Rstudio and Github — Sharing your R-code with the World

Reader Accompanying the Course Reaction Transport Modelling in the Hydrosphere

Lubos Polerecky and Karline Soetaert, Utrecht University

July 2021

Abstract

Once you start working on a code or program with several people, it becomes worthwhile to invest some time into learning about the tools that are available for sharing, editing and tracking changes in such codes. Here, we briefly illustrate how to work with Github in Rstudio.

1 Getting started with git

Git is a free and open source distributed version control system. To start sharing a code via *git*, you need to install *git* on your computer, and choose a version control system manager. There are several options; in this reader we will focus on GitHub.

1.1 Install git on your computer

Download *git* from https://git-scm.com/downloads for your operating system (e.g., `Git-2.32.0.2-64-bit.exe`, or some newer version, if you are a Microsoft Windows user), and install it.

During the installation process, select the default option for most questions (unless you know what you are doing), except:

- select *main* when asked about the name for the initial branch of a new repository;
- choose *none* when asked to *Choose a credential helper*.

If you forget to disable the credential helper, proceed as described here:

credential manager for windows

1.2 Create a GitHub account and project

Follow the steps on https://github.com/ to create an account.

Create a new repository. Make sure that you set up your repository as *private*, so that only you and a closed circle of invited developers will be able to see it. Then, choose *Settings* → *Manage access* → *Invite teams or people* to add developers to the project.

Note the URL of the repository. In most cases it will be in the form:

https://github.com/YOUR_USERNAME/YOUR_PROJECT.

2 Using git within Rstudio

Before you can actually work on the project, you need to add the project locally on your computer. To do this in *Rstudio*,

- choose from the menu File → New Project → Version Control → Git → Repository URL, and
- paste the URL of your project, e.g., **https://github.com/YOUR_USERNAME/YOUR_PROJECT**.
- When choosing a working directory, do not choose a directory where you already have files you are working on. Move these files first to a temporary location.

You now have a local copy of the remote repository and are ready to contribute to the project development.

If you want to make a personal copy of an existing project on which you don't have a developer access, you have to *fork* it first to your own account, and start working from there.

- In the GitHub website, go to the repository and click *fork*.
- You can now go back to your own account, find the URL of this project and proceed as described above.

2.1 Normal workflow in Rstudio

Once you have set up the project in *Rstudio* to be developed with *git*, you will find a *Git* tab in the top-right panel of *Rstudio*. In this tab, you will see a number of buttons, including *commit*, *pull*, *push*, and *New Branch*. Additionally, you will see there a pull-down menu with the different development branches of the project (if multiple branches exist).

In *Rstudio*, the "normal" *git* workflow goes like this:

- At the beginning, select the *main* branch of the project from the pull-down menu with the (possibly several) branches of the project, and *pull* the code from the remote repository.
- It is assumed that the version of the project you just pulled is correct and working, and you want to develop it further. However, you do not want to potentially "mess up" the working project by your changes. Therefore, it is recommended that you first create a *new branch*. This is done by clicking the *New Branch* button in the *Git* tab, giving the new branch a name (we assume here the name *branch1*), and clicking *Create*.
- From this moment, be sure that you only modify your code in this newly created branch. That is, work on your local files in *Rstudio* as you would normally do without *git*, but be sure that the *branch1* is selected in the *Git* tab. Specifically, do not change branches and edit files there, unless you know what you are doing. You will avoid headaches later.
- During the development, regularly *commit* the changes you have made, so that your changes are locally tracked by *git*.
- Every now and then, *push* your changes on-line (after committing). When you do so, your local changes will be uploaded to the GitHub server and will thus be visible by other developers. They will see them locally on their computer if they *pull* from the server and choose *branch1* in their *Git* tab.
- Similarly, you will see changes in their branch if you *pull* the project from the remote server.
- If you made changes that you (and your developers team) believe should be kept, it is time to *merge* the development branch *branch1* with the *main* branch. This is done by selecting the *main* branch in the *Git* tab, and then by typing

  ```
  git merge branch1
  ```

in the *terminal* within *Rstudio*.[1]

- After this step, you need to *push* the *main* branch to the remote server, so that others can *pull* it and work from there.
- Finally, it is a good idea to *delete* the *branch1* after it has been *merged* with the *main* branch. This will keep your development less confusing. To do this locally, type in the terminal

  ```
  git branch -d branch1
  ```

  or

  ```
  git branch -D branch1
  ```

  To do this remotely, type in the terminal

  ```
  git push origin --delete branch1
  ```

  Alternatively, you can delete a branch from within the GitHub website. All you need to do is display an overview of all branches and click on the delete icon next to it.
- Occasionally, you will need to *prune* to keep things tidy on your computer and on the github repository. This is done by typing in the terminal:

  ```
  git remote prune -n origin
  ```

  to see what will be *pruned*, and then

  ```
  git remote prune origin
  ```

  to actually do the *pruning*. Effectively, this will remove traces of your now-removed branches from the pull-down menu in the *Git* tab in *Rstudio*.

These are the basics. Check intro git rstudio or search the internet for more information.

3 Useful tips

- When pushing your local changes to the GitHub repository, you will be asked for your password every time. If you want to modify this behaviour, check the instructions here:

[caching github credentials in git] (https://docs.github.com/en/get-started/getting-started-with-git/caching-your-github-credentials-in-git)

For example, to cache the credentials for 2 hours (7200 s), type in the terminal:

```
git config --global credential.helper 'cache --timeout=7200'
```

- Check this website for updates in the authentication protocols:

authentication for git

To configure ssh authentication for Github, type in the Rstudio terminal:

```
git config --global user.name "your_user_name"
git config --global user.email "your_email_address"
ssh-keygen -t rsa -C "your_email_address"
ssh -T git@github.com
```

[1] You can find the terminal in the lower-left tab of *Rstudio*. Do not confuse the terminal with the R-console.