Unix 2

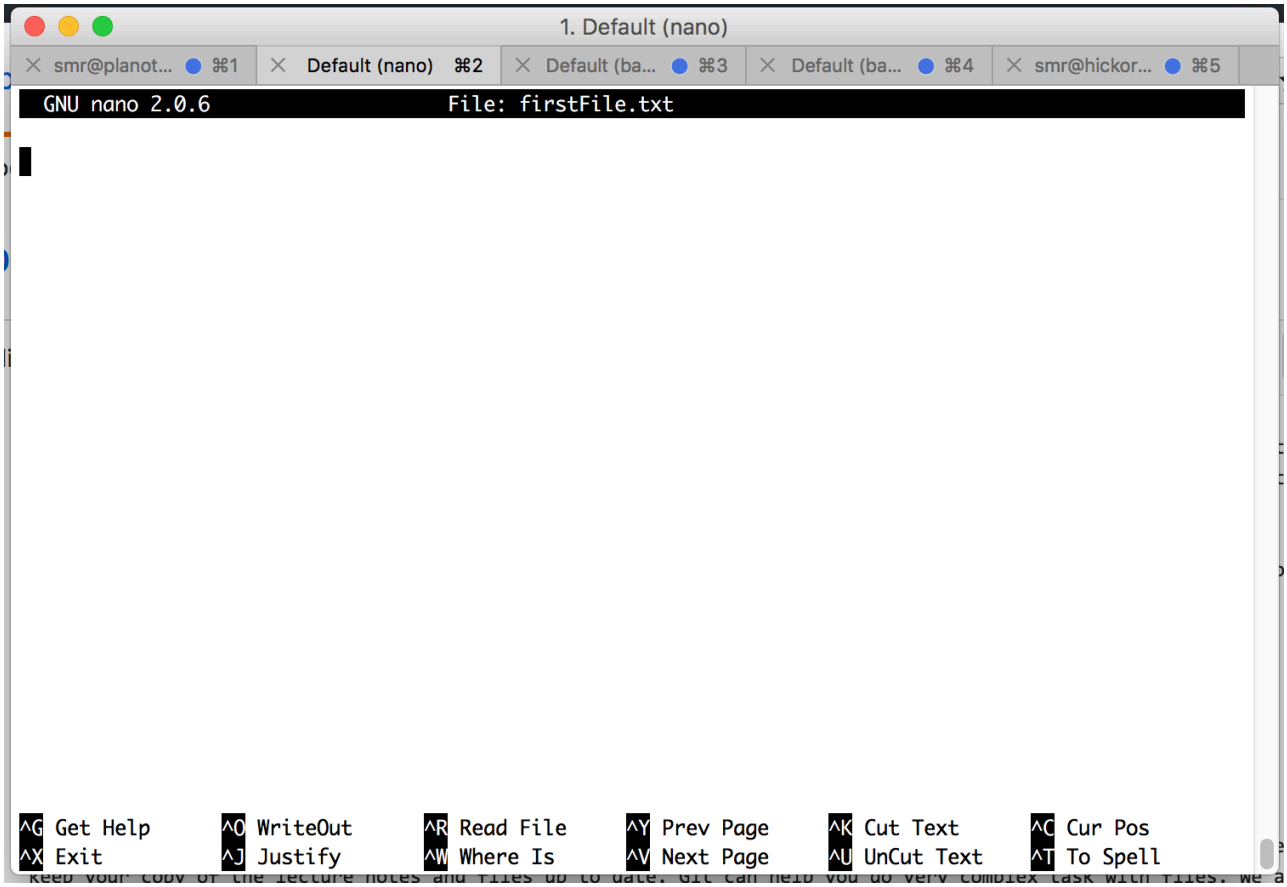
Text Editors

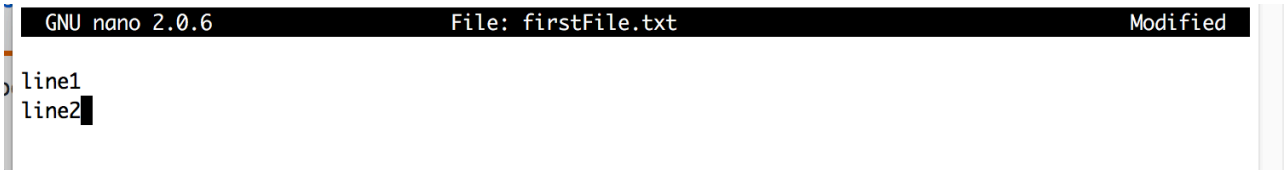
It is often necessary to create and write to a file while using the terminal. This makes it essential to use a terminal text editor. There are many text editors out there. Some of our favorite are Emacs and vim. We are going to start you out with a simple text editor called `nano`

