

Summary Workshop - Intro to GitHub in RStudio

J. N. Tendeiro

14 May 2020

Good reads

I started by showing two online sources that I found particularly instructive:

- [Hadley Wickam's](#) tutorial is very clearly written. Take a small hour to go through it; it pays off *very* quickly!
- [Happy Git with R](#) was a nice surprise from Google (thanks G!). Very useful to set things up and run some basics. As you progress the reading becomes increasingly thick, so feel free to stop at a point where you feel it's going over your head (this happens to me all the time, don't feel embarrassed).

Set up your system

This is what you'll need:

- Install [R](#).
- Install [RStudio](#).
- Install [Git](#).
- Set up the SSH keys (see [Happy Git with R](#), [section 11](#)). This will save you time and patience by not typing your credentials *ad nauseam* each time you push things to GitHub.

Now open RStudio. Make sure it has access to R: Your console (low-left corner) should look like [Figure 1](#).

Is Git installed and can RStudio find it? Check it ([Figure 2](#)).

Then, set up Git initially. You only do this *once* per computer. In RStudio, go to **Tools > Shell** and add your name and email using the two commands below:

```
git config --global user.name "YOUR FULL NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
```

Then test whether it worked out ([Figure 3](#)).

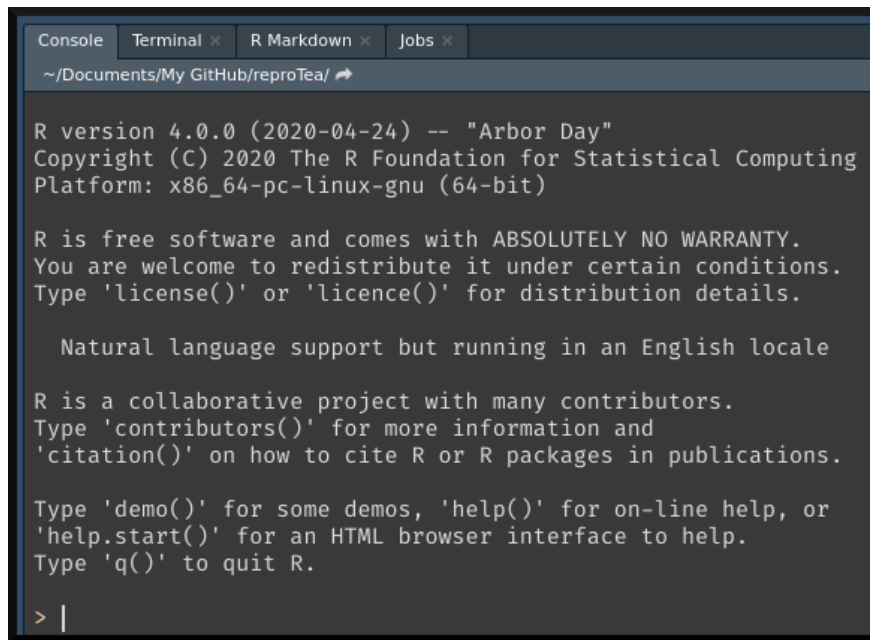
Did you set up the SSH keys correctly? You can quickly check it by going to **Tools > Global Options > Git/SVN** and make sure that the last field, **SSH RSA key**, is filled in ([Figure 4](#)). Also, remember to copy the public key to GitHub via the browser!

Finally, and if not done yet, create a [GitHub account](#). Remember to use the same email as the one you used above when setting up Git.

Phew! At this point you are *good to go*!!

