Learning to Love Version Control

Manodeep Sinha

Swinburne University

What will you learn?

- **Why** you should use version control

What will you learn?

- Why you should use version control
- When you should use version control

What will you learn?

- Why you should use version control
- When you should use version control
- How you should use version control

What is version control?

What is version control?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later

Why should you version control?

- Makes tracking your own work easier

Why should you version control?

- Makes tracking your own work easier
- Lets you recover from “oopsies”

Why should you version control?

- Makes tracking your own work easier
- Lets you recover from “oopsies”
- Collaborating is significantly easier (esp. with modern tools)

Why should you version control?

- Makes tracking your own work easier
- Lets you recover from “oopsies”
- Collaborating is significantly easier (esp. with modern tools)
- Note: Version Control is not *backup*. Backups also protect you from hardware failures

When should you version control?

- Working on a document (code/paper) that you can not easily recreate

When should you version control?

- Working on a document (code/paper) that you can not easily recreate
- For me, I use git + github as version control + backup. For all substantial work (>= 1 day effort), I create repos first on GitHub and then start working

When should you version control?

- Working on a document (code/paper) that you can not easily recreate
- For me, I use git + github as version control + backup. For all substantial work (>= 1 day effort), I create repos first on GitHub and then start working
- The standard VCS is designed for many developers working simultaneously on parts of a big project. Astronomers are typically 1 person/project - almost zero overhead

When should you NOT version control?

When should you **NOT** version control?

- Large binary files (> few MB)

When should you NOT version control?

- Large binary files (> few MB)
- Where the file does not change (use backup)

How to version control

- git (https://github.com/git/git)

How to version control

- git (https://github.com/git/git)
- mercurial (hg) https://www.mercurial-scm.org/

git/hg ?

git/hg ?

- Choice of VCS may be made for you

git/hg ?

- Choice of VCS may be made for you
- I will recommend git + GitHub ecosystem. hg uses python and I have run into python installation issues

git/hg ?

- Choice of VCS may be made for you
- I will recommend git + GitHub ecosystem. hg uses python and I have run into python installation issues
- Either choice is fine

Let’s create a repo

Let’s create a repo

- git init

Let’s create a repo

- git init
- hg init

Add the files

- git add <list of files>

Add the files

- git add <list of files>
- hg add <list of files>

Commit the files

- git commit -m “message”

Commit the files

- git commit -m “message”
- hg ci -m “commit message”

Check the commit was made

- git log

Check the commit was made

- git log
- hg log

Make and commit another change

- git add -u . && git commit -m “message”

Make and commit another change

- git add -u . && git commit -m “message”
- hg ci -m “message”

Make and commit another change

- git add -u . && git commit -m “your commit message”

Make and commit another change

- git add -u . && git commit -m “your commit message”
- hg ci -m “your commit message here”

github/bitbucket

- At minimum, stores your codes/papers

github/bitbucket

- At minimum, stores your codes/papers
- Also, a backup

github/bitbucket

- At minimum, stores your codes/papers
- Also, a backup
- You do not need github/bitbucket to have VC

github/bitbucket

- git clone <github url>

github/bitbucket

- git clone <github url>
- git add <list of files>

github/bitbucket

- git clone <github url>
- git add <list of files>
- git commit -m “initial commit”

github/bitbucket

- git clone <github url>
- git add <list of files>
- git commit -m “initial commit”
- git push

github/bitbucket

- git remote add origin <github url>

github/bitbucket

- git remote add origin <github url>
- git pull origin master —allow-unrelated-histories

github/bitbucket

- git remote add origin <github url>
- git pull origin master —allow-unrelated-histories
- git push