Git best practices

October 2, 2021

1 Make sure you are in the right directory in the terminal

In order to use `git`, you must be inside the `stat-471-fall-2021` directory in your terminal. Note that your directory in the terminal is not necessarily the same as your directory in RStudio! The instructions to navigate to a directory are in `getting-started.pdf`. You can check your current directory in the terminal by typing `pwd`.

2 When in doubt, type `git status`

The `git status` command will tell you what the state of affairs is, e.g. if you have made changes since your last commit and which files have been changed. If you did not successfully navigate to the `stat-471-fall-2021` directory (see above), typing `git status` will let you know by printing the message `fatal: not a git repository (or any of the parent directories): .git`.

3 Order of operations

When working with `git` and Github, you will be committing, pulling, and pushing. While you may or may not need to use all three of these operations every time, you should

- **commit before pulling**: You should no new changes to be committed (a "clean directory") before pulling. You can verify this by typing `git status`, and you should get the message `nothing to commit, working tree clean`. If you have `Changes not staged for commit`, then commit these changes before pulling.
- **pull before pushing**: You should have pulled all new changes from Github before pushing. You can just type `git pull` before typing `git push`.

4 Committing regularly

As you work on your homework, commit your changes every time you are about to take a break or switch to another task. For example, suppose you work for a couple hours and make some progress on HW1 Q1. You are about to break for lunch. Before doing so, commit your changes by typing the following commands in the Terminal:

```
git add --all                                  # stage changes for commit
git commit -m 'made some progress on HW1 Q1'   # commit your changes
```

The commit message is mandatory. Do not try committing without a commit message. Also, make sure to run the `git add --all` command together with the `commit` command. In particular, do not try to commit with first typing `git add --all`, and note that typing `git add --all` only *stages* your changes for a commit but does not actually commit them.

5 Pull regularly and commit before pulling

To stay up to date on the course materials, pull them regularly (e.g. daily) from the course Github repository by typing

```
git pull
```

Make sure you have committed all your changes before pulling!

6 Push regularly and pull before pushing

To keep your personal copy of the `stat-471-fall-2021` Github repository up to date, push your changes regularly by typing

```
git push
```

Make sure you pull before pushing, and commit before pulling. A common workflow is to make some changes, then commit them, then pull any new changes from Github, then push everything back to Github:

```
git add --all                                    # stage changes for commit
git commit -m 'made some progress on HW1 Q1'     # commit your changes
git pull                                         # pull updates from Github
git push                                         # push your changes to Github
```

7 Merging changes

Sometimes, the teaching staff may make changes in the course Github repository to one or more files that you have also changed on your computer. In this case, a *merge* is required after you pull. There are two situations to be aware of: *automatic merge* and *merge conflict*.

7.1 Automatic merge

In most cases when merges are necessary, you and the teaching staff will have edited different lines of the same file. In this case, Git is smart enough to automatically merge these changes. You will see a screen like in Figure 1 pop up. This is Git opening up the Vim text editor and prompting you to write a little not about why this merge was necessary. However, since a default message is already present (`Merge branch 'main' of https://github.com/Katsevich-Teaching/stat-471-fall-2021.git into main`), there is no need to do anything. We just want to exit the Vim editor by typing `:q` and then pressing `Enter`. If this does not work, press the `Esc` key prior to typing `:q`; you can also try `:wq` instead of `:q`. Once you exit the Vim editor, you will see a message like the following in the Terminal:
This just says that an auto-merge was done and that you are all set.

7.2 Merge conflict

In rare cases, you and the teaching staff will make changes to the same line of code in the same file (to minimize the risk of this, avoiding changing the instructions or problem statements in the