Create a new repository in GitHub

I always create a new repository in GitHub first and then bring it to RStudio (the other way around is also possible but I don't usually do that). So login into GitHub, press on **Repositories** on the top panel, and then press the green **New** button on the top-right ([Figure 5](#)).



```
Console Terminal x R Markdown x Jobs x
~/Documents/My GitHub/reproTea/ ↗

R version 4.0.0 (2020-04-24) -- "Arbor Day"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

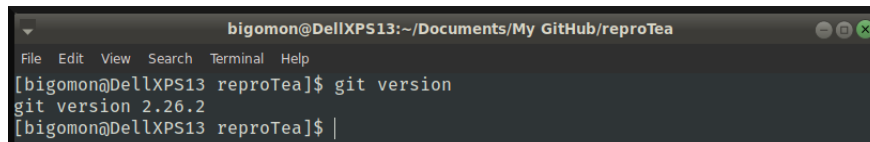
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

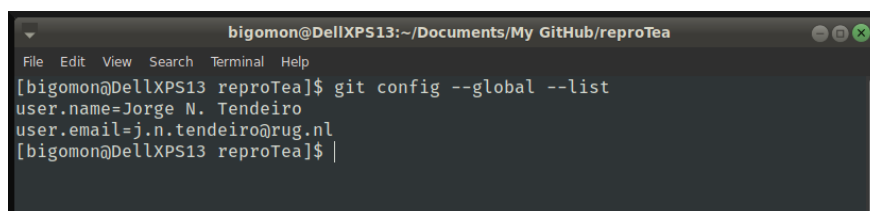
> |
```

Figure 1: Console showing R.



```
bigomon@DellXPS13:~/Documents/My GitHub/reproTea
File Edit View Search Terminal Help
[bigomon@DellXPS13 reproTea]$ git version
git version 2.26.2
[bigomon@DellXPS13 reproTea]$ |
```

Figure 2: Are you there, Git?



```
bigomon@DellXPS13:~/Documents/My GitHub/reproTea
File Edit View Search Terminal Help
[bigomon@DellXPS13 reproTea]$ git config --global --list
user.name=Jorge N. Tendeiro
user.email=j.n.tendeiro@rug.nl
[bigomon@DellXPS13 reproTea]$ |
```

Figure 3: Initial Git setup.

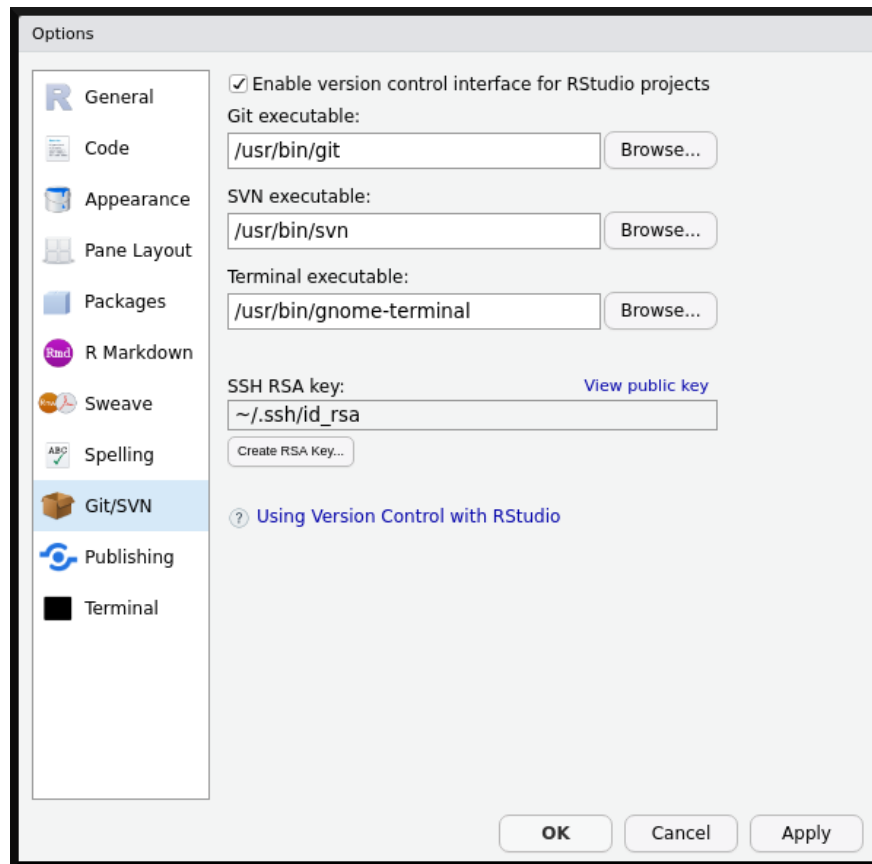


Figure 4: SSH keys.

