git enaR: a workflow summary

M.K. Lau

September 5, 2013

1 What is git?

- A system for keeping track of changes to a project.
- Git is noted for its decentralized structure, so multiple people can use be working on a project simultaneously.
- Git can seem a bit esoteric at first, but is intuitive once you get into it.

2 What is github?

- An organizing server that provides space and interfaces for projects that use git.

3 What's the scoop?

Say you have a project and you want to keep track of changes made to it. The nucleus of git is that it does this by starting with an original project (called the "origin" in

git parlance) and then only keeping track of the changes made to it (as opposed to keeping entire copies of new versions of the same files).

The gist of git is:

1. Initiate a new project or link in to an existing project, aka. repository, (git init or git clone)
2. Tell git to track files that you are changing (git add –all)
3. Edit old files and create new files
4. Tell git to log changes made to your files (git commit -m 'Insert annotation here!')
5. Merge your changes with the project (git merge origin)

4 The perks of using github

- GUI
- Track and control users (add/remove collaborators)
- Web hosting
- Automatic page generator
- 1GB storage
- wiki
- Can it play with CRAN?

5 What do I do first?

- Register with github by creating an account here. Note that it's free as long as you stay below 1GB and keep it all open-source.
- Get git here.
- Setup your git account.

6 Try it out

1. Make a new directory called "git_test" (i.e. "mkdir git_test")
2. Change to this new directory (i.e. "cd git_test")
3. Initialize the new repository with "git init"
4. Make a new file called "README"
5. Tell git to track README with "git add README"
6. Now, see what git is doing with "git status"
7. Add the words "I made this README file." to README
8. Check the status "git status"
9. You can now "commit" these changes to the project with "git commit -a -m 'Created README.'
10. Check the status.

You now know the basics of git. The "add-commit" procedure is repeated every time you are making changes to the project.

Don't worry if you make some changes before adding files. If you add them later, git will recognize the changes that you've made.

Make sure to annotate the changes that you are committing. There's no hard rules, but generally changes should be committed grouped together based on tasks (e.g. "Fixed a bug in a function."). Google "git best practices" for more info.

7 How do I git with enaR?

.gitignore

Unless you tell it otherwise, git will pay attention (although not necessarily track) all of your files in your repository. In general, you should only be tracking the files that you yourself are changing and not files that are being autogenerated.

To do this, create a ".gitignore" file that will list which files git will ignore. Let's create one so that git will play nicely with emacs.

- Create an empty file named "∼/.gitignore"
- Add one line that says "∗∼". This stops git from tracking emacs' recovery files.
- Note the syntax with the use of the wild card symbol ∗. This tells git that any file name that ends with a tilde (i.e. ∼) should be ignored.

Clone enaR

To bring a copy onto your local machine run:

```
git clone git://github.com/MKLau/enaR
```

Note: It is also possible to do this on the github using the "fork" option. This is useful for working on other peoples projects in a way that will keep their code untouched but allow you to save to the github server.

Checkout a branch

Now, you will need to isolate your work from the "master" copy of the repository. To do this you will need to create a branch by running:

```
git checkout -b mycopy
```

You can also delete a branch, which you are no longer on, with:

```
git branch -D mycopy
```

Push a branch to the origin

Once you have made your changes and committed them (see above), you will want to "push" them onto the server so that other users can access them. To do this use:

```
git push
```

Depending on what instructions you used to setup git, this may produce an fial, error and/or warning. Hopefully not though.

If this succeeds, your information from your branch should be propogated up onto the github server. These changes can then be merged into the main project repository.

Pull enaR origin and merge

If you have already cloned a version of the project onto your local machine, you will want to update it when you start working on it again at some point. To do

this you will need to "pull" the origin off the server and merge it with your local version. This can be accomplished by first switching to your local repository and then checking out your branch:

```
git checkout mybranch
```

And then you can pull from the server:

```
git pull origin
```

Again, depending on your setup, you might need to alter the previous line of code for this to work.

Using SSH

By default, git uses HTTPS for security reasons. To switch to SSH and not have to constantly enter your security info use the following:

```
git remote set-url origin git@github.com:MKLau/enaR.git
```