

# Using Git and GitHub with R

Kevin Shook and Rob Chlumsky

June 3, 2020



# Outline

- ▶ Version control
- ▶ What are git and GitHub?
- ▶ How to set up
- ▶ 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



# 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 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 centralised 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 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 versions of the files into the working directory

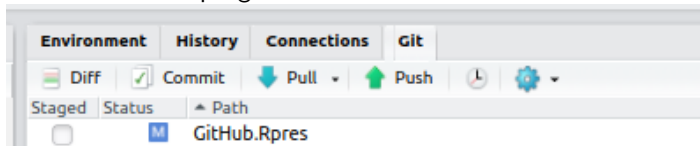


- ▶ When you clone a repository from GitHub, a local repository is automatically created
- ▶ You can also set up git for a local project in Rstudio using the menu

Tools | Project Options ...

# Working with git

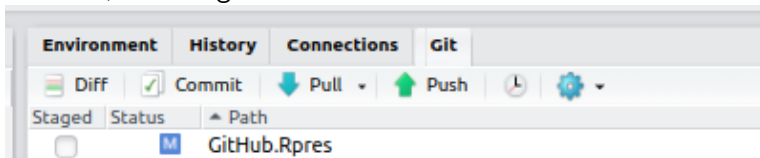
- ▶ git is a command-based program
- ▶ There are many GUIs for git, including Rstudio
  - ▶ makes working with git much easier
  - ▶ uses Git tab in top-right



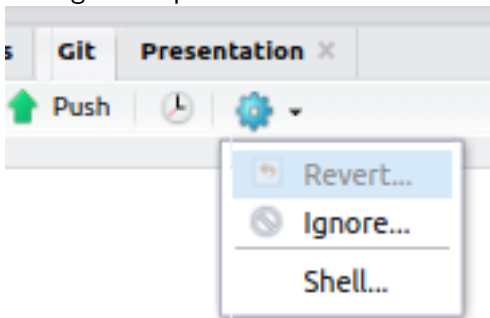
- ▶ you will still have to type commands occasionally

# Typing in commands

- ▶ In Linux or Mac OS, you can type in git commands in any terminal, including the terminal tab in Rstudio



- ▶ In Windows, you have to use the git shell, which is accessed through a drop-down menu



# 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

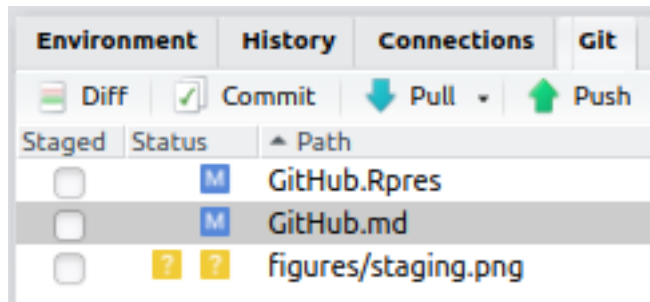
```
git config --list
```

# Version control

- ▶ As you create code, you will want to add it to the repository
  - ▶ generally 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

Select the files to be added

- the icons of the new files will change) - and 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. At the top, there are tabs for 'Changes' and 'History', and a dropdown menu set to 'master'. To the right of these are icons for 'Stage', 'Revert', and 'Ignore'. Further right are 'Pull' and 'Push' buttons. Below the tabs, there is a table with columns 'Staged', 'Status', and 'Path'. The table lists three files: 'GitHub.Rpres' (status M), 'GitHub.md' (status M), and 'figures/staging.png' (status A). To the right of the table is a large text area for the 'Commit message'. Below the message area is a checkbox for 'Amend previous commit' and a 'Commit' button. At the bottom of the window, there is a 'Show' section with radio buttons for 'Staged' and 'Unstaged', a 'Context' dropdown set to '5 line', and checkboxes for 'Ignore Whitespace' and 'Unstage All'. Below this is a list of lines of code, with line numbers 93 through 104 visible. The code is color-coded: red for commands and green for explanatory text.

Changes History master [G] [✓] Stage [↶] Revert [↷] Ignore [⇅] Pull [⇅] Push [⇅]

