

EEB603_Github_Tutorial

October 18, 2018

Contents

Set-up in R	1
Aims of this tutorial	1
Introduction	2
What is GitHub	2
Why use Github/RStudio?	2
Creating an account	2
GitHub account	2
Create a test repository:	2
Installing Git on your computer	4
<pre>{r} # check.for.updates.R(notify_user = TRUE, GUI = TRUE, # page_with_download_url = "https://cran.rstudio.com/bin/windows/base/", # pat = "R-[0-9.]+.-win\\.exe") #</pre>	4
Connect RStudio with GitHub	4
Repository options-Possibly Day 2???	5
Creating a branch	6
Pulling, pushing, etc.	6
Paid repository options	7
Student Developer Pack	7
Recommended web resources	8
Glossary	8
References	8

Set-up in R

```
base_data_directory<- "H:/Coursework/EEB603/EEB603Tutorial" result_path <- "H:/Coursework/EEB603/EEB603Tutorial  
setwd(base_data_directory)
```

Load necessary R packages

Aims of this tutorial

-Understand what GitHub is, and when and why you would use it -Set up your own GitHub account, install Git, connect with RStudio

Introduction

What is GitHub

Git is a version control system for tracking files and collaborating with others.

Version control is the management of changes to whatever (code, document, website, etc.). It's sort of like Track Changes in Microsoft Word but much better.

GitHub is the web-based hosting service for Git.

Git was started in 2005 originally for Linux. GitHub launched in 2008.

Why use Github/RStudio?

Manage evolution and changes of data files - reports, figures, code, data (i.e., repositories)

Allows others to see your work, adopt your code in their projects, or let others you are collaborating with make changes

Allows you to track your changes easily, don't need ten million versions saved on your computer/drive, can find changes you made months ago,

It is FREE

Pretty user-friendly

All the cool kids are doing it (seriously, take ecology out of the dark ages)

Creating an account

Online

GitHub account

First register for a GitHub account at www.github.com

Next, we will create a test repository that we will return to later.

Create a test repository:

1. Click on your profile
2. Click "Repositories" at the top
3. Click the green button "New"
4. Name your repository "test_github", write a short description of the repository, make it public, and select initialize with a README, finally select "Create repository" (Refer to Figure 1)

Again, we will return to this repository later in the tutorial.


GitHub, Inc. [US]

https://github.com/new


☆

Owner

Repository name

 katemarkham


/

test_github 


Great repository names are short and memorable. Need inspiration? How about **effective-octo-chainsaw**.

Description (optional)

This repository is only for test purposes

☒  **Public**

Anyone can see this repository. You choose who can commit.


☐  **Private**

You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

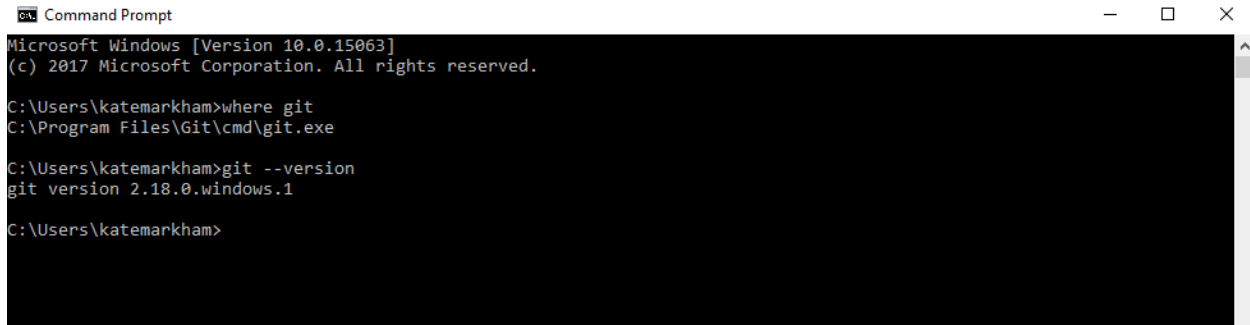
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None** 

Create repository

Figure 1: Figure 1. Creating a repository



```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\katemarkham>where git
C:\Program Files\Git\cmd\git.exe

C:\Users\katemarkham>git --version
git version 2.18.0.windows.1

C:\Users\katemarkham>
```

Figure 2: Figure 2. What to Type in the Command Prompt

Installing Git on your computer

In our opinion, the best guide is ‘happygitwithr’ by Jenny Bryan: <http://happygitwithr.com/> and we recommend having it handy nearby

Follow directions exactly. Note that install instructions differ for Mac, Windows, and Linux.

