

First Steps with your Cloned Repo: Pull, Commit, Push

Helene Wagner, University of Toronto

This tutorial assumes that you:

- have been invited to join a Github project,
- have installed git on your computer and made it work with RStudio, and
- have cloned the repo to your computer as an R project with version control.

If that's not the case, please see: [Installation](#).

Ok, you did your first "Pull" and "Push" in the previous tutorial on how to clone a repo, and nothing really happened. Here we'll start making things happen. The key is that you (i.e., your local R project) and your team (i.e., the remote repo) need to tango together to make this work. We'll start simple and add adrenaline as we go. In this tutorial, you'll be adding your own new file (i.e., no risk to hurt anyone).

As a beginner, you want to focus on these principles:

- Tackle that learning curve - it may all feel foreign at the beginning but it will get better.
- Make your first steps in a safe way so that you don't end up altering work by others.
- Avoid merge conflicts (unless you really are craving that adrenalin kick).

1. Create an R Notebook

- Always start your session with "Pull". Find the "git" tab again in your R project and pull.
- Create a new file: "File" > "New File..." > "R Notebook".
- A nice thing is that the R Notebook already contains some sample information:
 - A header (in yaml language)
 - Text (in white background, in R markdown language)
 - One chunk of R code (on a grey background)
- The default name is "Untitled1".

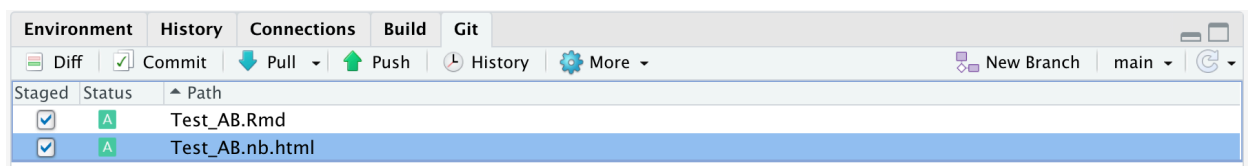
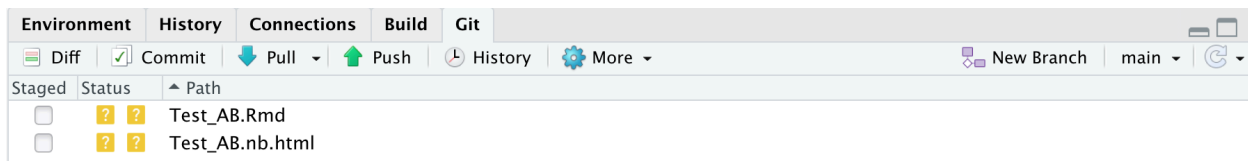
2. Save the file with another name

- Select "File" > "Save as..."
- Notice where R suggests that the file be saved. This should automatically default to the project folder. This is because in an R project, this folder is considered the "root", which means that it is the default location where R will look for files or where it will save them, unless you specify a different path.
- Save the file, using the ".Rmd" suffix. E.g., "Test_AB.Rmd", where "AB" are your initials - just for now to mark that this is your own test file and you can easily identify it again.

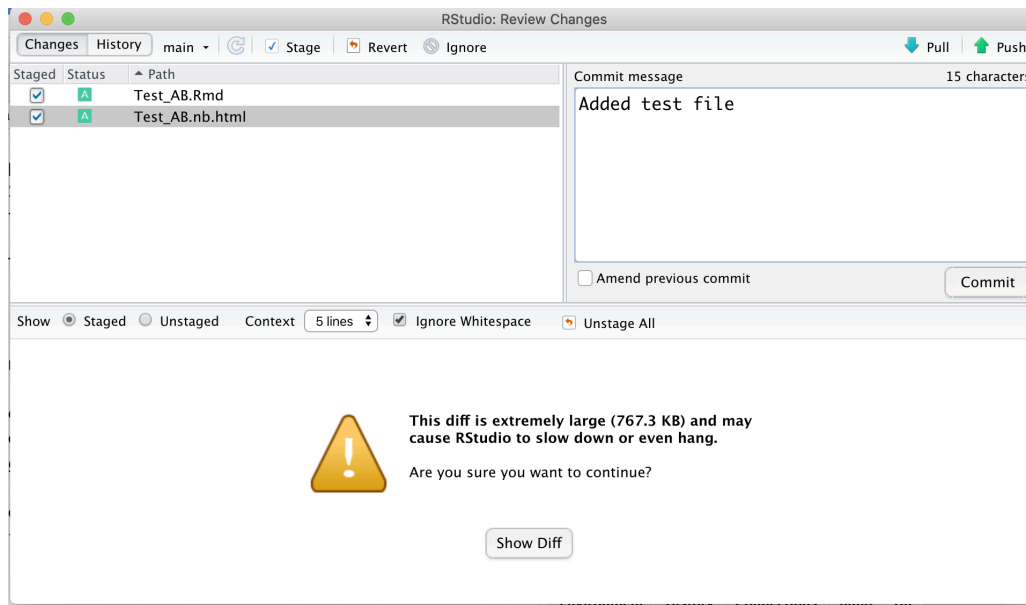
- In RStudio, find the "Files" tab. The file should now show up there with the new name.
- Most likely, a second file has been generated automatically: "Test_AB.nb.html". Click on it and select "View in Web Browser". This is a rendered ("knitted") version of the R Notebook.

