5/19/2021

Using GitHub for RStudio

Written by Justin Priest justin.priest@alaska.gov *and adapted by Sara Miller* sara.miller@alaska.gov

A common way to share and backup code is to use GitHub. The following instructions are heavily based upon the following resources. Refer to them for further information and troubleshooting. While it's optional, I find it easiest to make your computer show you hidden folders and file extensions. This will later allow you to see exactly what the computer sees. Google for specific instructions.

Part 1 – Connecting GitHub & RStudio

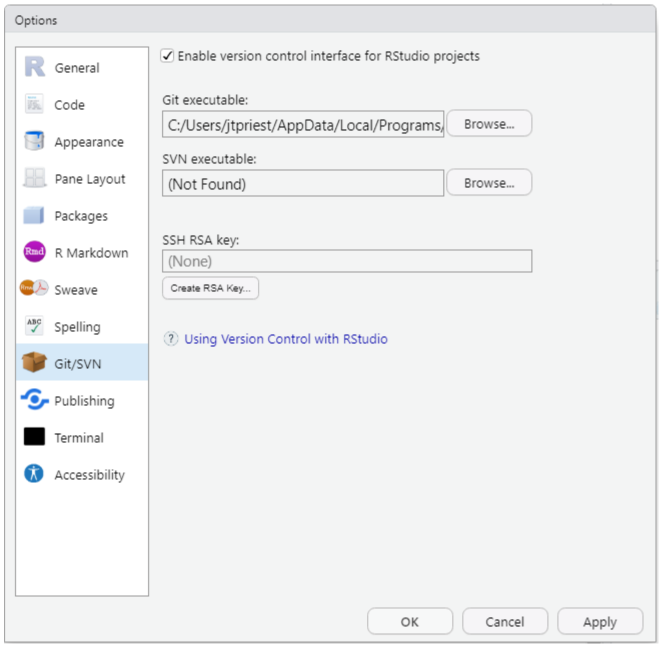
Daily usage of GitHub is easy! The hardest step is this first one: connecting RStudio to your GitHub account. Once up and running, it's only a few clicks a day!

1. **Register for a free GitHub account**
   Go to github.com and register a free account. I recommend that you use the email you want associated with this account (work email) and possibly another email (personal) with it. If you would like to be added as a member to the commfish GitHub site (https://github.com/commfish), send your GitHub login name to sara.miller@alaska.gov, and she will send you an invite.

2. **Install Git to your computer**
   Go to https://git-scm.com/download/ and download then install Git to your computer. Take note of the file directory that it is installed to. Common directories for this might be:
   C:\Users\jtpriest\AppData\Local\Programs\Git (change to your username)
   C:\Program Files\Git
   usr/local/git/bin/git (Mac OS)

   Accept all default settings and click through until installed. (Git won't show up as a program).

3. **Connect GitHub to RStudio**
   Open RStudio; Under Tools, click Global Options, then select "Git/SVN" on the side. In the window that pops up, select "Enable version control interface for RStudio projects". Under "Git executable" click browse and select the correct folder as shown before. You'll then want to select the git.exe file, typically found under folder "bin". (This is where it can be helpful to have Windows show you file extensions).


Options

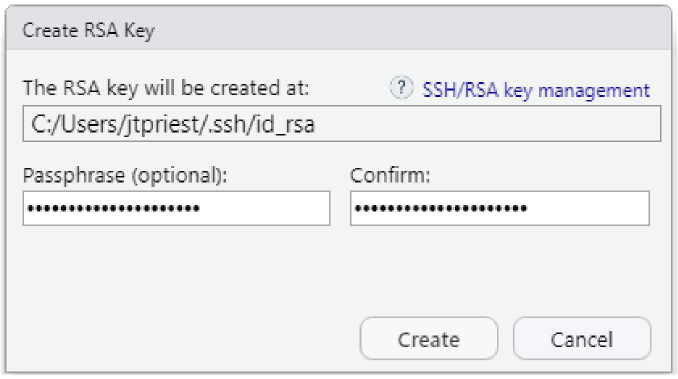
☑ Enable version control interface for RStudio projects

Git executable: C:/Users/jtpriest/AppData/Local/Programs/

SVN executable: (Not Found)

SSH RSA key: (None)

Next click “Create RSA Key”. Note the folder location. Give it a memorable password, then click Create. Once it is done, you’ll see some random art and file info. Click Close.
(What we’re doing here is a creating a “key” that will allow RStudio to know that this computer is definitely you!)

