Software Development for Scientists

UCSF Practice of Science Seminars

Peder Larson

February 18, 2020

Key Ingredients for Software Work

1. Languages
   a. Capabilities - Project dependent
   b. Support
2. Computational Resources
   a. Wynton
   b. Dedicated hardware
   c. Cloud
3. Development Tools
   a. Integrated Development Environment (IDE)
   b. Best coding practices
   c. Version Control
4. Sustainability
   a. Documentation
   b. Version Control
5. Sharing
   a. Distributing code
   b. Free cloud hosting - Binder/colab/Azure
   c. Version Control

Languages

One of the first choices to make

Often dictated by type of project

Most Popular

- Python - scientific computing and machine learning
- R - powerful statistics and visualization

UCSF Library resources (Python, R)

Many many others (web or app development, computational efficiency, CPU vs GPU)

Learning to Code

UCSF Library Classes and Resources https://www.library.ucsf.edu/ask-an-expert/data-science/

Software Carpentry https://software-carpentry.org/ “Teaching basic lab skills for research computing” Unix, python, R, git

Sample code from labmates or other researchers

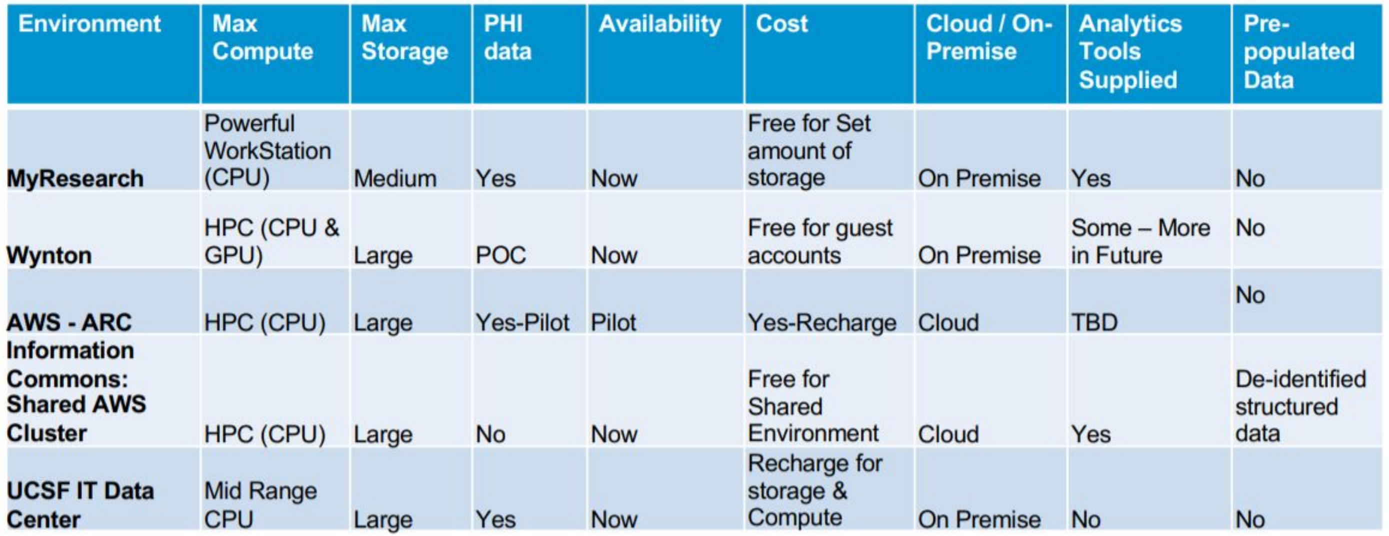
Task-based - best to have a goal of something to program to drive learning

Computational Resources

Where are you going to run the code?

- Personal Laptop - easy, flexible, secure (if managed by UCSF IT) but resources can be limited
- Personal/lab computer - build in more power and functionality, but likely requires more system administration and maintenance
- UCSF On-premise Shared computational resources - even more power and functionality, software and libraries often pre-determined by system administrators
- Cloud
  - AWS, Azure, Google Cloud.
  - AWS will provide free credits for project proposals. For larger work, UC negotiated rates available.

