizable, includes packages that adjust to individual coding preferences and needs.
- **Powerful:** Provides powerful features such as syntax highlighting, auto-completion, shortcuts, etc.

Introducing Sublime Text

- Sublime Text is a popular, cross-platform text editor known for its speed and design.
 - Highly customizable through plugins
 - Keyboard shortcuts to make it easier to write code
 - Syntax highlighting support in most languages
 - Widely used in programming, web development, and data science.
- Download Sublime Text [here](#). 
- Other popular text editors
 - Vim
 - Emacs
 - Atom (created by GitHub)

Sublime Text Shortcuts

- Let me show you around Sublime Text real quick!
- I'll demonstrate some keyboard shortcuts you can reference here:
 - [Keyboard shortcuts for Mac](#) 
 - [Keyboard shortcuts for Windows/Linux](#) 

Coding Session 1: Practicing Sublime Text Shortcuts

Quiz

Introduction to the Unix Shell


Introduction to the Unix Shell

What is the Unix Shell?

- A command-line interface (CLI) for interacting with the system.
- Allows executing commands, running scripts, and managing system processes.


Why Use the Unix Shell?

- **Efficiency:** Automate tasks and execute complex operations quickly.
- **Control:** Direct access to manage system processes and resources.
- **Scriptability:** Automate repetitive tasks with scripts.

Reference this [tutorial on swcarpentry.com](#)  as you become more comfortable with the Unix shell.

History of the Command Line Interface (CLI)

The Evolution of the Command Line

- Developed in the early days of computing (1960s-1970s).
- Evolved from teletype interfaces to sophisticated terminal interfaces (see a demonstration of a teletype interface used with Linux [here](#) )

Importance in Early Computing

- Essential for system management before the advent of graphical user interfaces (GUIs).
- Is the basis of modern operating systems such as Linux and Mac OS.

Accessing the Unix Shell

How you access the Unix shell will depend on your operating system. For **Mac OS**:

- The **Unix kernel** forms the foundational layer of Mac OS and Linux, borrowed from the Unix Operating system.
- As such, for Mac OS and Linux operating systems, you will already have access to it natively (in the application “**Terminal**” on Mac OS)
- To access it search “Terminal” in Spotlight (command + Space).

Accessing the Unix Shell (continued)

How you access the Unix shell will depend on your operating system. For **Windows**:

- You will need to [install Git Bash](#)  to access the Unix shell.
- Open Git Bash after installation to use the shell.

Shell Commands for File Management

Navigating the File System

- `pwd` - Print Working Directory
- `ls` - List directory contents
- `cd` - Change Directory

Manipulating Files and Directories

- `mkdir` - Make Directories
- `touch` - Create empty files
- `rm` - Remove files or directories
- `cp` - Copy files or directories
- `mv` - Move or rename files or directories

File Permissions and Ownership

- `chmod` - Change file modes or Access Control Lists
- `chown` - Change file owner and group

Viewing and Editing Files

- `cat` - Concatenate files and print on the standard output
- `nano`, `vim` - Editors to modify files directly

Downloading and Transferring Files

- `wget`, `scp` - Commands for downloading and secure copying

Why the Unix Shell is Essential for Researchers and Developers

- The Unix Shell is the gateway to the **Python interpreter**, a tool useful for debugging and development.
- For many computationally-intensive tasks, we will not want to set our computers ablaze running them locally!
- Instead, we will use a **remote server**.
- **Cloud computing services** like Google Cloud or Amazon Web Services (AWS) allow near-instant creation of remote servers that meet our computing needs.

Let's use the Unix Shell!

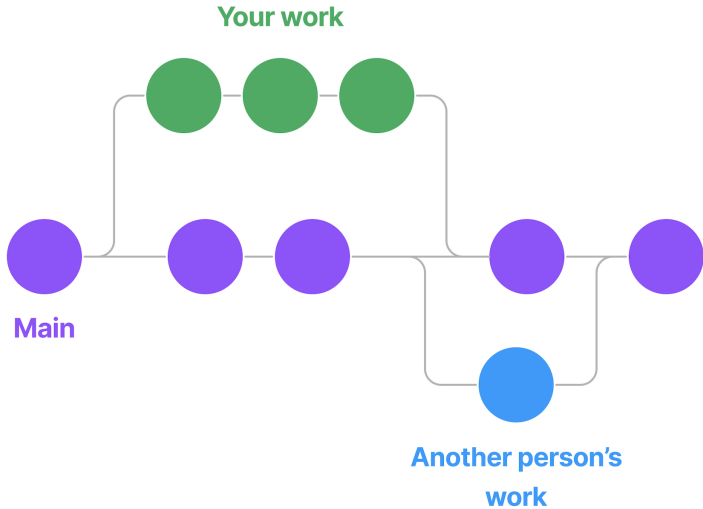
Let's try it out for ourselves!

