

Using Git and GitHub with R

Rob Chlumsky and Kevin Shook

June 3, 2020





cshs.cwra.org

- Promote the science of hydrology and its sound application in effective water management
- Courses (Kananaskis, Waterloo & Sacré-Coeur)
- Modelling Tools (GreyJay and hydrology R package)
- Webinars
- Conferences
- Awards and Scholarships

- R package of functions for Canadian hydrologists
github.com/CSHS-CWRA/CSHShydRology
- Undergoing development, hoping to deploy on CRAN this fall
- Many people interested in helping, but are having issues with git/GitHub
- Please join us!

Purpose of this webinar...

... is to get you comfortable *using* git and GitHub for your R projects, and enable you to collaborate on projects such as **CSHShydRology**.

It is *not* to make you an expert in version control software.

Webinar components

- ➊ Introduction to git/GitHub (slides) - Kevin Shook
- ➋ Using git/GitHub (demonstration) - Rob Chlumsky
- ➌ Questions and wrap-up

All files are available at
github.com/CentreForHydrology/git_for_R

1. Introduction to git and GitHub

Outline

- Version control
- What are git and GitHub?
- How to set up git/GitHub
- Using git in R
- Working with GitHub

"FINAL".doc



FINAL.doc!



FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10. #@\$%WHYDID
ICOMETOGRADSCHOOL????.doc

JORGE CHAN © 2012

WWW.PHDCOMICS.COM

Version control programs

- When you create R files (code, notebooks, documents), there are always changes
- Changes sometimes damage the files
 - ▶ need to go back to older versions
- Need to add/test new features without damaging current version
- Especially true when working with other people
- Version control programs allow you to manage the versions of the files that you create.

- Most popular version control program
- Written by Linus Torvalds, creator of Linux
- Free Open Source Software (FOSS)
- *Distributed* version control
 - ▶ doesn't require a centralized server like SVN

- Website running git
- Allows you to backup your git repository
- Also allows collaboration with others
- There are other similar sites like GitLab: <https://about.gitlab.com/>

Getting git

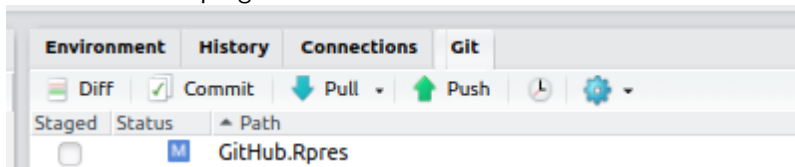
- Built into Linux
- For MacOS or Windows, you can download git from <https://git-scm.com/>

How git works

- A folder called **.git** is created in the directory holding your your project, the working directory
- This is the repository
 - ▶ It contains all versions, current and old, of your files
- When you make changes to the files, you add them to the repository
- You can retrieve old/different versions of the files into the working directory

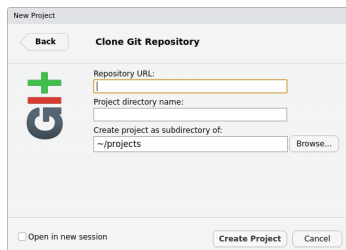
Working with git

- git is a command-based program
- There are many GUIs for git, including Rstudio
 - ▶ makes working with git much easier
 - ▶ uses Git tab in top-right



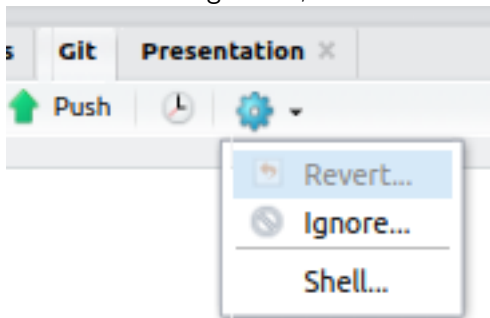
- ▶ you will still have to type commands occasionally

- When you create a new project in Rstudio, you can add a repository
 - ▶ you can also add a repository to an existing project
- When you clone a repository from GitHub through RStudio, a local repository is automatically created

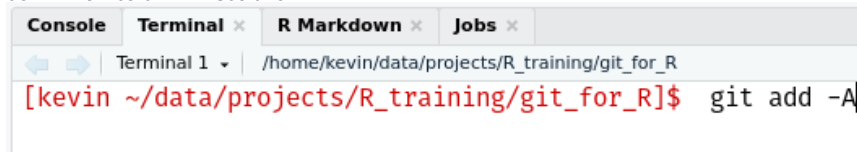


Typing in commands

- You can use the git shell, which is accessed through a drop-down menu



- Or you can type in git commands in any terminal, including the terminal tab in Rstudio