UCSF Compute Resources



Environment	Max Compute	Max Storage	PHI data	Availability	Cost	Cloud / On-Premise	Analytics Tools Supplied	Pre-populated Data
MyResearch	Powerful WorkStation (CPU)	Medium	Yes	Now	Free for Set amount of storage	On Premise	Yes	No
Wynton	HPC (CPU & GPU)	Large	POC	Now	Free for guest accounts	On Premise	Some – More in Future	No
AWS - ARC	HPC (CPU)	Large	Yes-Pilot	Pilot	Yes-Recharge	Cloud	TBD	No
Information Commons: Shared AWS Cluster	HPC (CPU)	Large	No	Now	Free for Shared Environment	Cloud	Yes	De-identified structured data
UCSF IT Data Center	Mid Range CPU	Large	Yes	Now	Recharge for storage & Compute	On Premise	No	No

MyResearch - secure data hosting + multiple software programs (SAS Stata, SPSS, R, Matlab …)

Wynton HPC - high performance computing cluster Free to all UCSF users! https://ucsf-hpc.github.io/wynton/index.html

ARC - “AWS Research Cloud” (pilot)

Computational Resources - Storage

Where are you going to store data?

- Personal Laptop - easy, UCSF provided backup via Code42, but limited
- Personal/lab computer - flexible for building up storage space, requires setup of dedicated backup and potentially encryption
- UCSF Shared computational resources - can request resources (10gb free on MyResearch, 500 GB on Wynton but not backed up)
- Cloud
    - Box - unlimited storage
    - AWS etc

Box tips

Synchronize with Box Sync or Box Drive

Transfer (e.g. backup) large numbers of files via FTP: https://app.box.com/services/box_ftp_server/

Wynton -> Box: https://ucsf-hpc.github.io/wynton/transfers/ucsf-box.html Includes tip for passwordless transfer for automatic processes

Notebooks

Interactive Environment for alternating between coding and data output (e.g. plotting)

Jupyter Notebooks

- Most popular
- Support a variety of programming languages (with proper kernels installed)
- Anaconda - tool for installing jupyter as well as kernels, libraries, and more

Integrated Development Environment (IDE)

What does it do?

- Understands code syntax
- build code
- testing and debugging

Eclipse - popular

Xcode - Mac

Atom or Sublime Text

Python

- PyCharm
- JupyterLab

R

- RStudio

Best Practices/Coding Style

Version Control

Git

Python

[PEP 8 -- Style Guide for Python Code]

Have your IDE tell you when you are conforming to style: https://realpython.com/python-pep8/#tips-and-tricks-to-help-ensure-your-code-follows-pep-8

Basic Style Tips

- Use Descriptive variable and function names
- Modular code (try not to copy/paste! instead write a function to repeat this operation)
- Unit tests - test key parts of your code with simple functions that verify operation or output is correct
- Comment code (although with descriptive names this is less important)
- Requires extra time, but worthwhile in the long-term

What is git?

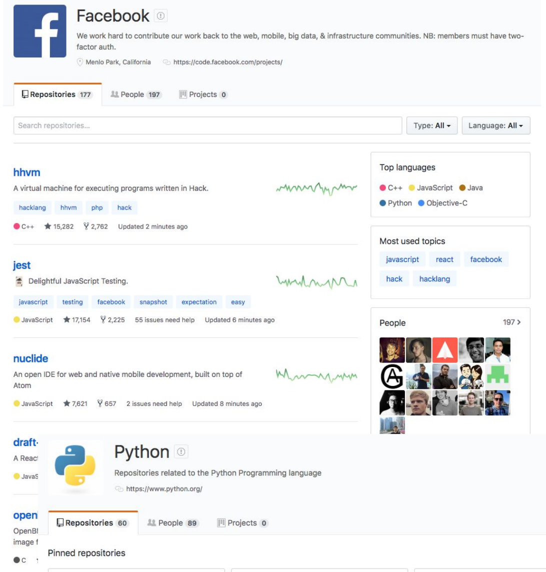
- Version control
  - Track changes you make to software or other documents
  - Go back to old versions, or look at changes
- Shared development platform
  - Designed to support projects with multiple developers
  - Create branches, merge and track individual user changes, report and assign issues
- Distributed
  - Every version of the repository, whether local or hosted (e.g. github, Radiology gitlab) is a full repository, so access to host required
  - Transition to and from local copy to hosted repository, between hosted sites, and between local copies

Why Git?

