

Fork the class repo to your GitHub account

Connect to docker and mount a local volume

Start a new project in Rstudio.

Choose version control project.

Choose git.

Copy 'https:' repo url then press tab

<https://github.com/nlangholzuc1a/stat418-tools-in-datascience-2025.git>

Browse to the directory where you want to setup your project and open

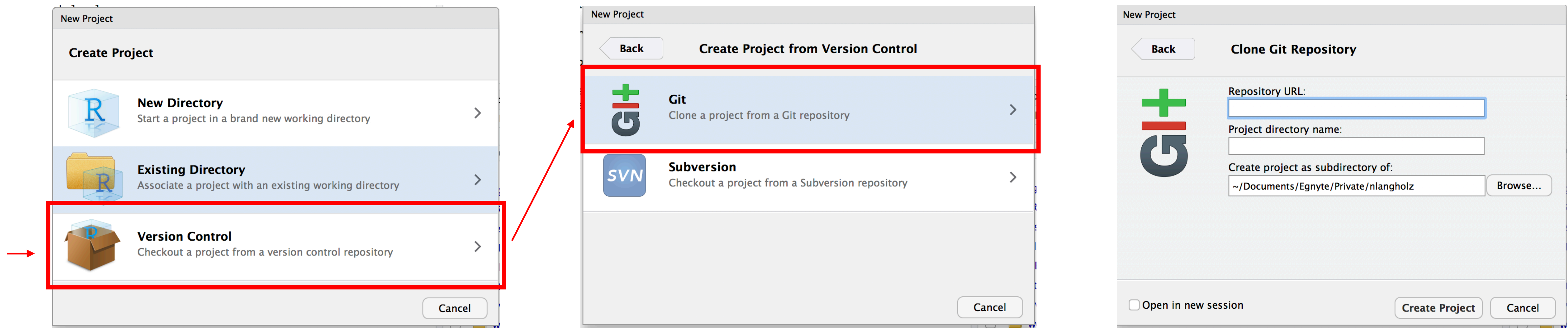
Start the new project

**This is my 'student'
repo; use your own for
this step**



Choose version control project. (1)

Choose git. (2)

