Applied Quantitative Methods for the Social Sciences II
MA in Social Sciences, UC3M, Spring 2026

Appendix to Assignment 1: Introduction to Git and GitHub

Contents

1 What is Git and GitHub?

Git is a version control system that tracks changes to your files over time. Think of it as an advanced "undo" system that remembers every version of your work, allowing you to go back to any previous state.

GitHub is a website that hosts Git repositories online. It allows you to:

- Back up your code in the cloud
- Share your work with others
- Collaborate on projects
- Access your files from any computer

In this guide, you will learn how to:

1. Install Git on your computer
2. Clone a repository from GitHub
3. Make changes to files
4. Commit and push those changes back to GitHub

We will show you two ways to do this: using **RStudio** (graphical interface) and using the **command line** (terminal).

2 Installing Git

2.1 macOS

Git is often pre-installed on macOS. To check, open the **Terminal** application (you can find it in Applications → Utilities, or search for it with Spotlight) and type:

```
git --version
```

If Git is installed, you will see something like `git version 2.39.0`. If not, macOS will prompt you to install the Xcode Command Line Tools—follow the prompts to install them.

2.2 Windows

Download Git for Windows from https://git-scm.com/download/win and run the installer.

Important installation options:

- When asked about the default editor, you can leave the default (Vim) or choose Notepad
- For "Adjusting your PATH environment", select **"Git from the command line and also from 3rd-party software"** (the recommended option)
- For all other options, accept the defaults

After installation, you can open **Git Bash** (search for it in the Start menu) to use Git from the command line.

3 First-Time Git Configuration

Before using Git, you need to tell it who you are. This information is attached to every change you make.

3.1 macOS

Open the **Terminal** and run:

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

Replace "Your Name" with your actual name and use the email address associated with your GitHub account.

3.2 Windows

Open **Git Bash** (search for it in the Start menu) and run the same commands:

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

3.3 Verify your configuration

To check that everything is set up correctly, run:

```
git config --global --list
```

You should see your name and email listed.

4 Setting Up RStudio for Git

Before using Git in RStudio, make sure RStudio knows where Git is installed:

1. Open RStudio
2. Go to **Tools** → **Global Options** → **Git/SVN**
3. Check that "Git executable" shows a path to Git:
    - On macOS: usually `/usr/bin/git`
    - On Windows: usually `C:/Program Files/Git/bin/git.exe`
4. If the field is empty, click "Browse" and find the Git executable
5. Click OK and restart RStudio

5 The Basic Git Workflow

The workflow we will follow is:

1. **Clone**: Download the repository from GitHub to your computer

2. **Modify**: Edit files (e.g., the README)
3. **Commit**: Save your changes with a descriptive message
4. **Push**: Upload your changes to GitHub

We will now show you how to do this using RStudio (Section 6) and from the command line (Section 7).

6 Option 1: Using RStudio

6.1 Step 1: Clone the Repository

1. Go to your repository on GitHub
2. Click the green **Code** button
3. Make sure **HTTPS** is selected
4. Copy the URL (e.g., `https://github.com/YOUR-USERNAME/your-repo.git`)

Now in RStudio:

1. Go to **File** → **New Project**
2. Select **Version Control**
3. Select **Git**
4. Paste the repository URL in "Repository URL"
5. Choose where to save the project on your computer
6. Click **Create Project**

RStudio will download the repository and open it as a project. You should see a **Git** tab in the upper-right panel.

6.2 Step 2: Modify a File

Open the `README.md` file from the Files panel and make a change. For example, add a new line:

```
This is my first edit using Git!
```

Save the file.

6.3 Step 3: Commit Your Changes

1. Go to the **Git** tab in RStudio
2. You will see `README.md` listed with a blue "M" (modified)
3. Check the box next to `README.md` to stage it
4. Click **Commit**
5. In the popup window, write a commit message (e.g., "Update README with first edit")
6. Click **Commit**

6.4 Step 4: Push to GitHub

1. Still in the Git panel, click the **Push** button (green upward arrow)
2. If prompted for credentials, see Section 8
3. Once complete, your changes are now on GitHub!

Go to your repository on GitHub and refresh the page—you should see your changes.

7 Option 2: Using the Command Line

7.1 Opening the Terminal

macOS: Open the **Terminal** application (Applications → Utilities → Terminal, or search with Spotlight).

Windows: Open **Git Bash** (search for "Git Bash" in the Start menu). Do *not* use Command Prompt or PowerShell—Git Bash provides a Unix-like environment that matches these instructions.

7.2 Step 1: Clone the Repository

First, navigate to the folder where you want to save your repository. For example, to go to your Documents folder:

macOS:

```
cd ~/Documents
```

Windows (Git Bash):

```
cd ~/Documents
```

Now clone the repository:

```
git clone https://github.com/YOUR-USERNAME/your-repo.git
```

Replace the URL with your actual repository URL (get it from the green "Code" button on GitHub).

Then enter the repository folder:

```
cd your-repo
```

7.3 Step 2: Modify a File

You can edit the README file with any text editor. Or, to quickly add a line from the command line:

macOS:

```
echo "This is my first edit using Git!" >> README.md
```

Windows (Git Bash):

```
echo "This is my first edit using Git!" >> README.md
```

To see what changed:

```
git status
```

You should see `README.md` listed as modified.

