



Git & RStudio : Practical training

Authors : Philip Ashton - Tristan Maunier

Session : 08/08/2018

Exercise 1.1 : Create Github account & Install git (RStudio)

Create Github account

GitHub is a remote web hosting service using git. GitHub has additionnal features support for collaboration, such as bug tracking, feature requests, or task management.

Note that there are other git-based hosting websites, such as GitLab, or BitBucket.

Go to <https://github.com/> and follow the instructions for creating and activating your account.

Install git on your machine

- You can verify if git is already installed on your machine:
To check whether or not you have git installed, simply open a terminal window and type "git --version". If git is installed you'll see a message like "git version 1.9.5.msysgit.0"
- Otherwise, for all operating systems, go to the Git downloads web site, and click on the appropriate icon for your operating system : <https://git-scm.com/downloads>

Install git on RStudio

- Open RStudio
- Click *Tools -> Global Options -> Git/SVN*
- If Git executable shows (none), click Browse and select the git executable installed on your system
- On a Mac, this will likely be one of
 - /usr/bin/git
 - /usr/local/bin/git
 - /usr/local/git/bin/git
- On Windows, git.exe will likely be somewhere in Program Files
- Note : If you are not sure where the git executable lives, try this in a terminal:
 - which git (Mac, Linux)
 - where git (most versions of Windows)
- Click OK

Exercise 1.2 : Make your own repository & set it up with RStudio

Create a new repository on GitHub

- Go to github.com and login
- Click the green "New Repository" button
 - Repository name : myrepository
 - Public
 - Check Initialize this repository with a README
 - Click the green "Create repository" button
- Copy the HTTPS clone URL to your clipboard via the green "Clone or Download" button.

Clone the new GitHub repository to your computer via RStudio

In RStudio, start a new Project:

- File -> New Project -> Version Control -> Git. In the "repository URL" paste the URL of your new GitHub repository. It will be something like <https://github.com/tmaunier/myrepo.git>.
- Decide where to store the local directory for the Project. Don't scatter everything around your computer - have a central location, or some meaningful structure. For repositories you create for your different projects, you can setup a /projects directory and clone all your repos there.
- I suggest you check "Open in new session", as that's what you'll usually do in real life.
- Click "Create Project" to create a new sub-directory, which will be all of these things:
 - a directory on your computer
 - a Git repository, linked to a remote GitHub repository *an RStudio Project
- **Whenever possible, this will be the preferred route for setting up your R projects.**
- This should download the README.md file that we created on GitHub in the previous step. Look in RStudio's file browser pane for the README.md file.
- Why setup your R projects this way?
There's a big advantage to the "Github first, then RStudio" workflow: the remote GitHub repo is now the "upstream" remote for your local repo. In essence, you are already setup to push and pull commits to GitHub. There is no need to set anything else up through the shell or a Git client.

Plan B: Connect a local RStudio project to a GitHub repository

Sometimes you cannot always setup the GitHub repo first, or you already have an RStudio project you need to connect to a GitHub repo. This workflow is the reverse of the above and cannot be executed from within RStudio.

- Create a new RStudio project: File -> New Project -> New Directory -> Empty Project.
- Directory name: `my repo` (or whatever you named the GitHub repo)
- Decide where to store the local directory for the Project.
- YES check 'Create a git repository'
- I suggest you check "Open in new session", as that's what you'll usually do in real life.

- Click “Create Project” to create a new sub-directory, which will be all of these things:
 - a directory on your computer
 - a Git repository - note, linking to github this isn’t automatic, we still need to link it up
- Initiate the “upstream” or “tracking” relationship by adding a remote. Go to Tools -> Shell and do this, substituting the HTTPS URL for your GitHub repo.


```
git remote add origin https://github.com/tmaunier/myrepo.git
```
- Download all the files from the online GitHub repository (possibly just README.md, at this point).


```
git pull origin master
```
- Cement the tracking relationship between your GitHub repository and the local repo by pushing and setting the “upstream” remote:


```
git push -u origin master
```

Exercise 2.1 : Basics git commands - Make local changes, save and commit

Do this every time you finish a valuable chunk of work, probably many times a day !