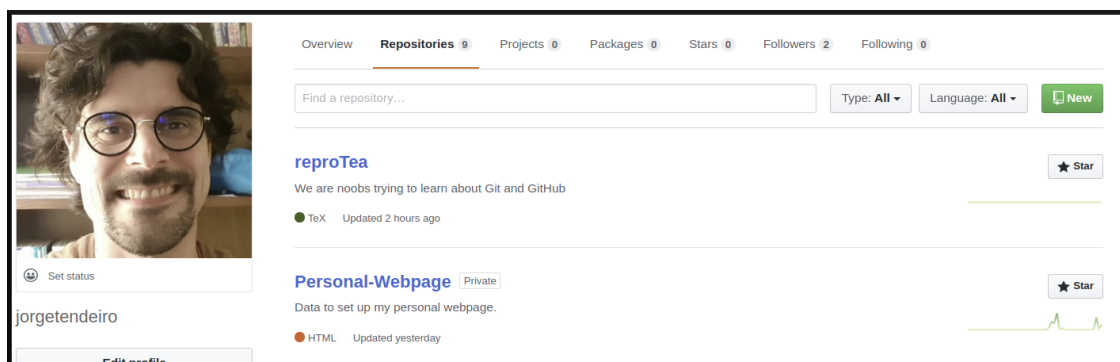
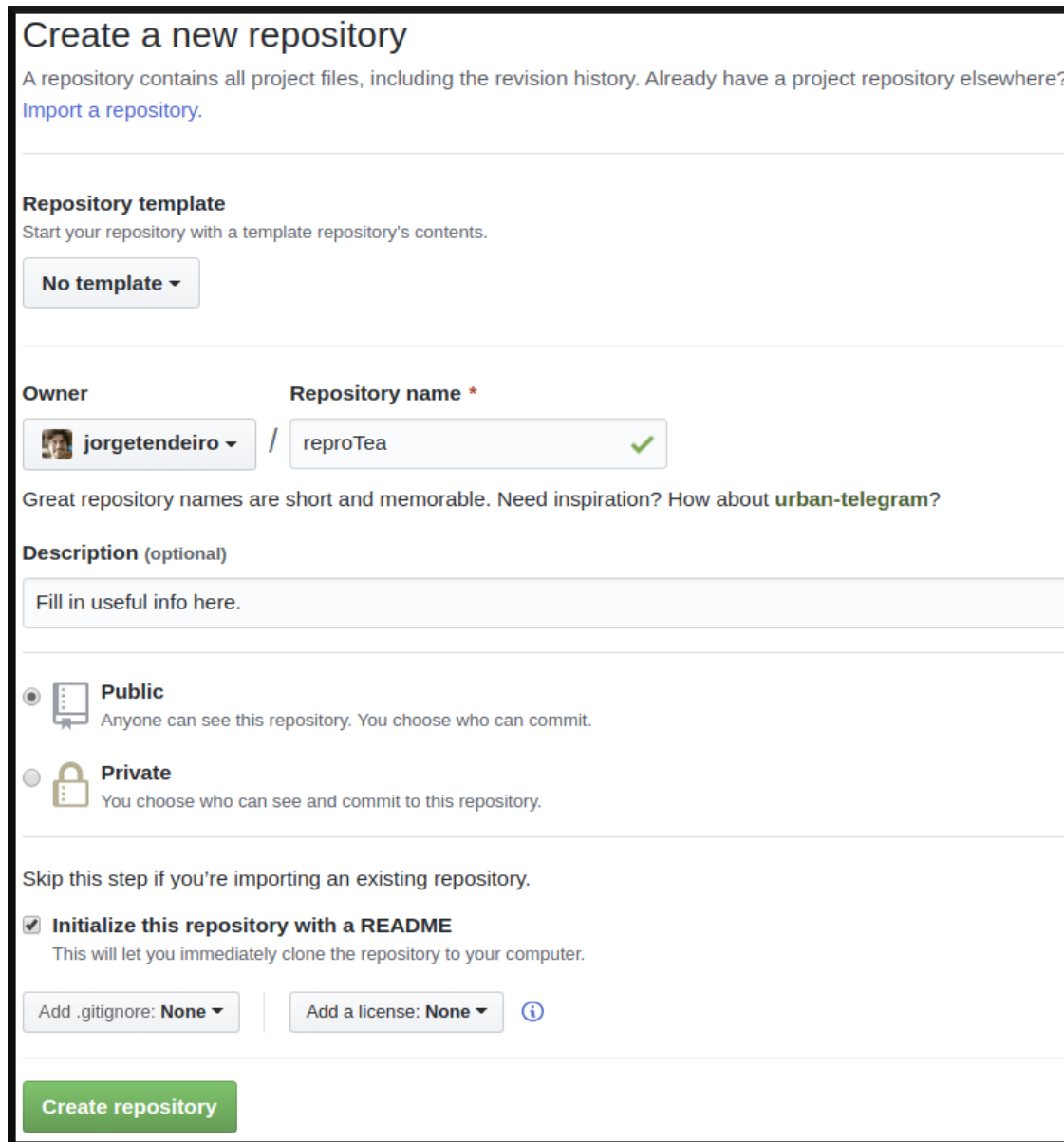


