Programming and Data Science for Biology (PDSB)

Session 2
Spring 2018

Syllabus updates

Office hours are now listed:

Professor Eaton: Fridays 10-12am (Scherm. ext. 1007)

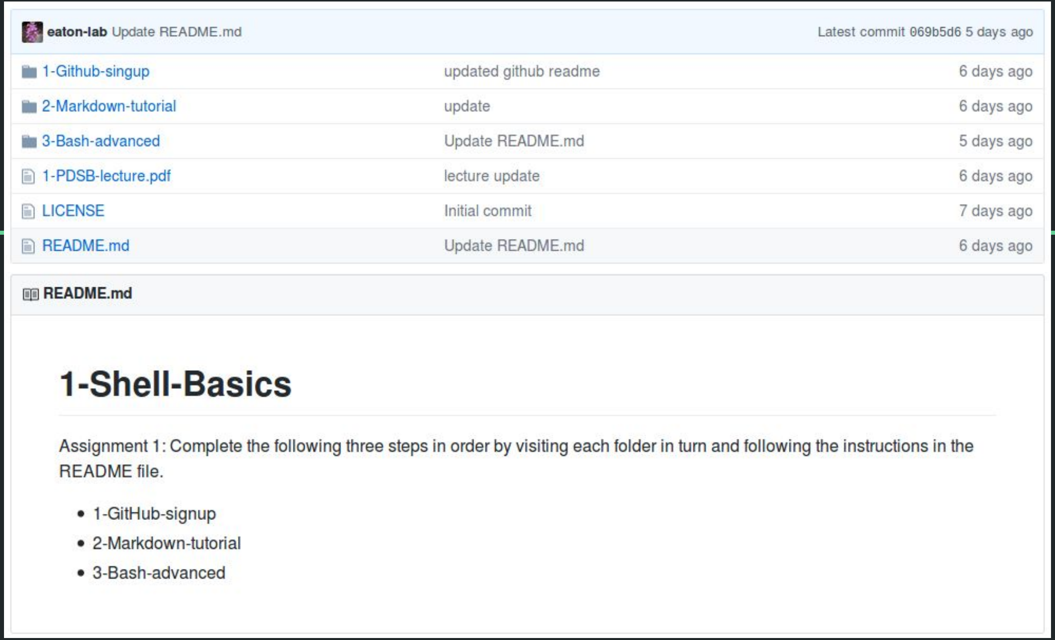
Patrick McKenzie: Tuesdays 2-4pm (Scherm. ext. …)

Reading the syllabus. Look at this week & next week.

git, github, and open source code

You should have completed the session 1 assignment

Link to session 1 repo



eaton-lab Update README.md — Latest commit 069b5d6 5 days ago

1-Github-singup	updated github readme	6 days ago
2-Markdown-tutorial	update	6 days ago
3-Bash-advanced	Update README.md	5 days ago
1-PDSB-lecture.pdf	lecture update	6 days ago
LICENSE	Initial commit	7 days ago
README.md	Update README.md	6 days ago

README.md

1-Shell-Basics

Assignment 1: Complete the following three steps in order by visiting each folder in turn and following the instructions in the README file.

- 1-GitHub-signup
- 2-Markdown-tutorial
- 3-Bash-advanced

How was your first coding assignment?

Just like with spoken language, there are many ways to accomplish a given task. That’s why we’ll use *code reviews* to examine each others code to learn different approaches to the problems that we tackle.

First, though, we’ll learn some efficient tools for sharing code with each other -- but before that, we’ll learn to install tools.

bash pitfalls

What hurdles did you encounter, and how did you get past them?

Getting stuck within a command.

Vagaries of different *bash* terminals?

If you used `sed -i` on OSX it is different from sed on Linux/gitbash. No problem if using sed without -i.

```
## how to escape an unclosed parentheses
## or unfinished command.
$ for i in
 >
 > <control-c to escape>

## in-place sed subst. on OSX vs GNU sed
$ sed -i ‘s/Iris/iris/g’ iris-data...
$ sed -i ‘’ ‘s/Iris/iris/g’ iris-data...
```

Where is your software?

Software can be installed *system-wide*, or *locally* in a *user* directory. We'll install locally whenever possible for the following advantages

-- Easy to update/remove
-- No permissions necessary to install
-- Same installation method can be used on your own computer as on computing clusters.