On your computer

For PCs: 1. Open the Command Prompt (use the magnifying glass to search for it if you’ve never used it before) 2. First, check that Git isn’t already installed by typing in your command prompt: `where git` If you have it installed, go to step 4.

3. If you don’t have Git, install it from this website: <https://gitforwindows.org/> *Install Git into the Program Files folder on your C drive so RStudio can find it.* After the “GNU General Public License”, keep the default options for the next two screens, then for “Adjusting your PATH environment”, select “Use Git from the Windows Command Prompt” (Figure 3), and then continue and keep the remaining defaults.
4. In your Command Prompt, type `git --version` to check that your install was successful (see Figure 2) *If you were not successful, please note that spaces are meaningful when working in Command Prompt and capitalization matters. Make sure you do not have any typos and/or extra spaces.*

In R

Make sure your version of R is updated and current

```
{r} # check.for.updates.R(notify_user = TRUE, GUI = TRUE, #
page_with_download_url = "https://cran.rstudio.com/bin/windows/base/",
#   pat = "R-[0-9.]+.-win\\.exe") #
```

Connect RStudio with GitHub

In R

1. Navigate to the repository you just created
2. Click the green box that says “Clone or download”
3. Copy the full https address in this box

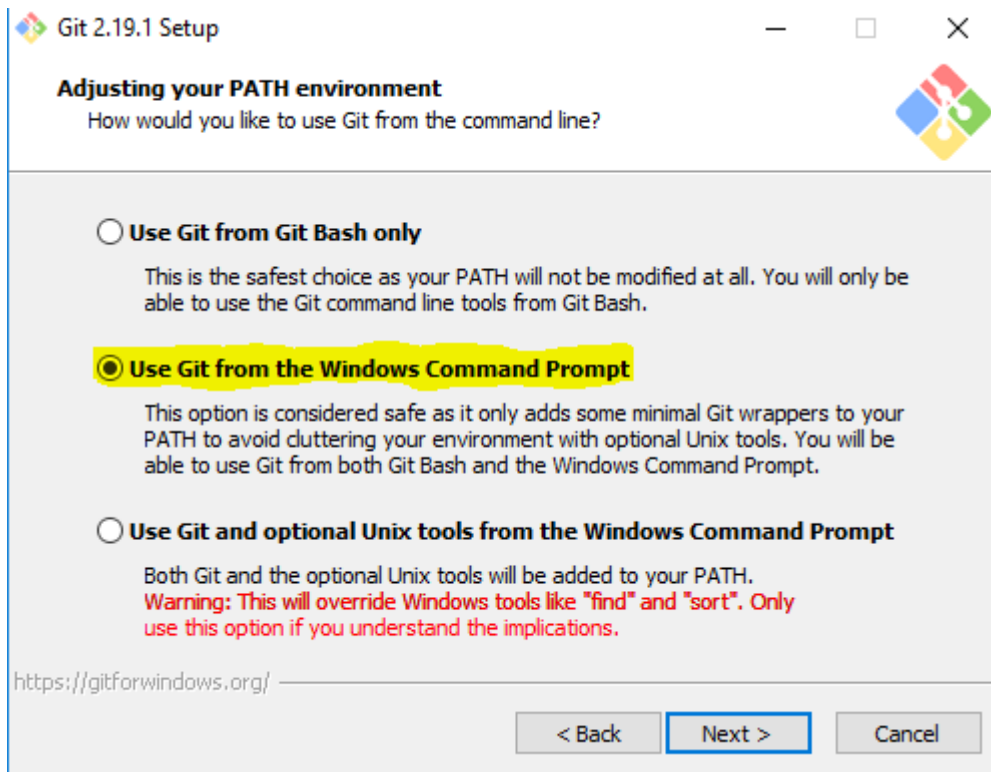


Figure 3: Figure 3. Adjusting your path environment

Return to Command Prompt

1.Type git clone AND THEN THE HTTPS YOU JUST GRABBED 2. Type cd test_github to change the directory to the one you just created 3. Type git remote show origin to check the connection to GitHub

If you were not successful, please note that spaces are meaningful when working in Command Prompt and capitalization matters. Make sure you do not have any typos and/or extra spaces.