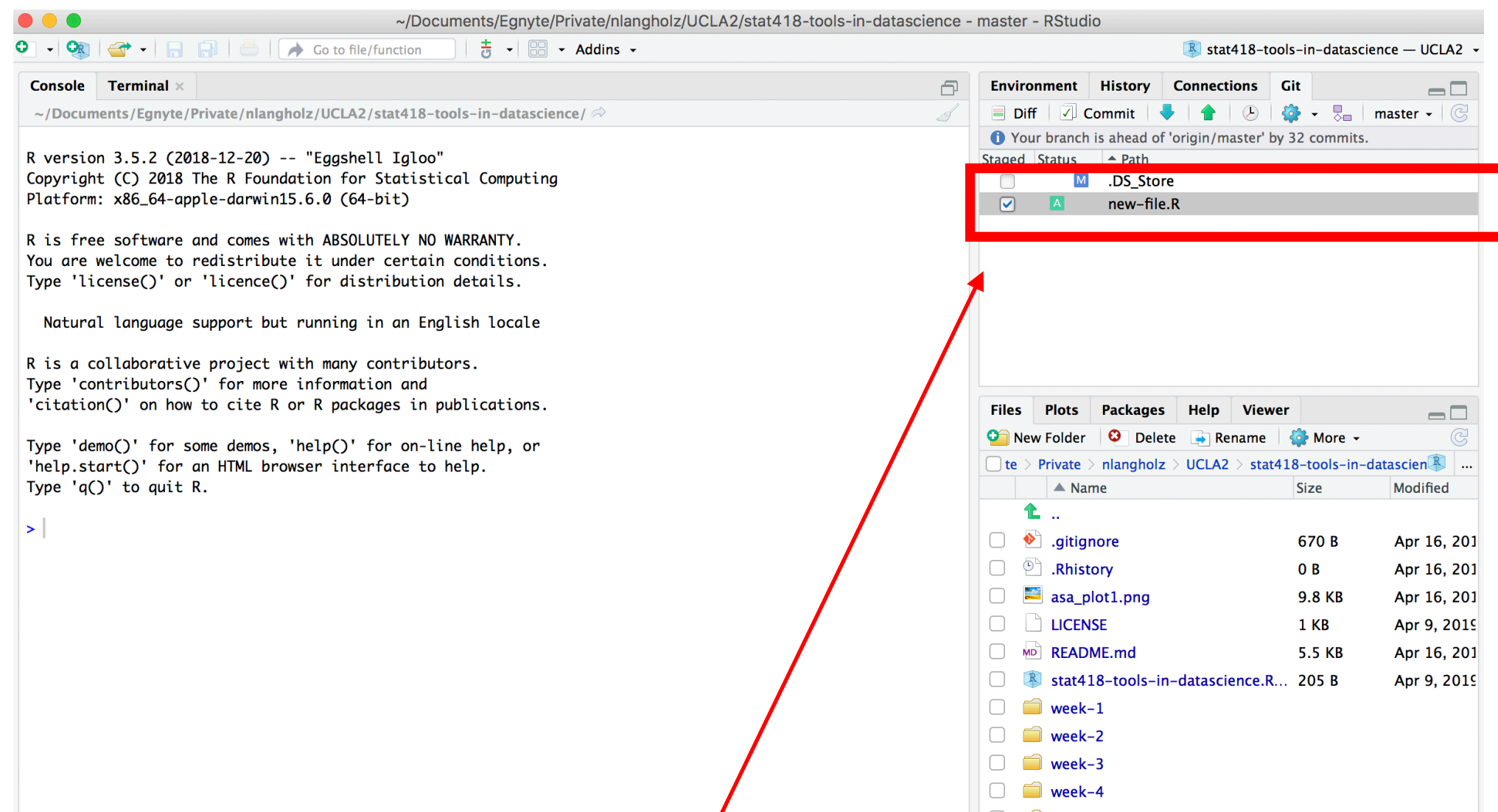


Copy 'https:' repo url then press tab (3)

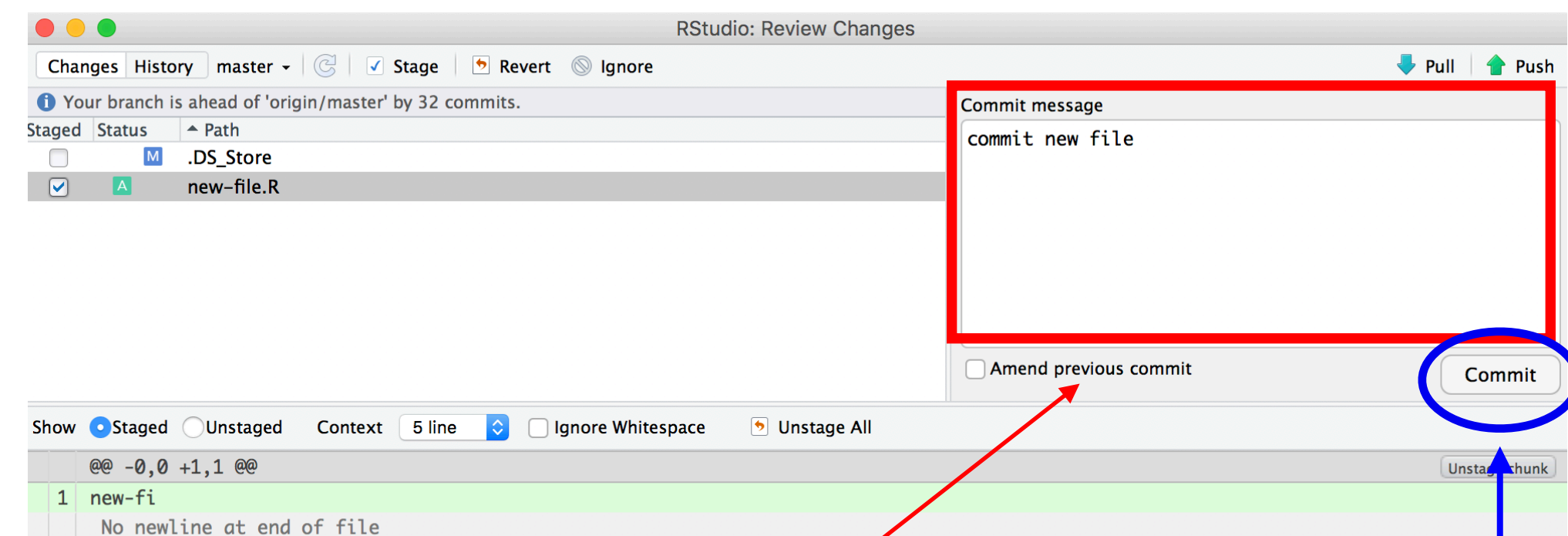
<https://github.com/nlangholzucla/stat418-tools-in-datascience-2025.git>

