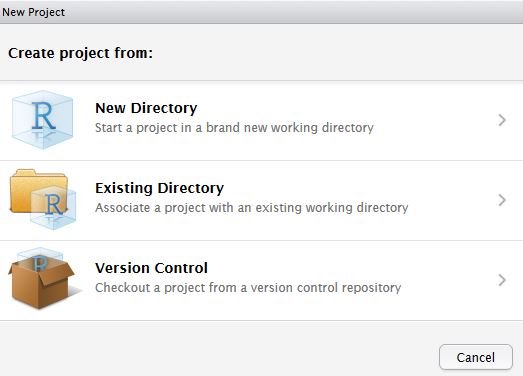
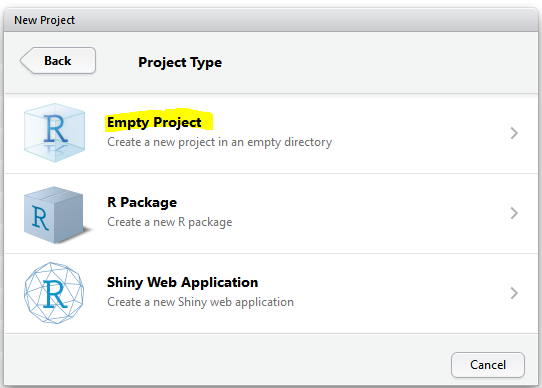
# SECTION 1 – Creating a NEW local project and pushing it to GitHub

Assuming git is correctly setup. Else refer to HELP

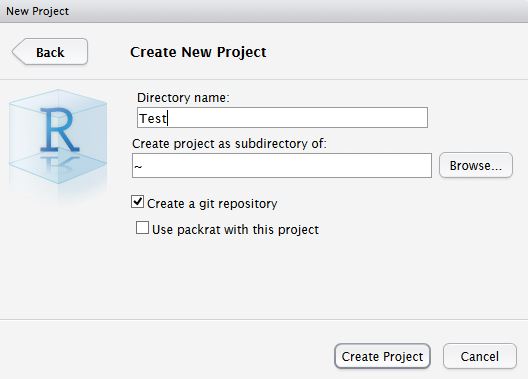
### CREATE NEW PROJECT AND GIT

1. In RStudio, click New project as normal. Click New Directory.





1. Name the project and check Create a git repository.

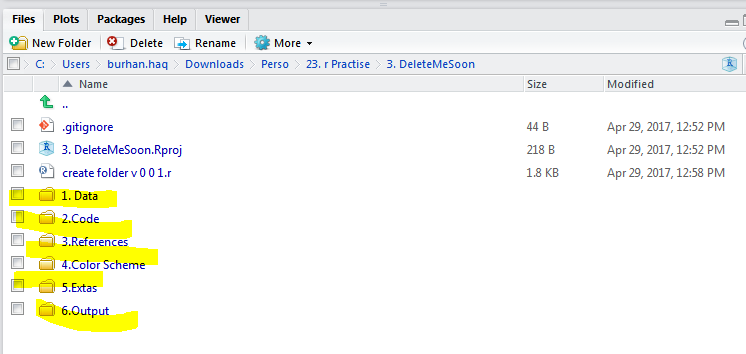


1. Now in RStudio, Copy “create folder v 0 0 1.r” to prepare the project structure.

If you don’t find it – get it from github (<https://github.com/patternproject/r.githubPractise1/blob/master/create%20folder%20v%200%200%201.R>) Click “RAW” to avoid any formatting or other issues.

Or copy from end of this document. (\*SECTION 4: CREATE FOLDER – R CODE\*)

Leave this file you created at the folder root. Do not copy it to the “2.Code” folder. It should have created your folder structure.



