A quick Github how-to

If you don't have a git software, go download Git Desktop (it will make things easier)

https://desktop.github.com/

If you want to have access to the cpue.rfmo package, ask Simon for access: simon.hoyle@gmail.com (give him your Github user name)

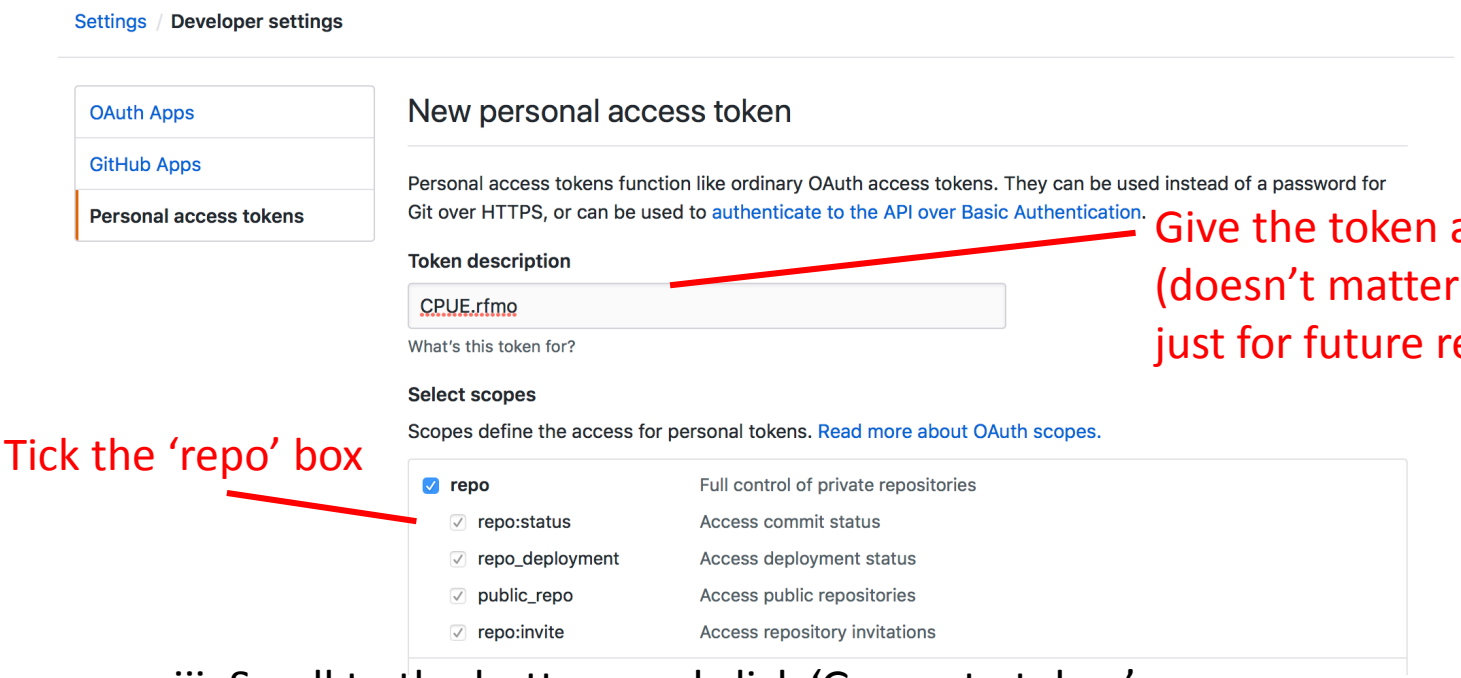
Outline:

> Install cpue.rfmo with a personal token
> Create a new personal git project and add files
> Add a collaborator to your project and go through conflicts

> Make a Github webpage

How to access the CPUE.rfmo library from R (1/2)

1. Create a github account (https://github.com/join)
2. It is a **private repository**, so you will need to be granted view access by Simon (you will get an email from github asking you to confirm)
3. Because it is a private directory, R needs a way to verify you are authorized to download it. The easy way to do that is to create a **personal access token** from within your github account:
   i. Sign-in to your github account
   ii. Go to: https://github.com/settings/tokens to make a token



Give the token a name (doesn't matter what it is… just for future reference)

Tick the 'repo' box

   iii. Scroll to the bottom and click 'Generate token'
   iv. Copy the token (you can save it to a .txt for later use if you want)

How to access the CPUE.rfmo library from R (2/2)

4. Within R, make sure you have the **devtools** library installed
5. Install cpue.rfmo in R by launching:

```
> library(devtools)
> install_github('hoyles/cpue.rfmo',
        username='yourgithubuser',
        auth_token='PASTEYOURTOKENHERE')
> library(cpue.rfmo) # this should work now
```

Boom!