The way you use nano to create a file is simply by typing the command *nano* followed by the name of the file you wish to create.

```
1 (~) 71% nano firstFile.txt
```

This is what you will see:





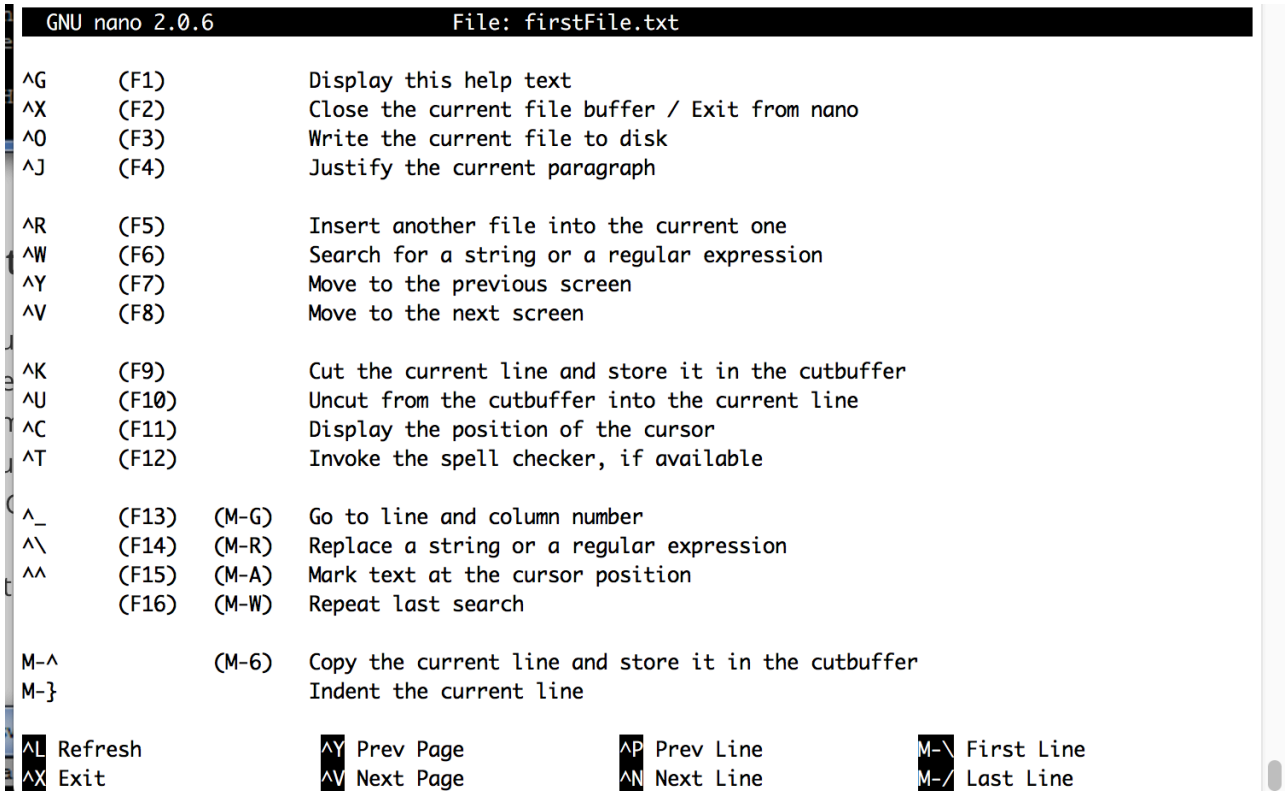
Things to notice:

- At the top

    - the name of the program (nano) and it's version number
    - the name of the file you're editing
    - and whether the file has been modified since it was last saved.
- In the middle
    - you will see either a blank area or text you have typed
- At the bottom
    - A listing of keyboard commands such as Save (control + o) and Exit (control + x)

Keyboard commands are the only way to interact with the editor. You cannot use your mouse or trackpad.

Find more commands by using `control g`:



The Meta key is <esc>. To use the Meta+key, hit <esc>, release, then hit the following key

Helpful commands:

- Jump to a specific line:
    - control + _ then line number
- Copy a block of highlighted text
    - control + ^ then move your cursor to start to highlight a block for copying
    - Meta + ^ to end your highlight block
- Paste
    - control + u

Nano is a beginners text exitor. vi and Emacs are better choices once you become a bit more comfortable using the terminal. These editors do cool stuff like syntax highlighting.

