14. Git On!

Z620: Quantitative Biodiversity, Indiana University

OVERVIEW

A major goal of Quantitative Biodiversity (QB) is to provide you with tools to conduct reproducible science. One of these tools is GitHub. GitHub helps ensure the integrity of a research project by managing changes to data and code. The version-control features of GitHub are useful for an individual working on a "solo" project, but also greatly facilitate collaboration among researchers.

In this handout, we present a workflow that will assist you with conducting reproducible science upon "graduation" from Quantitative Biodiversity.

After completing this exercise you will know how to:
1. Create a GitHub respository
2. Make a source-code file
3. Create and modify a README file
4. Properly license your work
5. Create a github.com account

1) CREATING A GITHUB RESPOSITORY

If you adopt GitHub as a part of your scientific workflow, you are going to want to create a new repository when you start a new project. A repository holds your files and records the history of changes that have been made throughout the life of your project.

This semester, you have been working with a repository that the instructors created for you (2015-QuantitativeBiodiversity). On the first day of class, you forked and cloned this respository to your local computer. After modifying the files on your local computer, you saved, added, and commited changes, which were then pushed to your origin. The origin is the version of the repository (e.g. QB2015_Smith) on your GitHub account. Last, you created a pull request that the instructors could merge into the "upstream" repository.

In the following section, we are going to walk through the steps involved in creating a repository in your GitHub account. You will be responsible for managing this repository, which can serve as the 'hub' for the code, text, and data of any project you wish.

1. Open a web browser and navigate to https://github.com/

2. At the top of page there is black banner. On the right-hand side there is **+** symbol.

3. Click on the **+** symbol and select "New Repository"

4. Give your repository a name. For this exericse, name your repository "QBTools".

5. Click the button that says "Initialize this repository with a README"

6. An additional option is to click on the "Add .gitignore" button and then select R. This allows you to identify certain files (e.g., .Rhistory) that will be ignored by Git when it looks for modified files to commit.

7. Click on the green "Create repository" button

8. Now, we are going to clone the newly created repository to your local machine. Open a Terminal window and type the following:

```
cd ~/GitHub
git clone https://github.com/User_Name/QBTools
cd ./QBTools
git status
```

9. Congratulations on successfully creating your first repo! Before we move on, it is good practice to check your remotes. You should have just one remote: origin. Open a Terminal window, `cd` to the correct directory, and type the code below to confirm.

```
git remote -v
```

2) CREATE A SOURCE-CODE FILE

Over the past eight weeks, we have created a number of functions in R. Some of them took a long time to write. In this part of the handout, we are going to compile some of these user-defined functions into an R source-code file. As you will recall, the benefit of a source-code file is that it contains "vetted" code, which can be used across multiple projects. Once created, you will be able to load your source-code file into a new project and use all of the functions that you created in Quantitative Biodiversity. Moreovoer, you can add new functions to this source-code file and effectively create a library of commonly used functions. The instructions below will guide you through the steps needed to create an R source-code file:

A. Create a Blank R File

Go to your menu bar in RStudio and choose: File > New File > R Script. Alternatively, you can use the following shortcut key: Ctrl + Shift + N. Note: you want this to be a .R file, not a .Rmd file. Save this blank file as `QBTools.R` in your QBTools respository directory.

B. Add a File Header

The header of your source-code file should provide necessary information that will allow others to use your script. What follows is an example header:

```
################################################################################
#                                                                              #
# Quantitative Biodiversity Functions Source Code                              #
#                                                                              #
# Written by: Rachel Smith                                                     #
#                                                                              #
# Last update: 2019/03/06                                                      #
#                                                                              #
# Notes: This file contains functions to cacluate metrics of taxonomic and     #
#        and phylogenetic diversity                                            #            #
#                                                                              #
################################################################################
```

There are a few things to note about the header. First, the comment character (#) prevents R from interpreting the information in the header. Second, the header is formatted for stylistic purposes. In our example, the header is 80 characters wide. As a rule of thumb, this is an appropriate width for both your header and code. We use the comment character (#) to box-off the top and sides of the header.

