

# Lecture 2: Data in Linguistics, Git/GitHub, Jupyter Notebook

LING 1340/2340: Data Science for Linguists

Na-Rae Han

# Objectives

---

- ▶ What do linguistic data look like?
- ▶ Tools:
  - ◆ Git and GitHub
  - ◆ Jupyter Notebook
  - ◆ DataCamp tutorials

**You should be  
taking NOTES!**



# First thing to do every class

---

1. Open up a Terminal/Git Bash window ("shell" window).

2. Move into your Data\_Science directory.

```
cd Documents/Data_Science
```

Hit TAB for auto-completion.

3. Make sure you are in the right directory.

```
pwd
```

"Print Working Directory"

4. Look at what's inside the directory.

```
ls
```

or

```
ls -la
```

ls for "list directory".  
-la for "long/all". Shows all hidden files in long output.

MINGW64:/c/Users/narae/Documents/Data\_Science

```
narae@T450s MINGW64 ~
$ cd Documents/Data_Science/

narae@T450s MINGW64 ~/Documents/Data_Science
$ pwd
/c/Users/narae/Documents/Data_Science

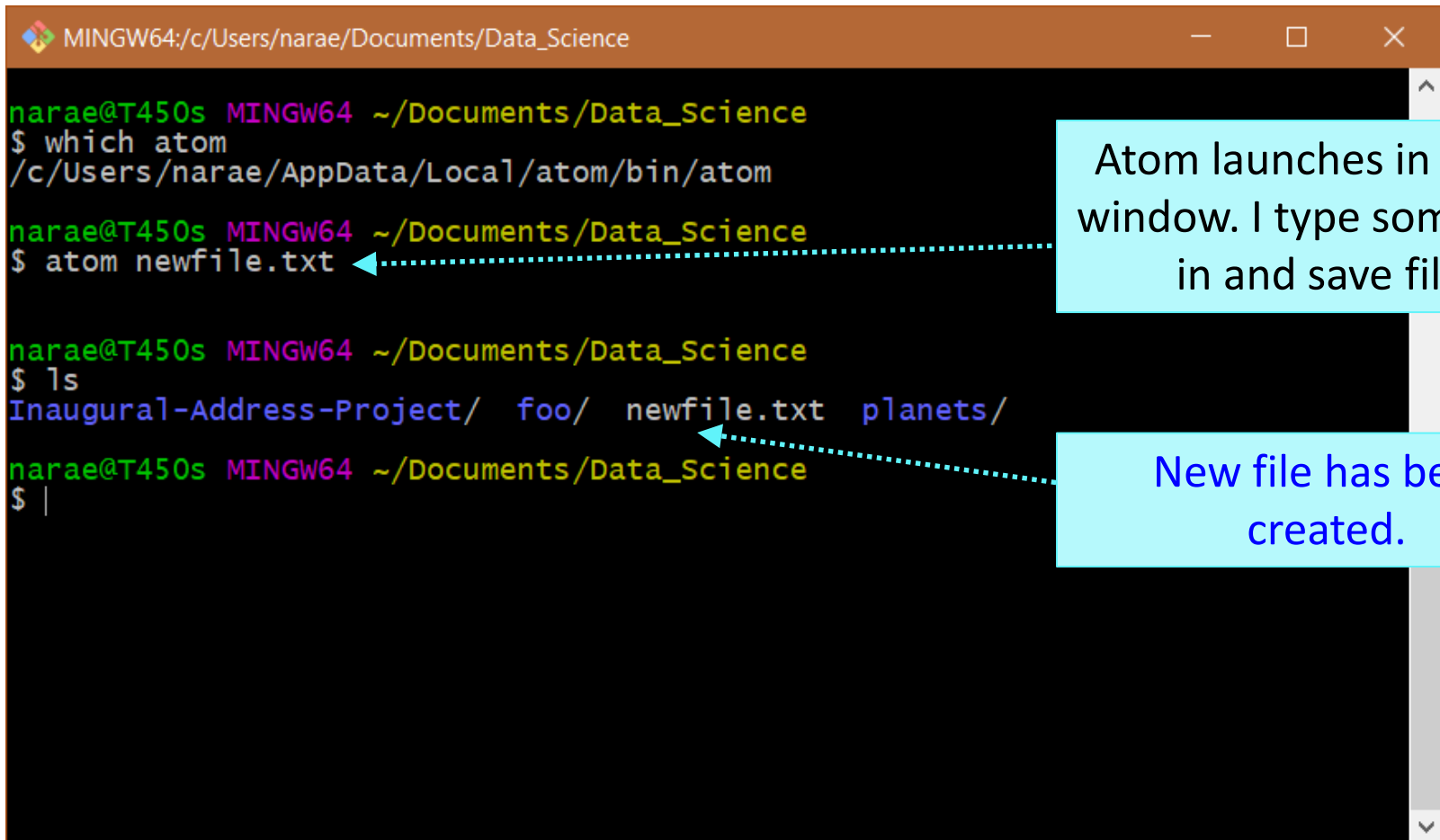
narae@T450s MINGW64 ~/Documents/Data_Science
$ ls
Inaugural-Address-Project/  foo/  planets/

narae@T450s MINGW64 ~/Documents/Data_Science
$ ls -la
total 12
drwxr-xr-x 1 narae 197121 0 Aug 30 22:43 ./
drwxr-xr-x 1 narae 197121 0 Aug 28 16:29 ../
drwxr-xr-x 1 narae 197121 0 Aug 28 16:32 Inaugural-Address-Project/
drwxr-xr-x 1 narae 197121 0 Aug 28 23:19 foo/
drwxr-xr-x 1 narae 197121 0 Aug 29 17:11 planets/

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

# Your text editor in shell

- ▶ You should be able to launch your text editor from shell and create a new text file in the directory.