7.4 Step 3: Stage and Commit Your Changes

First, stage the file (tell Git you want to include it in the next commit):

```
git add README.md
```

Then commit with a message:

```
git commit -m "Update README with first edit"
```

7.5 Step 4: Push to GitHub

```
git push
```

If prompted for credentials, see Section 8.

Once complete, go to your repository on GitHub and refresh—you should see your changes!

8 Authenticating with GitHub

When you try to push for the first time, Git needs to verify that you have permission to modify the repository. GitHub no longer accepts your regular password for this—you need to use a **Personal Access Token** or authenticate through a browser.

8.1 Windows: The Easy Way

If you installed Git for Windows recently (version 2.29 or later), it includes **Git Credential Manager**, which handles authentication automatically:

1. When you push for the first time, a browser window will open
2. Log in to GitHub
3. Authorize the application
4. Done! Your credentials are saved for future use

If no browser window opens, see Section 8.2.

8.2 macOS: The Easy Way

The simplest method on macOS is to create a **Personal Access Token** on GitHub and use it as your password:

1. Go to GitHub → Click your profile picture → **Settings**
2. Scroll down and click **Developer settings** (left sidebar, at the bottom)
3. Click **Personal access tokens** → **Tokens (classic)**
4. Click **Generate new token** → **Generate new token (classic)**
5. Give it a name (e.g., "My laptop")
6. Set expiration (e.g., 90 days or "No expiration" for convenience)
7. Under "Select scopes", check **repo** (this gives access to your repositories)
8. Click **Generate token**
9. **Copy the token immediately**—you won't be able to see it again!

Now, when Git asks for your credentials:

- **Username**: your GitHub username
- **Password**: paste the Personal Access Token (not your GitHub password)

To save the token so you don't have to enter it every time:

```
git config --global credential.helper store
```

The next time you enter your credentials, they will be saved to a file on your computer. **Note**: This stores your token in plain text. For a more secure method, see Section 11.

9 Quick Reference: Common Git Commands

Command	What it does
`git clone <url>`	Download a repository from GitHub
`git status`	Show which files have changed
`git add <file>`	Stage a file for commit
`git add .`	Stage all changed files
`git commit -m "msg"`	Save staged changes with a message
`git push`	Upload commits to GitHub
`git pull`	Download changes from GitHub
`git log --oneline`	Show commit history (compact)

10 Troubleshooting

10.1 "Permission denied" or authentication errors

- Make sure you are using the HTTPS URL (starts with `https://`), not SSH
- On macOS: Create a Personal Access Token (Section 8.2)
- On Windows: Make sure you have Git for Windows 2.29 or later

10.2 "Repository not found"

- Check that the URL is correct
- Make sure the repository exists on GitHub
- Make sure you have access to the repository (if it's private)

10.3 "Updates were rejected"

This means someone else (or you, from another computer) pushed changes that you don't have locally. Run:

```
git pull
```

Then try pushing again.

11 Extra: The Proper Way to Log In

The methods described above work, but there are more secure and convenient ways to authenticate with GitHub. This section describes the recommended setup for regular Git users.

11.1 macOS: Installing Homebrew

Homebrew is a package manager for macOS that makes it easy to install software from the command line. If you don't have it yet:

1. Open Terminal
2. Go to https://brew.sh and copy the installation command
3. Paste it into Terminal and press Enter
4. Follow the prompts (you may need to enter your password)
5. After installation, follow any instructions to add Homebrew to your PATH

Verify Homebrew is installed:

```
brew --version
```

11.2 macOS: Option A – GitHub CLI (gh)

The GitHub CLI is a tool that lets you interact with GitHub from the command line. It handles authentication automatically.

Install:

```
brew install gh
```

Authenticate:

```
gh auth login
```

When prompted:

- Select **GitHub.com**
- Select **HTTPS** as your preferred protocol
- When asked "Authenticate Git with your GitHub credentials?", select **Yes**
- Select **Login with a web browser**
- Copy the one-time code and press Enter
- Complete the authentication in your browser

Your credentials are now stored securely, and Git will use them automatically.

11.3 macOS: Option B – Git Credential Manager

Git Credential Manager (GCM) stores your credentials securely in the macOS Keychain.

Install Git (if not already installed via Homebrew):

```
brew install git
```

Install Git Credential Manager:

```
brew install --cask git-credential-manager
```

That's it! The next time you clone or push to a repository, a browser window will open for you to log in. Your credentials will be stored securely in the macOS Keychain.

11.4 Windows: Git Credential Manager

Good news: if you installed Git for Windows version 2.29 or later, Git Credential Manager is already included. There's nothing extra to install.

The first time you push or clone a private repository, a browser window will open for authentication. After you log in, your credentials are stored in the Windows Credential Manager.

To verify you have a recent version of Git:

```
git --version
```

If your version is older than 2.29, download the latest version from https://git-scm.com/download/win.

11.5 Why Use These Methods?

- **More secure**: Credentials are stored in your operating system's secure credential storage (Keychain on macOS, Credential Manager on Windows) rather than in a plain text file
- **More convenient**: You authenticate once through your browser and never have to enter credentials again
- **Supports 2FA**: If you have two-factor authentication enabled on GitHub, these methods handle it automatically