- From RStudio, create a new text file, File -> New File -> Text File. Write anything you want into it and save.
- It’s time to commit these changes to your local repo. How ?
- From RStudio:
 - Click the “**Git**” tab in the upper right pane.
 - Check the “**Staged**” box for any files whose existence or modifications you want to commit.
 - To see more detail on what’s changed in file since the last commit, click on “**Diff**” for a Git pop-up. Look at both the "Changes" tab and the "History" tab.
 - If you’re not already in the Git pop-up, click “**Commit**”
 - Type a message in “Commit message”. This should describe for a human what modifications you’ve made to the staged files.
 - Click “Commit”.
- Repeat the processus several times and take a look again at the commits history, you can have an idea about how version control is powerful.
- Pay attention to the additions and deletions between two commits.

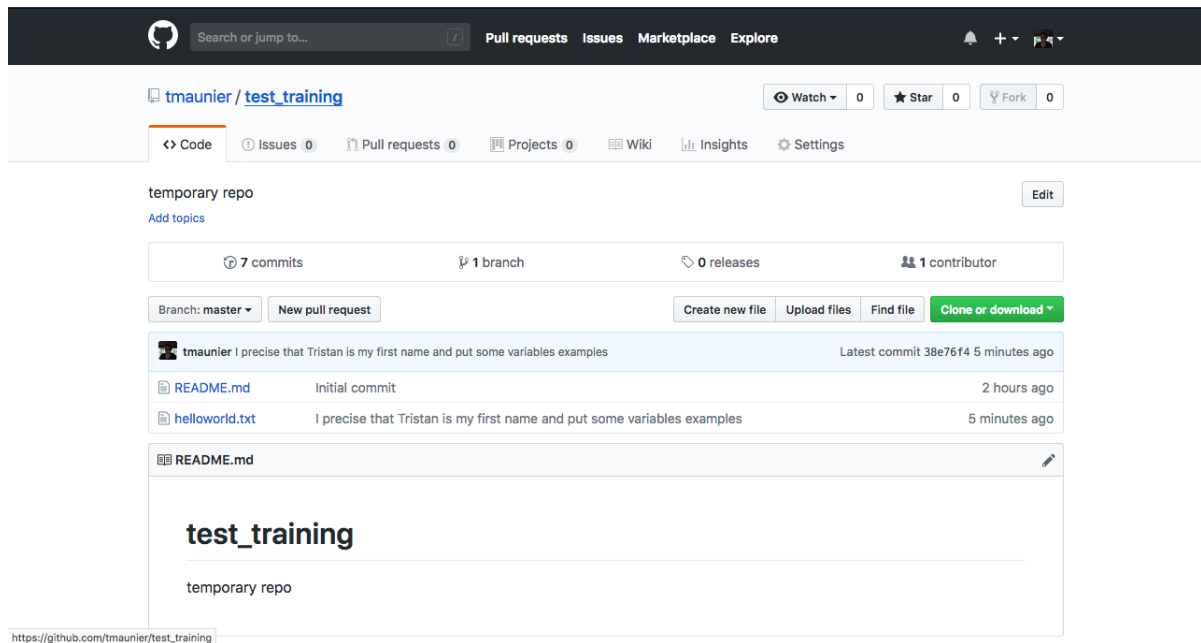
Exercise 2.2 : Basics git commands - push to distant repo (Github)

Do this a few times a day, but possibly less often than you commit.

- Once you committed some changes on your local repo, it’s time to update your distant repo. Now that you’ve obviously understood the whole theory of git, you had to guess, it’s time to **push** your commits to your remote GitHub repository. How ?
- Before you push your changes to GitHub, first you should pull from GitHub. Why? If you make changes to the repo in the browser or from another machine or (one day) a collaborator has pushed, you will be happier if you pull those changes in before you attempt to push.
- Click the blue “**Pull**” button (blue down arrow) in the “Git” tab in RStudio. I doubt anything will happen, i.e. you’ll get the message “Already up-to-date”. This is just to establish a habit.
- Now click the green “**Push**” button to send your local changes to GitHub.