```
MINGW64:/c/Users/narae/Documents/Data_Science
narae@T450s MINGW64 ~/Documents/Data_Science
$ which atom
/c/Users/narae/AppData/Local/atom/bin/atom
narae@T450s MINGW64 ~/Documents/Data_Science
$ atom newfile.txt
narae@T450s MINGW64 ~/Documents/Data_Science
$ ls
Inaugural-Address-Project/ foo/ newfile.txt planets/
narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

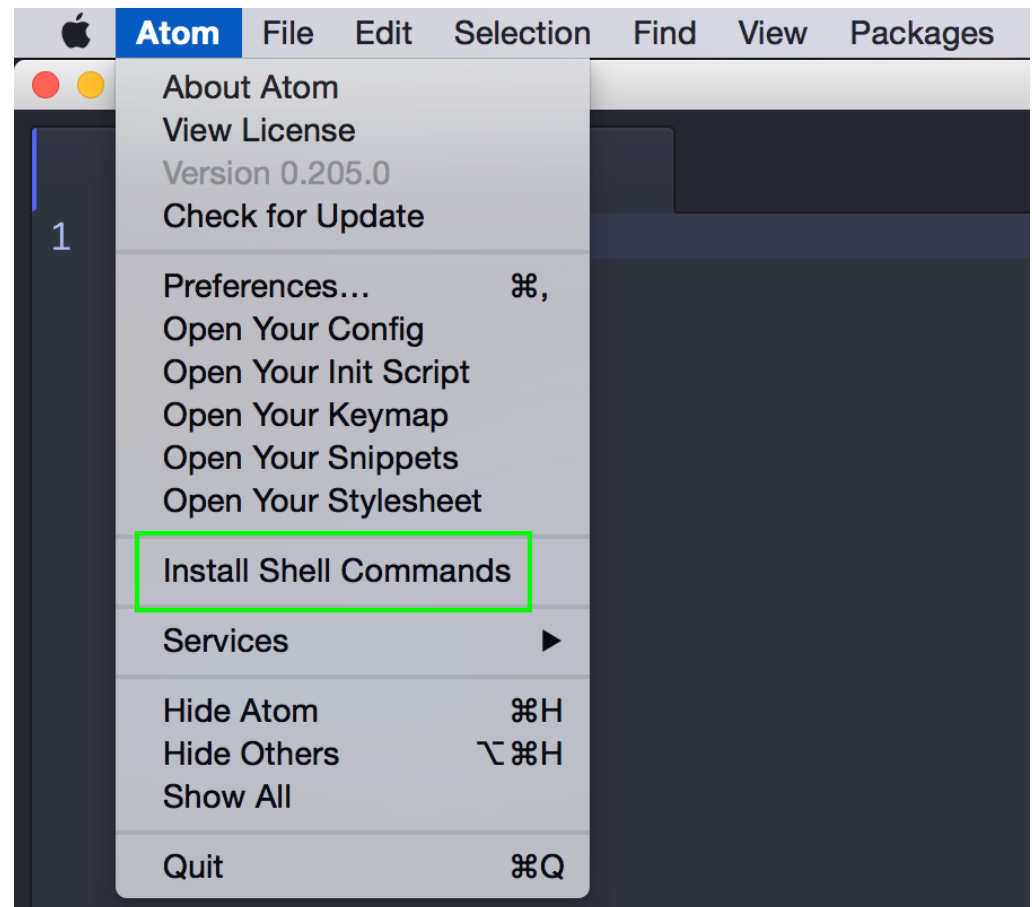
Atom launches in a new window. I type some stuff in and save file.