Figure 5: Create a new repository in GitHub.

Here, fill in the name for the repository under **Repository name** (in our workshop I used `reproTea`) and add a small description under **Description**. Choose whether the repository should be public or private. Finally, do check the option to initialize the repository with a README. When you are done, press the green **Create repository** button. See [Figure 6](#).



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a link to 'Import a repository'. Below this is the 'Repository template' section with a 'No template' dropdown. The 'Owner' is set to 'jorgetendeiro' and the 'Repository name' is 'reproTea', which is marked with a green checkmark. A note suggests repository names should be short and memorable, with a link to 'urban-telegram?'. The 'Description (optional)' field is empty. Under 'Visibility', the 'Public' option is selected. A note says 'Skip this step if you're importing an existing repository.' The 'Initialize this repository with a README' checkbox is checked. At the bottom, there are dropdowns for '.gitignore' (set to 'None') and 'Add a license' (set to 'None'), followed by an information icon and a large green 'Create repository' button.

Figure 6: Details in creating a new repository in GitHub.

The project is now created in GitHub. You will see something akin to [Figure 7](#), but yours should only show the README file whereas mine shows all the files that we created throughout the workshop.

Clone the repository

Now you can *clone* the project onto your computer. To do that, press the green button called **Clone or download** (see [Figure 7](#)) and copy the URL under the **Use SSH** option (at this point I am assuming you set up the SSH keys correctly).

