6-2020

Developing Open Source Software using Version Control Systems: An Introduction to the Git Language for Documenting Your Computational Research

Jared D. Smith  
*University of Virginia, js4yd@virginia.edu*

Jonathan D. Herman  
*University of California, Davis, jdherman@ucdavis.edu*

Follow this and additional works at: [https://digitalcommons.usu.edu/ecstatic_all](https://digitalcommons.usu.edu/ecstatic_all)

Part of the Civil and Environmental Engineering Commons, Computer Law Commons, Other Computer Sciences Commons, and the Systems Engineering Commons

**Recommended Citation**  
[https://digitalcommons.usu.edu/ecstatic_all/87](https://digitalcommons.usu.edu/ecstatic_all/87)

This Lecture Material is brought to you for free and open access by the ECSTATIC Repository at DigitalCommons@USU. It has been accepted for inclusion in All ECSTATIC Materials by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
Developing Open Source Software using Version Control Systems

An Introduction to the Git Language for Documenting your Computational Research

Jared D. Smith, University of Virginia
js4yd@virginia.edu
and
Jon Herman, University of California, Davis
jdherman@ucdavis.edu
Outline

1. What is Version Control? Open Source Software?
2. Why are they good for research?
3. The Git language for version control
4. How to install and use the Git language
5. Visualizing Git commands and commit history
6. Cloud-hosting Git code repositories
7. Legal Issues and Thoughts
8. Git Tutorial

Goal: Become more familiar with using git commands
Computational research: what are the “experimental steps”?

```
from __future__ import division
import numpy as np
from scipy.stats import norm
from util import read_param_file
import common_args

# Perform Sobol Analysis on file of model results
# Returns a dictionary with keys 'SI', 'S1_conf', 'ST', and 'S1T_conf'
# Where each entry is a list of size J (the number of parameters)
# Containing the indices in the same order as the parameter file

def analysis(pfile, output_file, column = 0, calc_second_order = True, num_resamples = 1000,
               delete = ' ', conf_level = 0.95, print_to_console=False):
    param_file = read_param_file(pfile)
    Y = np.loadtxt(output_file, delimiter=',' , usecols=(column,))
    D = param_file['num_vars']
    if calc_second_order and Y.size % (2*D + 2) == 0:
        N = int(Y.size / (2*D + 2))
    elif not calc_second_order and Y.size % (D + 2) == 0:
        N = int(Y.size / (D + 2))
    else:
        raise RuntimeError("Incorrect number of samples in model output file.
                           Confirm that calc_second_order matches option used during sampling."")
    if conf_level < 0 or conf_level > 1: raise RuntimeError("Confidence level must be between 0-1.")

A = np.empty(N)
B = np.empty(N)
AB = np.empty((N,D)) if calc_second_order else None
step = 2*D/2 if calc_second_order else D/2
```

Code!

J. Herman (2014)
Some familiar motivation

J. Herman (2014)
Have you ever …

1. Tried to improve a program and broken it so badly you wished you could abandon all your changes?

2. Left a program in a broken state, and needed new results right away?

3. Tried to work on a program at the same time as others?

The “manual copy” approach doesn’t work very well. Enter Version Control Systems (VCS) like git and mercurial.

Adapted from Lupton 2012, APC 524 course at Princeton

J. Herman (2014)
More Motivation: “The Reproducibility Crisis”

Assessing data availability and research reproducibility in hydrology and water resources

James H. Stagge\textsuperscript{1,2}, David E. Rosenberg\textsuperscript{1}, Adel M. Abdallah\textsuperscript{1,3}, Hadia Akbar\textsuperscript{1}, Nour A. Attallah\textsuperscript{1} & Ryan James\textsuperscript{1}

There is broad interest to improve the reproducibility of published research. We developed a survey tool to assess the availability of digital research artifacts published alongside peer-reviewed journal articles (e.g., data, models, code, directions for use) and reproducibility of article results. We used the tool to assess 360 of the 1,989 articles published by six hydrology and water resources journals in 2017. Like studies from other fields, we reproduced results for only a small fraction of articles (1.6\% of tested articles) using their available artifacts. We estimated, with 95\% confidence, that results might be reproduced for only 0.6\% to 6.8\% of all 1,989 articles. Unlike prior studies, the survey tool identified key bottlenecks to making work more reproducible. Bottlenecks include: only some digital artifacts available (44\% of articles), no directions (89\%), or all artifacts available but results not reproducible (5\%). The tool (or extensions) can help authors, journals, funders, and institutions to self-assess manuscripts, provide feedback to improve reproducibility, and recognize and reward reproducible articles as examples for others.
More Motivation: “The Reproducibility Crisis”

Many published studies are difficult or impossible to reproduce, given the paper and any supplied documentation alone.

Others require significant technical support from experts who have developed the code.

