Terminology

branch - version of repo for a specific task

commit - checkpoint that user can roll back to

head - most recent branch commit

merge - the process of combining two branches

pull - get latest version of a branch from the remote origin

push - send changes to the remote origin

remote origin - central location users changes are routed through

repository (repo) - collection of files / folders

save - a file must be saved to disc before it can be staged

stage - pre commit step

status - provides info on uncommitted changes and comparison to remote origin for example

Usage

We'll use git for three main things: (1) To create checkpoint we can roll back to. (2) To create branches for specific tasks we want to work on. (3) Merging branches.

Workflow

Our typical workflow will look like this:

(1) Create a branch on the remote origin (e.g. GitHub)
(2) Pull the branch onto our local directory
(3) Change something
(4) Stage the change
(5) Commit the change
(6) Repeat (3)-(5)
(7) Test all the changes work
(8) Push commits to branch (repeat)
(9) Submit merge request to master (e.g. via GitHub)

Best practice

- Create a new branch for each task
    - Branch names should concisely describe the task without being too general (not always easy). Use lowercase characters [a-z] for branch names with words separated by underscores.
    - Submit a merge request for the new branch (this will reduce the chances of a naming conflict)
    - Is the branch: (1) To be merged *to* the master once a feature has been added or updated? (2) To receive merges *from* the master in the future? (3) Never cross paths with the master again? Consider the implications for the branch.

- Commit often
    - Divide the large task into subtasks for example a new/updated plot, code-chunk or stylesheet compliance. Commit after each subtask.
    - Always add a comment when you commit, comments should concisely explain the change. Use lowercase alphanumeric characters [a-z, 0-9] only. If you cannot describe you commit in <50 characters the subtask is too long!

- Push often
    - Push before moving to a different branch (task).
    - Try to organize your subtasks so that you finish your work session at a good place to push.
    - It's ok to push code that doesn't work 100% (this is how your coworkers can review your code and help you out, perhaps they'll branch your code at this point).

- Merging with the master (we'll need to test this protocol as we go)
    - Open a merge request on GitHub
    - Ensure the master merges cleanly to the branch before merging the branch to the master (this way we should never get a bad merge to the master)

REMINDER OF TERMS AND WHAT TO DO

stage - comment - commit

as many times as required (to as many different files/folders as desired).

push

within R (the green arrow), push repo changes to the remote origin (now your local branch and the remote branch of the same name are in sync), all changes to as many things you've changed (files, folders, data…) will be pushed.

merge master into branch

using GitHub, check the master merges with your branch (i.e. pull *master* into *branch*), i.e. do any updates made to the master (since you branches) conflict with your updates? If there are conflicts resolve them (we haven't encountered this yet, so I'm not sure how we'll do this exactly, just let me know when it does).

merge branch into master

using GitHub merge your changes into the Master (i.e. pull *branch* into *master*)