

# Helicopter Introduction to Github<sup>1</sup>

Min-Yang Lee

Northeast Fisheries Science Center  
Woods Hole, MA 02536

March 7, 2022

---

<sup>1</sup>[https://github.com/NEFSC/READ-SSB-Lee-WorkingEfficiently/blob/main/presentations/github\\_overview.Rmd](https://github.com/NEFSC/READ-SSB-Lee-WorkingEfficiently/blob/main/presentations/github_overview.Rmd)

## Goal:

- ▶ Understand what github is and why it's a useful tool.
- ▶ Use the web editor to make a change to a repository.

## Pre-work:

- ▶ Get a github account.
  - ▶ This will take you about 15 minutes of actual time, plus some emails to ITD.
  - ▶ Follow the [github rules](#), which are long and complicated.
- ▶ Look through one of these two pages and find a mistake or a gap in the information.
  - ▶ <https://github.com/NEFSC/READ-SSB-Lee-metadata> or
  - ▶ <https://github.com/NEFSC/READ-SSB-Lee-WorkingEfficiently>
- ▶ Spend 10-15 minutes looking though your own code, emails, 3 ring binders, or documentation. Find a widget that you'd like to share.

# What is Github?

- ▶ Github is a tool to help you produce reproducible research.
- ▶ Github is Google Docs for code.
- ▶ Github is track changes for projects.
- ▶ Github has *some* lightweight project management tools. You can:
  - ▶ Track issues and assign them to people.
  - ▶ Aggregate related problems into a project.
  - ▶ Break down a long range project goal into smaller chunks.

# What are the disadvantages?

- ▶ Adds 2-3 steps to your workflow.
  - ▶ Yes, it's a little annoying at first.
  - ▶ Yes, the annoyance disappears quickly.
- ▶ Take care not to upload any sensitive information
  - ▶ Passwords, data, server addresses
  - ▶ Yes, you can automate this.

# What are the advantages?

- ▶ Collaborate with colleagues.
  - ▶ Work simultaneously and iterate quickly when developing code.
  - ▶ No emailing code back and forth.
  - ▶ Write collaboratively: [Paper](#) and [repository](#)
- ▶ Makes your life easier (maybe) when you do your revisions after 6 months in review.
- ▶ Project continuity when there is staff turnover
- ▶ You can't "break" someone's code.
  - ▶ Every version that you tell Github to save is saved.
  - ▶ You can always go back to a previous version. If you've written a good enough note that you can find that version quickly.

# Privacy and Control

- ▶ The owner of the repository can control who can see the repository
  - ▶ Anyone
  - ▶ Certain people
- ▶ The owner of the repository can control who can make changes to the repository:
  - ▶ Anyone
  - ▶ Certain people

# Getting Started with Editing a Document

- ▶ You can use github's online editor for simple things.
- ▶ Just need a github account.
- ▶ Workflow 1:
  - ▶ Edit a document using Github's online editor.
  - ▶ Write a *commit message*
  - ▶ "Save and *Fork*" the repository.
  - ▶ Submit a *pull request*: ask the owner to review and integrate changes.
  - ▶ **After** the changes are integrated, delete your Fork.
  - ▶ You can always do this



## Getting Started II

- ▶ Workflow 2:
  - ▶ Edit a document.
  - ▶ Write a *commit message*: a note about what you did
  - ▶ Save it by *committing* to the main *branch*.
  - ▶ This is my favorite for small things. What's small? – you know it when you see it.
- ▶ Workflow 3:
  - ▶ Create a *new branch* in the repository.
  - ▶ Edit a document.
  - ▶ Write a *commit message*
  - ▶ Commit to that branch.
  - ▶ Submit a *pull request*: ask the someone to review, approve, and integrate changes.
  - ▶ **After** the changes are integrated, delete the branch.

## For more complicated tasks:

- ▶ Ask IT-helpdesk to install “git” and either “Github desktop” or “Rstudio” on your computer.
- ▶ Workflow 4:
  - ▶ *Clone the project to your computer.*
  - ▶ Create a *new branch* in the repository.
  - ▶ Edit lots of documents.
  - ▶ Write a *commit message*
  - ▶ Commit to that branch.
  - ▶ *Push the changes up to Github.*
  - ▶ Submit a *pull request* for the owner to review and integrate your changes.
  - ▶ **After** the changes are integrated, delete the branch.
- ▶ Lots of guides on the internet on how to use git and github.

## A few guidelines

- ▶ *main* should always work. For everyone.
- ▶ No passwords, API keys, PII, or confidential data
  - ▶ Environment variables or
  - ▶ .gitignore
  - ▶ Load in data from Oracle or from locations on the network.
- ▶ Small data on the repository is fine.

# Working Efficiently

<https://github.com/NEFSC/READ-SSB-Lee-WorkingEfficiently> is a table of contents to some of the things I've collected, including links to

- ▶ Oracle metadata
- ▶ A project template with data extraction code
- ▶ Some code to run R on the NEFSC Servers.
- ▶ Code to construct Affiliated Firms for the RFA Analyses.
- ▶ Code to assemble custom rasters.