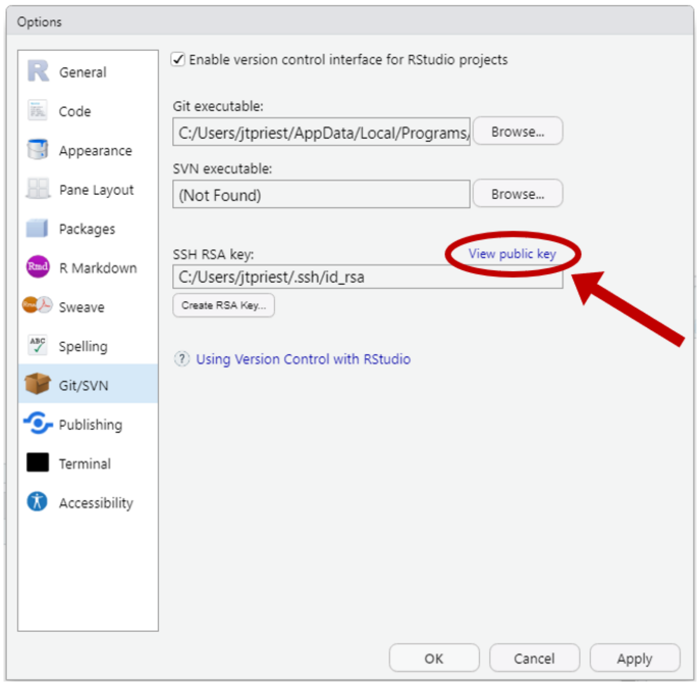

Create RSA Key

The RSA key will be created at: C:/Users/jtpriest/.ssh/id_rsa

Passphrase (optional): ••••••••••••••••••••••

Confirm: ••••••••••••••••••••••

Now you’ll want to open up and view this key. Click the small blue “View public key”



✓ Enable version control interface for RStudio projects

Git executable: C:/Users/jtpriest/AppData/Local/Programs/

SVN executable: (Not Found)

SSH RSA key: C:/Users/jtpriest/.ssh/id_rsa

At this point you'll see some random text that starts with "ssh-rsa". Copy this!

Open up github.com and go to your profile settings https://github.com/settings/profile
Click on "SSH and GPG keys" on the left. Click the green "New SSH key" button. Paste the key from RStudio in the key section. Give this key a unique title describing which computer you're using.

4. Set some default settings

In RStudio, we will want to save your name and email so that any edits that come from this computer are attributed to the correct name.
Run the following code in R, changing to your name and email:

```
install.packages("usethis")
library(usethis)
use_git_config(user.name = "Wilfred Higgins", user.email = "wfhiggins@alaska.gov")
git_sitrep() # this shows current name, directory, and other settings
```

Voila! Your GitHub account now knows that the computer you're on is linked to this account & it has saved your name and email to associate with changes you make.

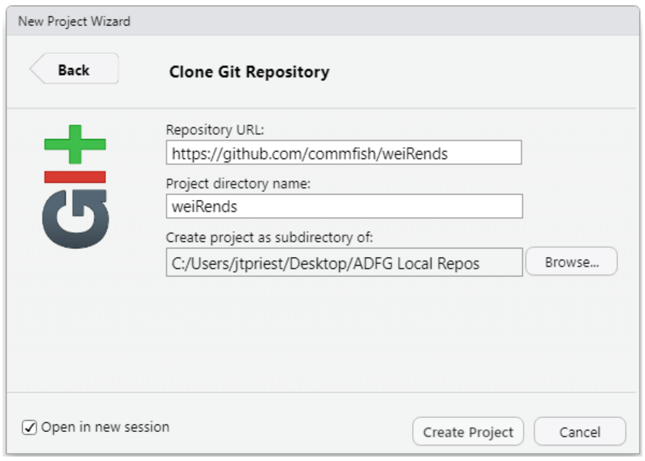
Part 2a – Connect to an Existing GitHub Repository

A “repository” or “repo” is just git-speak for a folder containing all files relevant to what you’re working on. It is easiest to connect to an existing GitHub repo. If your work already exists on your computer, it’s simplest to establish a new repo on github.com, following the instructions below, then copy/paste the files over. Connecting to GitHub only works with RStudio Projects (which is a good thing).

