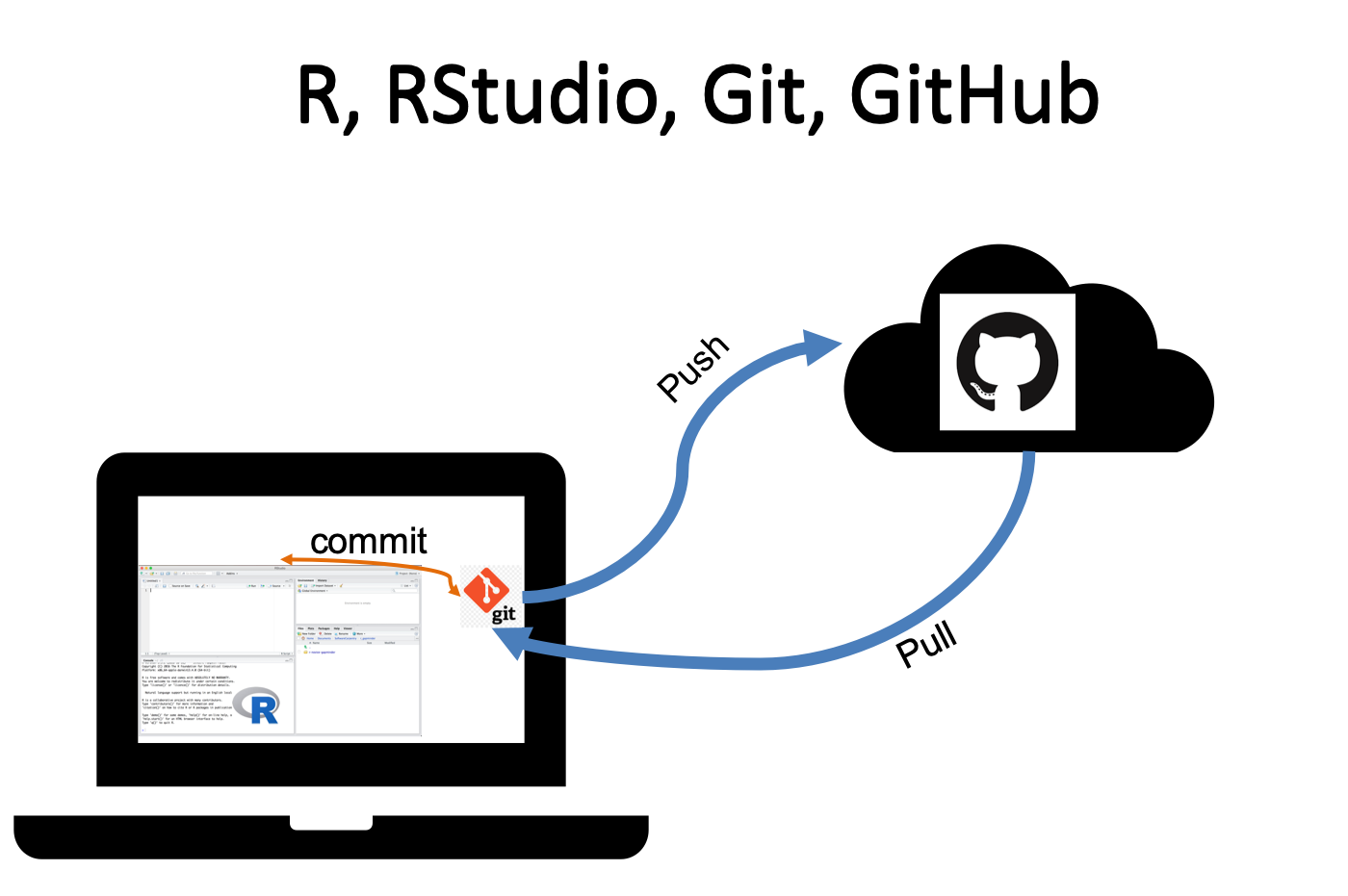
**How to use Github/Git/RStudio**

**Haldre Rogers**

**Last updated 6 September 2019**



* R and Git run in the background of Rstudio, and GitHub is in the cloud.
* Git provides version control, but the changes are only stored locally whenever you “commit”.
* GitHub is an online platform for sharing files (especially code) that is compatible with Git.
* Each GitHub Repository is the equivalent of a single R project.
* You should think of the version of the files on GitHub as the official, long-term version, and the version that you clone/download to your computer as a temporary local copy.
* When you start a work session, “pull” all new changes from GitHub, and when you finish a session, “push” all changes up to GitHub.

