Cheat Sheet

Working with Git & GitHub in R

THE BASICS

- Start every work session with a **pull** so you are up to date with collaborators
- Before first pull, run following Command Line Interface (CLI) code:

```
git config --global pull.rebase false
```

- Frequently while working, and especially at the end of every session, stage-commit-pull-push (see below) so you and collaborators can stay up to date

1. Stage ✓

RStudio: After edits are saved, check the box next to each file in the "Git" pane

CLI:

```
git add <specific files>
```

2. Commit ✓

Click the commit button under the "Git" pane

CLI:

```
git commit -m "<your informative commit message here>"
```

3. Pull ⬇

Click the downward blue arrow in the "Git" pane

CLI: `git pull`

4. Push ⬆

Click the upwards green arrow in the "Git" pane

CLI: `git push`

Congrats, your changes are now synched with Github!

Pro Tip: Google your error messages for helpful solutions!

Cheat Sheet

Working with Git & GitHub in R

For a deeper dive:

NCEAS LTER NETWORK

TIPS & TRICKS

Merge conflicts occur when Git cannot automatically merge changes for at least one file, and they require manual resolution.

Avoiding Conflicts

- Communicate early & often with your collaborators
- Make a plan for editing at the same time as others
- Write informative commit messages so you can diagnose conflicts if they occur
  - *Check out conventionalcommits.org & njlyon0.github.io/tips/commits for tips on writing good commit messages*

Resolving Conflicts

- Don't panic!
- Search for conflicted areas (maybe multiple) within file(s)
  - Look for lines with special symbols "<<<<", "====", ">>>>"
- Discuss the conflicted section(s) with your team and decide on how you want the code to look
- Delete special symbols & edit the code
  - Make necessary commits as you go
- Pull again to avoid another conflict
- Push your changes

Visit QR codes at top of cheat sheet for Intro to GitHub resources!