- Sharing and jointly developing code
- **Standard tool** for version control & software development, easy to work with others
- Open source and free
- Distributed – every clone/copy of the repository is the same/equivalent. E.g. work on local copy is equivalent, no need to be connected to server. Repositories can be easily moved between locations
- Jobs
    - Employers may consider your github/gitlab/bitbucket profile as part of your CV for tech jobs
    - Ask any graduate who works on software in industry – they must use version control
- Many Tools
    - e.g. github desktop client, probably many others
    - Web interfaces

Prominent Git Software Groups

- TensorFlow (Google) https://github.com/tensorflow
- Python https://github.com/python
- Facebook https://github.com/facebook
- LinkedIn https://github.com/linkedin



Facebook

We work hard to contribute our work back to the web, mobile, big data, & infrastructure communities. NB: members must have two-factor auth.

Menlo Park, California https://code.facebook.com/projects/

Repositories 177 People 197 Projects 0

Search repositories... Type: All Language: All

hhvm
A virtual machine for executing programs written in Hack.
hacklang hhvm php hack
● C++ ★ 15,282 ⑂ 2,762 Updated 2 minutes ago

jest
Delightful JavaScript Testing.
javascript testing facebook snapshot expectation easy
● JavaScript ★ 17,154 ⑂ 2,225 55 issues need help Updated 6 minutes ago

nuclide
An open IDE for web and native mobile development, built on top of Atom
● JavaScript ★ 7,621 ⑂ 657 2 issues need help Updated 8 minutes ago

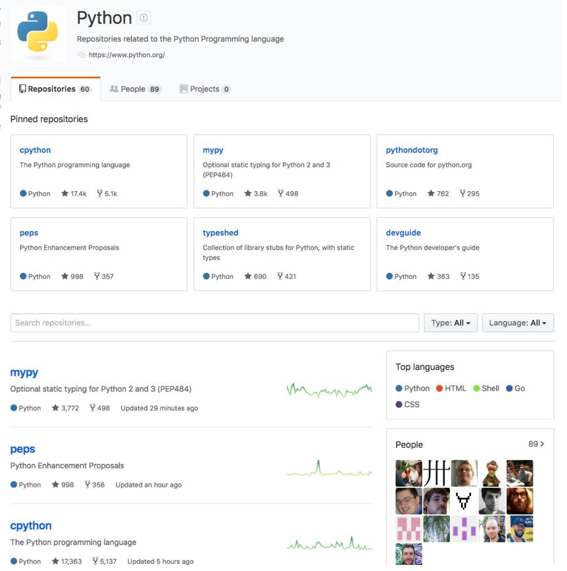
draft·
A Reac
● JavaS

open
OpenBl
image f
● C

Top languages
● C++ ● JavaScript ● Java ● Python ● Objective-C

Most used topics
javascript react facebook hack hacklang

People 197 >



Python

Repositories related to the Python Programming language

https://www.python.org/

Repositories 60 People 89 Projects 0

Pinned repositories

cpython The Python programming language ● Python ★ 17.4k ⑂ 5.1k	**mypy** Optional static typing for Python 2 and 3 (PEP484) ● Python ★ 3.8k ⑂ 498	**pythondotorg** Source code for python.org ● Python ★ 762 ⑂ 295
peps Python Enhancement Proposals ● Python ★ 998 ⑂ 357	**typeshed** Collection of library stubs for Python, with static types ● Python ★ 690 ⑂ 421	**devguide** The Python developer's guide ● Python ★ 363 ⑂ 135