**To get GitHub, Git, and RStudio setup (needs to be done just once, when you first start using Git/GitHub)**

Also see this site for more help: <https://happygitwithr.com/>

1. Download and install the latest versions of [R](https://www.r-project.org/) (use the CRAN mirror closest to you) and [R Studio](https://www.rstudio.com/products/rstudio/download/).
2. Download and install Git (<https://git-scm.com/downloads>)
   1. Choose all of the default options
3. Create a github account online at https://github.com.
   1. Verify your email address (you should receive an email, click on the link inside)
4. In R Studio
   1. Go to *Tools -> Global Options* 🡪 Git//SVN. On Macs, can also Go to RStudio🡪Preferences 🡪 Git/SVN
      * 1. Make sure "Enable version control" is checked.
        2. Make sure Git executable says /usr/bin/git, or navigate to the 'git' executable in the 'bin' folder. This can be tricky. See below if you have problems.
        3. (optional) Make sure SVN executable says /usr/bin/svn
        4. Link GitHub to your version of R Studio by doing the following
           1. Hit "Create RSA Key". This creates the key in a popup window. Press Close.
           2. Click "View Public key" (only apparent after you create a key) and copy the output
5. In GitHub
   1. Go to settings. SSH and GPG keys. Paste the copied public SSH key here. This should link your version of Git and R Studio to your GitHub account.
6. Link git to GitHub
   1. In the Terminal/Shell (to get there, go to RStudio, and select Tools-> Shell; this will open up the terminal)
      1. Tell Git your name and email address. These are used to label each commit so that when you start collaborating with others, it’s clear who made each change. In the shell, type:
         1. *git config --global user.email "youremailaddress"* (replace "youremailaddress" with your email that you used for GitHub)
         2. *git config --global user.name "yourgitname"* (replace "yourgitname" with your GitHub username)
   2. The [usethis package](https://usethis.r-lib.org/) offers an alternative approach. You can set your Git user name and email from within R:

install.packages("usethis")

**library**(usethis)

**use\_git\_config**(user.name = "Jane Doe", user.email = "jane@example.org")

**At this point, your accounts should be linked, however, you won't see anything different yet. You need to set up a new repository and check to see if the Git tab appears.**

1. In GitHub
   1. Create a new repository in GitHub to test whether you have linked the two accounts.
      1. Name it anything you want. Initialize with a ReadMe.
      2. Once it has been created, navigate to the repository.
      3. Press "Clone or Download"
      4. Copy the link.
2. In RStudio,
   1. Click "New Project"
   2. Choose "version control"
   3. Choose "Git"
   4. In "Repository URL", enter URL copied above
   5. Name the directory whatever you want
   6. Now you should see a Git tab on either the top right or bottom right window in R Studio.
3. **Handling problems in this process:** 
   1. When all else fails, try shutting down and restarting R Studio.
   2. If you can't figure out where you git executable file is.
      1. ask the shell. Run the following command:
         1. which git (Mac, Linux)
         2. where git (most versions of Windows)
      2. Or follow directions here: http://stat545.com/git03\_rstudio-meet-git.html
   3. If you see this error in your shell:
      * error: unable to read askpass response from 'rpostback-askpass'
      * fatal: could not read Username for 'https://github.com': Device not configured
      1. Go to here for the solution: https://github.com/jennybc/stat540\_2014/blob/master/seminars/seminar92\_git.md
      2. Or here
         1. https://github.com/OHI-Science/ohicore/issues/104
   4. Computer would not let you create RSA key - says "no such file exists". Go to command prompt and typed mkdir.ssh (see here for more details. https://jujucharms.com/docs/2.0/getting-started-keygen-win)
   5. Do you have a Mac upgraded to El Capitan? This upgrade moves an important directory that they need to re-install. Follow directions here.
      1. http://tips.tutorialhorizon.com/2015/10/01/xcrun-error-invalid-active-developer-path-library-developer-commandline-tools-missing-xcrun/
   6. If you are using an old operating system on a Mac, git might not show up for you. Follow instructions here to download the right version of git. <http://stackoverflow.com/questions/24215564/installing-git-on-mac-os-10-7-5>
   7. If R Studio does not seem to be talking with GitHub, you may need to use the SSH clone instead of the http. This is as simple as pushing the “clone as ssh” button when you go to clone/download in GitHub.
   8. If you get this error message, and don’t see a .Rproj file associated with your folder, could be that you have some space issues on your computer (possibly due to security restrictions). [GitHub: Clone succeeded, but checkout failed](https://stackoverflow.com/questions/22041752/github-clone-succeeded-but-checkout-failed)