How to create your own Github project (or ‘repository’)

Sign-in to your Github account
Click on the ‘Repositories’ tab
Click on the green button with New on the right-hand side
Pick a name for your directory and decide if it should be Public or Private

... *the other options can be changed later*

Click Create repository

Create a new repository

A repository contains all project files, including the revision history.

Owner **Repository name** *

lauratboyer / yellowfin2019

Pick a project name (it will become a folder in your computer)

Great repository names are short and memorable. Need inspiration? How about **bookish-octo-spork**?

Description (optional)

Processing data inputs for the 2019 yellowfin assessment

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

How to create your own Github project (or ‘repository’)

You have now created the ‘remote’ or ‘origin’ version of you directory. Now you need to make the ‘local’ version on your computer.

Get the link for your new repository in Github by clicking on the Clone or download button. Copy the provided link (or click on Open in Desktop if you are using Github Desktop)

Back to your computer, go to your Git software (e.g. Github Desktop, SourceTree, or the terminal)

Click on File > Clone repository > URL tab > Paste the provided link
Also pick a location for the repository folder to be located, e.g. C:/Projects or User/Documents/
Click on Clone

You have a Github project, now what?

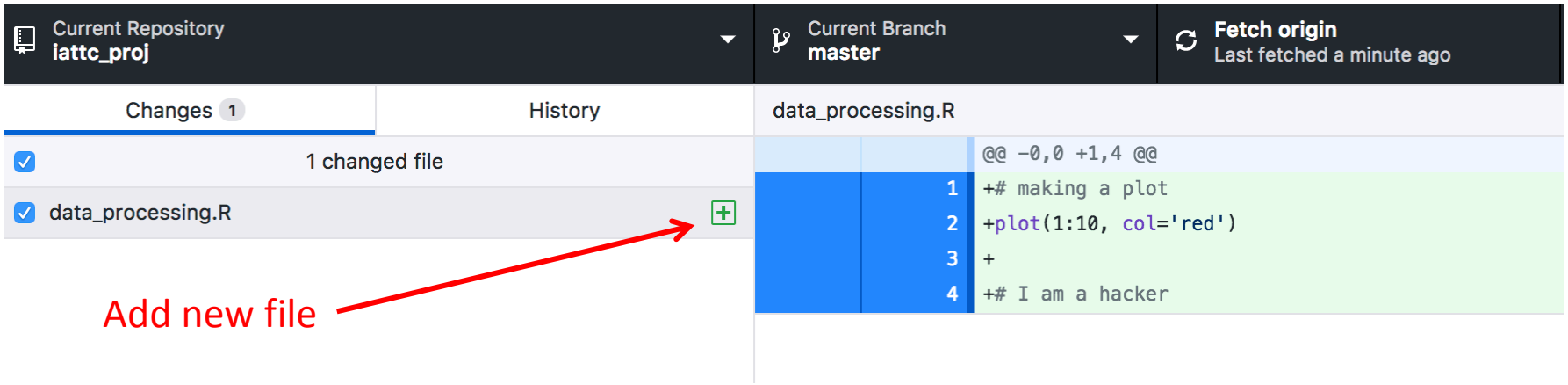
Save or copy existing files (e.g. R scripts) to the directory.