Search repositories... Type: All Language: All

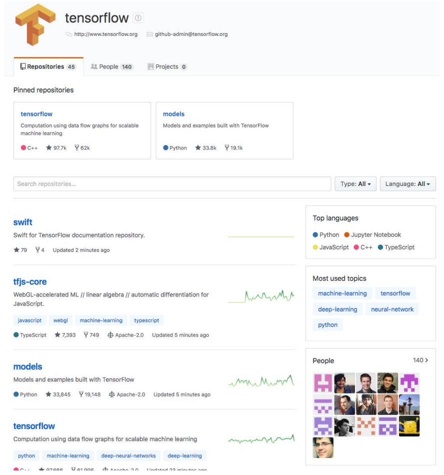
mypy
Optional static typing for Python 2 and 3 (PEP484)
● Python ★ 3,772 ⑂ 498 Updated 29 minutes ago

peps
Python Enhancement Proposals
● Python ★ 998 ⑂ 356 Updated an hour ago

cpython
The Python programming language
● Python ★ 17,363 ⑂ 5,137 Updated 5 hours ago

Top languages
● Python ● HTML ● Shell ● Go ● CSS

People 89 >



tensorflow

http://www.tensorflow.org github-admin@tensorflow.org

Repositories 45 People 140 Projects 0

Pinned repositories

tensorflow Computation using data flow graphs for scalable machine learning ● C++ ★ 97.7k ⑂ 62k	**models** Models and examples built with TensorFlow ● Python ★ 33.8k ⑂ 19.1k

Search repositories... Type: All Language: All

swift
Swift for TensorFlow documentation repository.
★ 79 ⑂ 4 Updated 2 minutes ago

tfjs-core
WebGL-accelerated ML // linear algebra // automatic differentiation for JavaScript.
javascript webgl machine-learning typescript
● TypeScript ★ 7,393 ⑂ 749 Apache-2.0 Updated 5 minutes ago

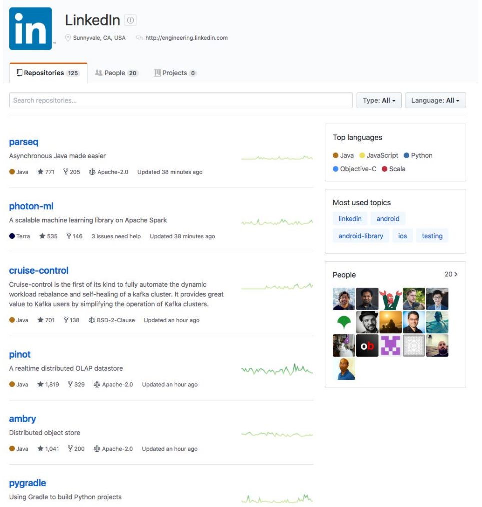
models
Models and examples built with TensorFlow
● Python ★ 33,845 ⑂ 19,148 Apache-2.0 Updated 5 minutes ago

tensorflow
Computation using data flow graphs for scalable machine learning
python machine-learning deep-neural-networks deep-learning
● C++ ★ 97,666 ⑂ 61,996 Apache-2.0 Updated 23 minutes ago

Top languages
● Python ● Jupyter Notebook ● JavaScript ● C++ ● TypeScript

Most used topics
machine-learning tensorflow deep-learning neural-network python

People 140 >



LinkedIn

Sunnyvale, CA, USA http://engineering.linkedin.com

Repositories 125 People 20 Projects 0

Search repositories... Type: All Language: All

parseq
Asynchronous Java made easier
● Java ★ 771 ⑂ 205 Apache-2.0 Updated 38 minutes ago

photon-ml
A scalable machine learning library on Apache Spark
● Terra ★ 535 ⑂ 146 3 issues need help Updated 38 minutes ago

cruise-control
Cruise-control is the first of its kind to fully automate the dynamic workload rebalance and self-healing of a kafka cluster. It provides great value to Kafka users by simplifying the operation of Kafka clusters.
● Java ★ 701 ⑂ 138 BSD-2-Clause Updated an hour ago

pinot
A realtime distributed OLAP datastore
● Java ★ 1,819 ⑂ 329 Apache-2.0 Updated an hour ago

ambry
Distributed object store
● Java ★ 1,041 ⑂ 200 Apache-2.0 Updated an hour ago