Git-bash users: for now some commands will be slightly different, we'll overcome this in a bit, e.g., `which.exe` and `where.exe`

```
## the path searched for executables
$ echo $PATH

## which one is in the front of $PATH
$ which grep

## find all versions in PATH
$ which -a git

## locate binary & source
$ whereis git          # [unix, osx]
$ where.exe git        # [git-bash]
```

Get conda

Conda is a tool for installing software locally. It is a package manager for installing Python packages, but can do much more than that. In fact, you can install Python, R, Julia, and many more languages in it as well.

Conda can easily install software into a local directory (e.g., in your $HOME) which does not require special permissions and can be easily removed. Go to this link to download conda

```
## use exe to install conda [git-bash]

## install from terminal [Linux, MacOSX]
$ curl
https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh -o
miniconda.sh

## install conda into $HOME [linux, osx]
$ bash miniconda.sh -b -p $HOME
```

Personalize your bash terminal

Whenever you open a bash terminal it first loads a file with your preferences from your home dir. We can add to that file to set `aliases` and to change the `$PATH`.

If we put miniconda/ at the front of $PATH then it will always find the version of a software in here **first**, and use it before the system version.

The file for your bash preferences is called ~/.bash_profile or on linux ~/.bashrc.

```
## use the `nano` text editor to edit
$ nano ~/.bash_profile


## [in ~/.bash_profile aliases]
alias ll='ls -l'


## [git-bash users only]
alias conda=
'/c/Users/deren/Miniconda3/Scripts/conda.exe'


## [in ~/.bash_profile linux/gitbash/osx]
PATH='/home/deren/miniconda3/':$PATH
PATH='/c/Users/deren/Miniconda3/':$PATH
PATH='/Users/deren/miniconda3/':$PATH
```

What did conda do?

Because conda is a Python program it just installed Python3 as well as a few basic programs that *conda* needs to be able to run. Let's look at what this looks like.

We need to run `source` to load changes made in .bash_profile.

To have go back to our system software and completely forget about the local conda software all we would need to do is remove the miniconda path from $PATH

```
## is conda in my $PATH
$ echo $PATH

## do source. And now?
$ source ~/.bash_profile
$ echo $PATH

## how many Pythons are in PATH?
$ which -a python

## which is the default/first?
$ which python
```

Install something with conda

Because conda is a Python program it just installed Python3 as well as a few basic programs that *conda* needs to be able to run. Let's look at what this looks like.

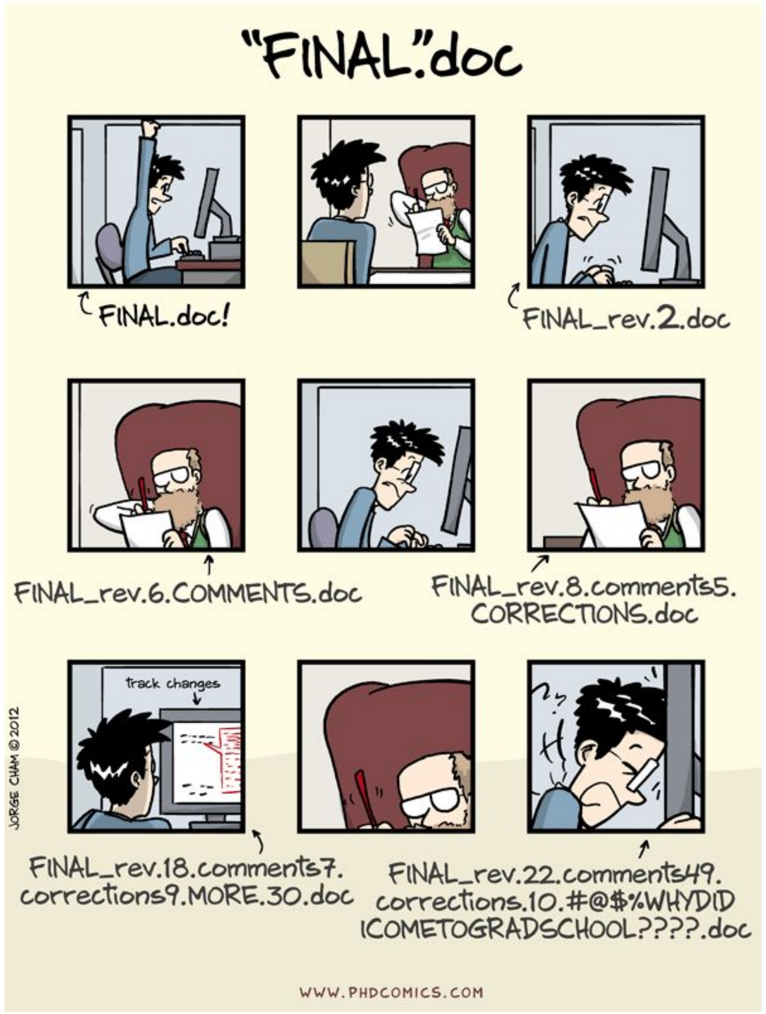
We need to run `source` to load changes made in .bash_profile.

```
## get info about conda distribution
$ conda info

## install something from main repo
$ conda install ipython

## install from a channel
$ conda install raxml -c bioconda

## install jupyter notebooks
$ conda install notebook
```