Configuring git

The first step is to tell git who you are:

```
git config --global user.name "John Doe"
```

```
git config --global user.email johndoe@example.com
```

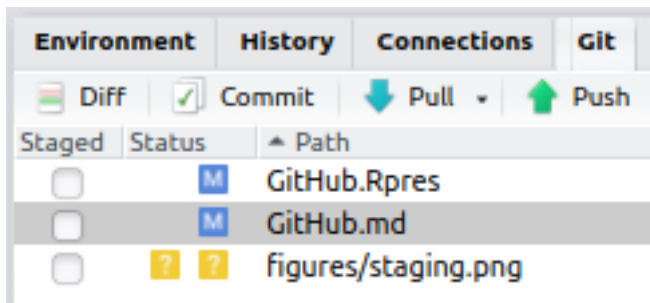
- You can list your current settings with the command

```
git config --list
```


- As you create code, you will want to add it to the repository
 - ▶ generally done each time you have made a significant change to any file
- Adding takes 2 steps:
 - 1 Staging (selecting the files to add), and
 - 2 Committing (adding the files to the repository)

Files available for staging

The Git tab shows all of the files which can be staged - 2 files have been modified (blue M icon), - 1 file is new (yellow ? icon)



Committing

- 1 Stage the files by clicking the check boxes beside the files to be added (the icons of the new files will change)
 - 2 click on the commit icon
- The commit window will pop-up, giving you a chance to review the files before committing

Commit window

The screenshot shows the Git commit window with the following components:

- Top Bar:** Includes tabs for 'Changes' and 'History', a branch selector set to 'master', and icons for 'Stage', 'Revert', and 'Ignore'. On the right, there are 'Pull' and 'Push' buttons.
- Staged Files List:** A table with columns 'Staged', 'Status', and 'Path'. It lists three files: 'GitHub.Rpres' (modified, M), 'GitHub.md' (modified, M), and 'figures/staging.png' (added, A).
- Commit Message:** A large text area for entering the commit message. Below it is a checkbox for 'Amend previous commit' and a 'Commit' button.
- Diff View:** A section at the bottom showing a diff of the staged files. It includes a 'Show' dropdown set to 'Staged', a 'Context' dropdown set to '5 line', and an 'Unstage All' button. The diff content shows changes to a file, with line numbers 93 through 102 visible. The changes include a new line about listing settings and a code snippet for `git config --list`.

Staged	Status	Path
<input checked="" type="checkbox"/>	M	GitHub.Rpres
<input checked="" type="checkbox"/>	M	GitHub.md
<input checked="" type="checkbox"/>	A	figures/staging.png

Commit message

☐ Amend previous commit Commit

Show Staged Unstaged Context 5 line ☐ Ignore Whitespace Unstage All

```
93 91
94 You can list your current settings with the command
92 - You can list your current settings with the command
95 93
96 94 ```
97 $ git config --list
95 git config --list
98 96 ```
99 97
98 Version control
99 =====
100 - As you create code, you will want to add it to the repository
101 - generally each time you have made a significant change to
102 any file
```

- The bottom pane (Diff) shows the changes in all of the files
 - ▶ you can select or discard changes
- You **must** add a comment in the top-right panel before clicking on Commit

Git history

In the Commit window, clicking on the History button shows the history of all of your commits to the repository

ChangesHistorymaster ▾(all commits) ▾🔄

🔍 Search

⬇


Subject	Author	Date	SHA
HEAD → refs/heads/master origin/master origin/HEAD revised folders	Kevin Shook <kevin.shook@usask.	2020-05-16	b25de385
18 slides	Kevin Shook <kevin.shook@usask.	2020-05-16	9846b2cb
more slides	Kevin Shook <kevin.shook@usask.	2020-05-16	b5967465
added more figures	Kevin Shook <kevin.shook@usask.	2020-05-16	80ff665d
added exercises	Kevin Shook <kevin.shook@usask.	2020-05-16	3e4cc8c1
First commit	Kevin Shook <kevin.shook@usask.	2020-05-16	784d0138

