

# Version Control with RStudio & GitHub

Dana Ghazal

## 1. Why do we use version control?

Version control makes it easy to share code, collaborate on the same project, and keep track of all changes in your code. In this document we will go through the steps of the git installation process and linking RStudio and GitHub.

## 2. Installation

Prior to using Git with RStudio you should install it on your system.

To install git, go to: <https://git-scm.com/downloads>

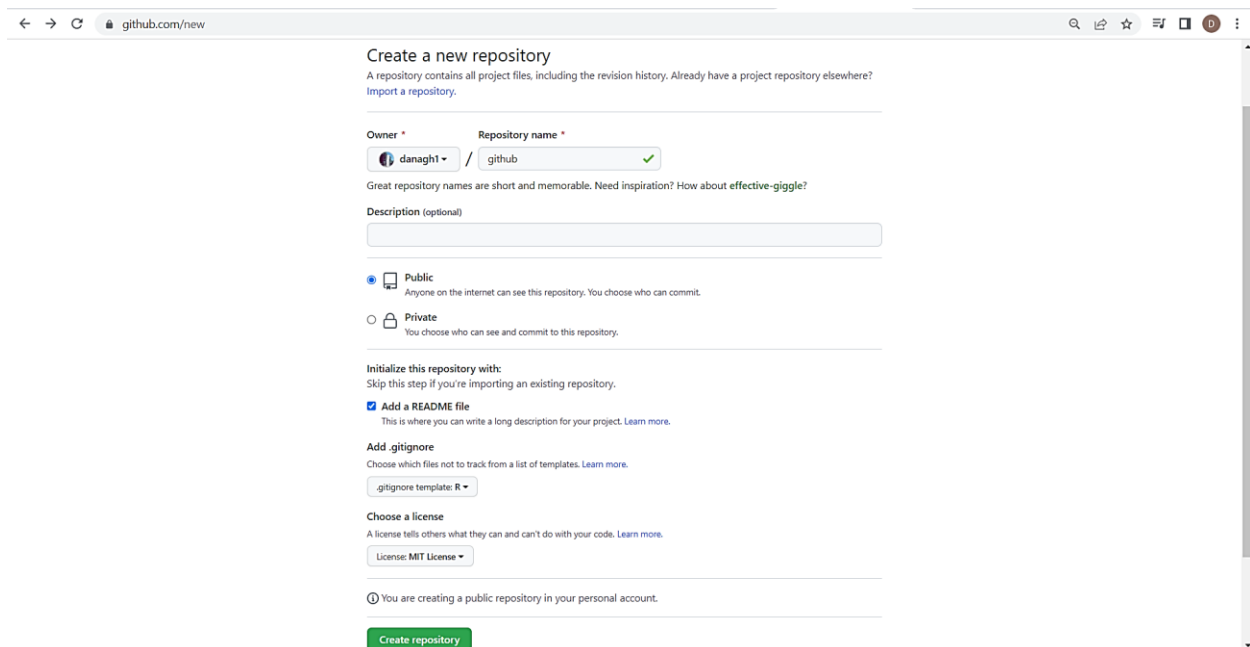
Then you should create a GitHub account. Get a GitHub account at: <https://github.com/>

Git is a version control system, which manages the evolution of files. GitHub is an online tool using the software git to store data and track changes. GitHub can be used with any files but works best with text files, for example R scripts. Here we will focus on using git and GitHub with RStudio.

## 3. Create New Repository & Project

### 1- Create a new repository in GitHub:

a) Log into GitHub and create a new repository.