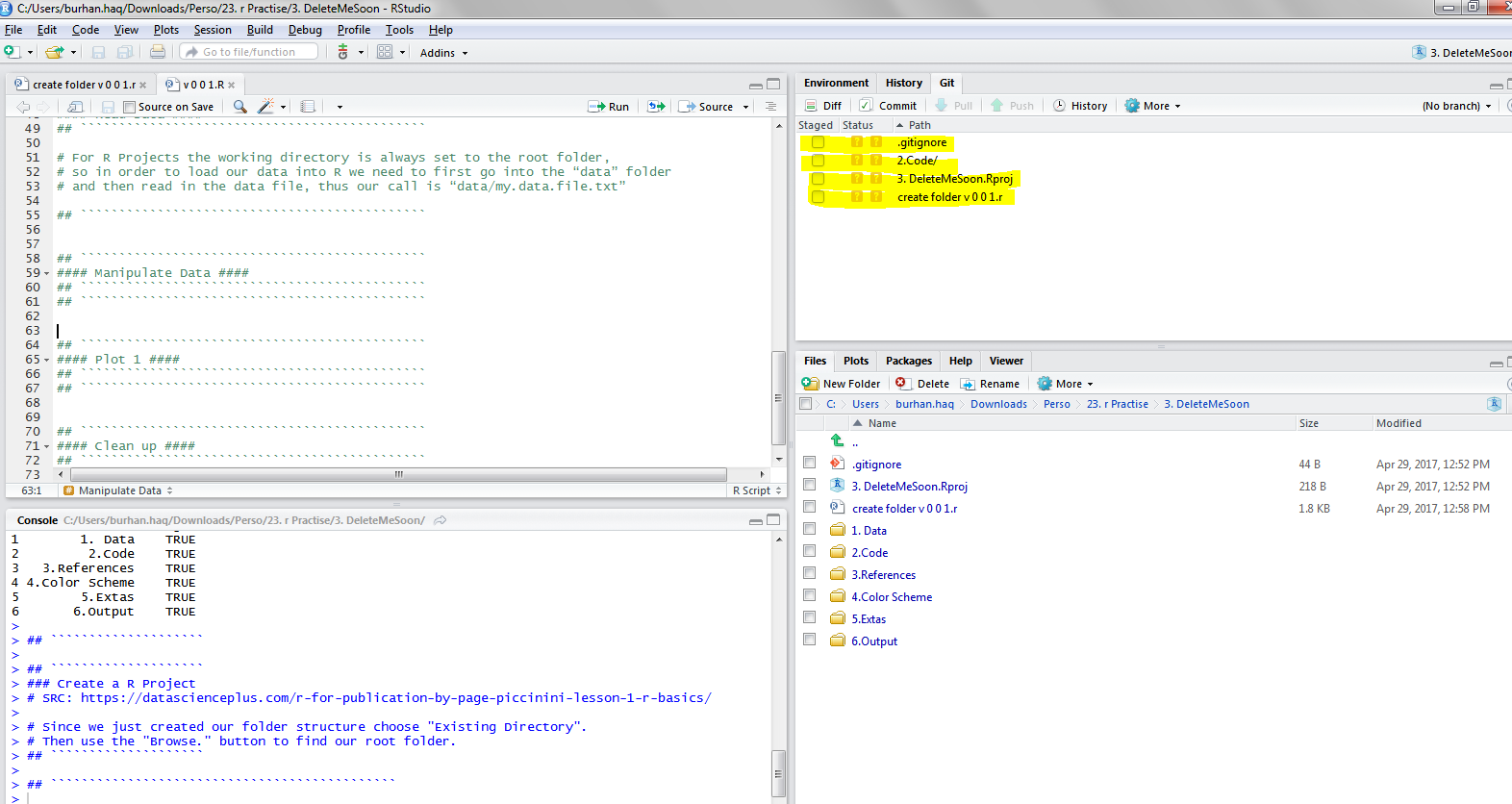
1. Now copy “[template v 0 0 2.R](https://github.com/patternproject/r.githubPractise1/blob/master/2.Code/template%20v%200%200%202.R)” to the folder “2. Source” and rename it as “v 0 0 1”.

If you don’t find it – get it from github (https://github.com/patternproject/r.githubPractise1/blob/master/2.Code/template%20v%200%200%202.R). Click “RAW” to avoid any formatting or other issues.