pygradle
Using Gradle to build Python projects

Top languages
● Java ● JavaScript ● Python ● Objective-C ● Scala

Most used topics
linkedin android android-library ios testing

People 20 >

Selected Imaging Repositories/Groups

- SIVIC - https://github.com/SIVICLab/sivic
- ANTS – Advanced Normalization Tools https://github.com/ANTsX/ANTs
- AFNI – **A**nalysis of **F**unctional **N**euro**I**mages https://github.com/afni/afni
- ISMRMRD – ISMRM raw data format
- BART – Berkeley advanced reconstruction toolbox https://github.com/mrirecon/bart
- Many more!

Git Terminology

- Repository - a set of code and other digital materials grouped together. Main grouping
- Branch - different versions of the repository, used for adding features, fixing bugs, or specialized versions. Main branch is the ‘master’
- Clone - make a copy of the repository
- Checkout - select a certain branch of the repository
- Commit - add changes into the repository
- Pull - get changes from a remote repository (e.g. GitHub -> your computer)
- Push - put changes into a remote repository (e.g. your computer -> GitHub)
-

Git Servers and Structure

- github.com
  - Most widely used service
  - Public and private repositories (“Repos”)
  - ***Request academic account for free upgrade to Pro account*** – https://education.github.com/
  - Free website hosting with GitHub Pages https://pages.github.com/
  - Good template: https://academicpages.github.io/
- UCSF GitHub Enterprise https://git.ucsf.edu/ - in early testing phase, more coming soon
- GitLab - popular alternative, similar features, including academic accounts with good benefits: https://about.gitlab.com/solutions/education/
- Individual accounts
- Groups – for groups with multiple shared projects, e.g. lab/research group, specific project/grant, organization

My Personal Git Ecosystem

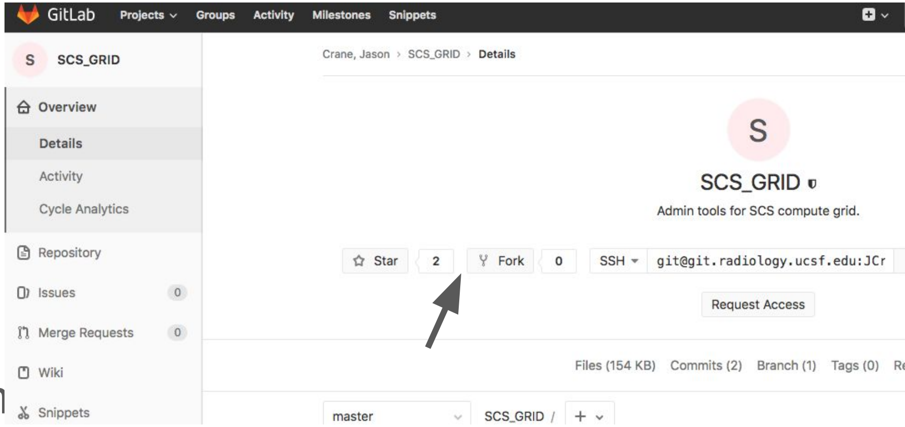
- Radiology git
  - plarson - personal account, for my own projects or initial development
    - matlab
    - EPSI processing
  - EPIC-MRI – group account for GE MRI EPIC programming projects (so far I'm the only user ☺)
    - 3dradial
    - prose_prostate
    - fidcsi_c13
    - 3dute
    - cones

Moving these to private git repos on GitHub.com

- GitHub.com
  - agentmess – personal account, personal projects, papers, playing around with code
  - LarsonLab – group account for shared projects and sustained projects
    - hyperpolarized-mri-toolbox
    - mripy (Python tools for MRI, including neural networks, originally from Peng Cao)
    - Spectral-Spatial-RF-Pulse-Design
    - MRI-education-resources
  - UCSF-EPIC-MRI
    - For sharing EPIC software with others
    - All private repositories (GE proprietary information)