New file has been created.

# Mac users: configure Atom for shell

- ▶ <https://stackoverflow.com/questions/22390709/how-to-open-atom-editor-from-command-line-in-os-x>

- ▶ "Install Shell Commands"
- ▶ After this, you can launch atom directly from your Terminal (bash shell).



# Git



## ▶ What is Git?

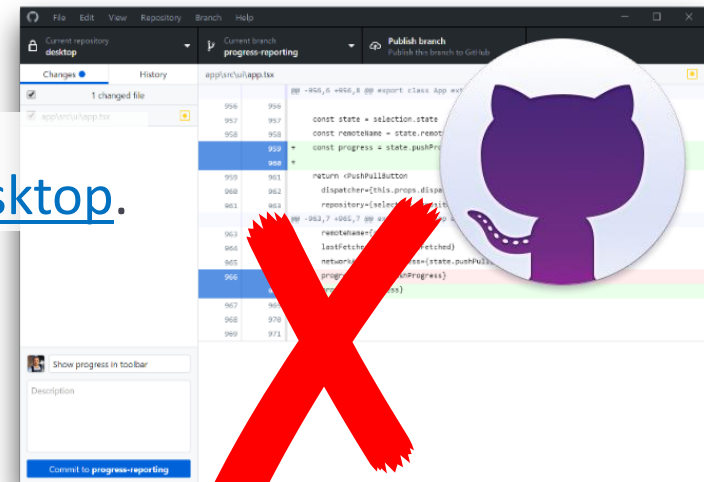
- ◆ One of the most popular *version control systems* in the coding community.

## ▶ Software Carpentry's tutorial:

- ◆ <http://swcarpentry.github.io/git-novice/>

## ▶ In this class, we will exclusively use the **command line** interface of git.

- ◆ Ignore the GUI clients.
- ◆ Likewise, do NOT install/use [GitHub Desktop](#).





# Configuring your Git

---



SW Carpentry's [2. Setting Up Git](#)

- ▶ Mac users: open up a [Terminal](#). 
- ▶ Windows users: open up a [Git Bash](#) terminal. 
- ▶ Display current configuration:
  - ◆ `git config --list`
- ▶ We will configure:
  - ◆ Your name
  - ◆ Your email (use **Pitt email!**)
  - ◆ Your editor (anything other than vim!)



# Your first local repository

---

► Follow steps in SW Carpentry's [3. Creating a Repository](#):