Browse to the directory where you want to setup your project and open. Start new project. (3)

Create your first commit. Make a file; save it to your local repo.



Click the box to stage
and then commit



Commit Message

Commit

Click the the box next to file to stage it. Then click commit and add a commit message.

Finally click the box to commit.

Finally click the box to commit.(continued...)

This won't work so you will have to set your global git user email and user name.

(If it does work then you have already set up these details at a previous time. Congrats!)

Click on the terminal tab next to the console in Rstudio.

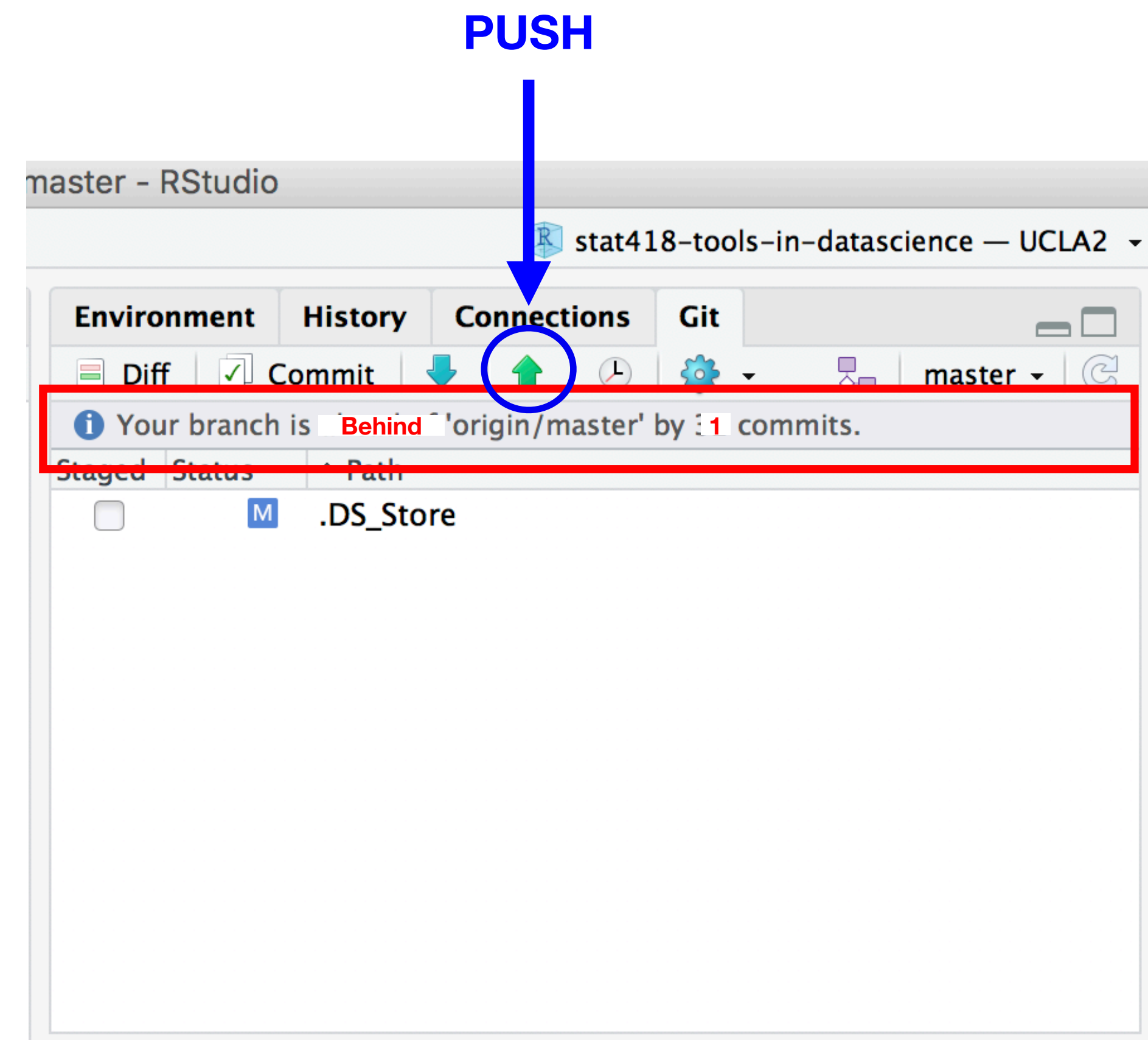
In the terminal

```
git config --global user.email "<your-github-email>"  
git config --global user.name "<your-github-username>"
```