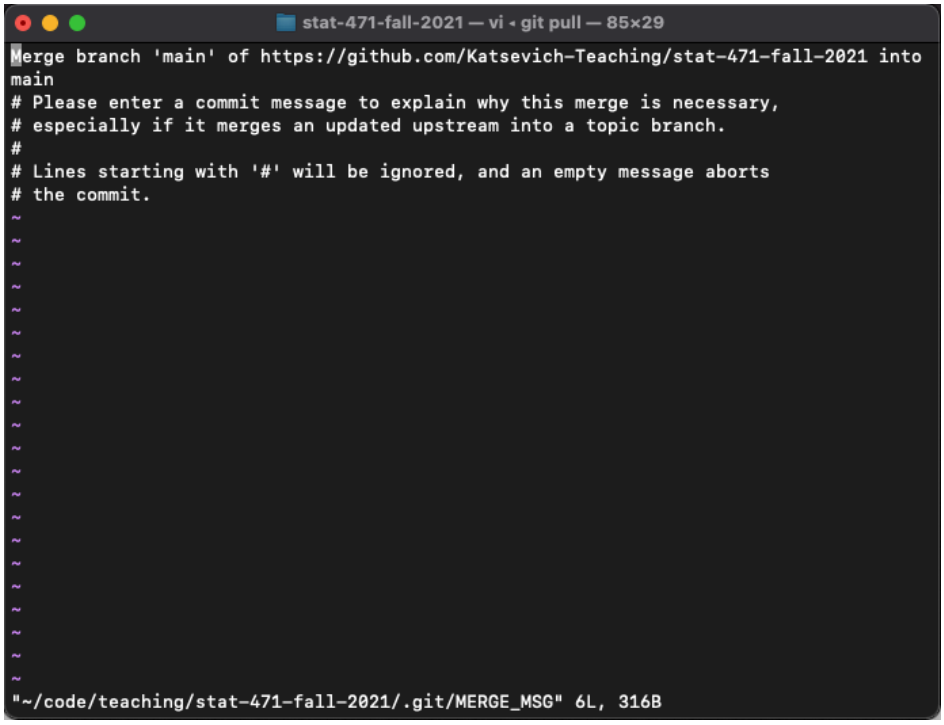

Figure 1: A message from Git about a merge conflict.

```
Auto-merging homework/homework-1/HW1.Rmd
Merge made by the 'recursive' strategy.
 homework/homework-1/HW1.Rmd | 2 ++
 1 file changed, 2 insertions(+)
```

```
Auto-merging homework/homework-1/HW1.Rmd
CONFLICT (content): Merge conflict in homework/homework-1/HW1.Rmd
Automatic merge failed; fix conflicts and then commit the result.
```

homework). In this case, you will have a merge conflict on your hands. You will be made aware of this by a message in the Terminal like the following:
In this case, Git cannot automatically do the merge and you will need to do so manually. In order to do this, open the file(s) where the conflict occurred (you can get a list of all conflicted files by typing `git status`). You will see in the file some weird parts that look like Figure 2. The text between `<<<<<<<< HEAD` and `========` is the version of the file that you created and the text between `========` and `>>>>>>>> 4fb72...` is the version of the file the teaching staff created. Edit the file to obtain the version you'd like to keep, and then commit your changes as usual to signal that the merge is complete.

8 If there are still issues, post on Piazza or come to office hours

The teaching staff are here to help you, so let us know if you are having issues.

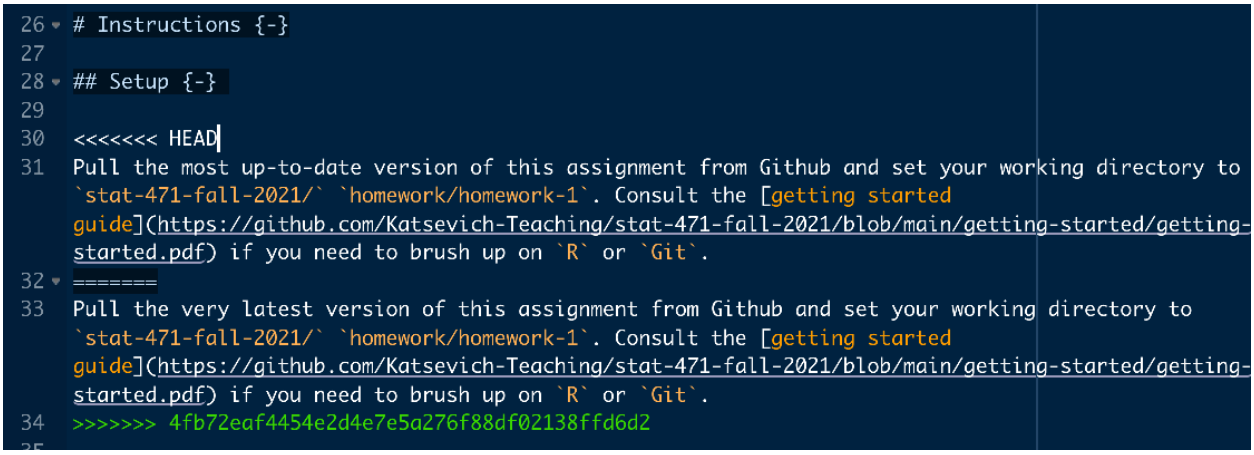

Figure 2: A file with a merge conflict.