Introduction to Git and GitHub

Version Control for Reproducible Computational Science

October 2025

Model-Based Decisions — Week 1 Practical

Computational Science Lab, University of Amsterdam

Motivation: Why Version Control?

- Research and code evolve — we need a record of changes over time.
- Collaboration: work safely with others without overwriting each other's files.
- Reproducibility: recover previous versions and ensure results can be traced.
- Backup: your work is stored remotely (e.g., on GitHub) as well as locally.

Motivation: Why Version Control?

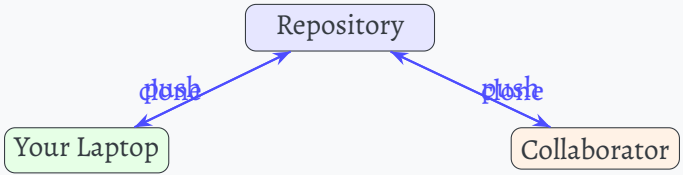
- Research and code evolve — we need a record of changes over time.
- Collaboration: work safely with others without overwriting each other's files.
- Reproducibility: recover previous versions and ensure results can be traced.
- Backup: your work is stored remotely (e.g., on GitHub) as well as locally.

Without version control

`final_report_v2_fixed_final_FINAL.docx`×

What is Git?

- **Git** is a distributed version control system.
- It records snapshots (“commits”) of your project over time.
- Every developer has a complete copy of the repository — no central lock-in.



Key Concepts

Repository (repo)	A project folder tracked by Git.
Commit	A snapshot of your project at a specific point in time.
Branch	A line of development (default is `main`).
Remote	A copy of the repo hosted elsewhere (e.g., GitHub).
Clone / Push / Pull	Commands to sync local and remote copies.

Creating and Initializing a Repository

Step 1: Create a new folder for your project

```
mkdir my_project
cd my_project
```

Step 2: Initialize Git

```
git init
```

This creates a hidden folder `.git/` that stores all version history.

Step 3: Check status

```
git status
```

Tracking and Committing Changes

```
git add my_notebook.ipynb
git commit -m "Initial commit: added notebook"
```

- `git add` stages files for the next snapshot.
- `git commit` permanently records the snapshot with a message.

Good commit messages:

- “Added visualization for cellular automaton”
- “Fixed bug in update rule”

Connecting to GitHub

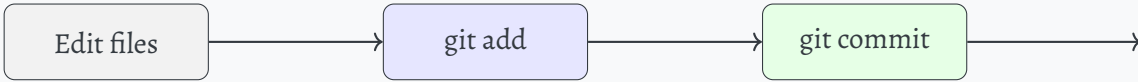
1. Create a free account at github.com.
2. On GitHub: “New Repository” → copy the HTTPS link.
3. In your terminal:

```
git remote add origin https://github.com/username/my_project.git
git branch -M main
git push -u origin main
```

4. You can now push updates with:

```
git push
```

The Basic Git Workflow



In short

`Edit → Add → Commit → Push`

Getting Updates from GitHub

If others have updated the repository, use:

```
git pull
```

This downloads and merges changes from the remote repository.

- Always `pull` before starting new work.
- Resolve any merge conflicts carefully — Git will mark conflicting lines.

Graphical Interfaces (Optional)

You do *not* need to use Git only via terminal — several excellent GUI tools exist:

- **GitHub Desktop** — free, simple interface for beginners https://desktop.github.com
- **VS Code** — built-in Git integration (source control tab).
- **Sourcetree** — Atlassian's free visual Git client.

All of these show:

- File differences before committing.
- Commit history and branches visually.
- Push/pull buttons instead of typing commands.

Common Pitfalls and Tips

- Remember to `git add` before committing — otherwise changes aren't saved.
- Never commit large data files or secrets (API keys, passwords).
- Use `.gitignore` to exclude unnecessary files (e.g., `.ipynb_checkpoints/`).
- Write descriptive commit messages — your future self will thank you.

Summary

- Git keeps a history of your work — like a time machine for your code.
- GitHub lets you share and collaborate safely.
- Learn the core cycle: `add` → `commit` → `push` → `pull`.
- Use GUI tools if you prefer a visual workflow.

Next: Practice by pushing your extended cellular automaton notebook to GitHub.