Only the code that ran the experiments can be used to exactly reproduce them. Modelers are biased by their education and experiences, and each programmer will make their own assumptions, which may or may not be documented in academic publications or reports.

Even code that is provided and reproducible may be impossible to understand – the ultimate goal of reproducibility is on out-of-sample tests to develop consensus (replicability).
What is the main contribution of computational research? The paper? The code?

“The idea is: An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.” David Donoho, 1998.
"Code isn’t very polished"
→ Performs algorithm described in paper? If yes, publish; if no, fix.

"It’s not common practice"
→ This should change

"It’s the IP of my institution"
→ Sometimes, if new discovery. Pay attention to licensing.

"People will criticize and demand technical support" (see left)

IP: intellectual property  J. Herman (2014)
**Version Control**
Keeping track of all code changes across developers

**Open Source**
Sharing your code online

**THE FUTURE**
(is now!)

J. Herman (2014)
Why are version control and open source software good for research?

• Always know where to find the latest version of a project
• Simple to share - no passing of zip folders by email
• Easy to modify without breaking the code
• Easy to work with collaborators from other institutions
• Good way to document the exact code used to reproduce studies – with complete revision history.
  • Easy to change in the future, if needed
• Promotes your work
• Saves time
• Cloud backup for your code

Modified from J. Herman (2013, 2014)
What is git?

- An open source distributed version control system
- Developed by Linus Torvalds in 2005 to keep track of Linux kernel development ("distributed" is important here)
- Now the most widely used VCS
- Originally a command line tool, but desktop versions are available

J. Herman (2014)
Useful Resources to Learn the Git Language

AAWATS (Ashley Watson?)

dictionary

Jon Herman’s Blog Posts

( Intro to Git, Working with Remotes )

Tutorials from Atlassian on All Git Commands

Rarely need more than the commands on the cheat sheet

Some visualizations of commands
How to install git

- On Debian/Ubuntu: `sudo apt-get install git`
- Also available as a Cygwin package
- Available on most clusters (run “module load git”)

Additional add-ins:
- Windows PowerShell with Git is nice

Modified from J. Herman (2013)
Git Commands
git commit: save the current state

Without git

With git

A change

Each commit (checkpoint) saves a message, the date/time, and ID of the person saving it.

The “current state” refers to all files in the repository. You can choose which files to include/exclude.

Modified from J. Herman (2014)
git log: view entire revision history

jdh366 /cygdrive/d/jon/code/pdfextract (master)
$ git log

commit dca3c012e391741d4c5c4e921b266af978815ac9
Author: Karl Jonathan Ward <karl.j.ward@gmail.com>
Date: Mon Jul 21 15:52:25 2014 +0100

  extract-bib example in readme

commit 358df84f6c434f9c6275f4caac5dccc0ff044e177
Merge: 4ade684 3358398
Author: Karl Jonathan Ward <karl.j.ward@gmail.com>
Date: Mon Jul 21 15:34:08 2014 +0100

  Merge pull request #11 from jdherman/feature/bibtex-resolve

  Fix reference resolver, and add option to output BibTeX with the CrossRef API

commit 3358398dfe6efb52e826b7d89bb08c6eaf4dc093
Author: jdherman <jdherman8@gmail.com>
Date: Mon Jul 21 10:20:35 2014 -0400

  add v1 to api link

J. Herman (2014)
git reset: **reset** to a prior state

The undone commits can either be kept (a “soft reset”) or removed (a “hard reset” – careful with this).

Example: `git reset --hard HEAD~1`

Reset back 1 commit. Can change 1 to any number of commits.

[Git undo resource](#)  

Modified from J. Herman (2014)
`git branch`: create a new branch

All commits done in the new branch do not affect master. This is incredibly helpful for testing large changes/additions.

Be aware that if you switch branches, your directory will change the files it has in it.

Modified from J. Herman (2014)
git merge: add changes from other branch

master

new-feature

Or, if the new feature doesn’t pan out, just go back to master and delete new-feature

Merge conflicts will only occur if the same code was edited in both branches. Resolve manually.

J. Herman (2014)
Pro Tips for Git

Entering ‘git’ will show a list of common commands

‘git status’ displays all modified and untracked files

Enter ‘q’ to escape from a long list of messages

When your repository has multiple branches, it can be useful to have the branches in separate directories

Only keep source code under version control. You can have other stuff in the folder (executables, data, etc.) but just don’t add it to be tracked by git. – J. Herman

.gitignore files are great for this!
Free Cloud Services/Hosts with Git Support

GitHub

GitLab - might be better for hosting private repositories

BitBucket – by Atlassian

And many others
Important: git and GitHub are not the same thing

- **git** is an open-source distributed version control system (http://git-scm.com/)

- **GitHub** is a company that provides hosting for git repositories (https://github.com/)
Ok, so what about GitHub?

How could you share a code repository with many people?

- GitHub and other cloud services provide hosting, bug tracking, and other tools to help you make sense of your revision history.

