# A Very Cursory Introduction to Github and Friends

An AGSO 15-minute lecture

Meredith L. Rawls • Aug 29, 2014

*Some slides are from SciCoder, courtesy of Demitri Muna*

# What You Want When Writing Code

- Backups
- Ability to see a previous version of your code
- Marking code that works / is stable
- Marking code that ran a particular analysis
- Access to your code from anywhere
- Synchronize changes to code across multiple computers.
- Share your code with people.

Most people try to accomplish some of these things by hand, but often forget to do (or just skip!) one more steps because it isn't easy or is time consuming.

(And really, you want most of these things for *all* your files!)

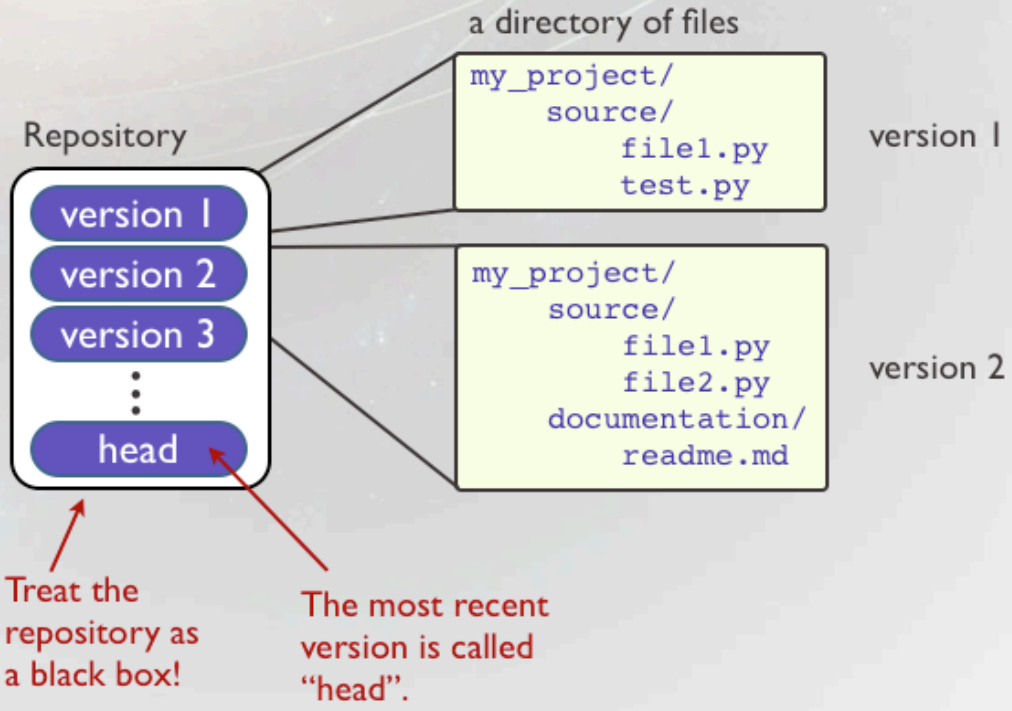
# Git vs. Github

- **Git** is a free *version control* program you can use from the command line
  - http://git-scm.com/
  - Learn the basic commands with this tutorial http://try.github.com/

- **Github** is a website that uses git and lets you save, share, and backup projects online
  - https://github.com/
  - It has a friendly GUI for Mac (Windows, too) https://mac.github.com/