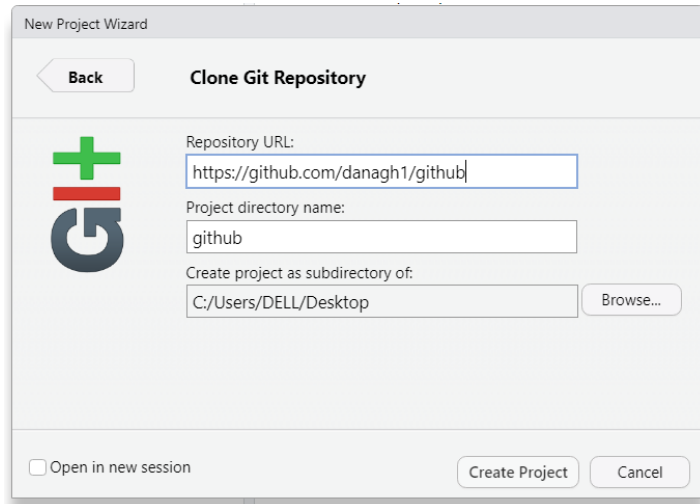


The screenshot shows the GitHub 'Create a new repository' page. The browser address bar shows 'github.com/new'. The page title is 'Create a new repository'. Below the title, it says 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' The form has two columns: 'Owner' and 'Repository name'. The 'Owner' field has a dropdown menu with 'danaghi' selected. The 'Repository name' field has a text input with 'github' and a green checkmark. Below these fields, it says 'Great repository names are short and memorable. Need inspiration? How about effective-giggle?'. The 'Description (optional)' field is a text input. Below the description field, there are two radio buttons for 'Public' (selected) and 'Private'. The 'Public' option says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option says 'You choose who can see and commit to this repository.' Below the visibility options, it says 'Initialize this repository with: Skip this step if you're importing an existing repository.' There are two checkboxes: 'Add a README file' (selected) and 'Add .gitignore'. The 'Add a README file' checkbox says 'This is where you can write a long description for your project. Learn more.' The 'Add .gitignore' checkbox says 'Choose which files not to track from a list of templates. Learn more.' Below the 'Add .gitignore' checkbox, there is a dropdown menu for '.gitignore template: R'. Below the 'Add .gitignore' checkbox, there is a section 'Choose a license' with a text input showing 'License: MIT License'. At the bottom, there is a green button labeled 'Create repository'.

b) Copy the URL of your repository from your browser's address bar

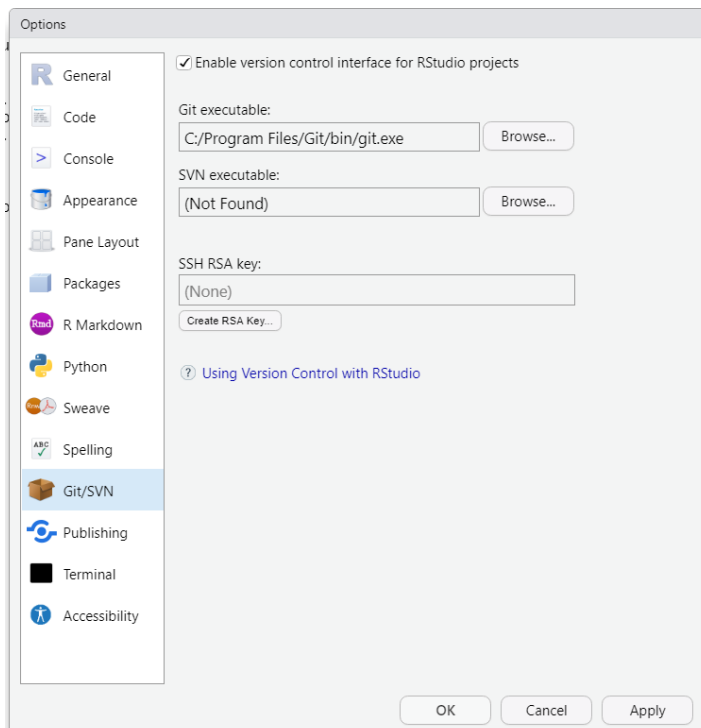
## 2- Create a new project from your GitHub repository in RStudio:

- a) In RStudio, go to File > New Project... > Version Control > Git
- b) Paste your GitHub repo link into the Repository URL box
- c) Optional: edit your Project directory name and select desired subdirectory for project
- d) Click 'Create Project'



## 4. RStudio Setup

In RStudio, go to Tools > Global Options > Git/SVN > make sure the box "Enable version control interface for RStudio projects" is checked.