1. Create a directory called **planets**

2. Initiate it as a git repository

```
git init
```

3. Create a new text file 'mars.txt'

4. Add files to be committed

```
git add mars.txt
```

5. Commit file

```
git commit -m 'first commit'
```

6. Edit the text file

7. Add files to be committed

```
git add mars.txt
```

8. Commit file

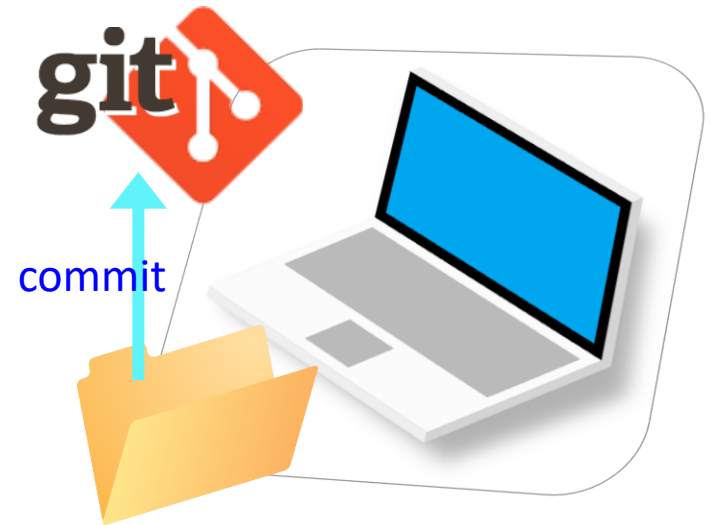
```
git commit -m 'changed x, y, z'
```

Check status  
between steps:  
`git status`

# Your first local repository

---

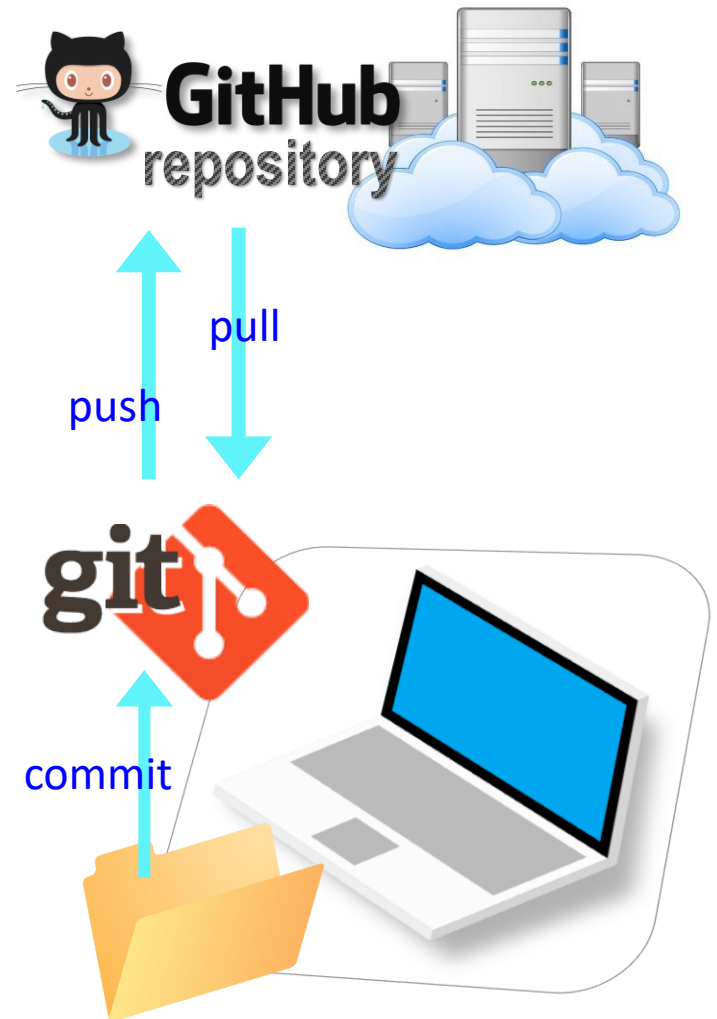
- ▶ Your directory `planets` was set up with a git repository.
- ▶ `planets` is now:
  - ◆ tracked by git
  - ◆ all changes will be documented
  - ◆ able to revert back to earlier version, if needs be
- ▶ But is this all?
  - ◆ How about backup? collaboration? social?