# The Repository

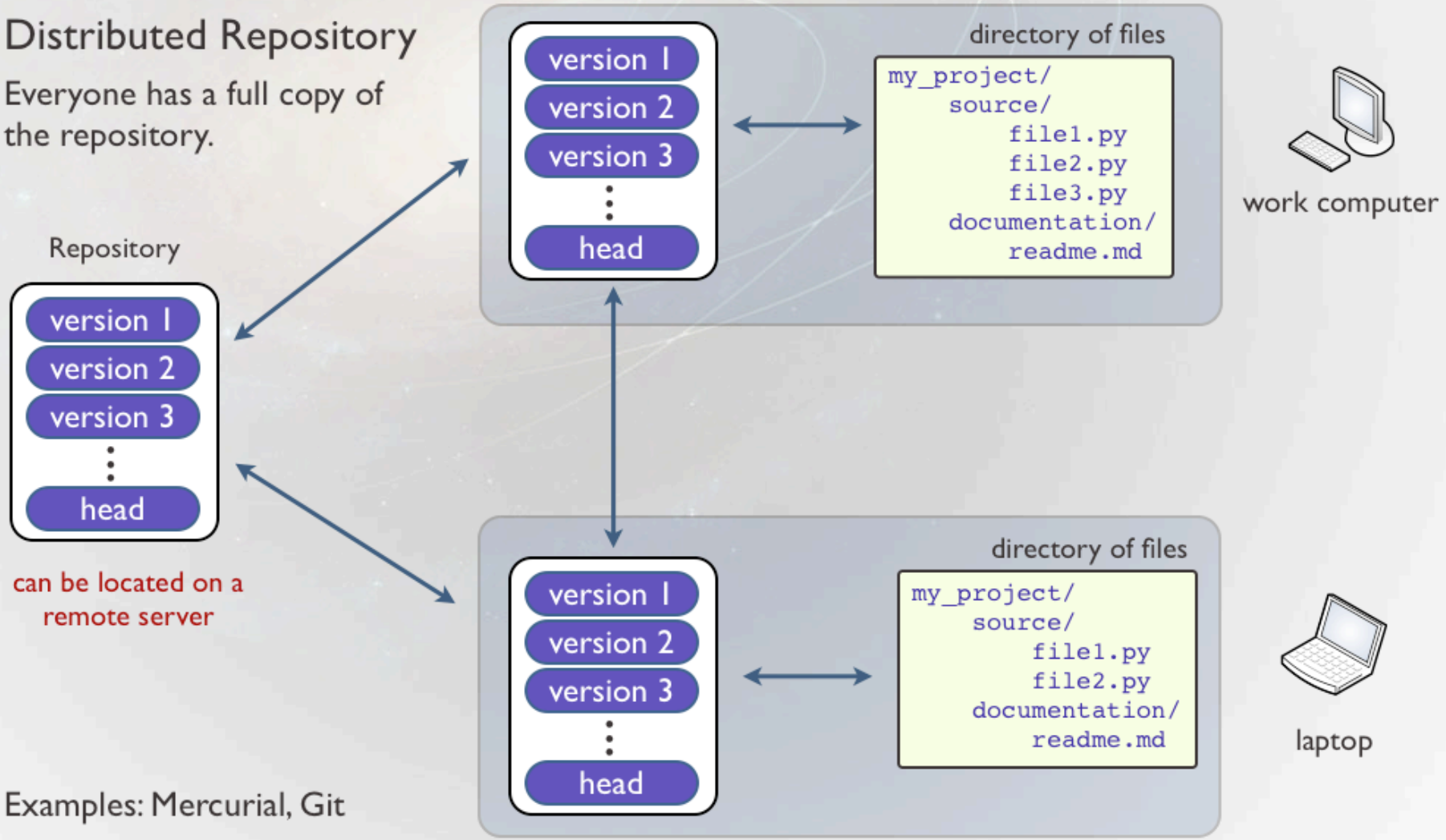
A place where all versions of all files are stored. With this, the current version or any prior version of any file in the repository can be recovered. Can be locally or remotely located. If only a local copy (i.e. on your own hard drive) is created, it doesn't provide a backup in case of computer failure.



# Kinds of Repositories

## Distributed Repository

Everyone has a full copy of the repository.