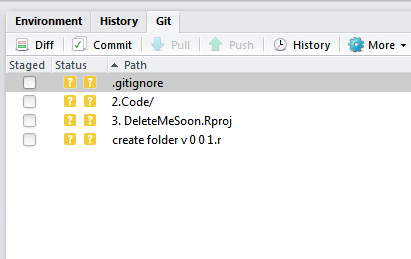
Or copy from end of this document. (\*SECTION 5: TEMPLATE – R CODE\*)

Copy it to the “2.Code” folder. Do not copy it to the source folder.

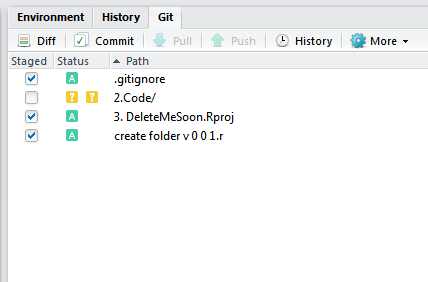
After saving your new scripts, these should appear in the Git tab on the Environment / history panel.

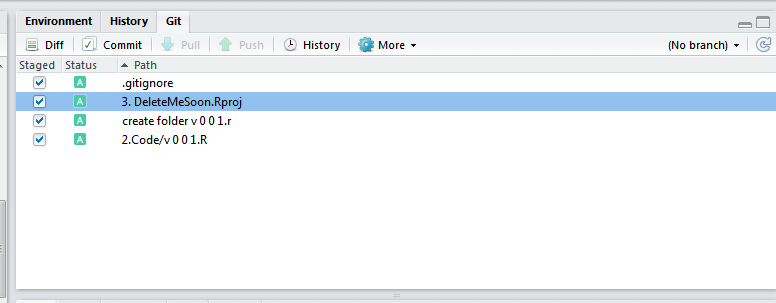


1. Let us now commit what we have done so far. Remember only the folders which have any files will be uploaded to git.

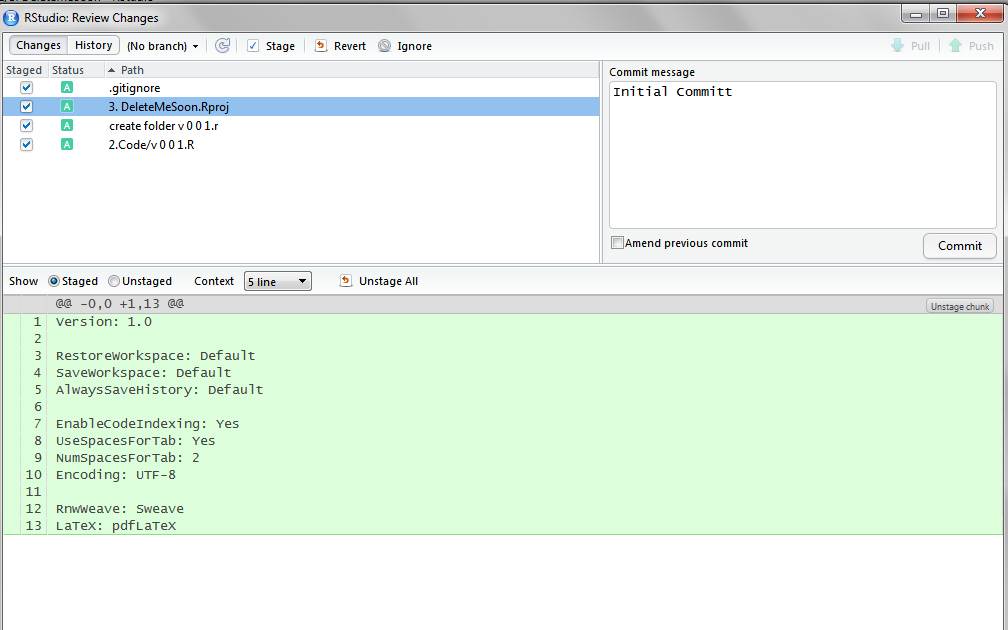


Click the file you wish to add, and the status should turn to a green ‘A’.