**Using R Studio with git and github to follow along but not suggest changes**

1. **Cloning**: If you just want to follow along with the R scripts from a repository on GitHub, and make changes on your local computer, but not suggest changes to the master file, you should do fine with a clone of the repository.
   1. In Github,
      * 1. Click on the repository you want to clone/download.
        2. Click on "Clone/Download".
           1. Note- typically this uses https, but if you are seeing errors associated with the phrase “askpass” when you try to enter this URL in RStudio below, then try using the SSH key instead.
        3. Copy the URL.
   2. In RStudio,
      1. Click "New Project"
      2. Choose "version control"
      3. Choose "Git"
      4. In "Repository URL", enter URL copied above
      5. Name the directory whatever you want
      6. If you want to make changes to the R file, and save them separately from the course file, then use “save as” and rename your R file. This allows you to play around with it, saving it locally on your computer, and avoids the problem of unstaged changes conflicting with any updates on GitHub.
      7. If you don’t want to save your own changes, then you can work off the files that you’ve cloned, but do not push up to GitHub.
2. **Branching: If you want to operate within the same repository, but work on a separate branch, and possibly merge it with the main branch at a later time, create a new branch.** 
   1. **More details later.**
3. **Forking**: If you want a duplicate repository in your personal GitHub account, so you can do whatever you want with the new files and do not have any expectation of pulling new changes down or merging it with the main branch, you might want to fork. It is still possible to contribute to the original master R script via a "pull request", but it would be better to branch if you are working collaboratively on something.
   1. In Github,
      1. Click on the repository you want to clone/download.
      2. Click on "Fork" in upper right side of page.
      3. Follow directions - it will ask you where you want to fork this repository to- pick somewhere in your personal GitHub account.
      4. Once you've forked it, go to that location, and follow directions for cloning/downloading above.
   2. In RStudio,
      1. Click "New Project"
      2. Choose "version control"
      3. Choose "Git"
      4. In "Repository URL", enter URL copied from GitHub "Clone/Download" button above.
      5. Name the directory whatever you want
      6. Now you can open the .R file and play around with it,

**To create your own repository**

1. You have two options:
   1. Option 1: Create a repository on GitHub, and clone to your local computer/RStudio
      1. Click “New Repository”
      2. Give your repository a name (short)
      3. Choose public or private
      4. Click “Initialize this repository with a ReadMe”. This is a very important step to make the rest of the process easier.
      5. Now your repository has been created. You can upload files, and clone/download just like any other repository.
   2. Option 2: Create a project on R Studio, and then push to a new repository on GitHub
      1. On your computer
         1. Start a “new project” on R Studio, in the folder set aside for this project. Be sure to click "create a git repository" when you create this project.
         2. Create a set of folders (e.g. "data", "analysis", "graphics", "wrangling") on your computer, and populate with data, documents, etc.
         3. Commit all changes (except .gitignore and .Rproj files). This doesn’t commit to GitHub, only to Git.
      2. On GitHub
         1. Create new repository. Name whatever you want. Do not initialize with a readme.
         2. Scroll down and copy the code following “existing repository”. If that doesn't work, use "create new repository" (the one just above existing repository).
      3. R Studio.
         1. Open Shell. Paste code from GitHub. Enter username and password if requested.
      4. This should create a link between Git and GitHub, and you should be able to push/pull now. To check, push the changes you committed earlier and then open GitHub to see if they have shown up.

**Pushing/pulling changes**

1. You can push changes in an Rproject from RStudio back to your GitHub repository. Let's practice this.
   1. Make some edits to R script.
   2. Save.
   3. Click on "Git" tab on bottom right panel.
   4. Check box next to the .R file.
   5. Click "commit". This will bring up a new window.
   6. Write a note in the commit box. Press commit.
   7. Keep working & committing every time you do something significant within a working session.
   8. At the end of your working session, press "push" to push your commits to the online repository.
   9. Go onto GitHub, and check to see if those changes are present. Voila!

**Guide to git codes in the R Studio Git pane:**

The Git pane lists each file that has been added, changed, or deleted.