Coding Session 2: Practicing and Installing the Unix Shell

Introduction to 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)



- **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...)

- **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

Key git commands

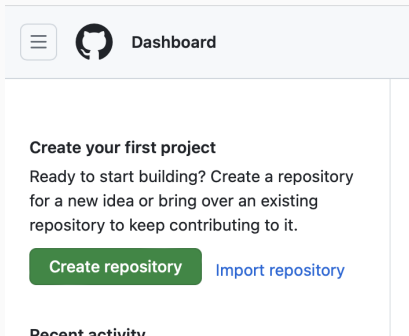
- `git clone ...`: Download online repository to local computer
- `git status`: See status of files in repository
- `git add .`: Stage all changes made (alternatively add distinct file names to be staged)
- `git commit -m 'some message'`: Commit (i.e. record) staged changes
- `git push`: Upload local 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 coding exercises

Coding Session 3.1: Creating a Github Account, SSH Authentication

Signing Up for Github



- If you don't have a GitHub account, sign up at <https://github.com/edu> 
- Login to your account.
- After logging in, click on the “Create Repository” button at the top left.

Creating Your Exercises Repository

- **Repository name:** python_fudan_exercises
- **Description:** (leave blank for now)
- **Choose visibility:** Private
- **Initialize this repository with:**
 - A README file (check this option)
- **Add .gitignore:** search for and select Python
- **Choose a license:** None
- Click "Create repository"

Coding Session 3.2: Github SSH Authentication

What is SSH?

- SSH keys provide a secure way to authenticate with GitHub.
- This tutorial will guide you through setting up SSH keys to clone the repository.

Step 1: Check for Existing SSH Keys

- Open your Terminal (Mac/Linux) or Git Bash (Windows).
- Enter the command:
 - `ls -al ~/.ssh`
- If you see an error message like `ls: cannot access '/home/user/.ssh': No such file or directory`, you do not have SSH keys. Move to step 2.
- If you do not see an error message, look for files named `id_rsa` and `id_rsa.pub`
- If they exist, you already have SSH keys. If you know the password, skip the next step, otherwise create a new key in the next step.

Step 2: Generate a New SSH Key

- If you don't have a SSH key called `id_rsa`, generate a new one, otherwise skip this step ().
- Enter the command (change your email to the email you used to sign up to Github):

```
ssh-keygen -t rsa -b 4096 -C "email@example.com"
```
- When prompted, press Enter to save the key in the default location.
- Enter a secure passphrase when prompted. Note you will not see the password as you type. If you make a mistake, press backspace several times and start over.

Step 3: Add SSH Key to SSH-Agent

- Start the SSH agent with:
 - `eval "$(ssh-agent -s)"`
 - You should see output similar to: Agent pid 12345
- Add your SSH key to the agent:
 - `ssh-add ~/.ssh/id_rsa`
 - You should see output similar to: Identity added:
/Users/username/.ssh/id_rsa (email@example.com)

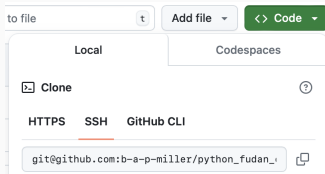
Step 4: Add SSH Key to GitHub Account

- Copy the SSH key to your clipboard:
 - `pbcopy < ~/.ssh/id_rsa.pub` (Mac)
 - `cat ~/.ssh/id_rsa.pub` (Windows/Linux)
 - If you are on Windows/Linux, you will need to manually select and copy the output.
 - On Mac, `pbcopy` has already copied the key to your clipboard.
- Go to GitHub and navigate to Settings. Click the top right profile icon, select “Settings”
- Click on “SSH and GPG keys” in the sidebar on the left.
- Click “New SSH key”
- Under “Title,” type a name that refers to your computer so you know which key corresponds to which device (when you add other keys later). For example “Blake’s LSE MacBook”
- Paste your key under “Key”.
- Click “Add SSH key”.

Step 5: Test SSH Connection

- Test the connection with:
 - `ssh -T git@github.com`
- You may see a warning message like:
 - The authenticity of host 'github.com (140.82.112.3)' can't be established... Are you sure you want to continue connecting (yes/no/[fingerprint])
 - If you do see this message, type "yes" and press enter.
- You should see a message like: Hi username! You've successfully authenticated, but GitHub does not provide shell access.

Step 6: Clone the Repository



- Navigate to the repository on GitHub.
- Click the "Code" button.
- Select the SSH tab.
- Copy the SSH URL.
- In your terminal, paste the copied text. It should look like this:
 - `git clone`
`git@github.com:yourusername/python_fudan_exercises.git`
 - The repository will be cloned to your local machine.