Examples: Mercurial, Git

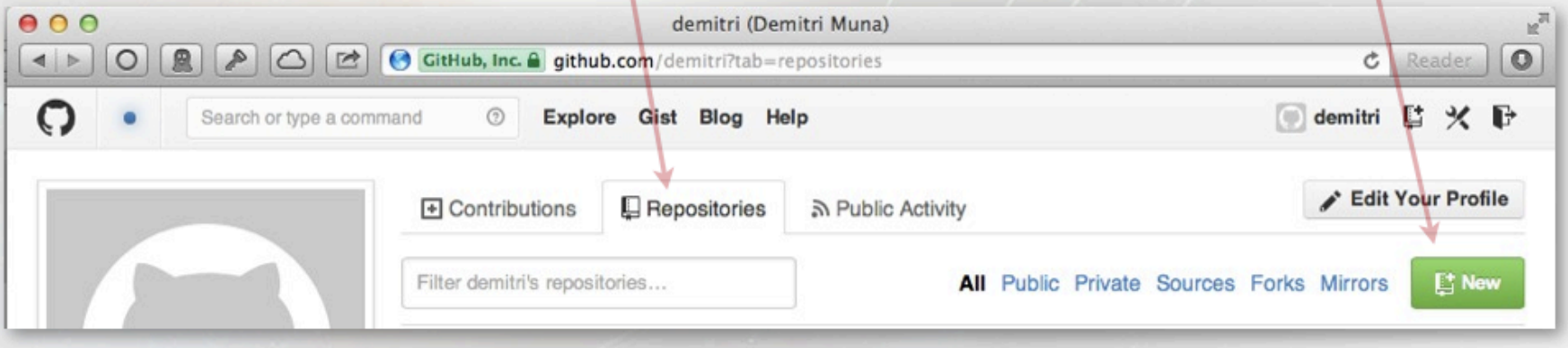
# Learn to speak git

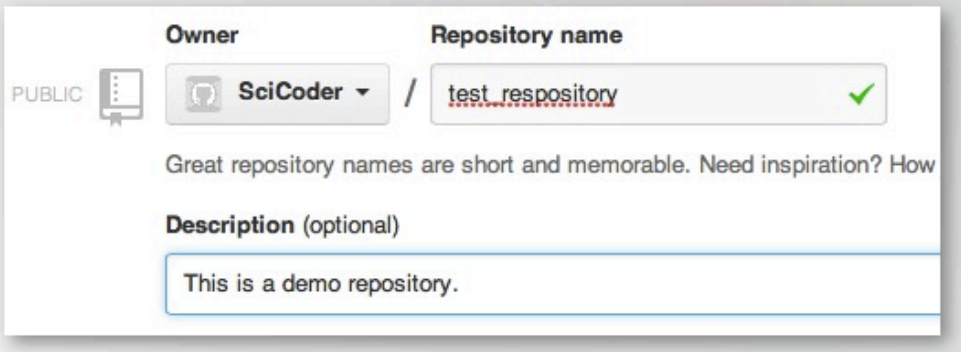
- **Repository (repo)**: like a folder for your project
  - **Remote**: version of a repo that lives on github
  - **Clone**: a local copy of your github repo
  - **Branch**: a separate parallel version of a repo
  - **Fork**: a personal copy of someone else's repo

# Creating a New Repository on GitHub

Go to your account on GitHub, select "Repositories".

Create new repository.



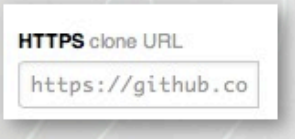




Fill in a name & description.


Initialize repository, select primary language to use (more on "ignores" laters).

