Getting Started with Git & GitHub

Cheat sheet for beginners

What this handout covers:
1) install/check Git • 2) create a GitHub repo • 3) clone it • 4) create a file • 5) *status* → *pull* → *add* → *commit* → *push*

Part 1 — Set up your folders

1. Create a folder on your computer **not on your Desktop**, for example:

   `gv4L3_2526`

This is the folder you'll be working out of this term.

2. Inside it, create a subfolder for this session, for example:

   `seminar1`

This folder will hold the GitHub repository you are about to clone. For future seminars, you'll create new folders.

Part 2 — Open a terminal (command line)

You will interact with Git using a *terminal*, which lets you type commands directly to your computer.

macOS

- Open **Terminal** (+ Space → type "Terminal").

Windows

- Use **Git Bash** (recommended for beginners).
- If Git Bash is not installed yet, install Git first (see Part 4).

> **Note:**
> "Terminal", "command line", "shell", and "bash" are all ways of typing commands directly to your computer.
> The commands you type will be the same.

Part 3A — Check whether Git is installed

In the terminal, type:

```
git --version
```

- If you see a version number, Git is installed.
- If not, install Git and try again.

Part 3B — Install Git (if needed)

Windows

- Install **Git for Windows** (this includes Git Bash).
- After installing, open Git Bash and run:

```
git --version
```

macOS

- Running `git --version` will often prompt macOS to install the required tools automatically.
- Follow the on-screen instructions, then re-run:

```
git --version
```

Part 4 — Create a GitHub account and repository

1. Go to **github.com**, create an account if needed, and log in.
2. Create a **new repository** called something like:

   `gv4L3_demo_repo`
3. Under **Add .gitignore**, select **R**.
4. Visibility can be **Private** (fine for class practice).
5. Click **Create repository**.

   What `.gitignore` does:
   It tells Git which files *not* to track (for example, temporary R session files like `.Rhistory`).

Part 5 — Clone the repository to your computer

Cloning means making a **local copy** of the GitHub repository on your computer.

Step 5A — Copy the repository URL

1. On GitHub, open your repository.
2. Click the green **Code** button.
3. Make sure **HTTPS** is selected.
4. Copy the URL.

Step 5B — Open a terminal in the correct folder

You want the terminal to be inside your `gv4L3_2526/seminar1` folder *before* cloning.

Option A (recommended on Windows): open Git Bash "here"

- Open File Explorer
- Navigate to `gv4L3_2526/seminar1`
- Right-click → **Show more options** → **Open Git Bash here**

This automatically puts you in the correct folder.

Option B: manually navigate using `cd`

You can also navigate by typing commands:

```
cd Dropbox
cd GV330GV4L3
ls
```

- `cd FolderName` → move into a folder
- `ls` → list contents of the current folder
- If you make a mistake:

```
cd ..
```

moves **up one level**.

Repeat until you are inside `seminar1`.

Step 5C — Clone the repository

In the terminal (inside `seminar1`), type:

```
git clone PASTE-THE-URL-HERE
```

> **Important (especially on Windows / Git Bash):**
> You usually **cannot use Ctrl+V** to paste into the terminal.
> Instead, **right-click** in the terminal and choose **Paste**.

Press **Enter**. A new folder (the repository) will appear inside `seminar1`.

Authentication note (might happen to you)

You may be asked to sign in via a browser.

Some students may hit an authentication error related to **tokens** or **passwords no longer being supported**.

If this happens: - Copy the **exact error message** - Paste it into ChatGPT or search Stack Overflow - Follow the steps for your operating system

(won't burn seminar time on this—these are fixable.)

Part 6 — Create a file in the repository

First: confirm you're saving in the correct repo folder

Make sure you are working **inside the newly cloned repository folder**, not somewhere else.

> **Note:**
> If you're working on multiple projects and creating multiple repos, this is the easiest place to get mixed up.
> Before editing/saving, double-check the repo name and folder path.

Create a file (recommended method)

1. Open **RStudio**
2. Create a new **Quarto (`.qmd`)** or **R Markdown (`.Rmd`)** file
3. **Save it inside the cloned repository folder**

   If the file is saved elsewhere, Git will not see it.

Where we are now (big picture)

Remember where we are in the workflow:

1. We created a **remote repository** on GitHub (in the cloud).
2. We **cloned** that repository to our local machine (first copy of the repo).
3. We made **local changes** (we created/edited files).

At this point:

- Your **local repository** is **ahead** of the GitHub version.
- GitHub does **not yet know** about your new files.

Now we need to **sync everything back up**.

Why `git status` matters

`git status` is the most important Git command.

It tells you: - where you are (which branch) - whether you are up to date with GitHub - which files are: - **untracked** (new files Git doesn't know about) - **modified** (changed since last commit) - **staged** (ready to be committed)

If you are ever confused, run:

```
git status
```

…and read it carefully before doing anything else.

Part 7 — Commit and push your work (the core routine)

Make sure your terminal is **inside the repository folder** (the folder created by `git clone`).

Step 7A — Check status

```
git status
```

Step 7B — Pull (good habit)

```
git pull
```

Even if nothing happens, this prevents future problems.

Step 7C — Stage your changes

```
git add .
```

(The dot means “everything in this folder”.)

Step 7D — Commit

```
git commit -m "Add seminar 1 files"
```

Step 7E — Push to GitHub

```
git push
```

Step 7F — Final check

```
git status
```

Then refresh your repository page on GitHub — your files should be there.

Common identity error (first-time setup)

If Git asks for your name and email, run:

```
git config --global user.email "your.email@example.com"
git config --global user.name "Your Name"
```

Then repeat the commit:

```
git commit -m "Your message"
```

tl;dr

Every time you work on a Git-tracked project:

1. `git pull`
2. Edit files
3. `git status`
4. `git add .`
5. `git commit -m "message"`
6. `git push`