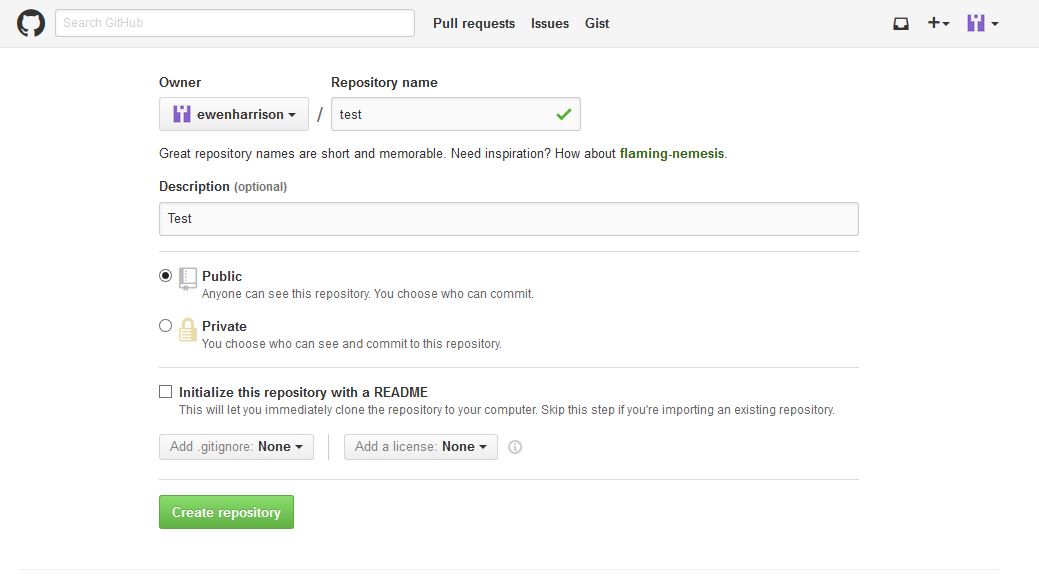


Now click Commit and enter an identifying message in Commit message.



You have now committed the current version of this file to your repository on your computer/server. In the future you may wish to create [*branches*](https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging) to organise your work and help when collaborating.

1. Now you want to push the contents of this commit to GitHub, so it is also backed-up off site and available to collaborators. In GitHub, create a New repository, called here test.



Do not check “Initialize this repository with a README” because:

The reason is that when you create a new project on GitHub, and leave tickbox "Initialise with readme" or choose .gitignore/GPL options, the new project already has a commit you do not have locally, thus the confusion caused by error above.

And you end up with git error:

! [rejected] master -> master (non-fast-forward)

error: failed to push some refs to 'git@github.com:519ebayproject/519ebayproject.git'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')

hint: before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

And remember - The branch name in Git is case sensitive.

So either do not “initialize read me” and follow step 7 where you can safely ignore the pull or look at 7b

1. In RStudio, again click Tools -> Shell … . Enter:

git remote -v

which should be blank as no remote setting are there

Now add

git remote add origin <https://github.com/ewenharrison/test.git>

i.e

git remote add origin <https://github.com/patternproject/DeleteMe.git>

where DeleteMe is the <repo-name>. Remember: case sensitivity with the repo name

For the next step – do not use the old style URL starting with git instead use those starting with https, as the previous step.