3. Make your first commit

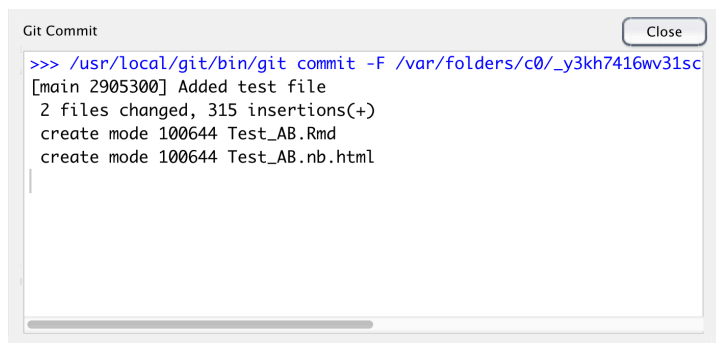
- Check out the "git" tab. Now the two new files should show up there. Time to commit - we want to keep the git tab as empty as we can.
- Check the boxes next to the file names, in the column "Staged". The status symbol will change to "A" (short for "Add").
- This means that you are telling git to include these files in your upcoming commit (i.e., you are "staging" them).



- Click on "Commit".
- Add a message that briefly describes the main change. The main point of this is so that you (or your team) can remember where a change was made if you need to revert it back. What text would help you find it again?
- Don't worry about the message "This diff is extremely large". As soon as you have a knitted file in there, R will think it is large.
- With "Diff", they mean the sum of all the changes in this commit. If you want, you can click on "Show Diff" to see the actual changes.



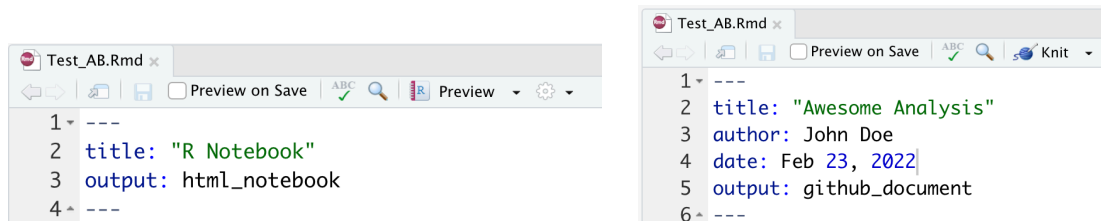
- Now click on "Commit" in the "Review Changes" window. You should soon see something like the window below.
- This means that R (or git) has saved the changes locally. They are not yet shared with your team on the remote repo.



4. Make further changes and commit them.

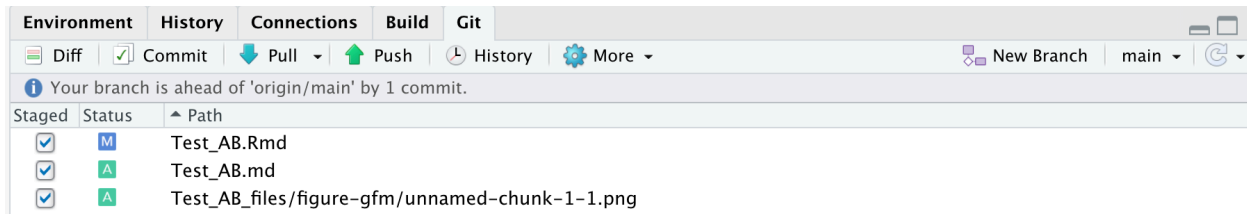
- Ok, this was nothing big yet. Let's make a few more changes before we let the world (i.e., the team) know.
- The "Test_AB.Rmd" file should still be open in the "Source" tab. If not, find it in the "Files" tab and open it.
- Change the header information. Insert your name, change the document title (this is not the file name but the title) and date. Change the document type ("output") that should be created to "github_document" as shown below (left: before, right: after).
- Save (use the menu or click on the floppy disk symbol).
- Note that this changed "Preview" to "Knit"!