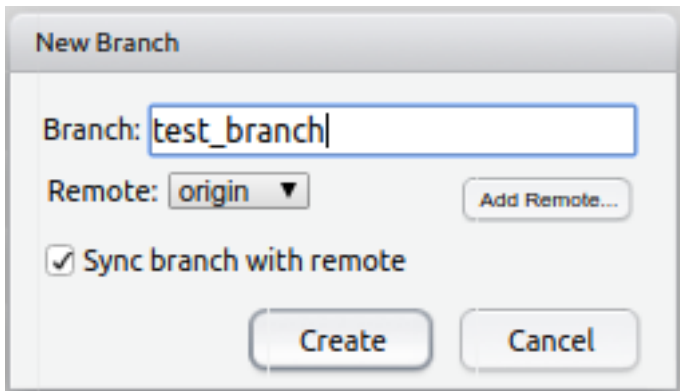
- Each commit is identified by a unique SHA number

- git uses *branches* to organize your code/documents
- Each repository always has a branch called **master**
 - ▶ most up-to-date, best version of the code
- Each branch is separate, and can be changed/deleted
- The current branch is shown in the Git tab
- You can add branches at any time
- When you change the branch, the files in the working directory are updated

Creating branches

You can create a new branch at any time

- Use the branch icon in RStudio:  to display the dialog box

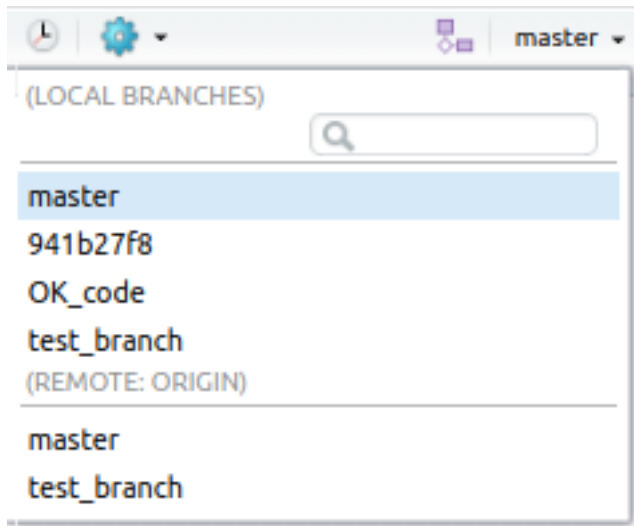


The image shows the 'New Branch' dialog box in RStudio. It has a title bar 'New Branch'. Inside, there is a text field labeled 'Branch:' containing the text 'test_branch'. Below it is a dropdown menu labeled 'Remote:' with 'origin' selected. To the right of the dropdown is a button labeled 'Add Remote...'. Below the dropdown is a checkbox labeled 'Sync branch with remote' which is checked. At the bottom are two buttons: 'Create' and 'Cancel'.

- Current versions of all files are added to the new branch

Changing between branches

- You can switch between branches by selecting the branch name



Recovering from mistakes

There are *lots* of ways of screwing up your code!

- accidentally deleting files
- accidentally deleting many lines in a file (and saving)
- overwriting files

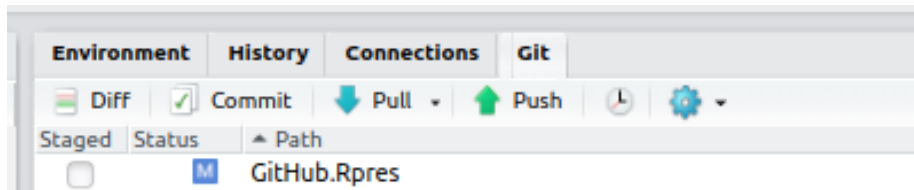
This is why it's a good idea to make a branch *before* making big changes to your project

sethrobertson.github.io/GitFixUm shows how to recover from many different types of mistakes

- The GitHub/GitLab repository linked to your local repo is referred to as the “Remote”
 - ▶ i.e. the repo that is online is the remote one, the repo on your desktop is local

Pulling

- Pulling downloads the GitHub repo to your local repo



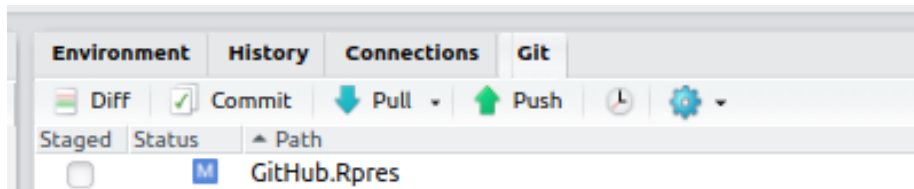
- It's a good idea to click on **Pull** to make sure that the local repo is up to date before doing any new work

