Guide on how to set up your computer to use GitHub to manage versions of MTcode.

# Why use GitHub?

(1) Without GitHub, if more than one person makes changes to files in MTcode, a “master” directory needs to be manually maintained and passed back and forth. With GitHub, the master directory is remotely stored and all users only have to send edits to the remote directory. This way, all users send their changes to the same place and there is less potential for human error in tracking files and versions.

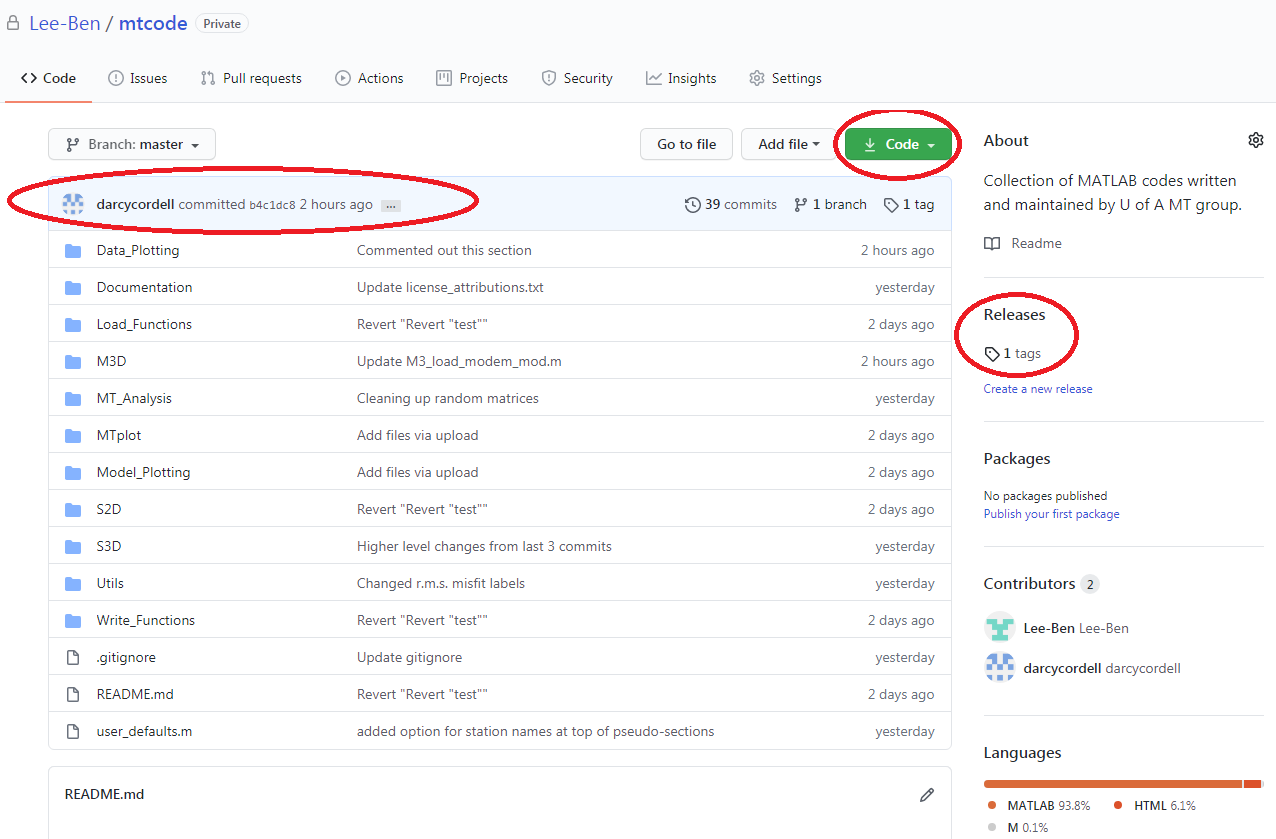
(2) With GitHub, any user can retrieve the latest version of MTcode at any time, without waiting for one person to compile the latest version on their local machine.

# How to get started with GitHub:

(1) Go to [github.com](http://github.com) and create an account.