- Click on "Knit". The tab "Render" will appear where the "Console" is, and some text in red and black will say that it is processing the file. If all goes well, you will see "Output created: Test_AB.md" at the end.
- Check the "Files" tab: instead of "Test_AB.nb.html", there is now a file "Test_AB.md". If you click on it, this one will open in RStudio, not in your browser, and it is nothing to write home about (yet). However, it is a great file format for Github because it will appear rendered on Github. Let's verify this in the next step.

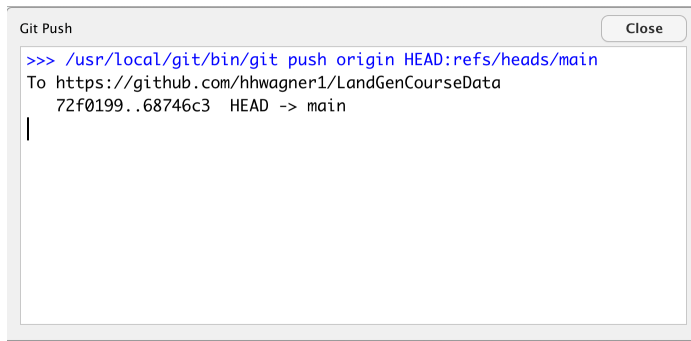


5. Share your change with the team: push to remote repo.

- Find the "git" tab and stage the files to be included in the commit.
- Notice what has changed:
 - The previously committed file "Test_AB.Rmd" has status "M" (for modified).
 - There are two new files: "Test_AB.md" and a ".png" image file associated with it.
This is the figure created in the grey R code chunk. Obviously they belong together, thus stage this file, too.
- Go ahead and commit your change. What commit message are you going to write?



- Once you've committed the changes, the "git" tab should again be empty.
- Now is a really good time to "Pull". I.e., check if anyone else has made changes in the meantime that might conflict with your changes. Pulling now can save you a lot of trouble (also known as merge conflicts).
- If you get the "Already up-to-date" message, you are good to go.
- Click "Push"! Deep breath... You should get a message like the one below (sorry, the example shows a different project folder).

A terminal window titled "Git Push" with a "Close" button in the top right corner. The terminal shows the command `>>> /usr/local/git/bin/git push origin HEAD:refs/heads/main` and its output: `To https://github.com/hhwagner1/LandGenCourseData`, `72f0199..68746c3 HEAD -> main`, followed by a cursor on a new line.

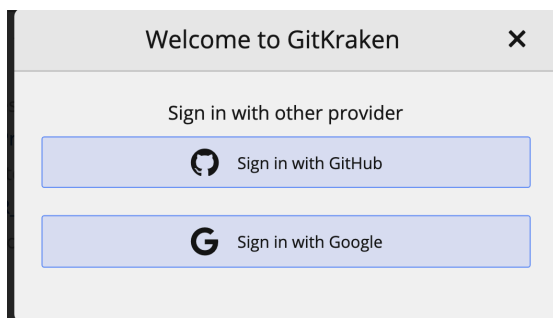
```
>>> /usr/local/git/bin/git push origin HEAD:refs/heads/main
To https://github.com/hhwagner1/LandGenCourseData
72f0199..68746c3 HEAD -> main
|
```

6. View your file on GitHub

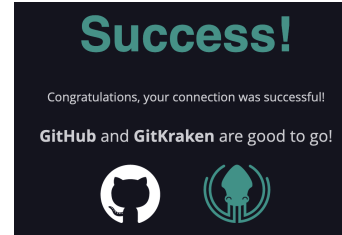
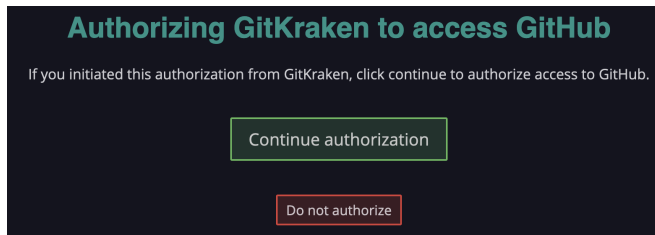
- In your web browser, go to the remote repo on GitHub. To get there, you can either use the URL you were given, or you log into your Github account and look for the list of your repos.
- Find the file "Test_AB.md" and click on it. There it is, the Awesome Analysis!
- Go back to the list of files and click on "Test_AB.Rmd". This is the code that produced the knitted ".md" file. Which one will be more fun to discuss at your next team meeting? You'll probably understand why we recommend creating the ".md" file for collaborative data analysis with GitHub.
- Notes on using ".html" vs ".md" files:
 - If you primarily want to share and discuss your R Notebook with people outside of your GitHub team, then knitting to .html is a better idea.
 - Also, certain types of interactive figures (e.g. interactive maps with "tmap") may not show up in ".md" files.
 - The ".html" files can't be viewed directly on GitHub, they need to be downloaded and opened in a web browser.
 - You can share ".html" files via email, and the recipients don't need to use R to read them.