Git for Beginners

Git is a tool for managing files and versions of files. It is a *Version Control System*. It allows you to keep track of changes. You are going to be using Git to manage your course work and keep your copy of the lecture notes and files up to date. Git can help you do very complex task with files. We are going to keep it simple.

The Big Picture.

A Version Control System is good for Collaborations, Storing Versions, Restoring Previous Versions, and Managing Backups.

Collaboration

Using a Version Control System makes it possible to edit a document with others without the fear of overwritting someone's changes, even if more than one person is working on the same part of the document. All the changes can be merged into one document. These documents are all stored one place.

Storing Versions

A Version Control System allows you to save versions of your files and to attach notes to each version. Each save will contain information about the lines that were added or alted.

Restoring Previous Versions

Since you are keeping track of versions, it is possible to revert all the files in a project or just one file to a previous version.

Backup

A Version Control System makes it so that you work locally and sync your work remotely. This means you will have a copy of your project on your computer and the Version Control System Server you are using.

The Details

git is the Version Control System we will be using for tracking changes in our files.

GitHub is the Version Control System Server we will be using. They provide free account for all public projects.

The Basics

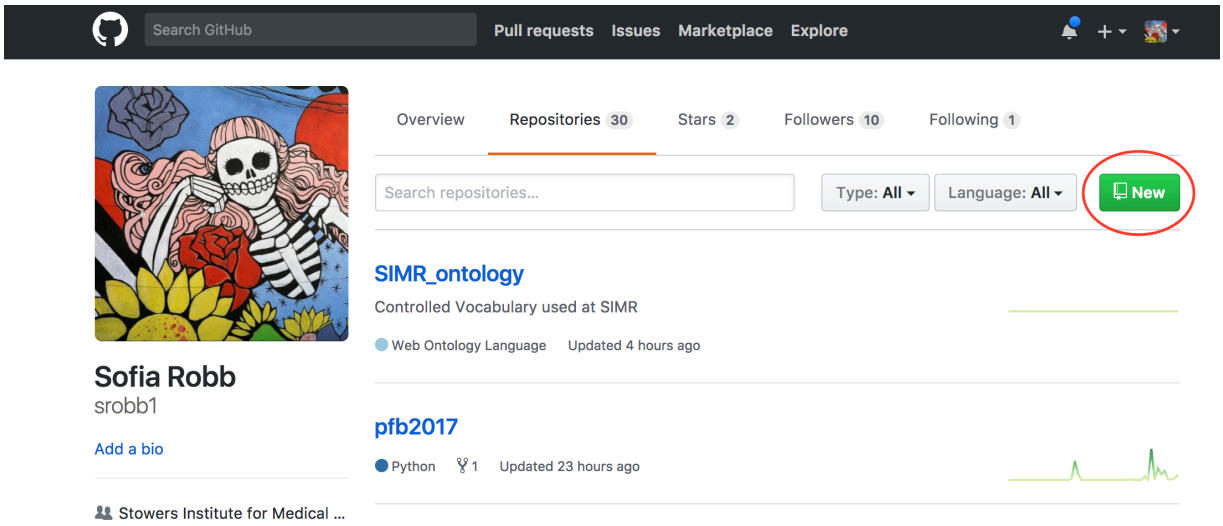
Creating a new repository

A repository is a project that contains all of the project files, and stores each file's revision history. Repositories can have multiple collaborators. Repositories usually have two components, one remote and one local.