(2) Once you have an account, tell me your username so that I can add you as a collaborator to the private mtcode repository. When you are added as a collaborator, on github.com you should see the mtcode repository home screen like the screenshot below.

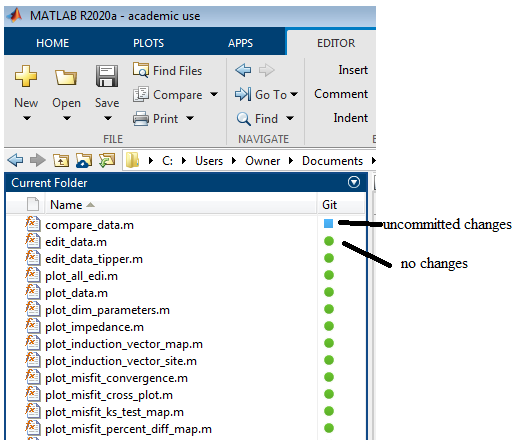
I circled some features to note. You can always see the latest changes at the top. The green button provides a link to download the repository. If you plan on being a contributor you should clone the repository in the next step. The repository will evolve over time, so we can compile releases once in a while to easily store versions of the code. These can be found under Releases.



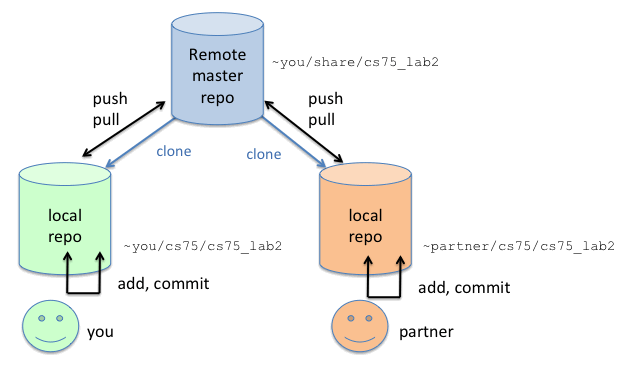
(3) Download the GitHub Desktop software from <https://desktop.github.com/> This is not necessary but I find it easier to use a gui than the command line… and instructions below assume you are using this software.

(4) After I add you as a collaborator, use GitHub Desktop to clone the mtcode repository (File -> clone repository). If you are a collaborator, the mtcode repository is listed under Your Repositories. Cloning the repository will create a local copy of the mtcode repository on your desktop. There is also a hidden .git folder that contains the files that git needs to keep track of files in the repository.

(5) In Matlab, navigate to the mtcode repository (and eventually set it on your path). In the current folder panel there should be a new column called Git with a little green dot next to each file (see figure below). This indicates if you have made any changes to the file from the last committed version on your local repository. If you see the Git column, I think your setup was successful.



Next, before you start editing files, it is important to understand the terminology of transferring files around. Here is a simple schematic of the workflow using GitHub:



*https://www.cs.swarthmore.edu/~adanner/cs40/f14/git.php*

**Repository (repo)**: a collection of files/folders, i.e. a project. In our case the mtcode repository contains the Matlab codes for mtcode.

**Remote master repo**: the remote repository, i.e. what you see on the mtcode page on github.com.

**Local repo**: the repository on your local computer (originally a clone of the master remote repository). The local and remote repositories will not always match - you will need to push/pull in order to get the latest version of the remote repository or send the latest version of your local repository.

**Add**: in our case, whenever you make edits to (or add or remove) a file in mtcode, you are adding changes to your local repository. While these edits are in the form of new, deleted, or edited files, git doesn’t commit these changes to your local repository until these edits are committed.

**Commit**: the action of confirming all your adds (edits) to update your local repository. This needs to be done before you move to the next step of updating the master remote repository.

**Push**: sending your local repository (after committed changes) to the master remote repository. Your pushed changes are implemented as the new version of the master remote repository.

**Fetch**: a safe way to get the latest version of the master remote repository, without updating your local repository. You can see what’s changed from your local repository before overwriting your local repository.

**Pull**: getting the latest version of the master remote repository, which updates your local repository. You might have conflicting changes between the remote and your local repositories that need to be resolved.

You now have a copy of the remote master repository as your local repository. Now it is time to start making your changes to mtcode.

