**Install Git**

(modified from <http://r-pkgs.had.co.nz/git.html>. A longer version is available at <https://aberdeenstudygroup.github.io/studyGroup/lessons/SG-T1-GitHubVersionControl/VersionControl/> )

1. Install Git:
   * Windows: <http://git-scm.com/download/win>.
   * OS X: <http://git-scm.com/download/mac>.
2. 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, run:

git config --global user.name "YOUR FULL NAME" [no spaces! A username]

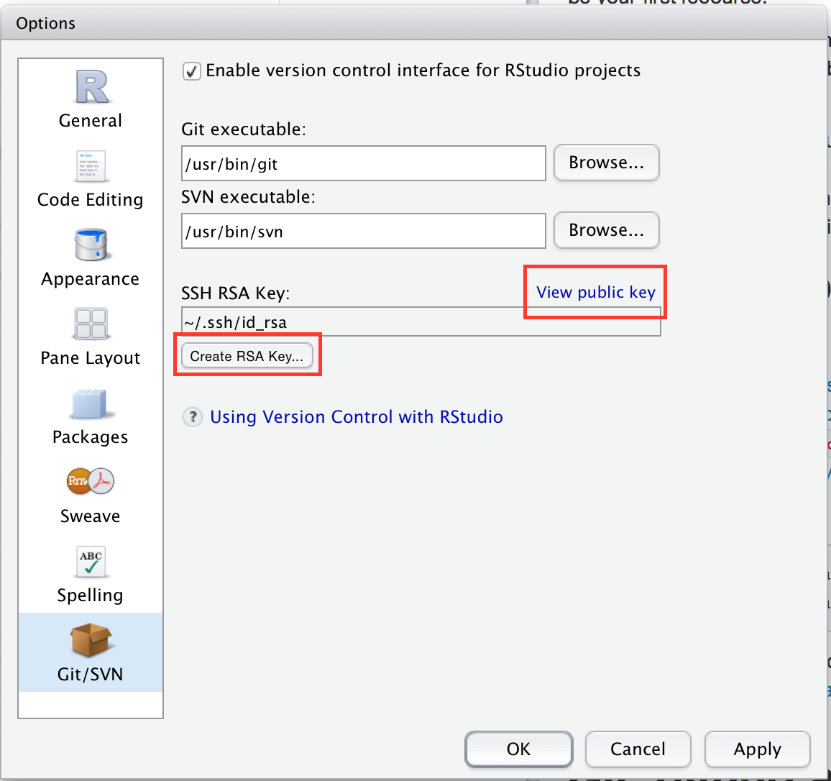
git config --global user.email "YOUR EMAIL ADDRESS"

(You can check if you’re set up correctly by running git config --global --list.)

1. Create an account on GitHub, <https://github.com>. Use the same email address as above.
   * Free plan requires public posting of files; paid plan allows private repositories.
2. If needed, generate a SSH key. SSH keys allow you to securely communicate with websites without a password. There are two parts to an SSH key: one public, one private. People with your public key can securely encrypt data that can only be read by someone with your private key.

From R, you can check if you already have an SSH key-pair by running:

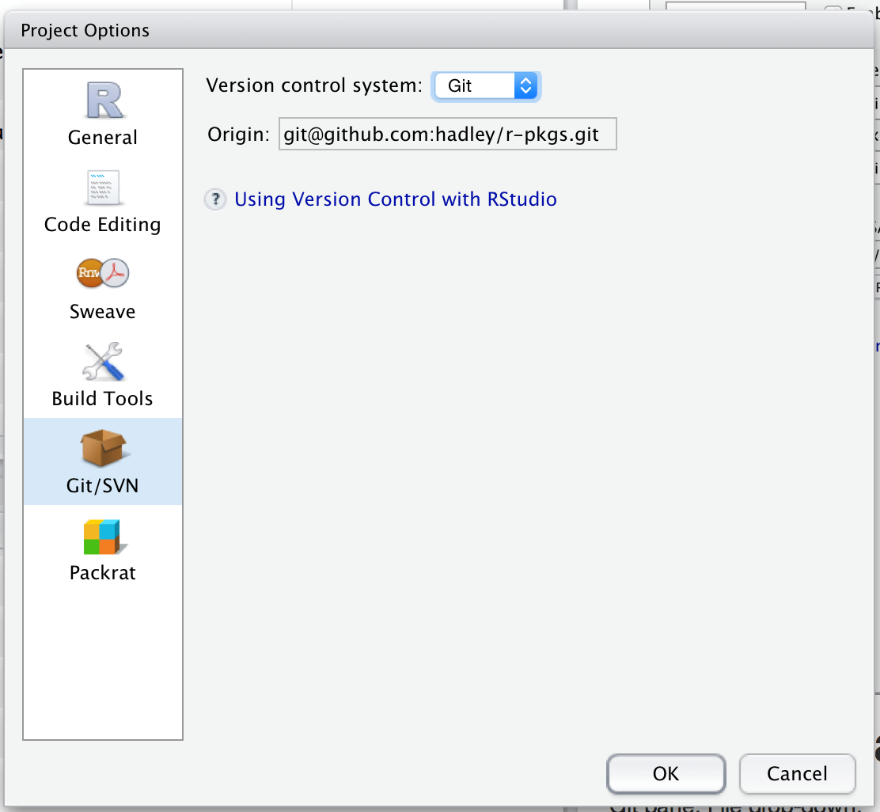
file.exists("~/.ssh/id\_rsa.pub")

If that returns FALSE, you’ll need to create a new key. You can either follow the instructions on GitHub (<https://help.github.com/articles/generating-ssh-keys>) or use RStudio.