- Any repositories that you’re willing to open source are free to host on GitHub.

- If you want private repositories, it costs (not much) money, and is free for students and academics with a *.edu email address.

J. Herman (2013)
GitHub – LinkedIn for Code

Jared D Smith
jds485

103 contributions in the last year

Contribution activity
August - September 2019

GitHub
–
LinkedIn for Code
What’s included in a GitHub Repository?

Full revision history

Browse files online

Your Code

Documentation!

Sensitivity Analysis Library (SALib)

Python implementations of commonly used sensitivity analysis methods. Useful in systems modeling to calculate the effects of model inputs or exogenous factors on outputs of interest.

Requirements: NumPy, SciPy

Methods included:

J. Herman (2014)
Detailed Contents of a Code Repository

1. README.md file. The .md is for the Markdown language. It’s simple to use, and there are many tutorials.
2. Your well-commented code
3. Test functions for your code (optional, but I think they should be required upon release)
4. Example data – may be better to host large files elsewhere
5. License file. GitHub and other services have built-in standard licenses
6. .gitignore file. This file lists all of the files (or extensions) to be excluded from git tracking. GitHub has pre-made files for all code languages, but you have to select them.
Getting Code from GitHub

git clone: Gets (clones) code from GitHub to your desktop

Modified from J. Herman (2014)
Committing and Gathering Code Changes using GitHub

`git push/pull`: when you own the repository

J. Herman (2014)
Sourcetree GUI – Visualizing your Commits

All of your repositories

Your Commit History

What changed since last commit
Sourcetree GUI – Visualizing your Commits

Can track all of your repositories in one place

Shows all of your branches, and how many commits ahead branches are from each other

Shows the difference between current changes and previous commits for all code that has changed

Can be used instead of command line
- Sometimes it’s better, especially for merging branches.
  The GUI has nice a side-by-side comparison of conflicts.

Sourcetree Tutorials: Introduction and Advanced
## Legal: Licensing Code

### GPL

GPL is the most widely used free software license and has a strong copyleft requirement. When distributing derived works, the source code of the work must be made available under the same license. There are multiple variants of the GPL, each with different requirements.

### MIT

A permissive license that is short and to the point. It lets people do anything with your code with proper attribution and without warranty.

### Licensing Code

- **GNU GPL v3.0**
  - **Required**
    - Disclose Source
    - License and copyright notice
    - State Changes
  - **Permitted**
    - Commercial Use
    - Distribution
    - Modification
    - Patent Grant
    - Private Use
  - **Forbidden**
    - Hold Liable
    - Sublicensing

- **GNU GPL v2.0**
- **GNU Affero GPL v3.0**

View full GNU GPL v3.0 license »

### MIT License

- **Required**
  - License and copyright notice
- **Permitted**
  - Commercial Use
  - Distribution
  - Modification
  - Private Use
  - Sublicensing
- **Forbidden**
  - Hold Liable

View full MIT License license »

[http://choosealicense.com/licenses/](http://choosealicense.com/licenses/)
Legal: Licensing Code

Default Licenses are available on GitHub, BitBucket, etc.

These licenses may be good as-is, but it may be in your best interest to modify them.

Working with a lawyer may be required for larger projects that become commercialized.

Many repositories have clauses that you must cite them if you use their code in your work. Be careful, and cite everything you use to avoid legal complications.
Git Tutorial #1: Clone Laurence Lin’s RHESSysEastCoast Ecohydrological Model GitHub Repository

1. Create a directory on your computer where you want to keep the repository
2. Navigate to that directory using the command line terminal of your choice (this can also be completed using a GUI of your choice [maybe not on Linux?])
3. For Linux, load the git module
4. Clone the repository into the directory, and change into the code directory
5. List the branches of that repository (use -a to see all branches)
6. Make your own branch in that repository
7. Check out your branch
8. Look at previous commits, and find the commit from Sept 13, 2019
9. Reset the HEAD back to that commit
   This is extremely useful to know how to do (mistakes in coding happen)
10. Check the status of the contents in the repository
11. Do a diff to see what changed
12. Add and commit the changed files to your branch (with a commit message)
13. Check the status again
14. List files in the repository
Tutorial Extras – adding and changing your files

1. Manually add a new .csv file on your branch and save it
2. Manually add a comment to a different file (e.g. README) in your branch and save it
3. Check the status of the contents in the repository
4. Pretend you don’t want to track files with a .csv extension. Let’s `gitignore` them!
5. Check the status again
6. Add the files to be staged for commit (tab complete helps)
7. Commit the files to your branch (with a commit message)
8. Check the status again
9. Undo that commit with a reset of the HEAD
10. (optional) Delete your branch
Git with Sourcetree Tutorial Videos

**Tutorial 1**: Introduction to git Commands Using Sourcetree

**Tutorial 2**: More Advanced git Commands with Sourcetree
- multiple branches
- working with remote repositories