Version Control: Backups and Improving Workflow

Dalton A. Hahn

Learning Machine Learning

Outline

- Version Control
  - Methods and Tools
- Git
  - Products and Tools
  - Process
  - Collaboration

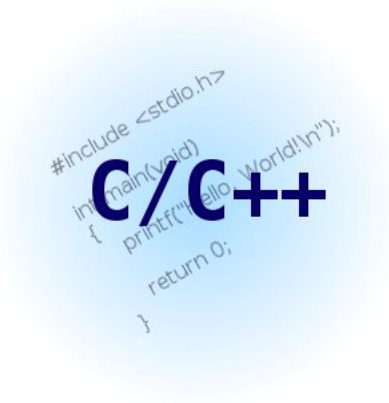
- SLIDES: https://ittc.ku.edu/~d987h530/git.pdf
- VIDEO: https://ittc.ku.edu/~d987h530/git_vid.mp4

Version Control

- Change tracking in files
- “History” of the changes made to a file
- As you edit the file, Git tracks changes and stores older versions
- Revert back to older versions
- Keep historical record of the changes you’ve made

Programming

- Python
- C/C++
- Java



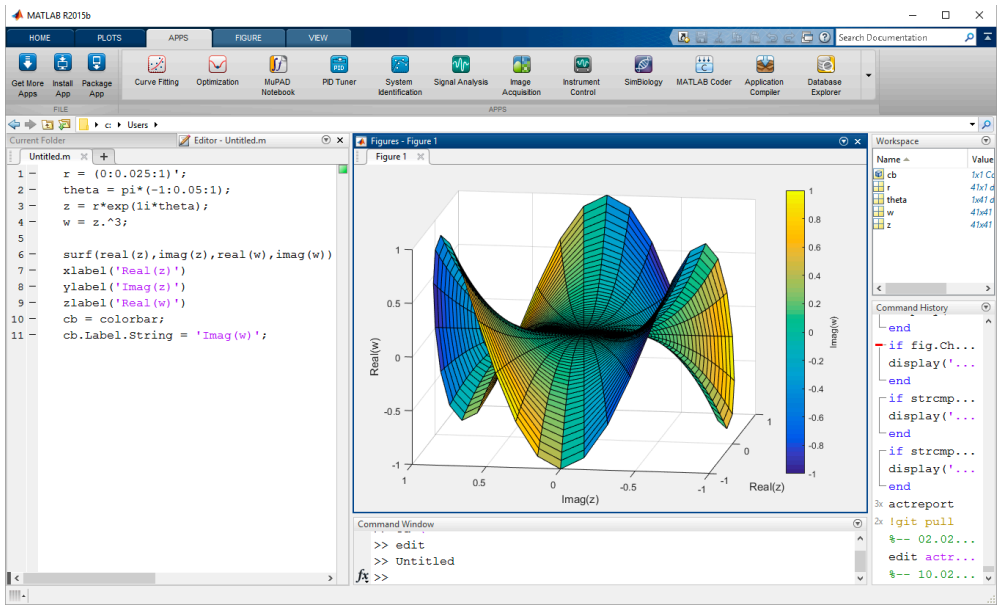
LaTeX

- Version Control for papers
- “*.tex” files
- Figures

- *Overleaf can be linked with a GitHub repository

Matlab

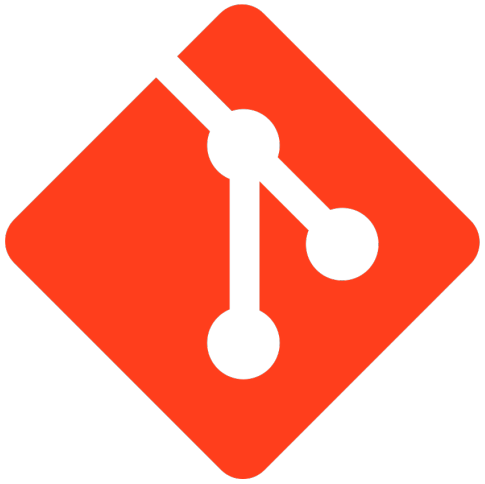
- Version control for “*.m” files
- Version control data****



****- Size limit, can’t be used as a database

Git

- Software that implements version control
  - Tracks changes in files
  - Implements functions that simplify collaboration
  - Provides mechanisms for splitting development and reincorporating splits