Go to RStudio’s global options, choose the Git/SVN panel, and click “Create RSA key…”:

Give GitHub your SSH public key: <https://github.com/settings/ssh>. The easiest way to find the key is to click “View public key” in RStudio’s Git/SVN preferences pane.

**Create a local Git repository**

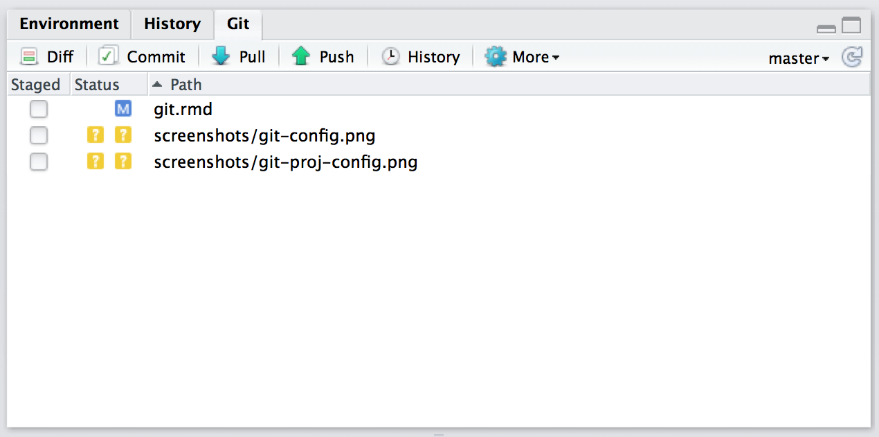
To use GitHub within RStudio, you’ll need to initialize a **local repository**, or **repo** for short. This creates a .git directory that stores configuration files and a database that records changes to your code. A new repo exists only on your computer; you’ll learn how to share it with others shortly.

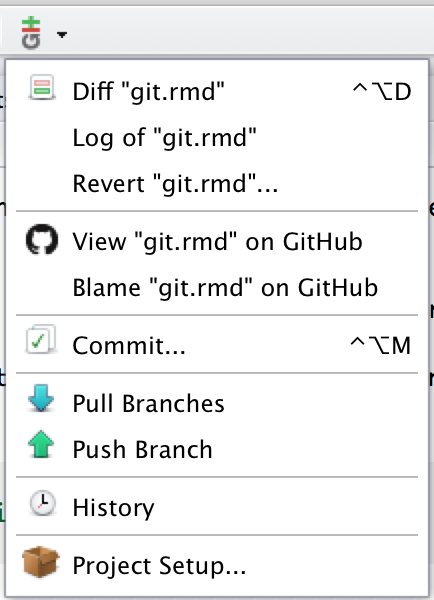
To create a new repo:

1. In RStudio, go to project options, then to the Git/SVN panel. Change the “Version control system” from “None” to “Git”:

You’ll then be prompted to restart RStudio.

1. In a shell, run git init. Restart RStudio and reopen your project.

Once Git has been initialised, you’ll see two new components:

* The **Git panel**, at the top-right, shows you what files have changed and includes buttons for the most important Git commands:
* The **Git dropdown** menu, found in the toolbar, includes Git and GitHub commands that apply to the current file:

**Synchronising with GitHub**

So far we’ve only been working locally, using commits to track the progress of a project and to provide safe checkpoints. However, Git really shines when you start sharing your code with others with [GitHub](http://github.com).

To publish, or **push**, your code to GitHub:

1. Create a new repo on GitHub: <https://github.com/new>. Give it the same name as your package, and include the package title as the repo description. Leave all the other options as is, then click Submit.
2. Open a shell, then follow the instructions on the new repo page. They’ll look something like this:
3. git remote add origin git@github.com:hadley/r-pkgs.git

git push -u origin master

The first line tells Git that your local repo has a remote version on GitHub, and calls it “origin”. The second line pushes all your current work to that repo.

Now let’s make a commit and verify that the remote repo updates:

1. Modify DESCRIPTION to add URL and BugReports fields that link to your new GitHub site. For example, dplyr has:
2. URL: http://github.com/hadley/dplyr

BugReports: http://github.com/hadley/dplyr/issues

1. Save the file and commit (with the message “Updating DESCRIPTION to add links to GitHub site”).
2. **Push** your changes to GitHub by clicking http://r-pkgs.had.co.nz/screenshots/git-push.png. (This is the same as running git push in the shell).
3. Go to your GitHub page and look at the DESCRIPTION.

Usually, each push will include multiple commits. This is because you push much less often than you commit. How often you push versus commit is completely up to you, but pushing code means publishing code. So strive to push code that works.

To ensure your code is clean, I recommend always running R CMD check before you push (a topic you’ll learn about in the chapter on [automated checking](http://r-pkgs.had.co.nz/check.html#check)). If you want to publish code that doesn’t work (yet), I recommend using a branch, as you’ll learn about below in [branching](http://r-pkgs.had.co.nz/git.html#git-branch).

Once you’ve connected your repo to GitHub, the Git pane will show you how many commits you have locally that are not on GitHub: http://r-pkgs.had.co.nz/screenshots/git-local-commits.png. This message indicates that I have 1 commit locally (my branch) that is not on GitHub (“origin/master”).