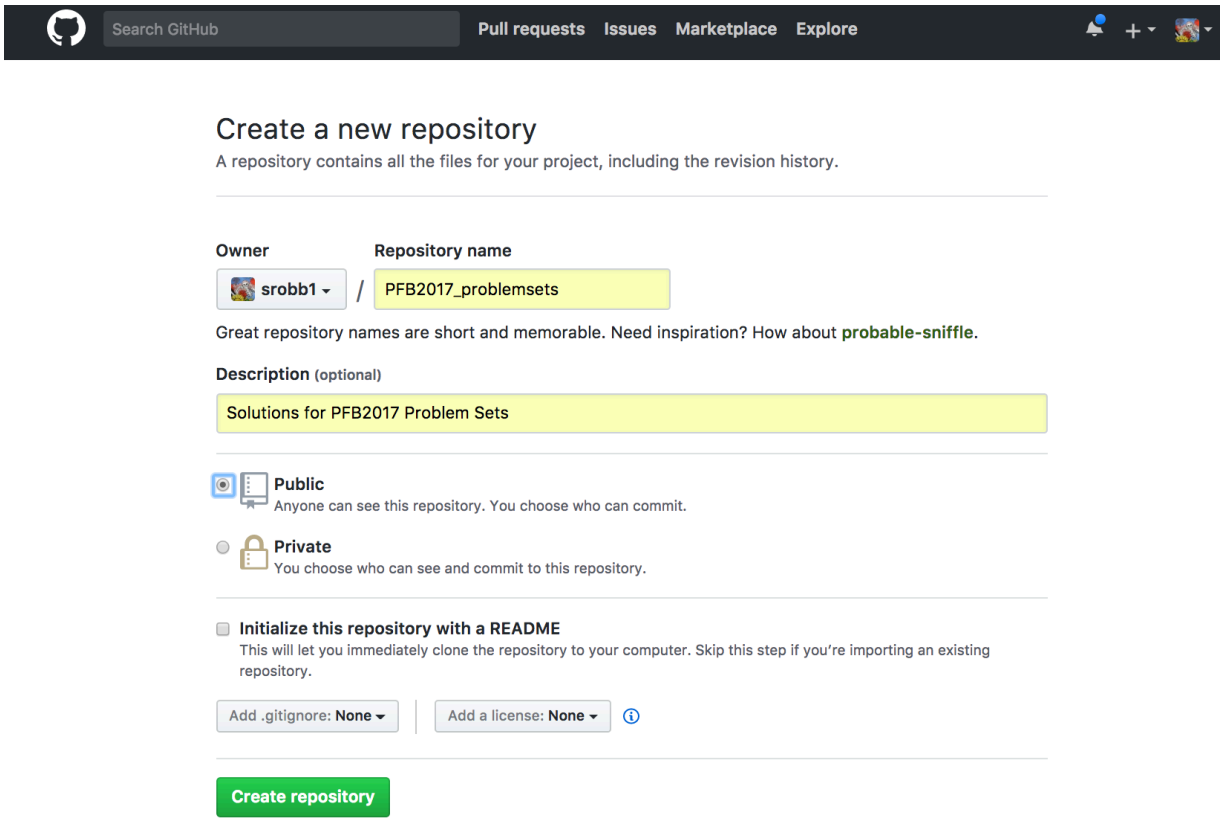
Let's Do It!

Follow Steps 1 and 2 to create the remote repository. Follow Step 3 to create your local repository and link it to the remote.

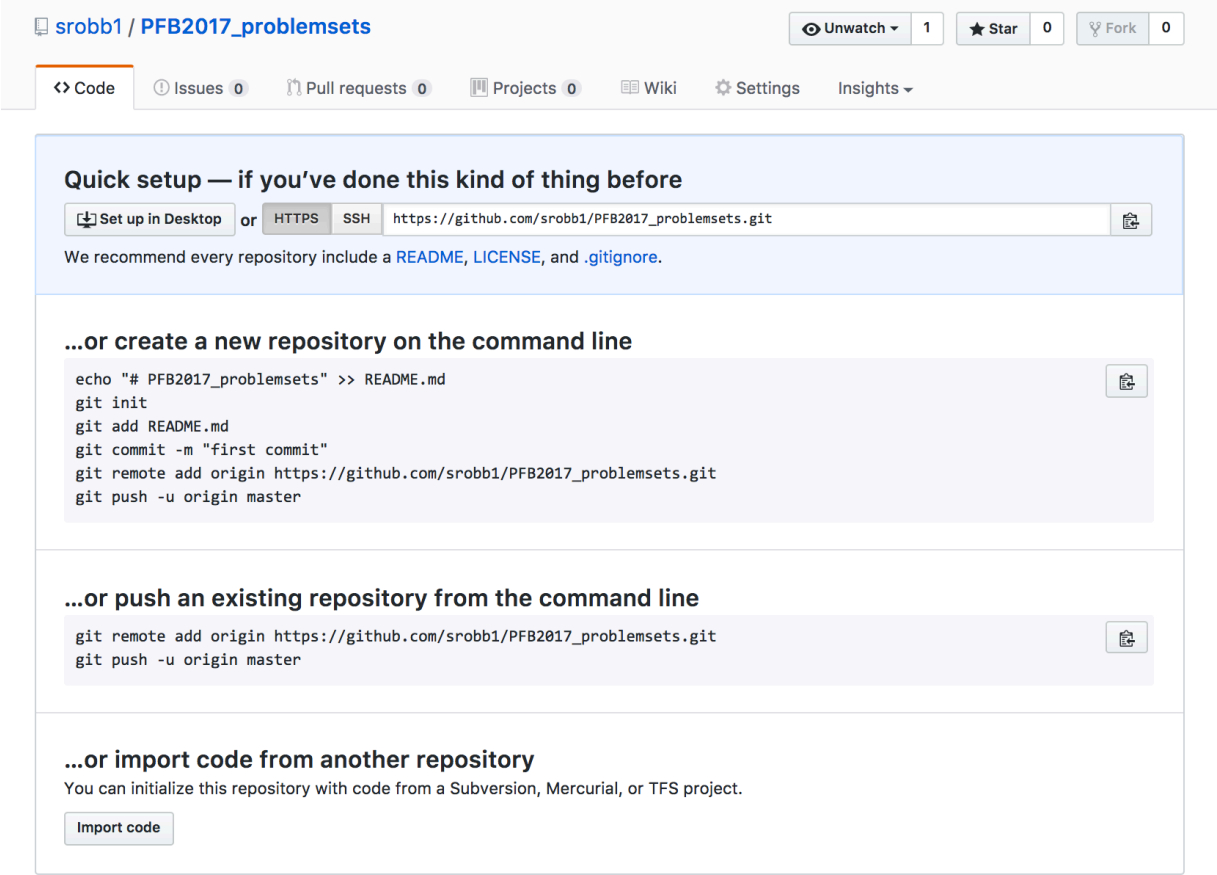
1. Navigate to GitHub --> Create Account / Log In --> Go To Repositories --> Click 'New'