OK, more Python in a bit, but first, more on git.

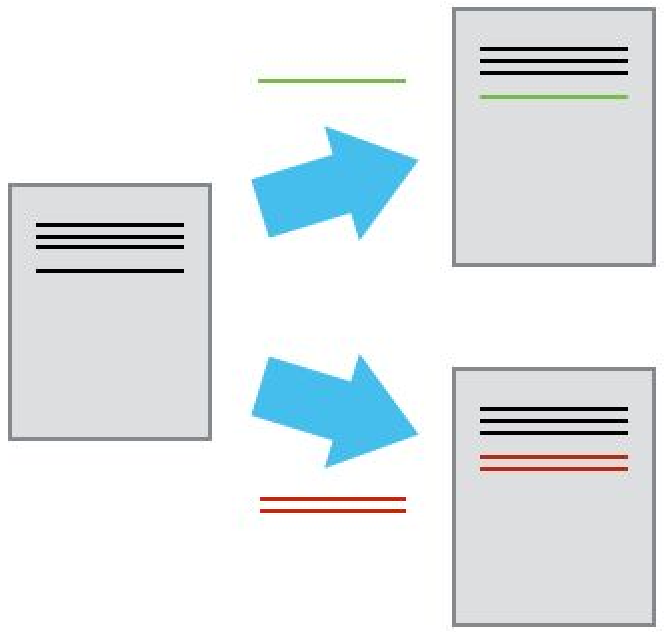
Let’s make sure we all have *git* installed.

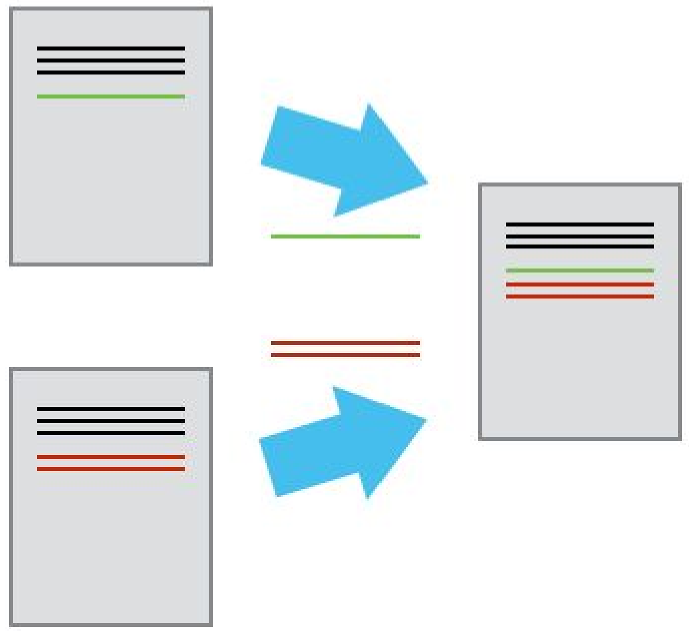
```
## linux and git-bash have git. OSX no.
$ git --version

## if you don’t have it on OSX it will
## ask you to install
```


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Track versions and changes w/ git

Revert to older versions at any time.





swcarpentry.github.io

You should all have a GitHub account now.