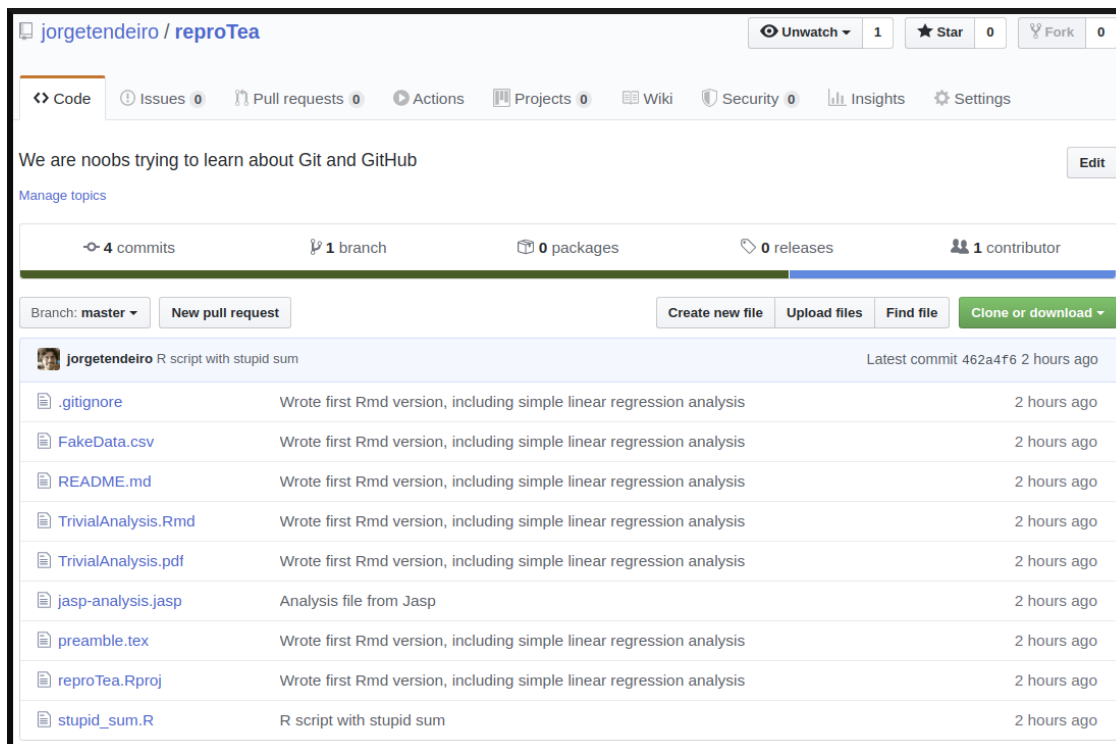


Figure 7: Repository page on GitHub.

Next, go to RStudio (top-left menu) and click on **File > New project ...**. In the **New Project** window, choose **Version Control**. In the **New project** window, choose **Git**. Finally, paste the URL that you copied from GitHub into the first field called **Repository URL**. Observe that by doing that the second field (**Project directory name**) is filled out automatically. Finally, make note of the location on your disk of this repository by looking at the last field called **Create project as subdirectory of** (change this location if desired via the **Browse** button). Finally, press **Create Project** (Figure 8).

Work on your project

Now it's time to get your hands dirty. Work on whatever files you want. Whatever files you create within RStudio (.R, .Rmd, .tex, .py, etc.) are automatically saved in the correct working directory (check the low-right corner in RStudio under the **Files** tab; my working directory is `Home/Documents/My GitHub/reproTea`, see Figure 8). As soon as a file in the working directory is added or modified, Git makes notice of it right away. You can see that by looking at the right-top panel of RStudio, under the **Git** tab. For instance, as I type these notes, I created an RMarkdown file called `Summary_Workshop.Rmd`. It knits into a PDF file with the same name. And, all the images you see in this document are screenshots that I placed inside a folder called `images`. I still didn't commit these changes, so Git is listing these files as being new or having been updated (Figure 9).

In case you work on files outside RStudio, *just make sure to save them all in the working directory*. RStudio will show all such files (low-right corner in RStudio under the **Files** tab), and Git will also notice them (right-top panel of RStudio, under the **Git** tab). So, feel free to e.g. use Jasp on a data file and save all files, or write a .docx file, or whatever. All such files are Git'able.

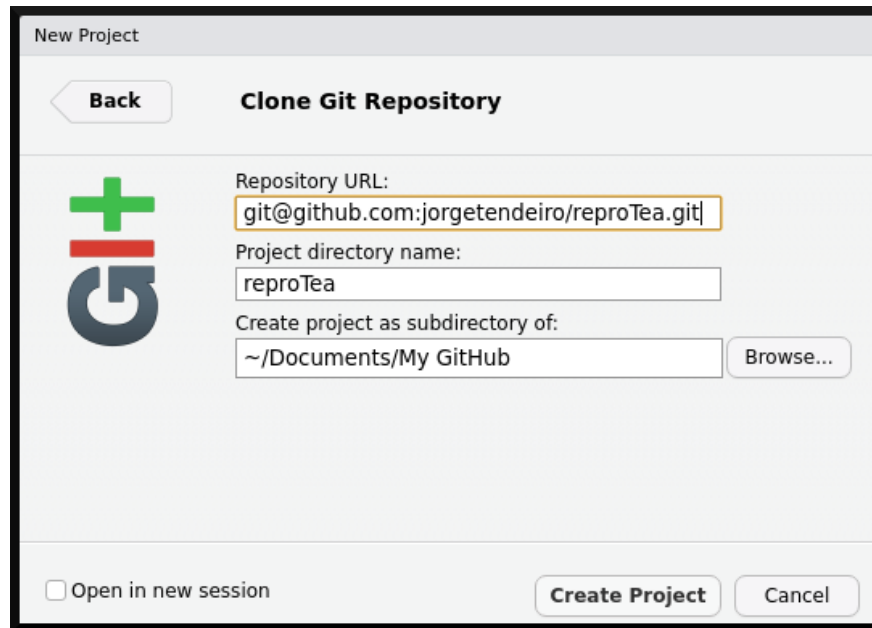


Figure 8: Import GitHub repository into RStudio.