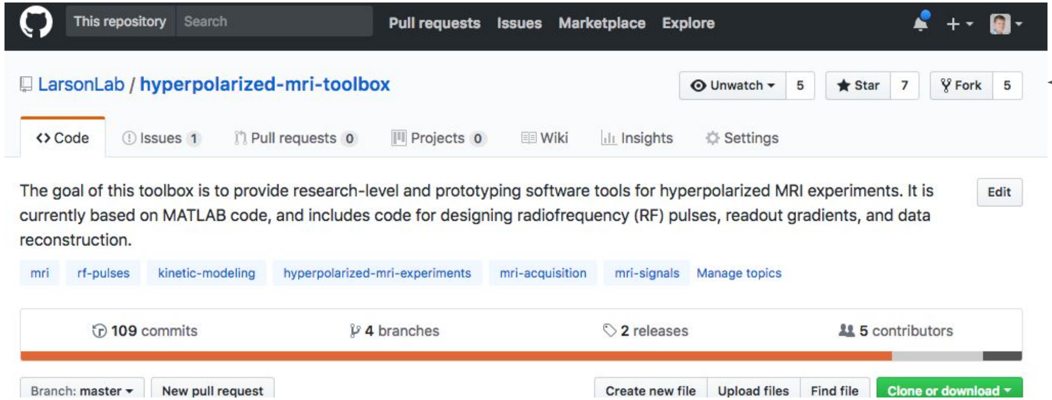
GitHub/GitLab features

- Star – any interesting code
- Watch – be notified of repo changes
- Fork – make your own copy to use and m



Admin tools for SCS compute grid.

git@git.radiology.ucsf.edu:JCr



The goal of this toolbox is to provide research-level and prototyping software tools for hyperpolarized MRI experiments. It is currently based on MATLAB code, and includes code for designing radiofrequency (RF) pulses, readout gradients, and data reconstruction.

Remember to …

- Commit every day!
- Practice, and don’t worry about mistakes since there’s a history of all your changes

Initialization

1. Login – git.radiology.ucsf.edu or github.com
2. Create New Repository/Project
3. Clone ‘git clone <address>’
   - Copy address from web
   - Clone to multiple places (laptop, SCS network) - I love this workflow to move between coding on my laptop and moving to shared computational resources
4. Add files or import in existing directory
5. Check file status ‘git status’, should show Untracked Files
6. Add these files to git repository with ‘git add’
7. Check file status ‘git status’, should show Changes to Be Committed
8. Commit ’git commit –m “<commit message>” ‘
9. Push changes to remote repository (e.g. github, radiology git) ‘git push’
10. Check web!

Daily Workflow

1. Pull changes from remote repo (in case others have added edits) ‘git pull’
    - If you are just a user (not developer) of repo/project, then this is all you need
2. Modify files
3. Check status ‘git status’
4. Add changes ‘git add’
5. Confirm status ‘git status’
6. Commit ‘git commit’
7. Push changes ‘git push’ (don’t need to do for every commit, but at least every day is best)

- Quick commit – ‘git commit –a –m <message>” stages all changes to be commited and then commits

Advanced Workflows - ignoring files

- .gitignore – this is a file within your git repository that can choose to ignore certain files. For example, large data files, temporary files, executables.
  - .gitignore templates at https://github.com/github/gitignore for templates for many programming languages
  - Copy into main directory
  - In GitHub Desktop app, right-click to add files to .gitignore
- Want to store large files?
  - Use “git-lfs” (large file storage) git-lfs.github.com

Advanced Workflows - Branching

- “Branch”
  - separate version of repository to work on
  - Create a branch to fix a bug, add a new feature, or play around without disrupting the “master” branch (default branch when you start a project
  - Easy to explore and visualize via web interfaces
  - First branch created is the “master”

Advanced Workflows – Creating and Editing New Branch

Via Web interface

Then switch to branch in repository

1. List all branches ‘git branch –a’
2. Checkout new branch ‘git checkout <branchname>’
3. Confirm you are now working on new branch ‘git branch’

Or Command Line

1. Check current branch ‘git branch -a’ (lists branches, * indicated curent branch_
2. Switch to starting branch if needed, e.g. ‘git checkout master’ to create branch from master
3. Create new branch ‘git branch <branchname>’
4. Checkout new branch ‘git checkout <branchname>’
5. Confirm you are now working on new branch ‘git branch’