Ensure you are logged into your account, we'll be using github in our activities in a few moments.

```
## open a tab to your repo profile
https://github.com/{your-user-name}

## open a tab to our class organization
https://github.com/programming-for-bio
```

1. git

The program *git* tracks versions of a document (software) as it changes, and allows multiple users to edit simultaneously and resolve conflicts.

A git repository is simply a normal directory, but contains a hidden subdirectory .git/ where info is stored.

```
## Let’s make a git repository
$ cd ~/PDSB/
$ mkdir gitrepo/
$ cd gitrepo
$ git init
Initialized empty Git repository in /...

## In here you now have a hidden dir
$ ls -a
.  ..  .git
```

1. git

When we call the *git* commands it is important to be aware of where you are in the file system, since git will execute on the .git/ subdirectory that is in your current directory.

```
## where are we?
$ pwd
```

1. Connecting to the world

remotes are locations where a copy of the repository is stored. There should always be a copy called origin, which is the main copy you have permissions to write to on GitHub.

```
## am I connected to a remote?
$ git remote show

## create a remote repo on GitHub
## add the remote (github .git address)
$ git remote add origin
https://github.com/{username}/{repo}.git

## make master branch from origin/master
$ git pull origin master

## tell master to track remote origin
$ git branch -u origin/master master
```

1. Connecting to the world

You can now push edits to this repository using ‘add’, ‘commit’ and ‘push’.

```
## edit a file
$ echo “New Text” > README.md

## add and commit changes, push to origin
$ git add README.md
$ git commit -m ‘edited the README’
$ git push
```

1. Cloning to start

The easiest way to start a GitHub repository is with cloning. This will ensure there is a remote version of origin/master on GitHub and clone a copy of master locally.

Let’s remove the last test repo and create a new one using cloning.

```
## clone from .git address on GitHub
$ git clone
https://github.com/{username}/{repo}.git

## our remote is now already configured
$ git remote -v

## add and commit changes, push to origin
$ git add README.md
$ git commit -m ‘edited the README’
$ git push
```

2. staging files

The easiest way to think of *git* is that it is like a combination of a camera and a time machine, which allows you to travel back to any point in time at which you took a **snapshot**.

Before we take a snapshot we always need to carefully *stage* the photo. The command **add** can be thought of as organizing the objects on a *stage*.

```
## let’s make some files in our dir/
$ echo “hello world” > hello.txt
$ echo “goodbye world” > goodbye.txt

## ‘add’ both files to git
$ git add hello.txt goodbye.txt

## check the status (2 staged files)
$ git status
...
    new file:   goodbye.txt
    new file:   hello.txt
```

3. commit changes

When we call the *git* commands it is important to be aware of where you are in the file system, since git will execute on the .git/ subdirectory that is in your current directory.

```
## let’s make some files in our dir/
$ pwd
```

4. Forking

Forking is not actually a *git* concept, but rather it is a GitHub concept, allowing arbitrary users to access each other’s code.

```
## Forking is not done from the terminal
## you need to go to the GitHub repo you
## wish to fork and fork a copy of the
## repo to your GitHub Profile.
```

4. Forking

In this example, I fork the repository for today's assignments. I then add my Markdown assignment to the repo and push the changes.

```
## Clone YOUR copy of the repo
$ git clone
https://github.com/pdsb-test-student/2-gi
t-and-more.git

## Make changes and push to origin
$ cd 2-git-and-more/
$ cp assignment.md assignment/
$ git add assignment.md
$ git commit -m "added my assignment"
$ git push
```

5. Pull request

However, the changes I made so far are only in **my** forked copy of the repository, and I want them to be in the main copy (upstream).

For this, I make a pull request on GitHub

```
## Just like forking, pull requests are
## something that is done on GitHub, not
## from a terminal.
```

6. Collaborative and open science tools

jupyter, rstudio, git, and github together provide tools for hosting, sharing, and collaboratively editing code.

```
## Install jupyter
$ conda install jupyter
```

6. Collaborative and open science tools

jupyter, rstudio, git, and github together provide tools for hosting, sharing, and collaboratively editing code.

```
## Install jupyter
$ conda install jupyter
```

Assignments and readings for next time

Assignment: Link to Session 2 repo
Readings: See syllabus
Collaborate: Work together in this gitter chatroom