1. Open the github.com webpage for the specific repo that you want. Copy the URL at the top.
2. For this example, we’ll use the ADF&G Commercial Fisheries account of “commfish” and the repo called “weiRends”: https://github.com/commfish/weiRends

3. Next go to RStudio. Under File click New Project then Version Control then Git. Paste in the github.com address into the Repository URL (https://github.com/commfish/weiRends). It will auto fill the Project Directory Name (this will also be the local folder name so make sure to change this if you don’t like it). You can now change the directory where this folder will be. I prefer to open in a new RStudio session so I can keep working. Example below:

   Click Create Project and it will sync all files on the GitHub repo. If you have a lot of files, this might take a minute or two.



Repository URL: https://github.com/commfish/weiRends
Project directory name: weiRends
Create project as subdirectory of: C:/Users/jtpriest/Desktop/ADFG Local Repos
☑ Open in new session

   Click Create Project and it will sync all files on the GitHub repo. If there are a lot of files in the repo, this might take a minute or two.

4. Success! Check your local destination folder to see & open the files.

Part 2b – Create a New GitHub Repository

Often, we'll need to create a new repo for us to use. This is most easily done starting on github.com, creating a repo there, then syncing it to a local RProject.

1. Create a new repo on github.com by clicking the green plus button or going to https://github.com/new
2. Give it a unique name, choose whether to make it public or private, and add a readme. (It's optional but I usually add a .gitignore then select the template of R).
3. Same as step 3 in Part 2a
4. Same as step 4 in Part 2a
5. Copy over any local files that you want in this folder.
6. Sync all local files to github.com using Part 3!

Part 3 – Daily Usage

You can easily use GitHub to keep track of your changes and backup files. On the top right pane, you'll now see "Git" between Connections and Tutorial. This will track your files. Once files are modified, they will be a different color. There are 3 commands to use daily:

Commit: Prep your local files for uploading to github.com. Commit OFTEN and add a descriptive summary of what changes you've done.

Pull: Get data from the github.com

Push: Send data from your local folders to github.com

Uploading from RStudio to remote repo (github.com)

1. To sync your local edits to the github.com repo, select the "Git" tab in the top right pane. On the left checkboxes, select all files you want to sync. A yellow "?" means that you've never told RStudio what to do with the file; a blue M box means that the file has been modified since last synced.
2. Once you check the boxes of files to upload, click Commit and the git dialog box will open.
3. In the git dialog box, add a short note to your future self (and other collaborators) in the Commit message at top right.
4. Click Commit (the "Commit" in this new git dialog box), and a status window will open. Once it completes (usually instantly), then you can close this status window box.
5. In the git dialog box, click Pull and make sure that you're up to date (close status box).
6. Finally, click Push and wait for it to upload (this might take 10-30 seconds). Once you see the message change and "HEAD -> master" in the dialog box, you're good! Close out of both windows. You've successfully edited locally and synced it to the web!

Other items

If you get an error during pushing data, there are most likely changed files on the github.com repo; pull first.

The .gitignore file can be set to tell Git to ignore those files and never try to sync them. I usually just leave this alone but it's helpful to tell Git to not prompt you to try to sync those files.

If you are collaborating with someone (both working on same github.com repo) commit and pull VERY often. Avoid working in the exact same file at the same time.

Sometimes you might be collaborating with someone who you don't want to grant "push" access to. They can still collaborate using a "Pull Request". Let's say they change code of yours that might break something, and you want to review it before accepting all changes. That's what a "Pull Request "is: they're requesting that you pull their code into the master document.

One great feature is "Issues" where you can flag known problems with the code / data files, keep track of remaining tasks, or assign sections to collaborators.

The worst thing that can happen is a "merge conflict" when two collaborators pushed to the same file without pulling first. Google for fixes on this. The best solution for this is to be diligent to prevent it (e.g., have collaborators work on different parts of the code; communicate consistently with collaborators that are pulling/pushing files to the repo).

General Resources:

https://usethis.r-lib.org/articles/articles/usethis-setup.html

https://dangitgit.com

https://happygitwithr.com

https://afredston.github.io/learn-git/learn-git.html

https://rstudio-conf-2020.github.io/r-for-excel/rstudio.html#github-brief-intro-config

https:// github.com/commfish/Biometric_Best_Practices

https://rawgit.com/nazrug/Quickstart/master/GithubQuickstart.html

https://r4ds.had.co.nz/tidy-data.htmlR Code – Best practices | R-bloggers

https://adv-r.hadley.nz/subsetting.html