Advanced Workflows – Creating and Editing New Branch

- Normally, a repository is a single directory and you can switch between branches (‘git branch’ to view branches, ‘git checkout <branchname>’ to switch)
- This can cause problems if you (like me) forget to check what branch you are working on
- Alternative
  - Keep one repository as a ‘master’
    - git clone git@git.radiology.ucsf.edu:PLarson/git-tutorial-test.git git-tutorial-test_master
  - Clone another version of repo for development(more like SVN)
    - git clone git@git.radiology.ucsf.edu:PLarson/git-tutorial-test.git git-tutorial-test_branch
    - cd git-tutorial-test_branch
    - git checkout <branchname>

Advanced Workflows – Merging Branches

- Create a Merge/pull request
  - When you want to put branches together, merge the changes
  - E.g. you have added feature in a feature branch, merge back into the master branch
  - I find this easiest via web interfaces
- When you push changes ‘git push’, message:
  - remote: To create a merge request for branch2, visit:
  - remote: https://git.radiology.ucsf.edu/PLarson/git-tutorial-test/merge_requests/new?merge_request%5Bsource_branch%5D=branch2
- Review and submit merge request
- Confirm request and submit (last steps are so you can review the changes carefully before continuing)

Fixing conflicts

Can arise when remote repo is out of sync with local copy, or during merging of branches

1. Find conflicting files ‘git status’
2. Edit with your favorite editor
    - Conflicting lines marked with <<<<, ====, >>>>
    - Choose appropriate changes, remove lines with <<<<, ====, >>>>
3. Mark resolution with ‘git add’
4. Commit
5. Push

Advanced Workflows - Other

- Multiple Remote Repos (e.g. github vs radiology git)
  - Can sync local copy with both
  - Move repository to other location

1. Create empty repository
2. Add as a remote ‘git remote add github https://github.com/agentmess/git-tutorial-test.git’ (can change “github” to be description of another remote repository, use ”origin” if you want to make this the new default repository)
3. Push to new remote ‘git push --all github’

Advanced Workflows - Other

- Tags/releases
  - When you’ve got a stable product
  - Allows others to easily find stable version or version that will work for them
- Issues
  - Keep track of bugs to fix or features to add

Remember to …

- Commit every day!
- Practice, and don’t worry about mistakes since there’s a history of all your changes

- git on it
- git your roll on
- git ‘er done
- everybody git together

Sharing Code and Data

GitHub or GitLab - share repositories (public, or private shared to specific users), build up your user profile (good for CV), enables community development

License - Recommended MIT + Creative Commons License is the reproducible research standard

Zenodo - get digital object identifier (DOI)

Available for citing code (easy integration with GitHub) https://zenodo.org/account/settings/github/

Available for citing datasets too, example: https://zenodo.org/record/3647820#.XkxtFmhKguU

Run your code online: Google Colab, Binder, and more

Live Notebooks

Google Colab: free, python-based, good GPU resources, files and code loaded through Google Drive

Demo: https://colab.research.google.com/github/mikgroup/extreme_mri/blob/master/colab-demo.ipynb

Binder: free, (Project Jupyter), flexible languages, resources also from Google Cloud, can link to GitHub repository as well as other sources

Demo:https://github.com/LarsonLab/hyperpolarized-mri-toolbox https://github.com/LarsonLab/MRI-education-resources

Distill journal https://distill.pub/

JupyterBook example https://qmrlab.org/t1_book/intro.html

Final Thoughts

- Invest in software skills such as version control, good coding style (e.g. modular code), good IDE, utilize available software & hardware resources
- UCSF compute & storage resources are evolving rapidly, pay attention to take advantage of latest offerings
- Share your code, from your own labmates to across the world
  - Take the time to help others use your code, this makes your work more impactful
  - Take the time to work with other's code, this makes all of our work more sustainable
  - Contribute to open-source initiatives
- Live and hosted notebooks are a powerful tool to share your work, enable others to use your code, and support reproducible research
- Did I mention Version Control?