2. Add a name (i.e., PFB2017_problemsets) and a description (i.e., Solutions for PFB2017 Problem Sets) and click "Create Repository"



3. Create a directory on your computer and follow the instructions provided.



- Open your terminal and navigate to the location you want to put a directory for your problem sets
- Create a new directory directory (i.e., PFB2017_problemsets)
- Follow the instructions provided when you created your repository. These are my instructions, yours will be differnt.

```
1 echo "# PFB2017_problemsets" >> README.md
2 git init
3 git add README.md
4 git commit -m "first commit"
5 git remote add origin https://github.com/srobb1/PFB2017_problemsets.git
6 git push -u origin master
```

You now have a repository!

Let's back up a bit and talk more about git and about these commands. For basic git use, these are almost all the command you will need to know.

Every git repository has three main elements called *trees*:

1. *The Working Directory* contains your files
2. *The Index* is the staging area

3. *The HEAD* points to the last commit you made.

> There are a few new words here, we will explain them as we go

command	description
`git init`	Creates your new local repository with the three trees on (local machine)
`git remote add remote-name URL`	Links your local repository to a remote repository that is often named *origin* and is found at the given URL.
`git add filename`	Propose changes and add file(s) with changes to the index or staging area (local machine)
`git commit -m 'message'`	Confirm or commit that you really want to add your changes to the HEAD (local machine)
`git push -u remote-name remote-branch`	Upload your commited changes in the HEAD to the specified remote repository to the specified branch

Let's Do it!

1. Make sure you are in your local repository
2. Create a new file with nano: `nano git_exercises.txt`
3. Add a line of text to the new file.
4. Save (control + o) and Exit (control + x)
5. (Add) Stage your changes. `git add git_exercises.txt`
6. (Commit) Become sure you want your changes your changes. `git commit -m 'added a line of text'`
7. (Push) Sync/Upload your changes to the remote repository. `git push origin master`

That is all there is to it! There are more complicated things you can do but we won't get into those. You will know when you are ready to learn more about git when you figure out there is something you want to do but don't konw how. There are thousands of online tutorials for you to search and follow.

Cloning a Repository

Sometimes you want to download and use someone else's repository.

Let's clone the course material.

Let's do it!

1. Go to our PFB2017 GitHub Repository
2. Click the 'Clone or Download' Button
3. Copy the URL

~Clone PFB2017

4. *Clone* the repository to your local machine
   `git clone https://github.com/srobb1/pfb2017.git`

Now you have a copy of the course material on your computer!

Bringing Changes in Remote to your Local Repository

If we make changes to any of these files and you want to update your copy you can *pull* the changes. `git pull`

Keeping track of differences between local and remote repositories

If you are ever wondering what do you need to add to your remote repository use the `git status` command. This will provide you a list of file that have been modified, deleted, and those that are untracked. Untracked files are those that have never been added to the staging area with `git add`

Links to *Slightly* less basic topics

You will KNOW if you need to use these features of git.

1. View Commmit History
2. Resolving Merge Conflicts
3. Undoing Previous Commits

Link To Unix 2 Problem Set