Now try to commit again. Use a descriptive comment. Will ask for your username and password.

You will see that your origin/master is one commit behind your local repo. To get your remote repo up-to-date you need to push. Click the green up arrow to push.

Now go check your remote repo (on GitHub.com) to make sure the changes are available there.



When you go to push it will now ask for your GitHub username and personal access code

The personal access code, can be created under your profile settings, developer settings, and personal access tokens.

Create a classic one for this use with here and so far I've selected all scopes. With that copied token, you can now input and push.

The screenshot shows the GitHub profile settings for user 'natelangholz' (Nate Langholz). The left sidebar contains a list of settings: 'Set status', 'Your profile', 'Your repositories', 'Your Copilot', 'Your projects', 'Your stars', 'Your gists', 'Your organizations', 'Your enterprises', 'Your sponsors', 'Try Enterprise' (with a 'Free' badge), 'Feature preview', and 'Settings' (highlighted with a red box). A blue arrow points from 'Settings' to the 'Developer settings' link in the right sidebar, which is also highlighted with a red box. Another blue arrow points from 'Developer settings' to the 'Personal access tokens' section in the main content area, which is also highlighted with a red box. The 'Personal access tokens' section shows 'Fine-grained tokens' and 'Tokens (classic)' (highlighted with a red box). The 'New personal access token (classic)' form is visible, with fields for 'Note', 'Expiration' (set to '30 days (May 12, 2025)'), and 'Select scopes'. The 'repo' scope is selected, providing full control of private repositories. Other scopes include 'repo:status', 'repo_deployment', 'public_repo', 'repo:invite', and 'security_events'.

natelangholz
Nate Langholz

Set status

Your profile

Your repositories

Your Copilot

Your projects

Your stars

Your gists

Your organizations

Your enterprises

Your sponsors

Try Enterprise Free

Feature preview

Settings

Scheduled reminders

Archives

Security log

Sponsorship log

<> Developer settings

Company

You can @mention your company's GitHub organization to link it.

Location

Santa Monica, CA

☐ Display current local time
Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information.

Update profile

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration

30 days (May 12, 2025)

The token will expire on the selected date

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events

Sync your forked repo to my master repo. Configure a remote that points to the upstream original repository in git to sync changes you've made with the original and allows you to sync changes made in the original repository with the fork.

List current configured remote repository for your fork

```
git remote -v
```

Specify a new remote upstream...the original repository

```
git remote add upstream https://github.com/natelangholz/stat418-tools-in-datascience-2025.git
```

Makes sure the new upstream repository is the correct one

```
git remote -v
```

Sync a fork of a repository to keep it up-to-date with the upstream repository. Fetch the branches and commits from the upstream repository. Commits to master will be stored in a local branch, upstream master. **Repeat this in your project when you want to sync new commits from upstream... should do this at least once a week (at the start of each class)**

List current configured remote repository for your fork

```
git fetch upstream
```

Check out your fork's local master branch

```
git checkout main
```

Merge changes from upstream/master into your local master branch. Brings your fork's master into sync with upstream repository, without losing local changes

```
git merge upstream/main
```