# Cloning the Repository

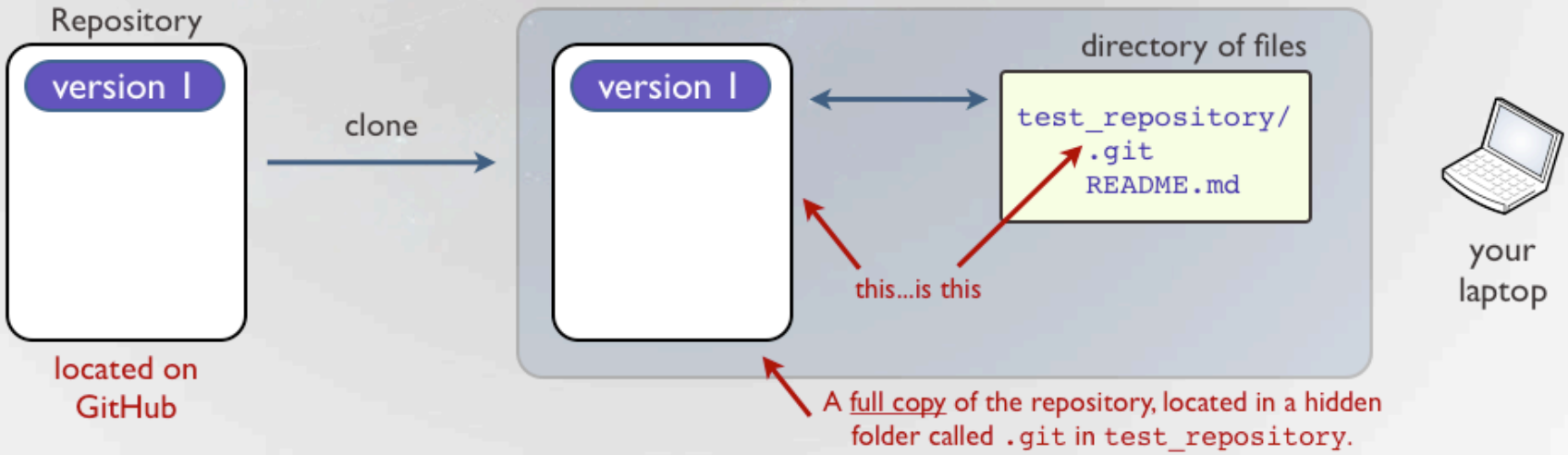
On the lower right on the next page, copy the "clone URL".




In the terminal, enter: `git clone <URL>`

```
blue-meanie [~/Documents/Repositories/tmp] % git clone https://github.com/SciCoder/test_respository.git
Cloning into 'test_respository'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
blue-meanie [~/Documents/Repositories/tmp] %
```

What does this do? You've made a local copy of the repository. Now there are two copies of the repository and one copy of the files (your "working directory").



# Committing Files

You need to be roughly aware staging happens. I do not recommend you use this feature. When you area ready to add files, place them into the repository IMMEDIATELY with the `commit` command.

```
blue-meanie [test_respository] % git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   newfile.txt
#
blue-meanie [test_respository] % git commit -m "First commit."
[master 9b8c29c] First commit.
 0 files changed
 create mode 100644 newfile.txt
blue-meanie [test_respository] % git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
nothing to commit (working directory clean)
blue-meanie [test_respository] %
```

i.e. in staging area limbo

list of files that will be committed

A description of the changes *must* be specified with every commit, easiest with the "-m" flag.

Yeah, that's helpful. We'll get to this.

This means that the latest version in the repository matches the files in the working directory.

NOTE: Committing files only saves them to your local repository.

# Learn to speak git

- **Repository (repo)**: like a folder for your project
  - **Remote**: version of a repo that lives on github
  - **Clone**: a local copy of your github repo
  - **Branch**: a separate parallel version of a repo
  - **Fork**: a personal copy of someone else's repo
- **Commit**: a change to a file with a note attached

# Saving To Another Repository

What did this line mean...?

name we call the remote repo

now we know this is the name of the main branch

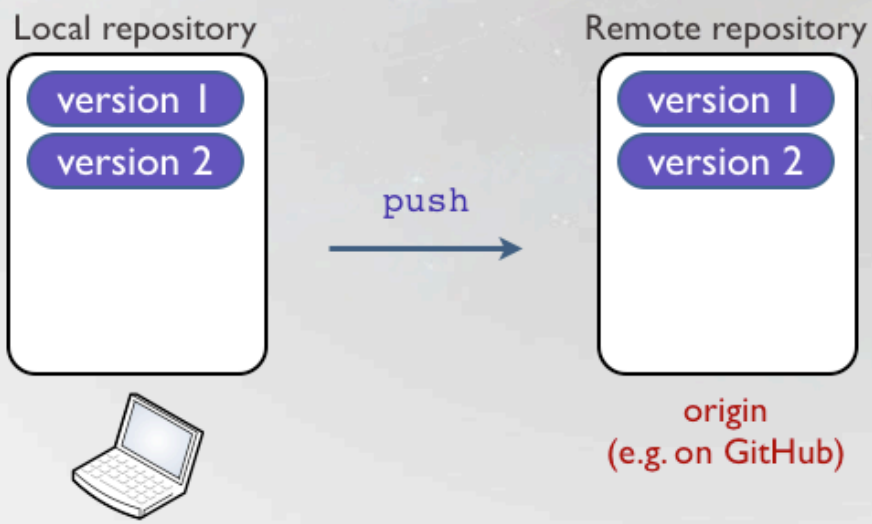
```
# Your branch is ahead of 'origin/master' by 1 commit.
```

The local repository (your computer) has a newer update than what is on the remote server (which we call "`origin`"). The line above means that one commit occurred after the last commit on the master branch on the remote repository `origin`.