C. Nuts and Bolts of the Source-Code File

i. Require commands

In order for functions in a source-code file to properly work, we need to make sure that the contributed packages are installed. By default, the following code will try to load a specificied package. If that package is not found, R will install the package.

```
require("vegan")||install.packages("vegan");require("vegan")
```

You can cusotmize this code for any package that you want loaded for our source file.

ii. Functions

We are now going to populate our source-code file with functions that we have written this semester. Go through your assignments and find the functions that you want to save. Copy those functions into the new source-code file. Using the comment character (#), add a brief description of each function before the code. This annotation will help you and others understand how the functions operate. Here is a hypothetical example:

```
# My_Function: returns estimate of beta biodiversity
# Inputs: x, y, z

My_Function <- function(x, y, z){
  ...
}
```

When you are done copying functions over, save your `QBTools.R` file to your new repository. In a Terminal window, add and commit the file using Git:

```
git status
git add ./QBTools.R
git commit -m "Initial commit"
git status
```

3) CREATE A README FILE

You may recall that early in the semester we had you add some personal inforamtion to a README file. README files are commonly used and contain important documentation about the files (code, data, etc.) contained in a directory. It's good practice to inclucde a README file for all of your GitHub respositories. A basic README file should contain the following:

1. General information on the content of the repository
2. The version of programs and software that are needed to run the code in the repository
3. Contact information for the owner of the repository (e.g., name and email)
4. A description of any potential bugs

When you initialized your new repository above, you created a blank README.md file. Now, we are going to open this file in RStudio and make changes to the README file to include the information needed for your QBTools repository. We have provided an example of a README.md file in the GitOn directory of your QB_2015. You may want to check this out as a reference.

Before moving on, save the modified `README.md` file in your new repository. In a Terminal window, add and commit the file using Git:

```
git status
git add ./README.md
git commit -m "Initial commit"
git status
```

4) LICENSING YOUR CODE AND CONTENTS

Licensing allows you to establish the rights, privelages, and liabilities that you want to assume or waive for the work that you make available to scienstists, journals, funding agencies, and the public. Licensing also allows you to establish intellectual primacy by putting your ideas and work into the academic or public domain while maintaining rights and restrictions to use and redistribution. To this end, we want to make you aware of how to license your code and data, and will walk you through the process of adding a license to your repository.

What can you license? In short, you can license the code you develop, data you collect, photos you take, and any intellectually novel or creative work The trade-off in choosing a license is that the more privelages you retain, the less freedom you give to others. This is important to consider if you have discovered a new algorithm, metric, model, or generated a dataset that you would like others to use.

Licensing your GitHub repository: We will obtain a license from the following Github hosted site: http://choosealicense.com. This site has been setup to allow users to quickly learn about and choose a license. We will then then add the license as a file to our repo.

For this exercise, we suggest the GNU General Public License version 3. This is a free and widely-trusted *copyleft* license. The definition of copyleft is an arrangement whereby software or artistic work may be used, modified, and distributed freely on condition that anything derived from it is bound by the same condition. Unless you plan to charge a fee for others to use your work, the GPL v3.0 comes highly recommended. Later, you can change the license to anything you like or, remove it altogether.

Use the following steps to create a GNU GPL v3.0 license for your respository:

1. Go to the following website: http://choosealicense.com/licenses/gpl-3.0/
2. Read about the GNU GPL v3.0 license
3. Click on the "Copy license text to clipboard" button at the top-right of the page
4. In Rstudio, create a new R Markdown file and base the licence into the file
5. Save the file as `license.md` in your QBTools respository directory

Note, that if you eventually include code from another project, e.g., if you borrowed code from R or used code you found online, then you will need attribute what you used to the owner, including the date that you added their code. This can be as easy as making a small statement in your source code or it may entail including or abiding by the license of the original project.

Licensing data and creative works: You can also license novel datasets or most any intellectual work (videos, music, art, photos). Licensing these less software-like products can be done through creative commons licensing: https://creativecommons.org/licenses/. This website: https://creativecommons.org/choose/ will help you determine what license is right for you.

Do due diligence: When it comes to licensing, copyright, or copyleft, be sure of your rights and the rights of others, and conduct due diligence. Additional information on licensing, storing, and sharing your data can be found in the White et al. (2013) paper, "9 Simple ways to make it easier to (re)use your data", which can be found here: https://peerj.com/preprints/7/. The writing of this paper was managed through a public github.com repository, where anyone in the public could see it as it was being written: https://github.com/weecology/data-sharing-paper.

5) OPEN VS. CLOSED RESPOSITORIES ON GITHUB.COM

Many scientists who seriously engage in reproducible science elect to make their respositories public. If for whatever reason you prefer not to do this, you have a few options:
1) Choose to make your repository private, at which time github.com is likely to ask for your credit card information (it generally costs money to maintain private repositories)
2) Make your respository public for now, and then request private repositories from github.com (see here: https://education.github.com/guide/private_repos)
3) Make your respository public for now, and then delete the repo (or github.com account) after class