~~git config remote.origin.url~~ [~~git@github.com:ewenharrison/test.git~~](mailto:git@github.com:ewenharrison/test.git)

~~i.e~~

~~git config remote.origin.url git@github.com:patternproject/ DeleteMe.git~~

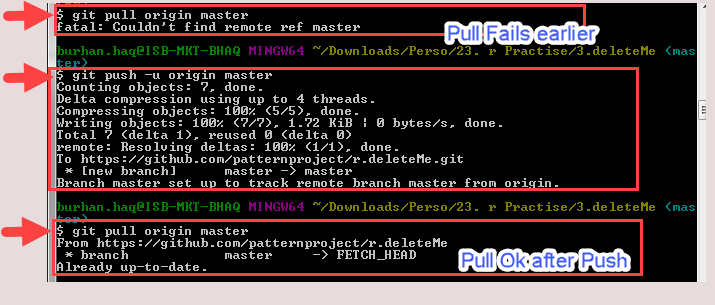
git config remote.origin.url https://github.com/patternproject/ DeleteMe.git

where DeleteMe is the <repo-name>

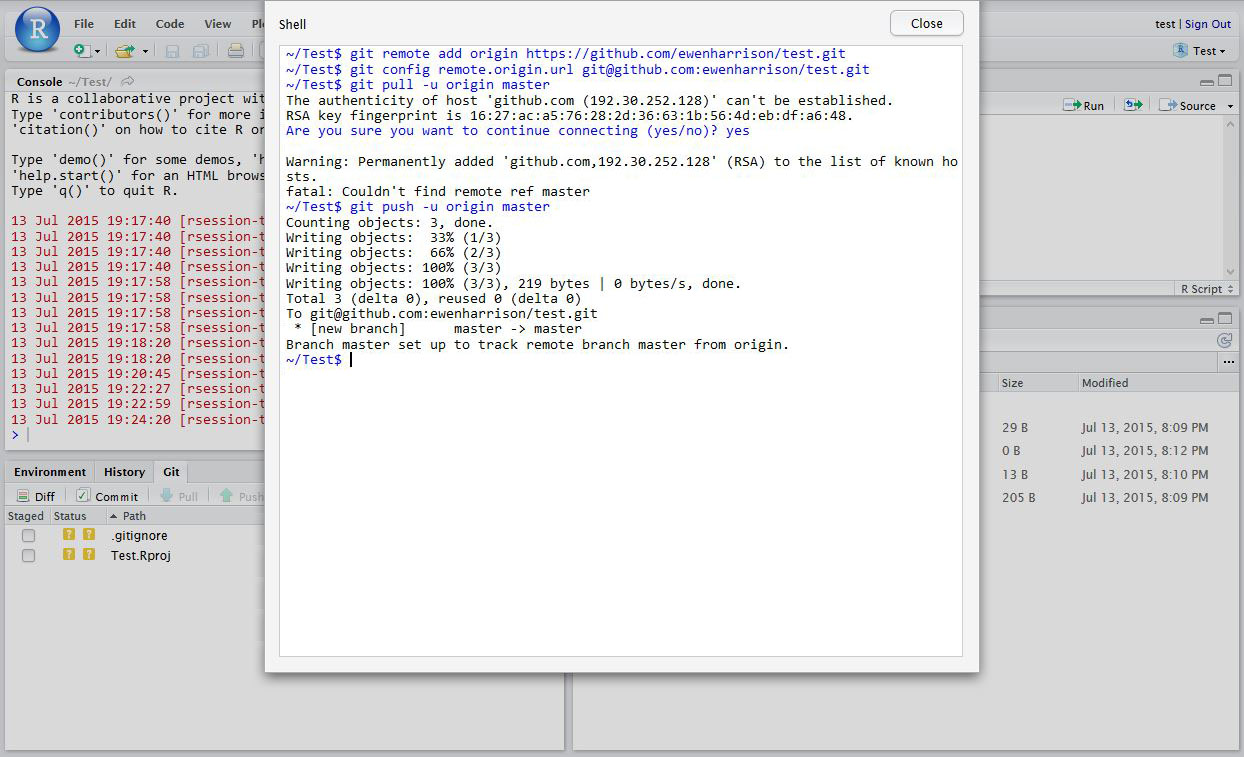
(SRC: <http://stackoverflow.com/questions/13509293/git-fatal-could-not-read-from-remote-repository>)

git push -u origin master

Here is my output from shell in rstudio



Here is the output from the expert in RStudio (his pull before push does not fail somehow)



7b.