* http://r-pkgs.had.co.nz/screenshots/git-modified.png, **Modified**. Contents of file have been changed since last commit
* http://r-pkgs.had.co.nz/screenshots/git-unknown.png, **Untracked**. New file that Git hasn’t seen before and isn’t being tracked by git
* http://r-pkgs.had.co.nz/screenshots/git-deleted.png, **Deleted**. File has been deleted
* http://r-pkgs.had.co.nz/screenshots/git-added.png: **Added.** This appears after you stage an untracked file.
* http://r-pkgs.had.co.nz/screenshots/git-renamed.png: **Renamed.** Git recognizes that a file has been renamed.
* http://r-pkgs.had.co.nz/screenshots/git-modified-staged.png. **Modified Modified.** Means you have some staged changes and some unstaged changes in a file.
* http://r-pkgs.had.co.nz/screenshots/git-commit-conflict.png **Merge conflict.**

**General notes about using R Studio with Git**

1. Making changes to existing files
   1. If you make any changes, you will need to first “Stage” the files that you want to commit.
      1. **Select files**. To stage (select) a single file for inclusion, tick its check box. To stage all files, press Ctrl/Cmd + A, then click http://r-pkgs.had.co.nz/screenshots/git-stage.png.
      2. As you stage each file, you’ll notice that its status changes. The icon will change columns from right (unstaged status) to left (staged status), and you might see one of two new icons:
         1. Added: http://r-pkgs.had.co.nz/screenshots/git-added.png: after staging an untracked file, Git now knows that you want to add it to the repo.
         2. Renamed: http://r-pkgs.had.co.nz/screenshots/git-renamed.png: If you rename a file, Git initially sees it as a deletion and addition. Once you stage both changes, Git will recognise that it’s a rename.
         3. Sometimes you’ll see a status in both columns, e.g. http://r-pkgs.had.co.nz/screenshots/git-modified-staged.png. This means that you have both staged and unstaged changes in the same file. This happens when you’ve made some changes, staged them, and then made some more. Clicking the staged checkbox will stage your new changes, clicking it again will unstage both sets of changes.
   2. Once you’ve staged everything you want to commit, press commit.
   3. When a new window pops up, write a commit message.
   4. Commit your changes.
2. Reverting changes
   1. Click Diff and then Revert. The erroneous change has been undone and the previous version restored.
3. Seeing what you’ve changed
   1. You can get more details about modifications with a “diff”, http://r-pkgs.had.co.nz/screenshots/git-diff.png. This opens a new window showing the detailed **diff**erences.
   2. The background colors tells you whether the text has been added (green) or removed (red). The grey lines of code above and below the changes give you additional context.
4. Undoing a mistake
   1. If you haven’t pushed to GitHub yet, this can be done simply in R Studio.
      1. Right click on the file in the Git pane and select “revert”. This will restore the version from before the most recent commit. This operation is permanent, so use carefully!
      2. You can also undo changes to just part of a file in the diff window. Use the discard chunk button above the block of changes that you want to undo.
5. Deleting a file
   1. Under the Files tab check the box next to the file name
   2. Click Delete
   3. Under the Git tab, a red D appears next to the deleted file
   4. Stage the change by clicking the checkbox and commit it
6. Ignoring files
   1. Sometimes you might want to ignore a file (i.e. not push it up to GitHub, just keep it on your desktop).
      1. You could either not stage it each time OR
      2. Add to .gitignore. Right click on the file in the Git pane, and select “Ignore”
7. Pushing to Github
   1. After you are finished working, you should push your changes to Github. You should have far fewer Push’s than Commits.
8. Merge conflicts.
   1. RStudio currently doesn’t provide any tools to help with merge conflicts, so you’ll need to use the command line.
   2. For more help, see: Working with Others section of this website: <http://r-pkgs.had.co.nz/git.html>
9. Dealing with large files
   1. If you try to push files >100 MB, it will break the system, leaving you & your file in purgatory. The way I figured out to fix it was to download and install (on the terminal) a program (https://git-lfs.github.com/) that allows you to store large files on GitHub. Then, I pushed the large file to GitHub manually (using terminal/shell), and then I deleted the file and pushed that change to GitHub so others don't have the same problem if they download and try to push it back up.
      1. I think this was useful: http://stackoverflow.com/questions/19573031/cant-push-to-github-because-of-large-file-which-i-already-deleted

**Useful resources**

1. A fantastic, readable overview of how to use GitHub, Git, and R Studio
   1. <http://r-bio.github.io/intro-git-rstudio/>
2. Hadley’s guide – good introduction to R Studio and Git/Github
   1. <http://r-pkgs.had.co.nz/git.html>
3. General overview of getting R studio set up with version control
   1. <https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN>
   2. <https://www.r-bloggers.com/rstudio-and-github/>
4. Overview of setup & common ways of using git with r studio
   1. <https://jennybc.github.io/2014-05-12-ubc/ubc-r/session03_git.html>
5. Useful for considering conflicts.
   1. <http://eriqande.github.io/rep-res-web/lectures/conflicts-stashing-remotes.html>
6. Great graphic showing how git works (especially useful for understanding shell commands)
   1. <https://www.git-tower.com/blog/workflow-of-version-control>