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
95 92 - You can list your current settings with the command
96 93
97 94 ``
98 95 $ git config --list
99 96 git config --list
100 97 ``
101 98 Version control
102 99 =====
103 - As you create code, you will want to add it to the repository
104 - generally each time you have made a significant change to
    any file
    - Adding takes 2 steps:
    1. Staging (selecting the files to add). and
```

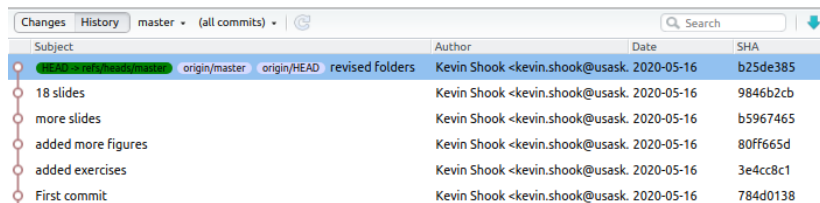


# Commit window

- ▶ 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



The screenshot shows the 'History' tab in a Git client. At the top, there are tabs for 'Changes' and 'History', and a dropdown menu showing 'master' and '(all commits)'. A search bar is on the right. Below the tabs is a table of commit history. The first commit is highlighted in blue. To the left of the table is a vertical timeline with circular markers representing the commit history.

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

# Branches

- ▶ git uses *branches* to organise your code/documents
- ▶ Each repository always has a brance 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

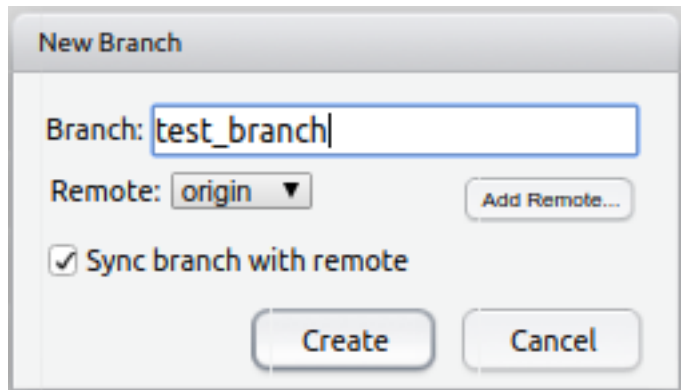
# Creating branches

- ▶ You can create a new branch at any time

- ▶ Use the branch icon in RStudio:



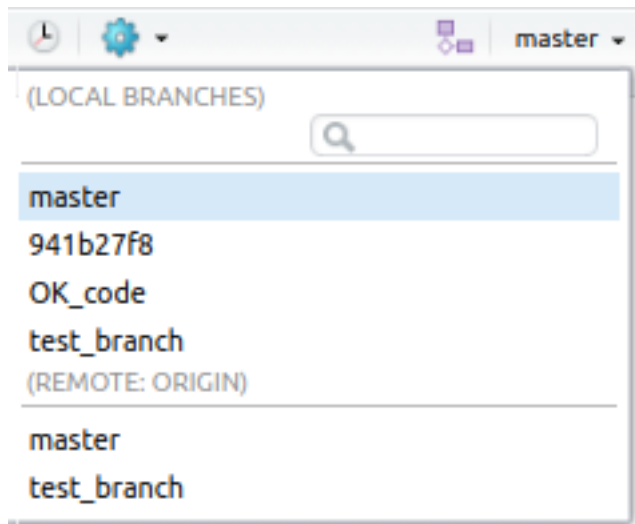
to display the dialog box

A screenshot of the 'New Branch' dialog box in RStudio. The dialog has a title bar 'New Branch'. Inside, there is a text field labeled 'Branch:' with the text 'test\_branch' entered. Below it is a 'Remote:' dropdown menu currently showing 'origin'. To the right of the dropdown is a button labeled 'Add Remote...'. Below the dropdown is a checked checkbox labeled 'Sync branch with remote'. At the bottom of the dialog 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](http://sethrobertson.github.io/GitFixUm) shows how to recover from many different types of mistakes

# Working with GitHub

- ▶ The GitHub/GitLab repository linked to your local repo is referred to as the “Remote”

# 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

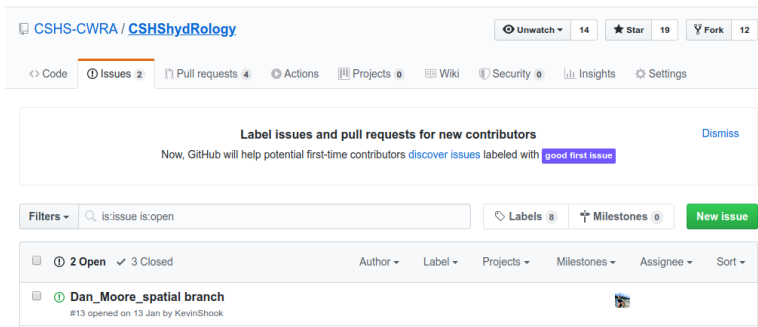
```
>>> git push origin HEAD:refs/heads/master  
To github.com:CentreForHydrology/git_for_R.git  
3e4cc8c..b596746 HEAD -> master
```

# Working with others

- ▶ 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 items, 19 Stars, and 12 Forks. The navigation bar includes links for Code, Issues (2), Pull requests (4), Actions, Projects (0), Wiki, Security (0), Insights, and Settings. A banner for new contributors is visible, stating "Label issues and pull requests for new contributors" and "Now, GitHub will help potential first-time contributors discover issues labeled with good first issue". Below the banner, there are filters for "is:issue is:open", 8 Labels, 0 Milestones, and a "New issue" button. The issue list shows 2 Open issues and 3 Closed issues. The first issue is titled "Dan\_Moore\_spatial branch" and was opened on 13 Jan by KevinShook.

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

# Forking

A *fork* is complete copy of a GitHub repo

- It lets you copy other work to use as a basis for your own
- It 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

- ▶ 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
- ▶ part of Linux and Mac OS
- ▶ to add to Windows

<https://jcutrer.com/windows/install-openssh-on-windows10>