We want to send those changes to the remote repository... this is called a *push*:

```
git push <remote repo name> <branch name>
```

note default remote name is "origin" and default branch is "master", defined in .git/config

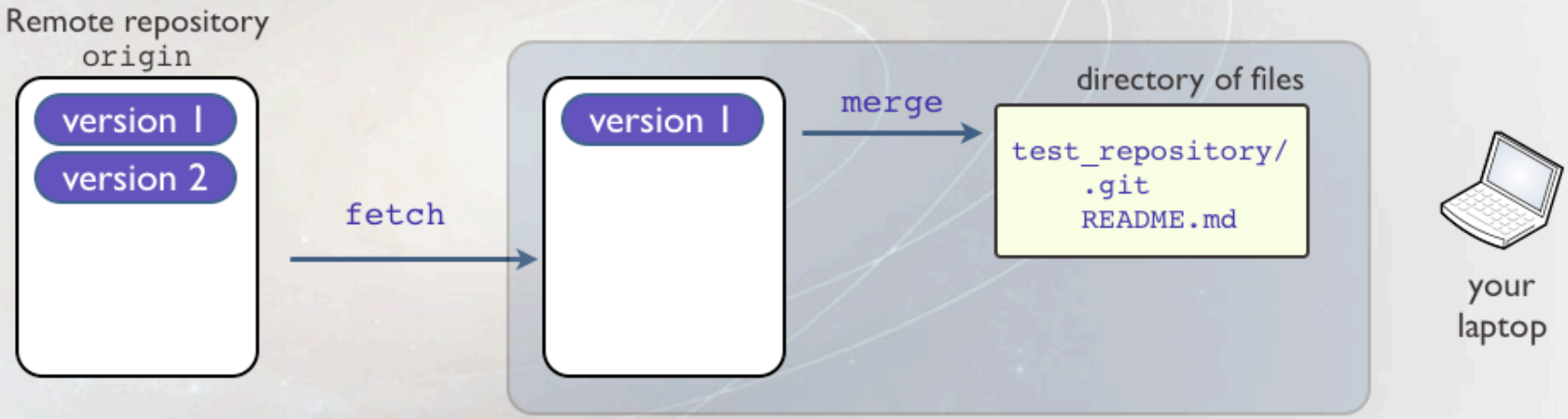



In our example from before, this would be:

```
git push origin master
```

```
blue-meanie [test_respository] % git push
Username for 'https://github.com': demitri
Password for 'https://demitri@github.com':
Counting objects: 9, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 720 bytes, done.
Total 8 (delta 2), reused 0 (delta 0)
To https://github.com/SciCoder/test_respository.git
   e5aa6ac..2377659  master -> master
```

# Getting Changes from the Remote Repository




This command will retrieve changes from the remote repository to your local repository:

```
git fetch
```

This does not update the files in your working directory. To do that, follow the `fetch` with:

```
git merge
```

Or, if you actually have better things to do with your time, use this command:

```
git pull
```

```
git pull
```

=

```
git fetch
git merge
```

# Learn to speak git

- **Repository (repo)**: like a folder for your project
  - **Remote**: version of a repo that lives on github
  - **Clone**: a local copy of your github repo
  - **Branch**: a separate parallel version of a repo
  - **Fork**: a personal copy of someone else's repo
- **Commit**: a change to a file with a note attached
- **Fetch**: retrieve latest changes from a github repo
- **Merge**: take changes from one branch and apply them to another
- **Pull**: Fetch + Merge
- **Push**: send your committed changes to a github repo