In R

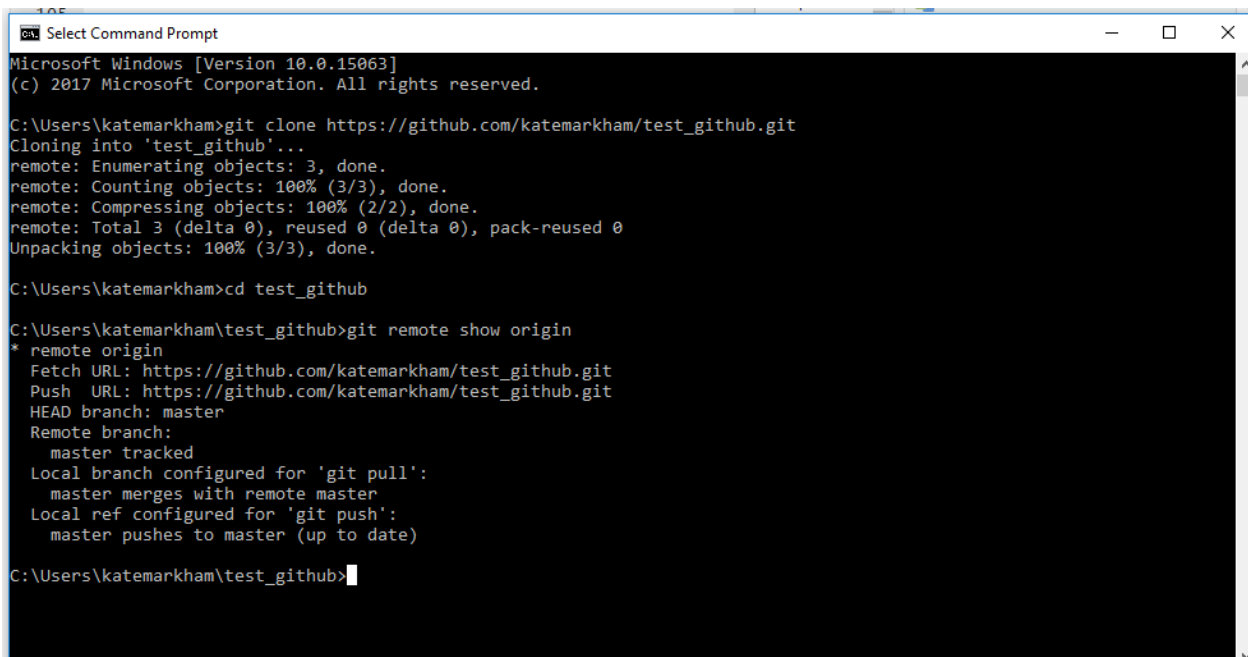
1.File > **New Project** > Version Control > Git 2. Paste repository URL (This autofills project directory name) 3. Pick a directory on your computer and remember where this directory is 4. Create project

Repository options-Possibly Day 2???

1. Go to your online github account and navigate to your repositories
2. Click your test_github repository
3. Click Settings

To change the name of your repository

1.The first option under Settings allows you to type in a new name for your repository

A screenshot of a Windows Command Prompt window titled "Select Command Prompt". The window shows the following text:

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\katemarkham>git clone https://github.com/katemarkham/test_github.git
Cloning into 'test_github'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.

C:\Users\katemarkham>cd test_github

C:\Users\katemarkham\test_github>git remote show origin
* remote origin
Fetch URL: https://github.com/katemarkham/test_github.git
Push URL: https://github.com/katemarkham/test_github.git
HEAD branch: master
Remote branch:
  master tracked
Local branch configured for 'git pull':
  master merges with remote master
Local ref configured for 'git push':
  master pushes to master (up to date)