7. View your history in GitKraken

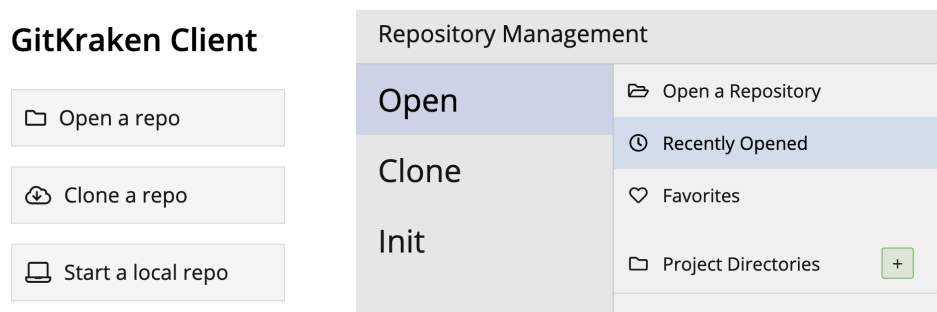
- Open GitKraken.
- Use the option to sign it with GitHub!



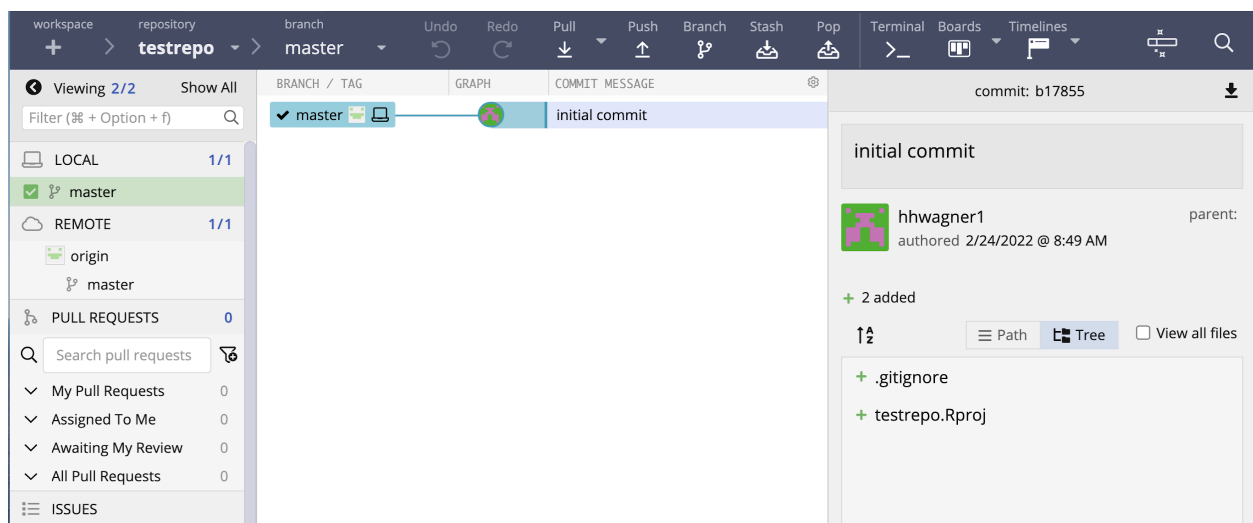
- This will open a page in your browser.
 - Click "Continue authorization"
 - You should then see the "Success" message
 - Return to GitKraken



- In GitKraken, click on "Open a repo".
- Under 'Open', click "Open a Repository" (if this is the first time). Navigate to the project folder in your file system, click "Open".
- When returning, you should see the repo listed under "Recently Opened".



- As a minimum, it will look something like this. However, your group's project may have a much more extensive history. We'll look at this in the next tutorial.



8. Keep going

- You just learned the basic workflow for using GitHub as version control for your coding .
- Practice a few times before taking it to the next level with the next tutorial.
- Watch how the history changes on GitKraken.
 - What happens when you add a file, before you commit it?
 - What happens when you commit?
 - What happens when you pull?
 - What happens when you push?

9. Summary of best practices: (not based on research, just our two cents)

- Commit often, and with commit messages that will allow you to find things later.
- Push when you've done a few commits and completed a minor task.
- Before you push, commit all changes and pull (one day you'll know why we said so).
- Consider using R Notebooks and knitting them to GitHub documents to make it easy for team members to see what is going on in your piece of code.
- Break up your code into chunks and annotate them (similar to the worked examples) so that team members (or your future you) can understand what's going on and why.