# GitHub: a *remote* repository

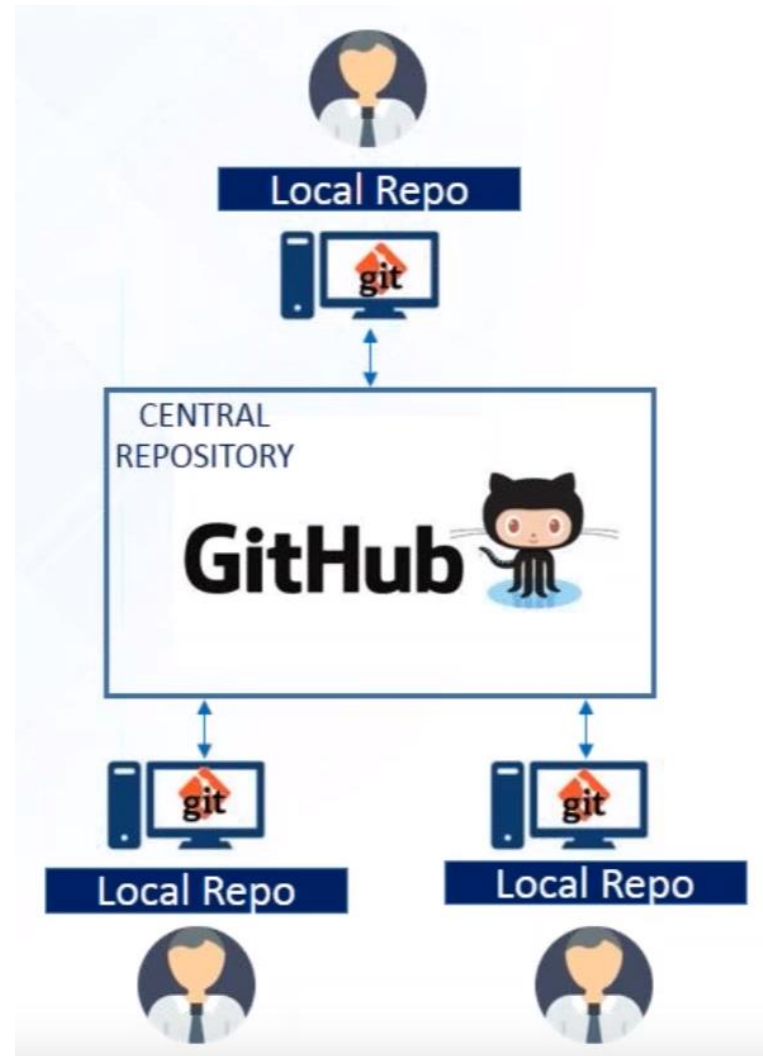
---

- ▶ This is where GitHub comes in.
- ▶ GitHub is a **repository hosting service**.
  - ← A website where you can keep a copy of your Git repository.
  - ← **REMOTE** repository on GitHub, **LOCAL** repository on your laptop.



# GitHub: a *social*, remote repository

- ▶ GitHub also works as a **central remote repository** among a group of **collaborators** working on a shared project.
  - ◆ Everyone works on their own local copy of the repository, making changes.
  - ◆ git is able to keep track and merge changes submitted by everyone.



# Creating a GitHub repo

---

▶ There are TWO main methods of initiating a GitHub repo.

**Method 1:** You have an **existing LOCAL git repo** on your laptop, and you link that up ("push") to GitHub's remote repo.

- ◆ SW Carpentry's [7. Remotes in GitHub](#) goes this route.
- ◆ We could push our planets git repo to a GitHub repo this way.

**Method 2:** You start from scratch. Create a whole **new repository on GitHub**, and then **clone it onto your laptop** as a brand-new local repository.


- ◆ [This YouTube tutorial](#) shows you how.

← Let's try this.

# Your first GitHub repo


---


- ▶ On GitHub, create a new repository called "practice-repo".
  - ◆ Provide a short description.
  - ◆ Keep it public.
  - ◆ Initialize it with a README.

Owner:  narae-student / Repository name: practice-repo ✓

Great repository names are short and memorable. Need inspiration? How about [urban-adventure](#).

Description (optional)  
Will be using this repository for Git/GitHub practicing.

 **Public**  
Anyone can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

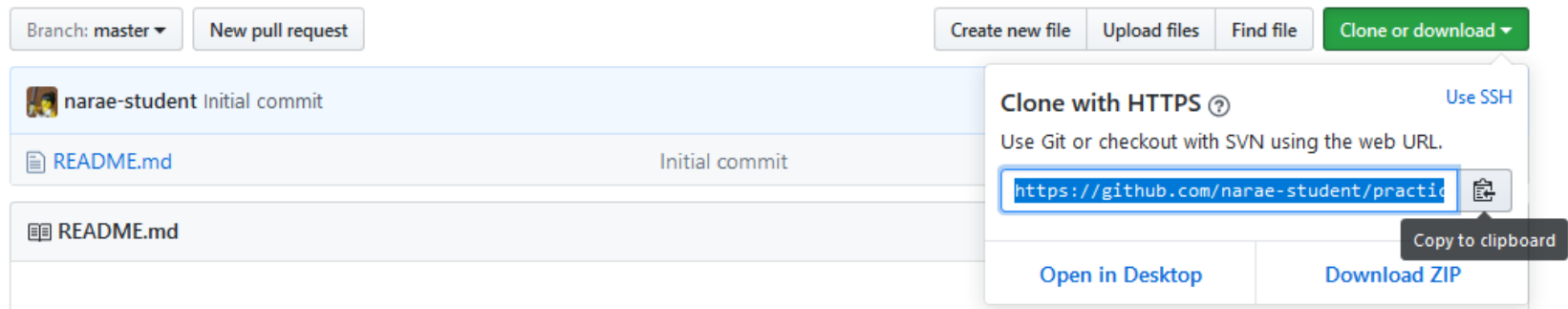
Add .gitignore: **None** | Add a license: **None** ⓘ

**Create repository**

# Cloning first GitHub repo

---

- ▶ GitHub shows a URL to use in cloning. Copy to clipboard.