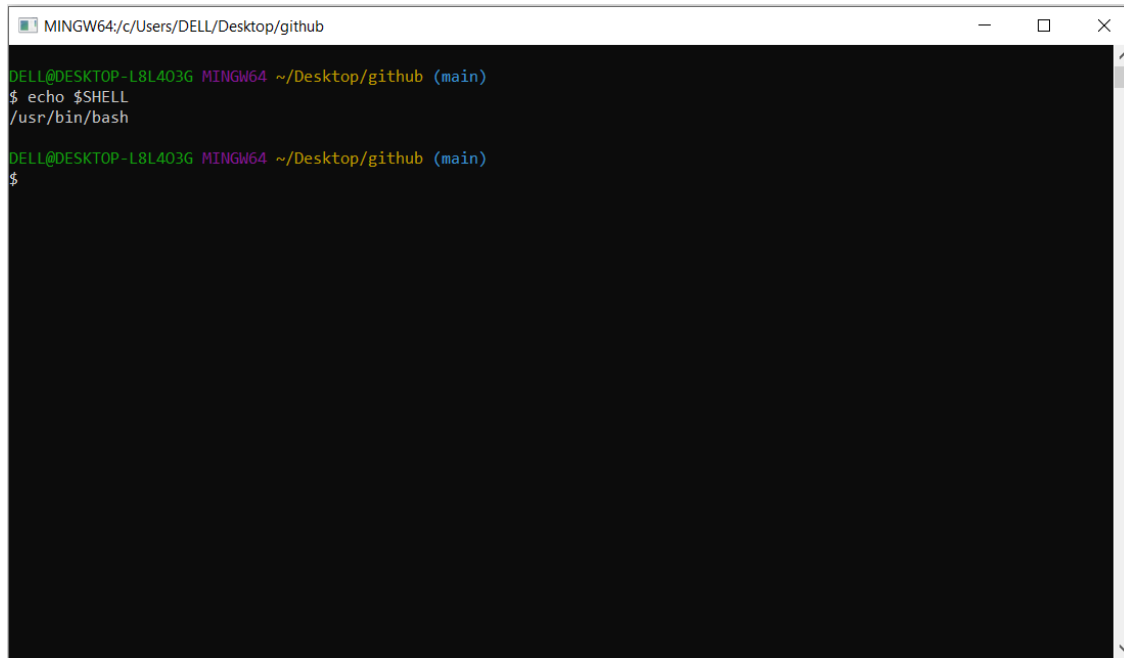
## 5. Git Setup

1- Open Git Bash shell from RStudio - Tools > Shell...

2- Confirm that RStudio is using Git - The shell window name at the top should say something like 'MINGW64' or 'MYSYS.' To double check, you can type the following:

```
echo $SHELL
```

Hit enter. Your window and output should look something like this:

A screenshot of a terminal window titled 'MINGW64:/c/Users/DELL/Desktop/github'. The prompt is 'DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)'. The user enters '\$ echo \$SHELL' and the output is '/usr/bin/bash'. The prompt returns to 'DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)'.

```
MINGW64:/c/Users/DELL/Desktop/github
DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)
$ echo $SHELL
/usr/bin/bash
DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)
$
```

### 3- Configure Git

In your Git Bash shell:

a) Type the following, replacing Your Name with your actual name, and hit Enter:

```
git config user.name "User Name"
```

Git will use your name to keep track of the identity of the person who is making the changes to the files.

b) Type the following, replacing youremail@email.com with the email associated with your GitHub account, and hit Enter:

```
git config user.email "youremail@email.com"
```

c) To check that Git has been properly configured, type the following:

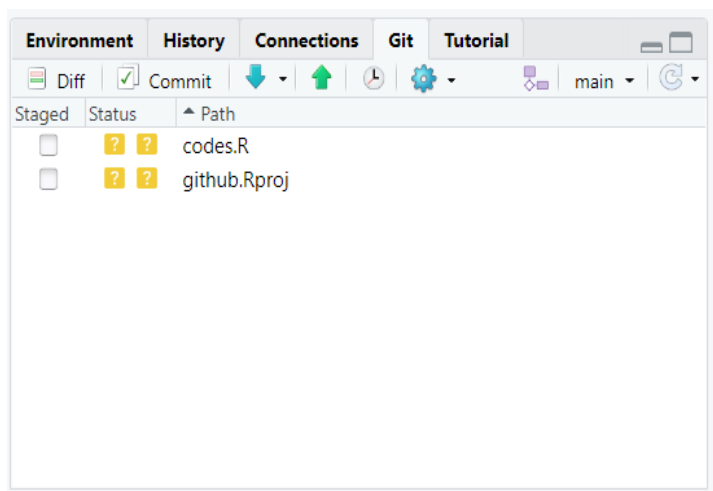
```
git config --list
```

Hit enter and scroll to the bottom to make sure user.name and user.email are correct. If you would like to make any changes, repeat steps a) to c).

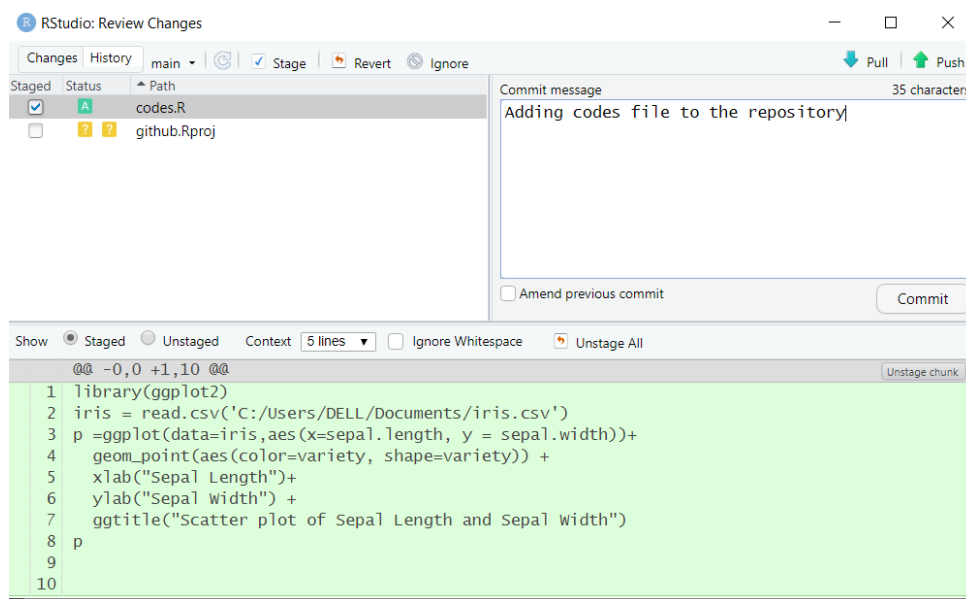
```
MINGW64/c/Users/DELL/Desktop/github
DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)
$ git config user.name "danagh1"
DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)
$ git config user.email "danaghazal2018@outlook.sa"
DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://github.com/danagh1/github
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
user.name="danagh1"
user.email="danaghazal2018@outlook.sa"
DELL@DESKTOP-L8L403G MINGW64 ~/Desktop/github (main)
$
```

## 6. Committing and Pushing Files to GitHub

- 1- Check all of your windows and make sure your Git Bash shell window is not open.
- 2- Go to the Git tab on the top right panel of RStudio. Any changes you make to the project (e.g. edits of existing files, creation of new files, etc.) that have not been committed should show up in the panel.



3- Select the boxes under 'Staged' of the files you want to commit, then click 'Commit'. An 'RStudio: Review Changes' window will pop up. Here, you can see the changes you've made since the last commit. Type a brief description of the changes in the 'Commit message' text box and click 'Commit.'



4- Push the files that committed to GitHub by click 'Push'.

5- Enter your GitHub login credentials as prompted. Your files should be successfully pushed to GitHub.