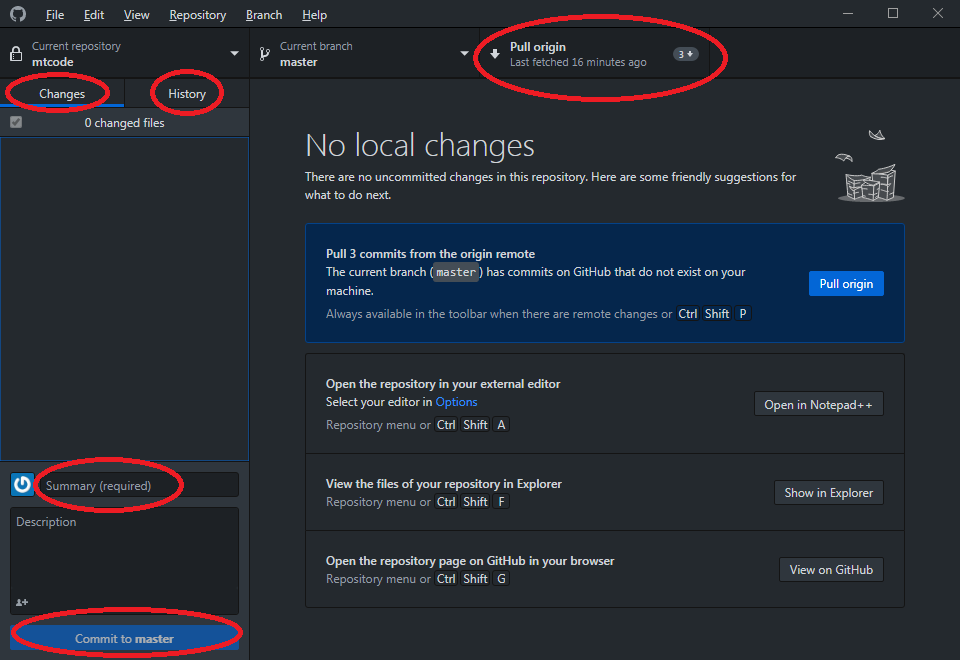
(6) If you make changes to a code in Matlab, the dot next to the file should change to a blue square. You won’t always have the mtcode directory open in Matlab, so you can also see what files are changed in GitHub desktop under the “Changes” tab (see next figure). Note that these are changes from your last committed version of your local repository. You will need to commit these changes in order to update your local repository. To do this in Matlab, right click on the file in the “Current Folder” panel, select Source Control, then View and Commit Changes. This confirms changes to your local repository.

(7) You can also commit changes in GitHub Desktop (I find this easier than in Matlab). Under the Changes tab, you will see the files that have been changed. Click on one to see a summary of the code that was changed. You can check/uncheck the box next to each changed item to include or not include it in your next commit action. In the summary box, write a summary of changes. Include a description if desired. Then click the “commit to master” button to commit the changes to your local repository.

(8) Next, push the changes to the master remote repository. There is a button near the top of GitHub Desktop to push, fetch, or pull to the master remote repository (it is called the origin here). If you have committed changes to your local repository, then use this button to push the changes. Otherwise, check this button once in a while to see if other users have pushed changes. If they have, you need to click this button to fetch or pull from the master remote repository.

You can see a history of changes in the History tab. This is updated every time you fetch/pull from the master remote repository.

(9) Start by making an edit to the GithubTest.m file in the mtcode directory. When you save your edits, the file should appear under “Changes” in GitHub Desktop. Following the instructions above, commit your changes to your local repository (either in Matlab or GitHub Desktop) and push the changes to the master remote repository. Your changes should appear on GitHub.com if successful.



# Important Notes

The user\_defaults file contains parameters that are customized by each MTcode user. These parameters might be specifically set for each user’s survey area and data set. Therefore, it is not practical to push this file in each edit to MTcode. When you edit your local user\_defaults file, make sure to uncheck the file under Changes in GitHub Desktop. This excludes user\_defaults from your committed changes and prevents it from being pushed to the master remote repository.

If someone can figure out how to use the .gitignore feature, we could automate this process.