C:\Users\katemarkham\test_github>
```

Figure 4: Figure 4. Connecting your repository to Git

To add a collaborator

1. On the lefthand side of Settings, select “Collaborators”
2. Include either the username or email address of the person you wish to collaborate with **Email addresses will only work if the user has chosen to make their email address public

Creating a branch

Branches of a repository allow you to work on the code without changing the master branch.

For example, many packages on GitHub will have a master branch and a developer branch. The developer branch is often a work in progress whereas the master branch is the version users not looking to edit the package will download.

To create a branch from GitHub 1. Navigate to the repository on GitHub 2. Click Branch:master 3. In the drop-down menu, type in whatever you want the name of your new branch to be

Pulling, pushing, etc.

Once you’ve edited the file and are ready to share it, click the Git button in RStudio

1. Select Commit
2. First, check that you aren’t missing any changes from collaborators and pull. This small step makes sure you don’t jump ahead of your colleagues and then need to backtrack and merge projects that no longer line up correctly.
3. Select the changes you want to share by staging them. Additions are in green and deletions are in red. Note: you can stage entire files or you can stage chunks of code or even lines of code. What is staged is shared, what is unstaged is not.
4. Comment your changes. Tip: write something meaningful you will understand later and that is useful to collaborators.

RStudio and Git

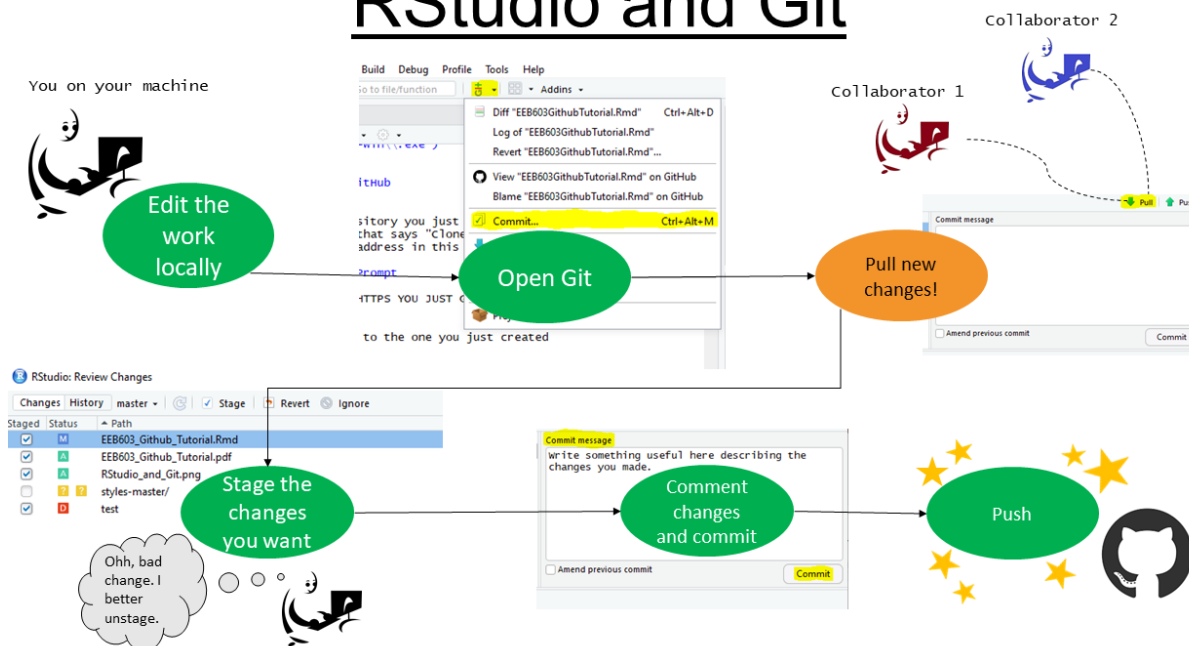


Figure 5: Figure 5. Sharing your code with RStudio and Git

5. Commit your changes

6. Push your changes. **This is when your work goes to the GitHub repository!**

##.gitignore If you have files that you want to keep on your local machine but do not want to share to GitHub for whatever reason, you can do this through your .gitignore file within your repository.

On your local machine 1. Navigate to where your repository is on your machine and open .gitignore in a text editor 2. Add the file you want to ignore 3. Save it

On GitHub 1. Navigate to your repository and open .gitignore 2. Click the pencil next to the trashcan symbol 3. Add the file you want to ignore

Paid repository options

Some of the features on GitHub require a paid plan. <https://github.com/pricing>

1. Private repositories
2. Team permissions
3. Backups

Student Developer Pack

<https://education.github.com/pack>

With the student developer pack, users have 1. Unlimited private repositories 2. Unlimited collaborators

Recommended web resources

<http://happygitwithr.com/> <http://r-bio.github.io/intro-git-rstudio/> <https://services.github.com/on-demand/resources/> <https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet> <https://guides.github.com/activities/hello-world/>

Glossary

Branch-Parallel version of a repository. Working on a branch allows you to work without changing the live version on the master branch. **Clone**-Copy of a repository that is on your machine rather than on the server. (Can also act as a verb.) You can sync changes in your cloned copy to remote versions when you are online. **Commit**-Changes to a file. It's similar to a "save as" but you do not need to rename the file (Git automatically keeps track of each change). **Fetch**-Grabbing the latest version from an online repository-does not automatically merge this version with your own. **Fork**-Your own copy of another user's repository that remains on your own account. This allows you to make changes without affecting the original. You can pull updates from the original user's repository. **Pull**-Grabs new changes and merges them with your own copies. If you need to grab new changes but not merge them, see "fetch." **Push**- Send your changes to a remote repository. If you make changes and you want your collaborators to see them, you need to push your changes. **Repository**-The place where all of the project's files are stored. A repository also records the file's revision history. Can be public or private and can have multiple collaborators.

For more terms, see <https://help.github.com/articles/github-glossary/>

References