When you create a new repository on GitHub, GitHub may ask you to create a readme file. If you create a readme file directly on GitHub, then you will need to first make a ‘pull’ request before the ‘push’ request will be successful. These commands will ‘pull’ the remote repository, merge it with your current files, and then ‘push’ all the files back to GitHub:

git pull https://github.com/thomas07vt/MyFirstRepo.git master

git push https://github.com/thomas07vt/MyFirstRepo.git master

(Src: <http://stackoverflow.com/questions/10298291/cannot-push-to-github-keeps-saying-need-merge>)

You have now pushed your commit to GitHub, and should be able to see your files in your GitHub account. The Pull Push buttons in RStudio will now also work. Remember, after each Commit, you have to Push to GitHub, this doesn’t happen automatically.

-=-=-=

# SECTION 2 - NEED TO ENTER PASSWORD AGAIN AND AGAIN

<http://happygitwithr.com/credential-caching.html#credential-caching>

You may proceed when

* You have a test repo.
* You know where it lives on your local computer. Example:
  + /home/jenny/tmp/myrepo
* You know where it lives on GitHub. Example:
  + https://github.com/jennybc/myrepo
* You know local is tracking remote. In a [shell](http://stat545.com/git09_shell.html) with working directory set to the local Git repo, enter:

git remote -v

Output like this confirms that fetch and push are set to remote URLs that point to your GitHub repo:

origin https://github.com/jennybc/myrepo (fetch)

origin https://github.com/jennybc/myrepo (push)

Now enter:

git branch -vv

Here we confirm that the local master branch has your GitHub master branch (origin/master) as upstream remote. Gibberish? Just check that your output looks similar to mine (or whatever was your last commit comments):

master b8e03e3 [origin/master] line added locally

### Verify that your Git is new enough to have a credential helper

In a [shell](http://stat545.com/git09_shell.html), do:

git --version

and verify your version is 1.7.10 or newer. If not, update Git (chapter [7](http://happygitwithr.com/install-git.html#install-git)) or use SSH keys (chapter [12](http://happygitwithr.com/ssh-keys.html#ssh-keys)).

### Turn on the credential helper

#### Windows

In the shell, enter:

git config --global credential.helper wincred

### Trigger a username / password challenge

Change a file in your local repo and commit it. Do that however you wish.

Now push!

git push -u origin master

One last time you will be asked for your username and password, which hopefully will be cached.

Now push AGAIN.

git push

You should NOT be asked for your username and password, instead you should see Everything up-to-date.

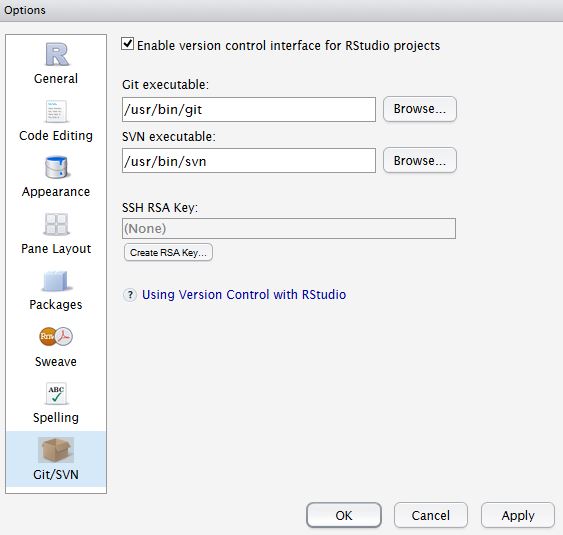
Rejoice and close the shell.