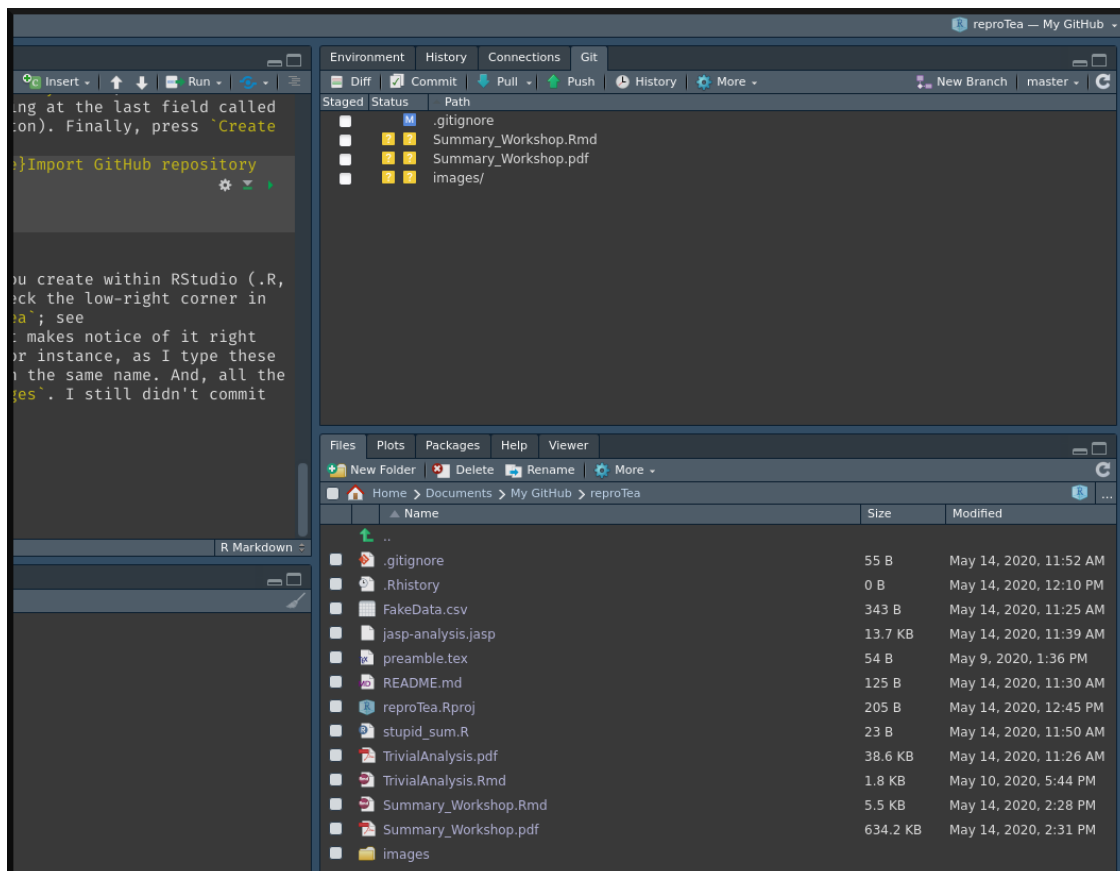


Figure 9: Git in RStudio keeping track of the files that suffered changes (top-right).

Stage, commit, and push

Stage

Now let's use Git (we still haven't, by the way). Please focus on the right-top panel of RStudio, under the Git tab. To **stage** the files (i.e., to select what files you want to keep track), check them under the **Staged** column; see Figure 10. In case you have a lot of files, try doing `Ctrl+A` to select all files at once and then check-mark one of them; all will be checked in one go.