Git Pull

```
>>> git pull  
Already up to date.
```

Pushing

- Pushing uploads your local repository to GitHub



- You should only push to your *own* GitHub repository

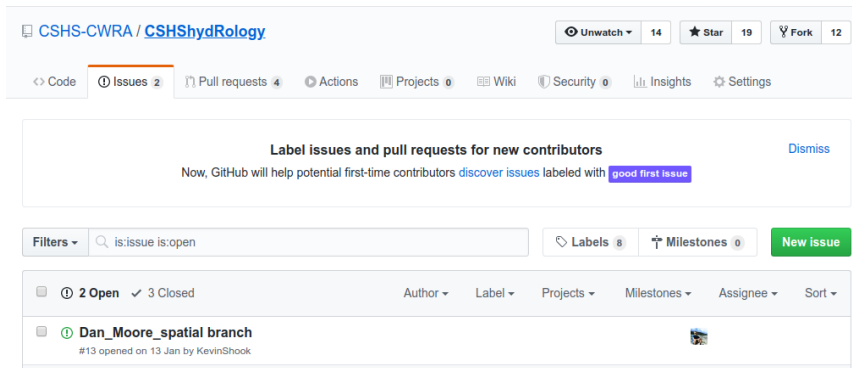
Git Push

```
>>> git push origin HEAD:refs/heads/master
To github.com:CentreForHydrology/git_for_R.git
   3e4cc8c..b596746  HEAD -> master
```

- The most important feature of GitHub is the way it enables people to work together on projects
- Each project will typically have an owner, and one or more people who can approve changes
- If you aren't one of these people (and even if you are!), you shouldn't be pushing changes to the **master** branch directly.

Bug reports (Issues)

- One of GitHub's most important features.
- Very easy to submit an Issue



The screenshot shows the GitHub repository page for **CSHS-CWRA / CSHShydRology**. The repository has 14 Unwatched issues, 19 Stars, and 12 Forks. The **Issues** tab is selected, showing 2 open issues. A notification banner states: "Label issues and pull requests for new contributors. Now, GitHub will help potential first-time contributors discover issues labeled with **good first issue**". The search bar shows the query `is:issue is:open`. The issue list shows 2 Open issues and 3 Closed issues. The first issue is titled **Dan_Moore_spatial branch**, opened on 13 Jan by KevinShook, and is labeled as a **good first issue**.

- Writing a *good* bug report is an art - see github.com/rstudio/rstudio/wiki/Writing-Good-Bug-Reports

- A *fork* is complete copy of a GitHub repo
 - ▶ lets you copy other work to use as a basis for your own
 - ▶ also lets you make a working copy the repo files, without affecting the original repo
- A good way to create new features or fix bugs
- When you are finished, you can then submit a Pull Request

Pull requests

- Pull requests are submitted through GitHub
 - ▶ tell members of the project about your suggested changes
 - ▶ allows discussion
- Files can then be merged with the specified branch

- ssh is short for “secure shell”
- Provides secure, encrypted communication between 2 computers
- If you set it up on your computer, you can avoid having to type in your user name and password every time you push to GitHub
- Part of Linux and Mac OS
- to add to Windows
docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse
- Once installed, you have to configure it to create a key and set up your GitHub account to use the key
help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh

Typical Workflow

- ➊ Fork a repository of interest to your own GitHub account (creates a copy of this repo on your own account).
- ➋ Checkout your version of this repo locally.
- ➌ Make updates/ changes/ new branches/ etc. on your local account. Preferably in a new branch.
- ➍ Merge/ push changes to the remote branch on your forked repo.
- ➎ Make a pull request to bring these changes from your repo back to the original.
- ➏ Celebrate in having made a contribution to another project!

2. Git and R Exercises

This will walk through the typical workflow above with two examples:

1. This presentation repository ([CentreForHydrology/git_for_r](#))
2. The CSHS-hydRology package ([CSHS-CWRA/CSHShydRology](#))

3. Questions?

- **lots** of resources online for git, GitHub, and R support
- many cool things that can be done with R, RStudio, and Git:
 - ▶ presentations, papers, reports, web pages, animated htmls. . .
 - ▶ Git pages, project landing pages websites, etc.

- please fill out our post-webinar survey (click on the url):
docs.google.com/forms/d/e/1FAIpQLSeUf7iG_GZoxrKHkKrh6cqVs_baFS
- support the **CSHShydRology** project, as you can:
 - ▶ join our newsletter signup and monthly meetings
 - ▶ contribute function ideas, review code, tell your colleagues about this group
- stay tuned for free webinars on R and hydrology in the future