# SECTION 3: SETUP GIT ON RSTUDIO AND ASSOCIATE WITH GITHUB

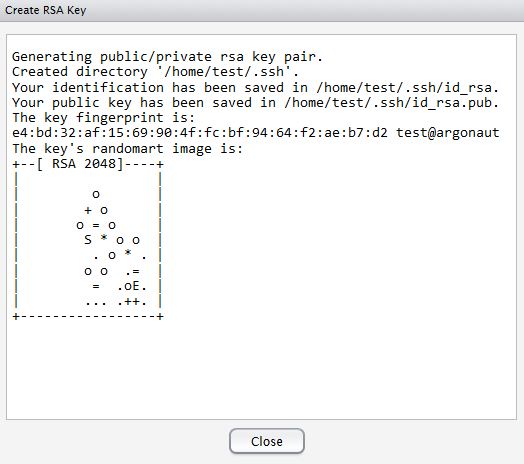
Src: <http://www.datasurg.net/2015/07/13/rstudio-and-github/>

In RStudio, *Tools -> Version Control,*select Git.

In RStudio, *Tools -> Global Options*, select Git//SVN tab. Ensure the path to the Git executable is correct. This is particularly important in Windows where it may not default correctly (e.g. C:/Program Files (x86)/Git/bin/git.exe).

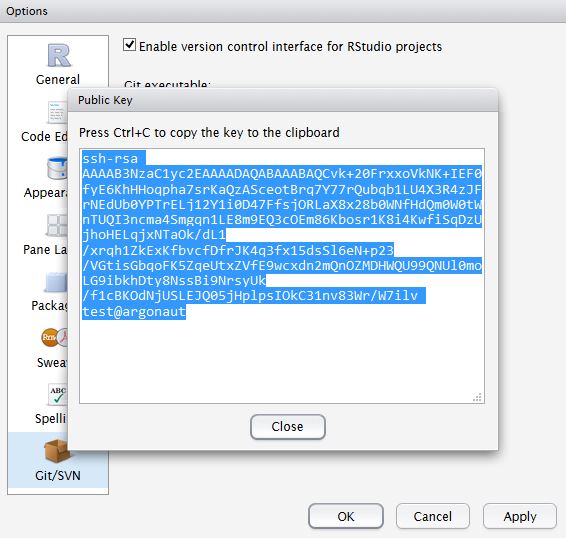


Now hit, Create RSA Key …

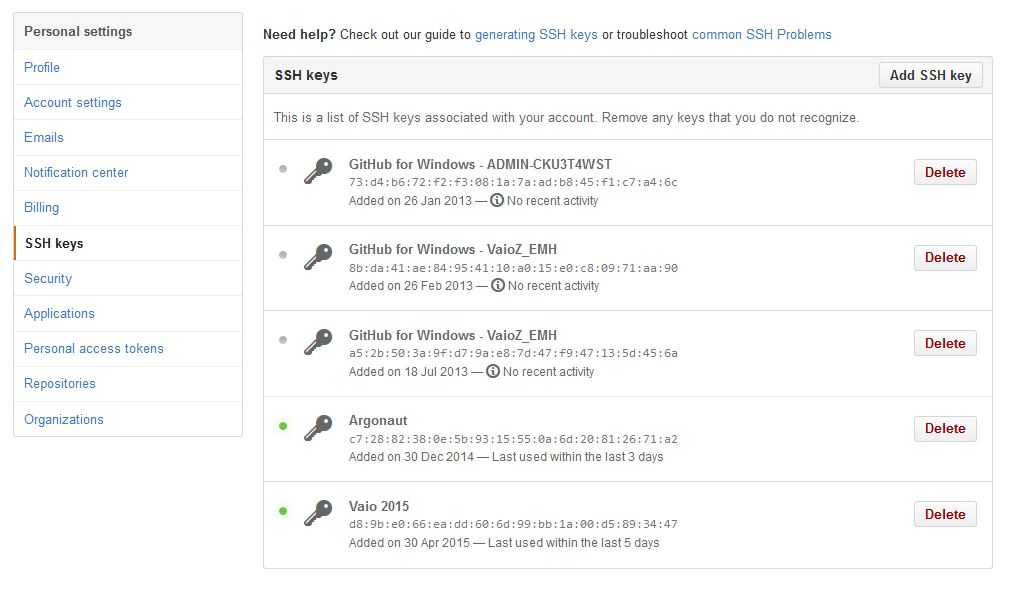


Close this window.