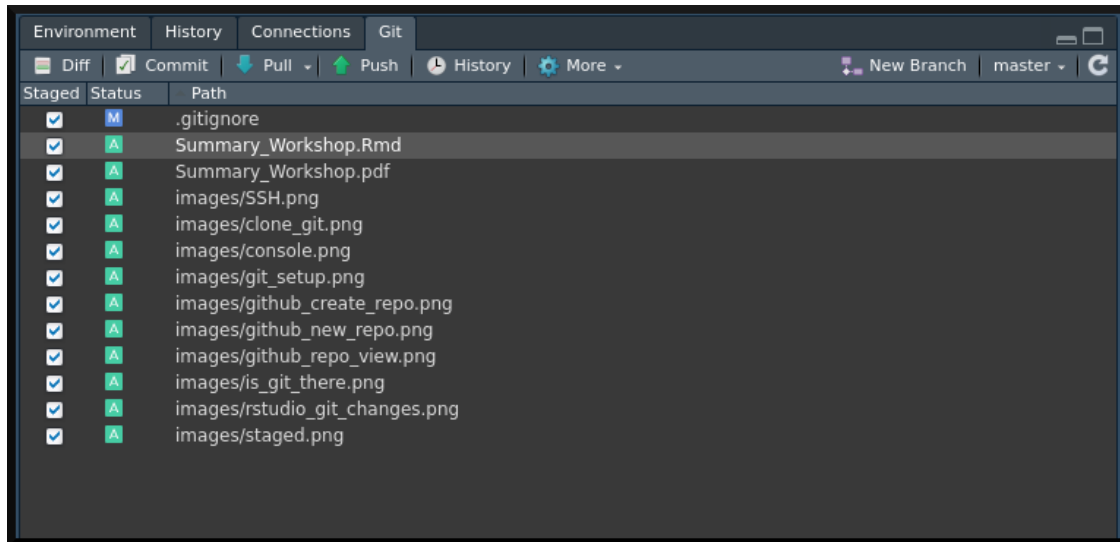


Figure 10: Staged files in Git.

Commit

Next, press the **Commit** button (Figure 10); a new window opens (Figure 11). Here, you can see the changes that the files suffered. This at least works for text files like the .Rmd that I am typing on now (it won't work for .png images, for instance). Green color means new text, red color means erased text. I typically don't look at this, to be fully honest. Anyway, we will now *commit* this version of the repository. In wording, we will create a snapshot *in your local PC* of the current status of all files of the repository. I need to stress this again: This is a *local* backup of the current state of the project. This will not be seen by GitHub just yet (for that we do need to *push*, see below). Give the commit a sensible short description of what you did. Then press the **Commit** button (Figure 11).

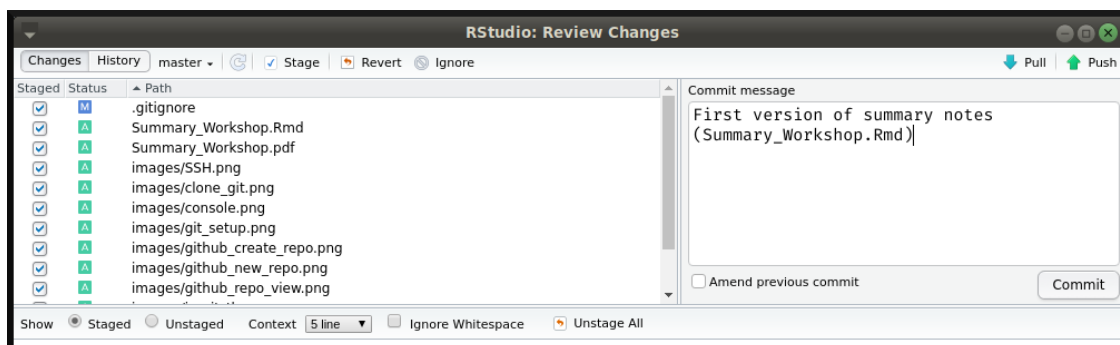


Figure 11: Committed files, done locally.

Afterwards, you will see that the files version on your PC is now more recent than the version on GitHub.

This is what the message Your branch is ahead of 'origin/master' by 1 commit means (Figure 12).

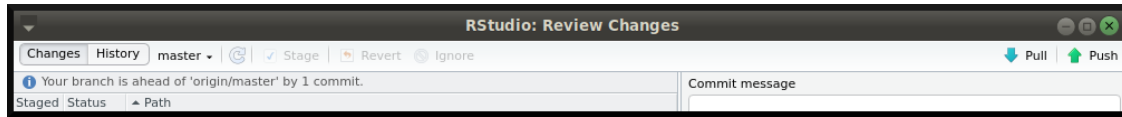


Figure 12: After doing the commit.

You can do as many of these (local) commits as you feel like. I suggest you do a commit each time you reach a certain milestone (after finishing a section, or after editing a particular file). The commits don't take a lot of space, and they will keep a nice story of the development of the files.