Exercise 2.3 : Become familiar with the GitHub graphical user interface

- Open your project page on Github (<https://github.com/youraccount/yourrepo>), it should look like this:

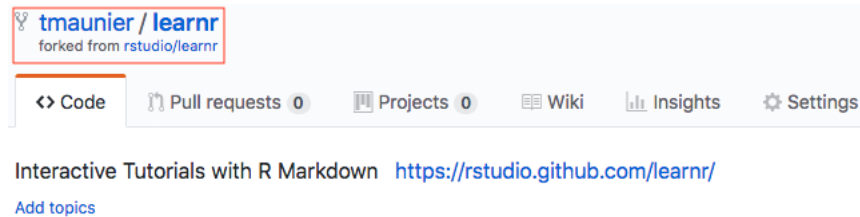


- As you can see there are all your files stored in your repo, you can read and modify them from this interface and commit changes exactly as through RStudio.
- You can also look at every commits that have been done on your files, it will be ordered by date.
- If you click on one specific commit, every information related to this commit will appear on your screen (additions/deletions, commit's parent, commit's sha, who committed and when, ...)
Note that here you can see only the difference between the current commit and his parent.
- GitHub's graphical interface offers so much more useful features, I let you discover it on your own if you want to go further.

Exercise 3.1 : Advanced features - Fork a repository

It's not a secret for you anymore, GitHub is a collaborative tool and you can have access to every public repository existing on GitHub. The act of forking a repository means that you'll copy this repository on your account. Every commits, branches and files are included into this copy.

After a fork, you can initiate a new project on your local side with RStudio, pull this repository from GitHub to your project and work directly with all those files.



Exercise 3.2 : Advanced exercise - Practice with code

Scenario: you have been using a published R package which is on GitHub, you discover a bug while using it and you want to fix it for yourself and others working with it.

- From your Github account:
 - Fork this repository : https://github.com/tmaunier/git_training
- From RStudio:
 - Clone the repo to local
 - Follow the following path
git_training -> codes -> JEV_indo_Fer.R
open the file in RStudio.
 - Find the error and fix it (write something random)
 - Update your remote repository with the right version of the file
- From Github:
 - Click on the "New pull request button", select branches of interest. You should be able to see the difference between the original repo and yours.
 - Send a pull request and explain what is the point of this request, give some arguments.

Exercise 3.3 : Advanced features - Branches

Scenario: Imagine that you have a working version of your code, and you want to continue to use this working version while you add some advanced new features.

You can do this with branches. We will leave the 'master' branch intact and working while we modify and potentially break the 'feature' branch.


- From Github:
 - Create a new branch, Click on the branch tab and write the name of the new branch you want to create and click "Enter".
- From RStudio:
 - Pull your distant repo, it will specify that a new branch has been created. Then change your branch to the new one thanks to the branch button on the right of Push and Pull buttons.
 - Make some changes on a file, commit and push.
- go back to Github:
 - You'll see a message on your project's home page, it says that a commit has been done on a secondary branch and it offers you the possibility to compare and pull a request.


Your recently pushed branches:


🔗 **training_test_branch** (less than a minute ago)


🔗 **Compare & pull request**

- If you click on this button, you are able to look at the differences between two branches and also to ask for a **pull request**.
- If you want to **merge** your secondary branch with the master branch i.e. to apply the changes on the main branch of the project. Send a pull request with a message, then GitHub will automatically verify if there are conflicts between the branches.

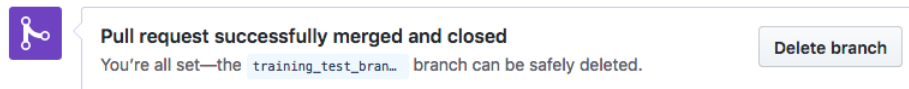


**Continuous integration has not been set up**
Several apps are available to automatically catch bugs and enforce style.

**This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- After a successful merging to the master branch, GitHub note that you can safely delete your branch.



- Note : In case of merge conflicts, you have to choose between the different versions which is the right one. You only have to delete the wrong version and push the modifications. If you are collaborating on a project and your version of a file has a conflict with that of someone else, be human talk with that person and choose between your two versions which is the best.
- On your Github repository's home page, go to the 'Insight' tab and look at the network graph ('Network' tab), you'll have the history of your branches over the time, including the different commits and the merge steps.
- From RStudio:
 - change your local branch to master, and pull again. Changes you made in the secondary branch will appear now on the master branch.
- Note : Usually pull requests are used for collaborative work, indeed pull requests let you tell others about changes you've pushed to a GitHub repository. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.
In the case of merging branches you just send a pull request to yourself and use GitHub's tools to verify the state of your branches.

References :

- https://jennybc.github.io/2014-05-12-ubc/ubc-r/session03_git.html
- <https://cfss.uchicago.edu/git05.html>