Click, View public key, and copy the displayed public key.



If you haven’t already, create a [GitHub](https://github.com/) account. Open your account settings and click the SSH keys tab. Click Add SSH key. Paste in the public key you have copied from RStudio.



Tell Git who you are. Remember Git is a piece of software running on your own computer. This is distinct to GitHub, which is the repository website. In RStudio, click Tools -> Shell … . Enter:

(The greyed part above will not work – because Shell will only open once you have a project. Instead:

Open git bash from the icon ("D:\Program Files\Git\git-bash.exe" --cd-to-home) And enter the following commands.

git config --global user.email "mail@ewenharrison.com"

git config --global user.name "ewenharrison"

Use your GitHub username.

where: user.email will be patternproject based credentials

-=-=-

# SECTION 4: CREATE FOLDER – R CODE

## `````````````````````````````````````````````

#### Read Me ####

## `````````````````````````````````````````````

## This file creates folder setup in the project directory

##

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Load Libraries ####

## `````````````````````````````````````````````

if (!require("pacman")) install.packages("pacman")

pacman::p\_load(tidyverse)

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Global Settings ####

## `````````````````````````````````````````````

## ````````````````````

### Setting Working Dir

# SRC: http://stackoverflow.com/questions/17605563/efficiently-convert-backslash-to-forward-slash-in-r

# 1. At the prompt copy and paste windows path of project root folder

# x <- readline()

# 2.

# my.dir <- gsub("\\\\", "/", x)

# 3.

# setwd(my.dir)

# 4. check

getwd()

## ````````````````````

## ````````````````````

### Creating Sub folders or Setup Project Folders

# SRC: http://stackoverflow.com/questions/42435225/creating-folders-using-walk-and-purrr/42441681

my.folders <- data.frame(folder = c('1. Data','2.Code','3.References','4.Color Scheme','5.Extas','6.Output'))

my.folders %>%

by\_row(

function(x)

dir.create(as.character(x$folder),

showWarnings = FALSE),

.collate = "rows",

.to = "success"

)

## ````````````````````

## ````````````````````

### Create a R Project

# SRC: https://datascienceplus.com/r-for-publication-by-page-piccinini-lesson-1-r-basics/

# Since we just created our folder structure choose "Existing Directory".

# Then use the "Browse." button to find our root folder.

## ````````````````````

## `````````````````````````````````````````````

-=-=-

# SECTION 5: TEMPLATE – R CODE

\***template v 0 0 2.R**\*

## `````````````````````````````````````````````

#### Read Me ####

## `````````````````````````````````````````````

## This is the template file. Rename it as v 0 0 1.R in Code dir

## And start coding

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Load Libraries ####

## `````````````````````````````````````````````

if (!require("pacman")) install.packages("pacman")

pacman::p\_load(tidyverse)

pacman::p\_load(readxl)

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Helper Function ####

## `````````````````````````````````````````````

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Global Settings ####

## `````````````````````````````````````````````

## ````````````````````

### Setting Working Dir

# SRC: http://stackoverflow.com/questions/17605563/efficiently-convert-backslash-to-forward-slash-in-r

# 1. At the prompt copy and paste windows path of project root folder

x <- readline()

# 2.

my.dir <- gsub("\\\\", "/", x)

# 3.

setwd(my.dir)

# 4. check

getwd()

## ````````````````````

## ````````````````````

### Set theme for plots

# theme\_set(theme\_minimal())

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Read Data ####

## `````````````````````````````````````````````

# For R Projects the working directory is always set to the root folder,

# so in order to load our data into R we need to first go into the “data” folder

# and then read in the data file, thus our call is “data/my.data.file.txt”

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Manipulate Data ####

## `````````````````````````````````````````````

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Plot 1 ####

## `````````````````````````````````````````````

## `````````````````````````````````````````````

## `````````````````````````````````````````````

#### Clean up ####

## `````````````````````````````````````````````

# rm(list=ls())