- ▶ In Terminal/Git Bash, move into your Data\_Science directory (`cd Documents/Data_Science`), then execute:  
`git clone https://github.com/yourid/practice-repo.git`  
← practice-repo directory is cloned as a local repository.

# Local repository ↔ remote repository

---

► After committing, we now need to *push* to remote repo.

1. Create a new text file 'notes.txt'

2. Add files to be committed

```
git add notes.txt
```

3. Commit file

```
git commit -m 'first commit'
```

4. **Push change to GitHub: git push**

5. Edit the text file

6. Add files to be committed

```
git add notes.txt
```

7. Commit file

```
git commit -m 'changed x, y, z'
```

8. **Push change to GitHub: git push**

Check status  
between steps:  
`git status`

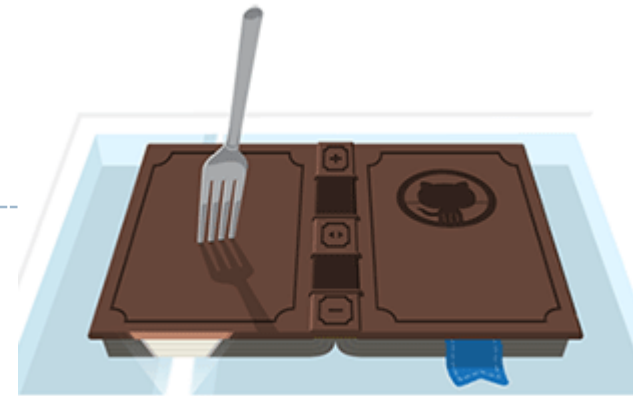
Push change to  
GitHub:  
`git push`



# Forking

---

- ▶ When you start with someone else's project.
- ▶ <https://help.github.com/articles/fork-a-repo/>
- ▶ You fork the original repo into your own account, creating your own "fork".
- ▶ You make changes in your own fork. Original repo is not affected.
- ▶ **pull request**: When you think the original project could benefit from your contribution, you ask the owner to "pull" from your fork.
- ▶ Owner of original ("upstream") will review your contribution, and then either merge it or reject it.



# Your first fork

---

- ▶ Go to narae's GitHub profile.
- ▶ Fork "Class-Practice-Repo". You will now have the exact same content in your own account.
- ▶ Clone your fork onto your local machine, via [git clone](#).
- ▶ Copy over your To-Do1 submission file into `todo1/` directory. Make sure the file name has your name in it.
- ▶ Commit, and then push to your fork.
- ▶ Confirm your GitHub fork now has your submission file.
- ▶ Create a merge request for Na-Rae.

# Up-close with linguistic data

---

Activity  
30 minutes



1. Everyone download this zipped archive:

[http://www.pitt.edu/~naraehan/ling1340/real\\_linguistics\\_data.zip](http://www.pitt.edu/~naraehan/ling1340/real_linguistics_data.zip)

2. There are two data sets; one requires downloading from source. Explore and discuss:

Discussion points → next slide

3. Team lead is in charge of submitting the info above.

- ◆ Inside activity1/ folder, created and edit [report\\_teamcolor.txt](#)
- ◆ Push change to your own fork, and then issue a pull request.
- ◆ Na-Rae will then merge the changes into the upstream (original) repo.

4. Finally, everyone sync their own local fork.

# Up-close with linguistic data

---

Activity  
30 minutes



## ► Discussion points:

- ◆ What kind of data set is this? Content? Format? Purpose?
- ◆ How was the data created, collected, and processed?
- ◆ How did the researcher organize the data?
- ◆ At present, what can you *do* with this data?
- ◆ After learning data-science methods, what do you think you will be able to do with the data?

# Wrapping up

---

- ▶ Homework #1 is out: due on Tuesday.
- ▶ Office hours
  - ◆ Mon/Wed 2:30 -- 4pm. G17 CL
  - ◆ Also by appointment
- ▶ Make sure you joined DataCamp group.
- ▶ I'll also be sending GitHub Organization invitation.
- ▶ Start learning:
  - ◆ Git, GitHub
  - ◆ Jupyter Notebook
  - ◆ numpy