New files need to be ‘added’ to the project so they get tracked (‘version-controlled’) (this will store the original version of the file, from which subsequent changes will be tracked)

Within Github Desktop new files will show under the Changes tab on the left-hand side with a green plus logo

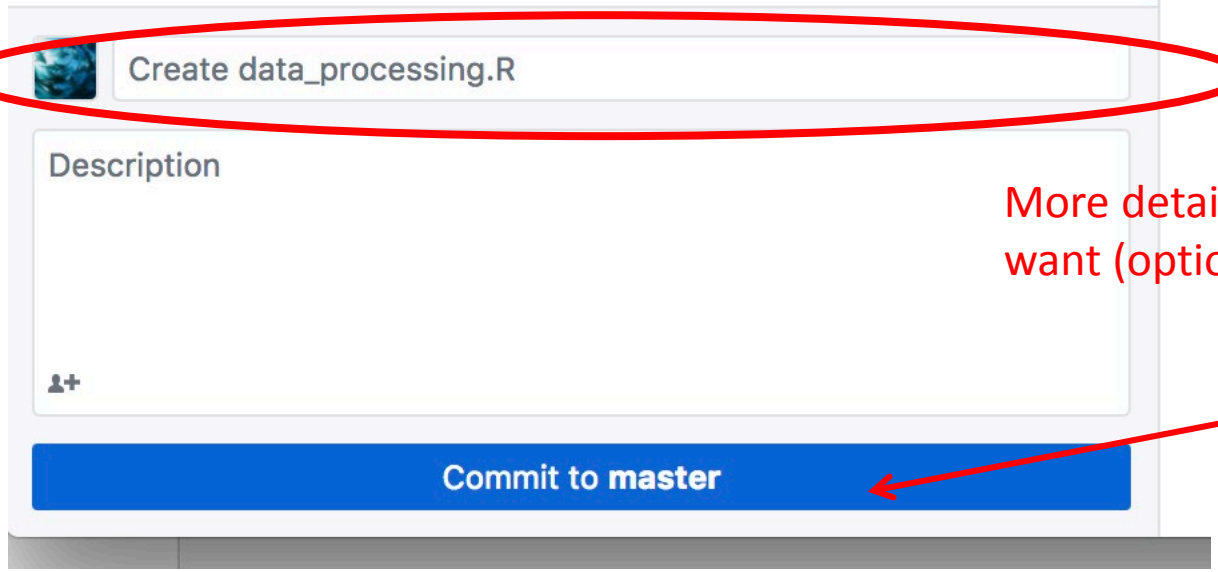
Commit (*add a message*) and Push the changes to the origin. You should now be able to see the new files online on the Github project.

You have a Github project, now what?



Add new file

Add a message to describe what your commit is about



More details if you want (optional)

Commit

(add changes to log!)

Then:

You have a Github project, now what?

Do the same thing for any changes to existing project files

1. Save the changes to the file(s)
2. Commit (with a useful message)
3. Push

* If you are working with a collaborator, make sure to pull changes (or 'sync') before starting to work on files, to make sure you have the latest version. Else you might have to solve a conflict between file versions.
* To add a collaborator to your project, go to

> Collaborators

Example project with a webpage
https://github.com/lauratboyer/iattc_tutorial/

Tell Github to create a webpage for the directory
(Settings --> scroll down to GitHub Pages 'Source')

Files needed by Github to be able to make a website:
_site.yml **** defines the structure and layout of your webpage*
index.html *(from index.Rmd if using rmarkdown)*
(+ any other .html listed in your _site.yml)
The webpage won't compile without those two files!

You can create all of the pages of your webpage with Rmarkdown (.Rmd) and use the function rmarkdown::render_site() to translate all of the .Rmd within a folder to .html (needs _site.yml + index.Rmd to work)

Commit + Push all the generated webpage files to your Github project and a webpage will automatically be created at:
https://yourusername.github.io/yourprojectname/ (there might be a <1min delay)

A simple _site.yml:

```
name: "lauras-website"
output_dir: "."
navbar:
  title: "A random website"
  left:
    - text: "Home"
      href: index.html
    - text: "Fish things"
      href: Web1.html
```

Where to look for webpage content? '.' means in the current folder, otherwise defaults to '_site'

Navigation bar for the website (by default, on top) with labels for each page (Home and Fish things)

index.html and Web1.html were created by render_site()

User only needs to create and edit **index.Rmd** and **Web1.Rmd**

[See the resulting webpage here: https://lauratboyer.github.io/iattc_tutorial/](https://lauratboyer.github.io/iattc_tutorial/)