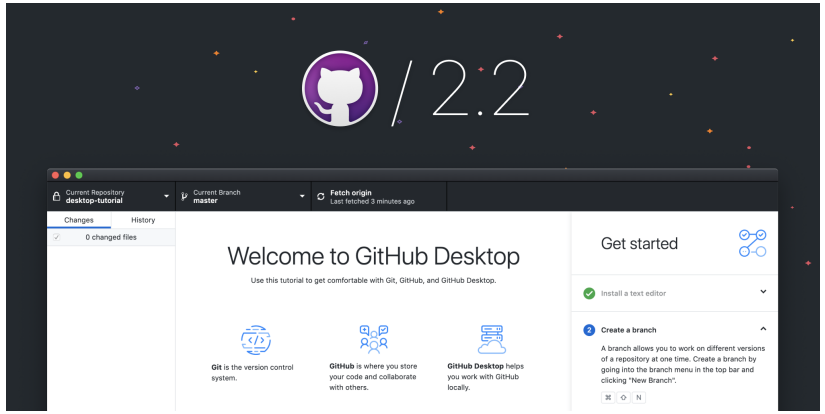


git

Tools

- GitHub - Unlimited private repositories (projects) for academics
  - Desktop Application - Graphical User Interface (GUI)

- Sign up for a Github account (https://github.com)
- Download Github Desktop (https://desktop.github.com/)



Git Workflow - Basics

- Initialize Repository/Project
- Publish Repository to Github

---------- (Start of Repeat)

- Do Some Work
- Commit Your Work
- Push Your Work

---------- (End of Repeat)

Git Workflow - Basics

- **Initialize Repository/Project**
- Publish Repository to Github

---------- (Start of Repeat)

- Do Some Work
- Commit Your Work
- Push Your Work

---------- (End of Repeat)

Git Workflow - Basics

- Initialize Repository/Project
- **Publish Repository to Github**

---------- (Start of Repeat)

- Do Some Work
- Commit Your Work
- Push Your Work

---------- (End of Repeat)

Git Workflow - Basics

- Initialize Repository/Project
- Publish Repository to Github

---------- (Start of Repeat)

- **Do Some Work**
- Commit Your Work
- Push Your Work

---------- (End of Repeat)

Git Workflow - Basics

- Initialize Repository/Project
- Publish Repository to Github

---------- (Start of Repeat)

- Do Some Work
- **Commit Your Work**
- Push Your Work

---------- (End of Repeat)

Git Workflow - Basics

- Initialize Repository/Project
- Publish Repository to Github

---------- (Start of Repeat)

- Do Some Work
- Commit Your Work
- **Push Your Work**

---------- (End of Repeat)

Git Workflow - Basics

- Initialize Repository/Project
- Publish Repository to Github

---------- **(Start of Repeat)**

- **Do Some Work**
- **Commit Your Work**
- **Push Your Work**

---------- **(End of Repeat)**

Git Workflow - Working with Collaborators

- Initialize Repository/Project
- Publish Repository to Github

---------- (Start of Repeat)

- Do Some Work
- Commit Your Work
- Push Your Work

---------- (End of Repeat)

Git Workflow - Working with Collaborators

- Initialize Repository/Project
- Publish Repository to Github

---------- (Start of Repeat)

- **Fetch+Pull Other's Work**
- Do Some Work
- Commit Your Work
- Push Your Work

---------- (End of Repeat)

Advanced Topics/Techniques

“I Broke It”

- Your super productive 4am coding session went great
- You wake up the next morning to test the project
  - **Nothing works**

- Git Revert to pull project back to a previously “good” commit

- *Git Checkout to go back and test an old commit*

Collaboration – “Branching” the Project

- Jim has a genius idea for a feature for your project
- You want to incorporate this feature, but you’re not sure if Jim’s “genius idea” will break the main project
  - Alternatively, don’t want to clutter the main project with untested features
- Have Jim “BRANCH” the repository and he can work on his changes there instead of the main branch (master branch)

Collaboration – Working the Branch

- Users must “checkout” the branch before they can make changes

- **Return to basic workflow presented in previous slides**

Collaboration – “Merging” the Branch

- Jim’s genius feature is complete and needs to be incorporated to the main project

- Create a “PULL REQUEST” and “MERGE” these changes to the main project

Keeping Secrets and Data Out

- Most projects require some data that’s not code
- For ML, this data may be HUGE
- Git ignore files can be used to exclude certain files from being tracked and uploaded to GitHub

Command-Line Interface vs. GUI

- All actions possible in GitHub Desktop can be performed in a Terminal/Command Line
  - For Linux users, you must use the CLI (Command-Line Interface)

- **Same exact workflow that was presented in previous slides**

Conclusions

- Version Control can be used as a backup for project code and documents
- Used to revert changes back to working conditions
- Changes and files can be tracked and edited by many people in a group and kept in one central location
- Not limited to just traditional “programming” files
  - Matlab, LaTeX, etc.