GitHub for Aggies

Chapter 1: Why You Are Using Terminal (and not the website)

The real reason (no sugarcoating)

You are using Terminal because **Git is fundamentally a command-line tool**. Everything else (GitHub website, GitHub Desktop, IDE buttons) is just a wrapper.

For your work — research, dissertation artifacts, reproducibility, IRB-sensitive materials **wrappers hide too much**.

Terminal gives you:

- **Transparency** — you see exactly what happens
- **Reproducibility** — the same commands work anywhere (Mac, Linux, servers)
- **Auditability** — every change is intentional and documented
- **Academic credibility** — this is how serious research software is managed

In short:
You're using **Terminal** because you're doing ***professional-grade work***, not because you're a programmer.

GitHub for Aggies

Chapter 2: What Git Actually Is (Plain English)

Git is **not** cloud storage.

Git is:

- a **versioned notebook**
- that lives **locally first**
- and optionally syncs to GitHub

Key principle:

Your local repo is the source of truth. GitHub is a backup + sharing mirror.

This is why you work locally and *push* when ready.

GitHub for Aggies

Chapter 3: The Three States of Every File

Every file in your repo is always in **one of three states**:

1. **Untracked / Modified**

   “I changed something, Git is aware, but it’s not saved yet.”

2. **Staged**

   “This exact version is ready to be saved.”

3. **Committed**

   “This snapshot is officially recorded forever.”

You control when files move between these states.

GitHub for Aggies

Chapter 4: The Core Workflow (This Never Changes)

This is the *entire* Git lifecycle you'll use 95% of the time:

```
Edit files → Review → Stage → Commit → Push
```

Terminal is how you move through that pipeline deliberately.

GitHub for Aggies

Chapter 5: The Essential Commands (Your Toolbox)

Below is the **canonical command list**, with *why* and *when*.

---

git status

What it does:

Shows what changed since the last commit.

Why it exists:

So you never commit blindly.

When to use it:

- Before every commit
- When confused
- When Git feels "off"

Think of it as:

"Show me the truth."

---

git pull

What it does:

Brings down the latest version from GitHub into your local repo.

Why it exists:

So you don't overwrite newer work (yours or collaborators').

GitHub for Aggies

When to use it:

- At the start of every work session
- Before pushing

Rule:

Pull before you push. Always.

---

git add

What it does:

Stages *all* current changes.

Why it exists:

To say: “This exact state is worth saving.”

When to use it:

- After reviewing changes
- When the work is logically complete

Academic analogy:

Selecting which pages go into the archival binder.

---

git commit -m "message"

What it does:

Creates a permanent snapshot with a description.

Why it exists:

Commits are your **research log**.

When to use it:

- After a meaningful unit of work
- Not necessarily after every file edit

Good commit messages:

- “Add README licensing section”
- “Reorganize root files for reproducibility”
- “Finalize env example for Docker setup”

Bad commit messages:

- “stuff”
- “changes”
- “fix”

---

git push

What it does:

Uploads your commits to GitHub.

Why it exists:

So your work is backed up, shared, and timestamped.

GitHub for Aggies

When to use it:

- When you want the world (or reviewers) to see it
- At logical milestones
- End of a work session

Rule:

If it’s not pushed, it doesn’t exist outside your laptop.

GitHub for Aggies

Chapter 6: Commands You'll Use Occasionally (But Should Know)

---

git log

Shows commit history.

Use it when:

- Writing a paper timeline
- Tracing when something changed
- You want evidence of development

---

git diff

Shows *what* changed line-by-line.

Use it when:

- Reviewing before a commit
- Unsure if a change is "too big"

---

git restore <file>

Reverts a file back to last commit.

Use it when:

- You broke something
- You want a clean slate

This is your **undo-with-a-safety-net**.

GitHub for Aggies

Chapter 7: Keeping the Repo Clean (Critical for You)

Golden rules

1. **Never commit secrets**
    - .env → ❌
    - .env.example → ✅
2. **Commit structure before content**
    - Folder organization first
    - Then code/data
3. **One purpose per commit**
    - Structure changes ≠ content changes
    - Docs ≠ experiments
4. **Local = messy, GitHub = clean**
    - Your laptop is the workshop
    - GitHub is the showroom

GitHub for Aggies

Chapter 8: What You Should Not Do

- ❌ Commit generated files (__pycache__, logs)
- ❌ Commit PDFs that are outputs unless intentional
- ❌ Rewrite history unless you understand it
- ❌ Panic — Git is recoverable

GitHub for Aggies

Chapter 9: The Daily Academic Git Routine

This is your **repeatable ritual**:

```
cd Wortschatzspiel-Pilot-2026

git pull

git status

# work in editor

git status

git add .

git commit -m "Clear, specific message"

git push
```

That’s it. That’s the whole dance.

GitHub for Aggies

Chapter 10: Why This Matters for Your Career

Using Git properly:

- strengthens your **methodology section**
- supports **open science**
- enables **reproducibility**
- signals **technical literacy** without performative coding
- aligns with **European research expectations** (JURE, EARLI, DFG, etc.)

This is not “learning Git.”

This is **learning how research artifacts live in the modern world**.

GitHub for Aggies

Final Takeaway (Write This on a Sticky Note)

Terminal is not about complexity — it's about clarity.

GitHub is not where you work — it's where your work is proven.