Whenever you are ready to upload all the commits to Github, you will then use the push option.

Push

Simply press the push button, to be seen on the right-end of Figure 12. Because we set up the SSH keys, you won't need to fill in the username and password at this point. Just wait a few seconds and it will be over (Figure 13). Then you can press the Close button.



Figure 13: Pushing the changes to GitHub.

And now, go back to GitHub and admire in wonder! See Figure 14.

Final advice

I suggest that at the end of each working session you always commit and push to Github.

Then, whenever you open RStudio and start working again, you always start by *pulling* the latest version from GitHub. Just go to the right-top panel of RStudio, under the Git tab, and press the Pull button (the button is to be seen, e.g., in Figure 9). I do this all the time because I work on different computers (in my office, laptop, home desktop). By keeping the latest version in GitHub, I can resume work on the latest version from any of my machines, without worries of mismatch of versions. Plus, if any of my computers dies, the code is all safe in GitHub!

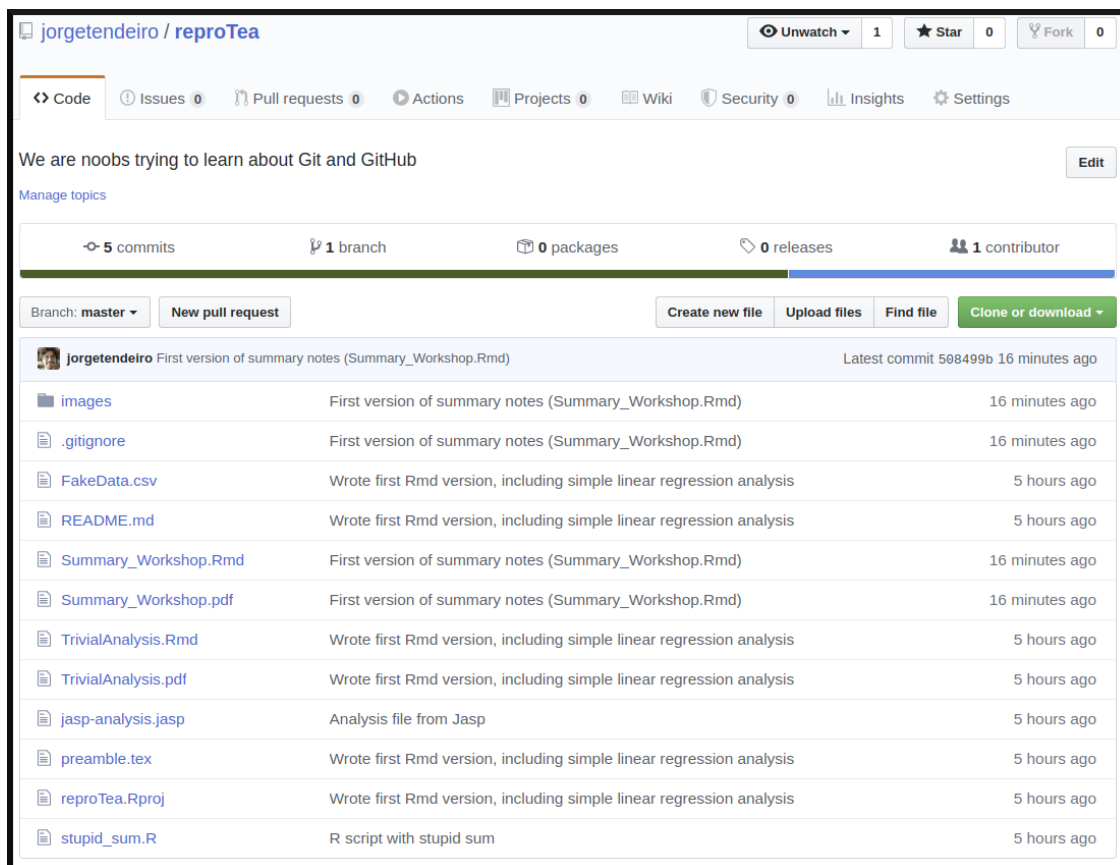


Figure 14: GitHub is now up to speed. In my case, see the files associated to the commit called ‘First version of summary notes (Summary_Workshop.Rmd)’.