This repository Search Explore Gist Blog Help mrawls +

# mrawls / kepler-makelc

Unwatch 1 Star 0 Fork 0

Basic light curve processing for all your Kepler EB needs. — Edit

16 commits 1 branch 0 releases 1 contributor

branch: master kepler-makelc / +

updated plotter and pretty LC plot!

mrawls authored on Jul 17 latest commit 384ab3a377

| | | |
|---|---|---|
| .gitignore | exclude big text outfiles from repo | a month ago |
| ELClcprep.py | Added long-term lc detrending | a month ago |
| README.md | Minor README update | 2 months ago |
| example2_9246715.png | Added long-term lc detrending | a month ago |
| example2_lcchunks.png | Added long-term lc detrending | a month ago |
| example_9246715.png | New program: ELClcprep.py | 2 months ago |
| example_lcchunks.png | New program: ELClcprep.py | 2 months ago |
| lc_finalplot1.png | updated plotter and pretty LC plot! | a month ago |
| lc_functions.py | Added long-term lc detrending | a month ago |
| lc_functions.pyc | I put .pyc in .gitignore but here this is anyway. | a month ago |
| lcplotter.py | updated plotter and pretty LC | |
| makelc.py | Added long-term lc detrending | |
| makelc_out.txt | Added long-term lc detrending | |
| mask_kepcotrend.txt | Kepler light curve detrender fo | |

README.md

Code
Issues 0
Pull Requests 0
Wiki
Pulse
Graphs
Settings

HTTPS clone URL
https://github.com/r
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP

```
[~/Astronomy/github/kepler-makelc]$ ls
ELC_lc0.txt             ELC_lcall.txt          lc_functions.pyc
ELC_lc1.txt             ELClcprep.py           lcplotter.py
ELC_lc2.txt             README.md              login.cl
ELC_lc3.txt             example2_9246715.png   makelc.py
ELC_lc4.txt             example2_lcchunks.png  makelc_out.txt
ELC_lc5.txt             example_9246715.png    mask_kepcotrend.txt
ELC_lc6.txt             example_lcchunks.png   pyraf
ELC_lc7.txt             lc_finalplot1.png      uparm
ELC_lc8.txt             lc_functions.py
[~/Astronomy/github/kepler-makelc]$
```

# Pro Tips

- You don't have to put **data** in your repository
- Use your .edu email for free **private** repositories
  - https://education.github.com/discount_requests/new
- *Nobody's code is perfect.*
  - **Share anyway**.
- Learn to **fork** others' code for your own use
  - Projects like **astropy** and **sunpy** live on github!
- A recommended, more powerful GUI: SourceTree
  - http://www.sourcetreeapp.com

# Write papers collaboratively: Authorea

| | |
|---|---|
| **It is web-native.** | We think that writing a scientific article should be as easy as writing a blog post. Authorea lets you write on the web, for the web. Every document you create becomes a beautiful webpage which you can choose to share with your coauthors and the public. |
| **Many formats.** | Authorea lets you write in LaTeX, Markdown, HTML, Javascript, and more. And you can work with your coauthors on the same document, using different formats. |
| **Collaboration.** | More and more often, we write together. A recent paper coauthored on Authorea by a CERN collaboration counts over 200 authors (!). If Authorea can solve large-scale collaboration for CERN, it can probably solve it for you too. |
| **Version control.** | Authorea uses Git, a robust versioning control system to keep track of document changes. Every change that you make to a document by you or your collaborators is recorded and can be undone at any time. |
| **Open science.** | Did you ever wish you could share with your readers the data behind a figure? Check. Authorea documents can take IPython notebooks, d3.js scripts, and Plot.ly images to make your articles shine with beautiful data-driven interactive visualizations. |

- https://authorea.com

# Authorea features

- **Free!** Like Google Docs for writing papers
- LaTeX and BibTeX-friendly
- Automatic backups with behind-the-scenes git
- No more `\begin{figure}[h!!!!!dammit!]`
- No more emailing PDFs back and forth
- Developed by astronomers who “get it”
- And when you’re done… export in the format for your journal of choice (!)

Great example: https://authorea.com/users/23/articles/249/_show_article